Bag-of-Words Representations for Computer Audition

DISSERTATION

for the attainment of the degree of
Doctor of Engineering (Doktor-Ingenieur)
at the
Faculty of Applied Computer Science
of the
University of Augsburg

by

Maximilian Lukas Schmitt

2021

Referees:	Prof. Dr.-Ing. habil. Björn W. Schuller (University of Augsburg) Prof. Dr. rer. nat. Elisabeth André (University of Augsburg) Prof. Dr.-Ing. Dorothea Kolossa (Ruhr University Bochum)
Date of the oral exam:	18 March 2022

Acknowledgements

This thesis is based on the outcomes of my research at the *Chair of Embedded Intelligence for Health Care and Wellbeing* at the Faculty of Applied Computer Science of the University of Augsburg and at the *Chair of Complex & Intelligent Systems* of the University of Passau (January 2015 – September 2017). Parts of the research presented in this thesis were supported by the European Union's Horizon2020 programme under grant agreement No. 645094 (IA SEWA) and by the 7$^{\text{th}}$ Framework Programme through the ERC Starting Grant No. 338164 (iHEARu).

My very special thanks go to my supervisor, Prof. Dr.-Ing. habil. Björn W. Schuller, for his continuous inspiration, support, and his useful advice. Working in his research group, dealing with several interdisciplinary projects, leading to extraordinary results, were a key factor for the success of this thesis.

My further special thanks go to Prof. Dr. rer. nat. Elisabeth André and Prof. Dr.-Ing. Dorothea Kolossa for reviewing my thesis and being part of the examination board.

I would like to thank very much all my fellows and colleagues, from whom I have learnt a lot, for the great cooperation in Passau, Munich, Augsburg, and London. In particular, these are (in my order of meeting them for the first time): Dr. Shahin Amiriparian, Dr. Vedhas Pandit, Dr. Jing Han, Dr. Fabien Ringeval, Dr. Hesam Sagha, Dr. Simone Hantke, Dr. Zixing Zhang, Dr. Gil Keren, Dr. Anton Batliner, Dr. Ognjen Rudovic, Dr. Florian Eyben, Dr. Felix Weninger, Dr. Yue Zhang-Weninger, Dr. Eduardo Coutinho, Dr. Erik Marchi, Dr. Jun Deng, Prof. Kun Qian, Dr. Xinzhou Xu, Dr. Amr Mousa, Dr. Bogdan Vlasenko, Dr. Jouni Pohjalainen, Dr. Nicholas Cummins, Dr. Jude Dineley, Dr. George Trigeorgis, Dr. Panagiotis Tzirakis, Dr. Florian B. Pokorny, Dr. Katrin D. Bartl-Pokorny, Dr. Zhao Ren, Maryna Gavryukova, Alice Baird, Thomas Kehrenberg, Gerhard Hagerer, Zijiang Yang, Meishu Song, Lukas Stappen, Andreas Triantafyllopoulos,

Manuel Milling, Adrià Mallol Ragolta, Shuo Liu, Vincent Karas, Maurice Gerczuk, & Sandra Ottl, and our guest researchers Prof. Wen-Hsing Lai, Prof. Ziping Zhao, Dr. Ya'nan Guo, & Dr. Francesco Barone.

Very special thanks go to the office managers at the chairs, Andrea Tonk and Nadine Witek, who have always supported me on many issues.

During my research, I have had the pleasure of profiting from very fruitful collaborations with a number of external colleagues and students, whom I would like to thank, in particular: Dr. Christoph Janott, Dr. Clemens Heiser, & Prof. Werner Hemmert from the 'snore sounds' project; Prof. Maja Pantic, Dr. Jie Shen, Robert Walecki, Dr. Yannis Panagakis, Dr. Stavros Petridis, Dr. Jean Kossaifi, Teresa Ng, & Dionysia Kordopati, all of them working with me on the SEWA project; Dr. Stefan Steidl, Christian Bergler, Prof. Elmar Nöth, Prof. Jarek Krajewski, Prof. Sebastian Schnieder, Dr. Eva-Maria Meßner, & Prof. Stefanos Zafeiriou, with whom I was working on Interspeech ComParE from 2017 to 2020; Dr. Heysem Kaya, Dr. Albert Ali Salah, Dr. Mohammad Soleymani, Siyang Song, Leili Tavabi, & Dr. Michel Valstar, with whom I was working on AVEC from 2017 to 2019; Prof. Yoshiharu Yamamoto & Tomoya Koike from the University of Tokyo, and our former students Christoph Hausner & Stephan Wolf.

Most of all, I would like to thank especially my girlfriend Emi, my parents Ulrike and Gunter, my grandmother Irmtraud, and the rest of my whole family and all of my friends for their active support, their constant encouragement and love.

Linz (Austria), May 2021

Maximilian L. Schmitt

Abstract

Computer audition is omnipresent in everyday life, in applications ranging from personalised virtual agents to health care. From a technical point of view, the goal is to robustly classify the content of an audio signal in terms of a defined set of labels, such as, e. g., the acoustic scene, a medical diagnosis, or, in the case of speech, 'what' is said or 'how' it is said. Typical approaches employ machine learning (ML), which means that task-specific models are trained by means of examples. Despite recent successes in neural network-based end-to-end learning, taking the raw audio signal as input, models relying on hand-crafted acoustic features are still superior in some domains, especially for tasks where data is scarce. One major issue is nevertheless that a sequence of acoustic low-level descriptors (LLDs) cannot be fed directly into many ML algorithms as they require a static and fixed-length input. Moreover, also for dynamic classifiers, compressing the information of the LLDs over a temporal block by summarising them can be beneficial. However, the type of instance-level representation has a fundamental impact on the performance of the model. In this thesis, the so-called *bag-of-audio-words* (BoAW) representation is investigated as an alternative to the 'standard' approach of statistical functionals. BoAW is an unsupervised method of representation learning, inspired from the *bag-of-words* method in natural language processing, forming a histogram of the terms present in a document. The toolkit openXBOW is introduced, enabling systematic learning and optimisation of these feature representations, unified across arbitrary modalities of numeric or symbolic descriptors. A number of experiments on BoAW are presented and discussed, focussing on a large number of potential applications and corresponding databases, ranging from emotion recognition in speech to medical diagnosis. The evaluations include a comparison of different acoustic LLD sets and configurations of the BoAW generation process. The key findings are that BoAW features are a meaningful alternative to statistical functionals, offering certain benefits, while being able to preserve the advantages of functionals, such as data-independence. Furthermore, it is shown that both representations are complementary and their fusion improves the performance of a machine listening system.

Zusammenfassung

Maschinelles Hören ist im täglichen Leben allgegenwärtig, mit Anwendungen, die von personalisierten virtuellen Agenten bis hin zum Gesundheitswesen reichen. Aus technischer Sicht besteht das Ziel darin, den Inhalt eines Audiosignals hinsichtlich einer Auswahl definierter Labels robust zu klassifizieren. Die Labels beschreiben bspw. die akustische Umgebung der Aufnahme, eine medizinische Diagnose oder—im Falle von Sprache—*was* gesagt wird oder *wie* es gesagt wird. Übliche Ansätze hierzu verwenden maschinelles Lernen, d. h., es werden anwendungsspezifische Modelle anhand von Beispieldaten trainiert. Trotz jüngster Erfolge beim *Ende-zu-Ende-Lernen* mittels neuronaler Netze, in welchen das unverarbeitete Audiosignal als Eingabe benutzt wird, sind Modelle, die auf definierten akustischen Merkmalen basieren, in manchen Bereichen weiterhin überlegen. Dies gilt im Besonderen für Einsatzzwecke, für die nur wenige Daten vorhanden sind. Allerdings besteht dabei das Problem, dass Zeitfolgen von akustischen Deskriptoren in viele Algorithmen des maschinellen Lernens nicht direkt eingespeist werden können, da diese eine statische Eingabe fester Länge benötigen. Außerdem kann es auch für dynamische (zeitabhängige) Klassifikatoren vorteilhaft sein, die Deskriptoren über ein gewisses Zeitintervall zusammenzufassen. Jedoch hat die Art der Merkmalsdarstellung einen grundlegenden Einfluss auf die Leistungsfähigkeit des Modells. In der vorliegenden Dissertation wird der sogenannte *Bag-of-Audio-Words-Ansatz* (BoAW) als Alternative zum Standardansatz der statistischen Funktionale untersucht. BoAW ist eine Methode des unüberwachten Lernens von Merkmalsdarstellungen, die von der *Bag-of-Words*-Methode in der Computerlinguistik inspiriert wurde, bei der ein Textdokument als Histogramm der vorkommenden Wörter beschrieben wird. Das Toolkit openXBOW wird vorgestellt, welches systematisches Training und Optimierung dieser Merkmalsdarstellungen—vereinheitlicht für beliebige Modalitäten mit numerischen oder symbolischen Deskriptoren—erlaubt. Es werden einige Experimente zum BoAW-Ansatz durchgeführt und diskutiert, die sich auf eine große Zahl möglicher Anwendungen und entsprechende Datensätze beziehen, von der Emotionserkennung in gesprochener Sprache bis zur medizinischen Diagnostik.

Die Auswertungen beinhalten einen Vergleich verschiedener akustischer Deskriptoren und Konfigurationen der BoAW-Methode. Die wichtigsten Erkenntnisse sind, dass BoAW-Merkmalsvektoren eine geeignete Alternative zu statistischen Funktionalen darstellen, gewisse Vorzüge bieten und gleichzeitig wichtige Eigenschaften der Funktionale, wie bspw. die Datenunabhängigkeit, erhalten können. Zudem wird gezeigt, dass beide Darstellungen komplementär sind und eine Fusionierung die Leistungsfähigkeit eines Systems des maschinellen Hörens verbessert.

List of Publications and Awards

Author Profiles

- ORCID:
  `https://orcid.org/0000-0001-7453-5612`
- Google Scholar:
  `https://scholar.google.com/citations?user=bTLclOYAAAAJ`

Journal Articles

- **Maximilian Schmitt** & Björn Schuller: *openXBOW – Introducing the Passau Open-Source Crossmodal Bag-of-Words Toolkit*, The Journal of Machine Learning Research, volume 18, no. 96, pp. 1–5, 2017.
- Christoph Janott, **Maximilian Schmitt**, Yue Zhang, Kun Qian, Vedhas Pandit, Zixing Zhang, Clemens Heiser, Winfried Hohenhorst, Michael Herzog, Werner Hemmert, & Björn Schuller: *Snoring classified: The Munich-Passau Snore Sound Corpus*, Computers in Biology and Medicine, volume 94, pp. 106–118, Elsevier, 2018.
- Paul Buitelaar, Ian D. Wood, Sapna Negi, Mihael Arcan, John P. McCrae, Andrejs Abele, Cécile Robin, Vladimir Andryushechkin, Housam Ziad, Hesam Sagha, **Maximilian Schmitt**, Björn W. Schuller, J. Fernando Sánchez-Rada, Carlos A. Iglesias, Carlos Navarro, Andreas Giefer, Nicolaus Heise, Vincenzo Masucci, Francesco A. Danza, Ciro Caterino, Pavel Smrž, Michal Hradiš, Filip Povolný, Marek Klimeš, Pavel Matějka, & Giovanni Tummarello: *MixedEmotions: An Open-Source Toolbox for Multi-Modal Emotion Analysis*, IEEE Transactions on Multimedia, volume 20, no. 9, pp. 2454–2465, IEEE, 2018.
- Kun Qian, **Maximilian Schmitt**, Christoph Janott, Zixing Zhang, Clemens Heiser, Winfried Hohenhorst, Michael Herzog, Werner Hemmert, & Björn Schuller: *A Bag of Wavelet Features for Snore Sound Classification*, Annals of Biomedical Engineering, volume 47, no. 4, pp. 1000–1011, Biomedical Engineering Society, 2019.

- Emilia Parada-Cabaleiro, Giovanni Costantini, Anton Batliner, **Maximilian Schmitt**, & Björn Schuller: *DEMoS – An Italian Emotional Speech Corpus: Elicitation Methods, Machine Learning, and Perception*, Language Resources and Evaluation, volume 54, pp. 341–383, Springer, 2019.

- Christoph Janott, **Maximilian Schmitt**, Clemens Heiser, Winfried Hohenhorst, Michael Herzog, Marina Carrasco Llatas, Werner Hemmert, & Björn Schuller: *VOTE versus ACLTE: Vergleich zweier Schnarchgeräuschklassifikationen mit Methoden des maschinellen Lernens*, HNO Journal, volume 67, no. 9, pp. 670–678, Springer, 2019.

- Shahin Amiriparian, Jing Han, **Maximilian Schmitt**, Alice Baird, Adrià Mallol-Ragolta, Manuel Milling, Maurice Gerczuk, & Björn Schuller: *Synchronization in Interpersonal Speech*, Frontiers in Robotics and AI, section Human-Robot Interaction, volume 6, no. 116, pp. 1–10, Frontiers, 2019.

- Kun Qian, Christoph Janott, **Maximilian Schmitt**, Zixing Zhang, Clemens Heiser, Werner Hemmert, Yoshiharu Yamamoto, & Björn W. Schuller: *Can Machine Learning Assist Locating the Excitation of Snore Sound? A Review*, IEEE Journal of Biomedical And Health Informatics, volume 25, no. 4, pp. 1233–1246, IEEE, 2020.

- Vedhas Pandit, **Maximilian Schmitt**, Nicholas Cummins, & Björn Schuller: *I see it in your eyes: Training the Shallowest-possible CNN to Recognise Emotions and Pain from Muted Web-assisted in-the-wild Video-chats in Real-time*, Information Processing and Management, volume 57, no. 6, Elsevier, 2020.

- Stephan Wolf, **Maximilian Schmitt**, & Björn Schuller: *A Football Player Rating System*, Journal of Sports Analytics, volume 6, no. 4, pp. 243–257, IOS Press, 2020.

- Jean Kossaifi, Robert Walecki, Yannis Panagakis, Jie Shen, **Maximilian Schmitt**, Fabien Ringeval, Jing Han, Vedhas Pandit, Antoine Toisoul, Bjorn Schuller, Kam Star, Elnar Hajiyev, & Maja Pantic: *SEWA DB: A Rich Database for Audio-Visual Emotion and Sentiment Research in the Wild*, IEEE Transactions on Pattern Analysis and Machine Intelligence, volume 43, no. 3, pp. 1022–1040, IEEE, 2021.

- Kun Qian, **Maximilian Schmitt**, Huaiyuan Zheng, Tomoya Koike, Jing Han, Juan Liu, Wei Ji, Junjun Wei, Meishu Song, Zijiang Yang, Zhao Ren, Shuo Liu, Zixing Zhang, Yoshiharu Yamamoto, & Björn Schuller: *Computer Audition for Fighting the SARS-CoV-2 Corona Crisis – Introducing the Multi-task Speech Corpus for COVID-19*, IEEE Internet of Things Journal, pp. 1–12, IEEE, 2021.

Publications in Conference Proceedings

- **Maximilian Schmitt**, Fabien Ringeval, & Björn Schuller: *At the Border of Acoustics and Linguistics: Bag-of-Audio-Words for the Recognition of Emotions in Speech*, Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH), San Francisco, CA, USA, pp. 495–499, ISCA, 2016.

- **Maximilian Schmitt**, Christoph Janott, Vedhas Pandit, Kun Qian, Clemens Heiser, Werner Hemmert, & Björn Schuller: *A Bag-of-Audio-Words Approach for Snore Sounds' Excitation Localisation*, Proc. ITG Speech Communication, Paderborn, Germany, pp. 230–234, VDE, IEEE, 2016.

- **Maximilian Schmitt**, Erik Marchi, Fabien Ringeval, & Björn Schuller: *Towards Cross-lingual Automatic Diagnosis of Autism Spectrum Condition in Children's Voices*, Proc. ITG Speech Communication, Paderborn, Germany, pp. 264–268, VDE, IEEE, 2016.

- Johanna Böhm, Florian Eyben, **Maximilian Schmitt**, Harald Kosch, & Björn Schuller: *Seeking the SuperStar: Automatic Assessment of Perceived Singing Quality*, Proc. International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, pp. 1560–1569, IEEE, 2017.

- Nicholas Cummins, **Maximilian Schmitt**, Shahin Amiriparian, Jarek Krajewski, & Björn Schuller: *"You sound ill, take the day off": Automatic Recognition of Speech Affected by Upper Respiratory Tract Infection*, Proc. Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Jeju Island, Korea, pp. 3806–3809, EMBS, 2017.

- Tobias Geib, **Maximilian Schmitt**, & Björn Schuller: *Automatic Guitar String Detection by String-Inverse Frequency Estimation*, Proc. INFORMATIK 2017, Chemnitz, Germany, pp. 127–138, GI, 2017.

- Jun Deng, Nicholas Cummins, **Maximilian Schmitt**, Kun Qian, Fabien Ringeval, & Björn Schuller: *Speech-based Diagnosis of Autism Spectrum Condition by Generative Adversarial Network Representations*, Proc. Digital Health, London, UK, pp. 53–57, ACM, 2017.

- Björn Schuller, Stefan Steidl, Anton Batliner, Elika Bergelson, Jarek Krajewski, Christoph Janott, Andrei Amatuni, Marisa Casillas, Amanda Seidl, Melanie Soderstrom, Anne S. Warlaumont, Guillermo Hidalgo, Sebastian Schnieder, Clemens Heiser, Winfried Hohenhorst, Michael Herzog, **Maximilian Schmitt**, Kun Qian, Yue Zhang, George Trigeorgis, Panagiotis Tzirakis, & Stefanos Zafeiriou: *The INTERSPEECH 2017 Computational Paralinguistics Challenge: Addressee, Cold & Snoring*, Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH), Stockholm, Sweden, pp. 3442–3446, ISCA, 2017.

- Raymond Brückner, **Maximilian Schmitt**, Maja Pantic, & Björn Schuller: *Spotting Social Signals in Conversational Speech over IP: A Deep Learning Perspective*, Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH), Stockholm, Sweden, pp. 2371–2375, ISCA, 2017.

- **Maximilian Schmitt** & Björn Schuller: *Recognising Guitar Effects – Which Acoustic Features Really Matter?*, Proc. INFORMATIK 2017, Chemnitz, Germany, pp. 177–190, GI, 2017.

- Jing Han, Zixing Zhang, **Maximilian Schmitt**, Maja Pantic, & Björn Schuller: *From Hard to Soft: Towards more Human-like Emotion Recognition by Modelling the Perception Uncertainty*, Proc. ACM Multimedia, Mountain View, CA, USA, pp. 890–897, ACM, 2017.
- Yue Zhang, Felix Weninger, Boqing Liu, **Maximilian Schmitt**, Florian Eyben, & Björn Schuller: *A Paralinguistic Approach To Holistic Speaker Diarisation – Using Age, Gender, Voice Likability and Personality Traits*, Proc. ACM Multimedia, Mountain View, CA, USA, pp. 387–392, ACM, 2017.
- Hesam Sagha, **Maximilian Schmitt**, Filip Povolny, Andreas Giefer, & Björn Schuller: *Predicting the Popularity of a Talk-show based on the Emotionality of its Speech Content*, Proc. International Workshop on Affective Social Multimedia Computing (ASMMC), Stockholm, Sweden, 5 pages, 2017.
- Fabien Ringeval, Björn Schuller, Michel Valstar, Jonathan Gratch, Roddy Cowie, Stefan Scherer, Sharon Mozgai, Nicholas Cummins, **Maximilian Schmitt**, & Maja Pantic: *AVEC 2017 – Real-life Depression, and Affect Recognition Workshop and Challenge*, Proc. Audio-Visual Emotion Challenge and Workshop (AVEC), Mountain View, CA, USA, pp. 3–9, ACM, 2017.
- Christian Kohlschein, **Maximilian Schmitt**, Björn Schuller, Sabina Jeschke, & Cornelius J. Werner: *A Machine Learning Based System for the Automatic Evaluation of Aphasia Speech*, Proc. IEEE Healthcom, Dalian, China, 6 pages, IEEE, 2017.
- **Maximilian Schmitt** & Björn Schuller: *Deep Recurrent Neural Networks for Emotion Recognition in Speech*, Proc. DAGA, München, Germany, pp. 1537–1540, DEGA, 2018.
- Nicholas Cummins, Shahin Amiriparian, Sandra Ottl, Maurice Gerczuk, **Maximilian Schmitt**, & Björn Schuller: *Multimodal Bag-of-Words for Cross Domains Sentiment Analysis*, Proc. International Conference on Acoustics, Speech, & Signal Processing (ICASSP), Calgary, Canada, pp. 4954–4958, IEEE, 2018.
- Simone Hantke, Christian Cohrs, **Maximilian Schmitt**, Benjamin Tannert, Florian Lütkebohmert, Mathias Detmers, Heidi Schelhowe, & Björn Schuller: *Introducing an Emotion-Driven Assistance System for Cognitively Impaired Individuals*, Proc. International Conference on Computers Helping People with Special Needs (ICCHP), Linz, Austria, pp. 486–494, Springer, 2018.
- Vedhas Pandit, Nicholas Cummins, **Maximilian Schmitt**, Simone Hantke, Franz Graf, Lucas Paletta, & Björn Schuller: *Tracking Authentic and In-the-wild Emotions using Speech*, Proc. Asian Conference on Affective Computing and Intelligent Interaction (ACII Asia), Beijing, China, 6 pages, AAAC, IEEE, 2018.
- Shahin Amiriparian, **Maximilian Schmitt**, Nicholas Cummins, Kun Qian, Fengquan Dong, & Björn Schuller: *Deep Unsupervised Representation Learning for*

Abnormal Heart Sound Classification, Proc. Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Hawaii, HI, USA, pp. 4776–4779, EMBS, IEEE, 2018.

- Simone Hantke, **Maximilian Schmitt**, Panagiotis Tzirakis, & Björn Schuller: *EAT – The ICMI 2018 Eating Analysis and Tracking Challenge*, Proc. International Conference on Multimodal Interaction (ICMI), Boulder, CO, USA, 5 pages, ACM, 2018.

- Emilia Parada-Cabaleiro, **Maximilian Schmitt**, Anton Batliner, Simone Hantke, Giovanni Costantini, Klaus Scherer, & Björn Schuller: *Identifying Emotions in Opera Singing: Implications of Adverse Acoustic Conditions*, Proc. International Society for Music Information Retrieval Conference (ISMIR), Paris, France, pp. 376–382, ISMIR, 2018.

- Emilia Parada-Cabaleiro, **Maximilian Schmitt**, Anton Batliner, & Björn Schuller: *Musical-Linguistic Annotations of Il Lauro Secco*, Proc. International Society for Music Information Retrieval Conference (ISMIR), Paris, France, pp. 461–467, ISMIR, 2018.

- Jing Han, Zixing Zhang, **Maximilian Schmitt**, Zhao Ren, Fabien Ringeval, & Björn Schuller: *Bags in Bag: Generating Context-Aware Bags for Tracking Emotions from Speech*, Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH), Hyderabad, India, pp. 3082–3086, ISCA, 2018.

- Björn W. Schuller, Stefan Steidl, Anton Batliner, Peter B. Marschik, Harald Baumeister, Fengquan Dong, Simone Hantke, Florian B. Pokorny, Eva-Maria Rathner, Katrin D. Bartl-Pokorny, Christa Einspieler, Dajie Zhang, Alice Baird, Shahin Amiriparian, Kun Qian, Zhao Ren, **Maximilian Schmitt**, & Panagiotis Tzirakis and Stefanos Zafeiriou: *The INTERSPEECH 2018 Computational Paralinguistics Challenge: Atypical & Self-Assessed Affect, Crying & Heart Beats*, Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH), Hyderabad, India, pp. 122–126, ISCA, 2018.

- Fabien Ringeval, Björn Schuller, Michel Valstar, Roddy Cowie, Heysem Kaya, **Maximilian Schmitt**, Shahin Amiriparian, Nicholas Cummins, Denis Lalanne, Adrien Michaud, Elvan Çiftçi, Hüseyin Güleç, Albert Ali Salah, & Maja Pantic: *AVEC 2018 Workshop and Challenge: Bipolar Disorder and Cross-Cultural Affect Recognition*, Proc. Audio-Visual Emotion Challenge and Workshop (AVEC), Seoul, Korea, pp. 3–13, ACM, 2018.

- Jing Han, **Maximilian Schmitt**, & Björn Schuller: *You Sound Like Your Counterpart: Interpersonal Speech Analysis*, Proc. International Conference on Speech and Computer (SPECOM), Leipzig, Germany, pp. 188–197, Springer, 2018.

- Vedhas Pandit, **Maximilian Schmitt**, Nicholas Cummins, Franz Graf, Lucas Paletta, & Björn Schuller: *How Good Is Your Model 'Really'? On 'Wildness' of the In-the-wild Speech-based Affect Recognisers*, Proc. International Conference on Speech and Computer (SPECOM), Leipzig, Germany, pp. 490–500, Springer, 2018.

- **Maximilian Schmitt** & Björn Schuller: *End-to-end Audio Classification with Small Datasets – Making It Work*, Proc. European Signal Processing Conference (EUSIPCO), A Coruña, Spain, 5 pages, EURASIP, IEEE, 2019.

- **Maximilian Schmitt**, Nicholas Cummins, & Björn Schuller: *Continuous Emotion Recognition in Speech – Do We Need Recurrence?*, Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH), Graz, Austria, pp. 2808–2812, ISCA, 2019.

- Adrià Mallol-Ragolta, **Maximilian Schmitt**, Alice Baird, Nicholas Cummins, & Björn Schuller: *Performance Analysis of Unimodal Models in Valence-Based Empathy Recognition*, Proc. International Conference on Face & Gesture Recognition (FG) / OMG Challenge, Lille, France, 5 pages, IEEE, 2019.

- Christoph Janott, Christian Rohrmeier, **Maximilian Schmitt**, Werner Hemmert, & Björn Schuller: *Snoring – An Acoustic Definition*, Proc. Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Berlin, Germany, pp. 3653–3657, EMBS, IEEE, 2019.

- Julian Schiele, Fabian Rabe, **Maximilian Schmitt**, Manuel Glaser, Franziska Häring, Jens O. Brunner, Bernhard Bauer, Björn Schuller, Claudia Traidl-Hoffmann, & Athanasios Damialis: *Automated Classification of Airborne Pollen using Neural Networks*, Proc. Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Berlin, Germany, pp. 4474–4478, EMBS, IEEE, 2019.

- Björn W. Schuller, Anton Batliner, Christian Bergler, Florian B. Pokorny, Jarek Krajewski, Margaret Cychosz, Ralf Vollmann, Sonja-Dana Roelen, Sebastian Schnieder, Elika Bergelson, Alejandrina Cristia, Amanda Seidl, Anne S. Warlaumont, Lisa Yankowitz, Elmar Nöth, Shahin Amiriparian, Simone Hantke, & **Maximilian Schmitt**: *The INTERSPEECH 2019 Computational Paralinguistics Challenge: Styrian Dialects, Continuous Sleepiness, Baby Sounds & Orca Activity*, Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH), Graz, Austria, pp. 2378–2382, ISCA, 2019.

- Vedhas Pandit, **Maximilian Schmitt**, Nicholas Cummins, & Björn Schuller: *I Know How you Feel Now, and Here's why!: Demystifying Time-Continuous High Resolution Text-Based Affect Predictions in the Wild*, Proc. International Symposium on Computer-Based Medical Systems, Córdoba, Spain, pp. 465–470, IEEE, 2019.

- Kun Qian, Hiroyuki Kuromiya, Zhao Ren, **Maximilian Schmitt**, Zixing Zhang, Toru Nakamura, Kazuhiro Yoshiuchi, Björn W. Schuller, & Yoshiharu Yamamoto: *Automatic Detection of Major Depressive Disorder via a Bag-of-Behaviour-Words Approach*, Proc. International Symposium on Image Computing and Digital Medicine (ISICDM), Xi'an, China, pp. 71–75, ACM, 2019.

- Alice Baird, Shahin Amiriparian, Miriam Berschneider, **Maximilian Schmitt**, & Björn Schuller: *Predicting Biological Signals from Speech: Introducing a Novel Multimodal Dataset and Results*, Proc. International Workshop on Multimedia Signal Processing (MMSP), Kuala Lumpur, Malaysia, 5 pages, IEEE, 2019.
- Fabien Ringeval, Björn Schuller, Michel Valstar, Nicholas Cummins, Roddy Cowie, Leili Tavabi, **Maximilian Schmitt**, Sina Alisamir, Shahin Amiriparian, Eva-Maria Messner, Siyang Song, Shuo Liu, Ziping Zhao, Adrià Mallol-Ragolta, Zhao Ren, Mohammad Soleymani, & Maja Pantic: *AVEC 2019 Workshop and Challenge: State-of-Mind, Detecting Depression with AI, and Cross-Cultural Affect Recognition*, Proc. Audio-Visual Emotion Challenge and Workshop (AVEC), Nice, France, pp. 3–12, ACM, 2019.
- Björn Schuller, Shahin Amiriparian, Gil Keren, Alice Baird, **Maximilian Schmitt**, & Nicholas Cummins: *The Next Generation of Audio Intelligence: A Survey-based Perspective on Improving Audio Analysis*, Proc. International Symposium on Auditory and Audiological Research (ISAAR), Nyborg, Denmark, pp. 101–112, The Danavox Jubilee Foundation, 2019.
- Björn W. Schuller, Anton Batliner, Christian Bergler, Eva-Maria Messner, Antonia Hamilton, Shahin Amiriparian, Alice Baird, Georgios Rizos, **Maximilian Schmitt**, Lukas Stappen, Harald Baumeister, Alexis Deighton MacIntyre, & Simone Hantke: *The INTERSPEECH 2020 Computational Paralinguistics Challenge: Elderly Emotion, Breathing & Masks*, Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH), Shanghai, China, pp. 2042–2046, ISCA, 2020.

Book

- Shahin Amiriparian, Andreas Bühlmeier, Christoph Henkelmann, **Maximilian Schmitt**, Björn Schuller, & Oliver Zeigermann: *Einstieg ins Machine Learning – Grundlagen, Prinzipien, erste Schritte*, entwickler.press, 2019.

Bookchapters

- **Maximilian Schmitt** & Björn Schuller: *Machine-based Decoding of Paralinguistic Vocal Features*, The Oxford Handbook of Voice Perception, Oxford University Press, 2018.
- Vedhas Pandit, Shahin Amiriparian, **Maximilian Schmitt**, Amr Mousa, & Björn Schuller: *Big Data Multimedia Mining: Feature Extraction Facing Volume, Velocity, and Variety*, Big Data Analytics for Large-Scale Multimedia Search, John Wiley and Sons Ltd., 2019.
- Shahin Amiriparian, **Maximilian Schmitt**, Simone Hantke, Vedhas Pandit, & Björn Schuller: *Humans Inside: Cooperative Big Multimedia Data Mining*, Innovations in Big Data Mining and Embedded Knowledge, Springer, 2019.

- Shahin Amiriparian, **Maximilian Schmitt**, Sandra Ottl, Maurice Gerczuk, & Björn Schuller: *Deep Unsupervised Representation Learning for Audio-based Medical Applications*, Intelligent Systems Reference Library (ISRL), Springer, 2020.

Popular Press

- Shahin Amiriparian, **Maximilian Schmitt**, & Björn Schuller: *Exploiting Deep Learning: die wichtigsten Bits und Pieces – Sieh zu und lerne*, Java Magazin, pp. 46–53, JAXenter, 2018.
- **Maximilian Schmitt**, Shahin Amiriparian, & Björn Schuller: *Maschinelle Sprachverarbeitung: Wie lernen Computer unsere Sprache zu verstehen?*, Entwickler Magazin, pp. 30–38, Software & Support Media GmbH, 2018.

Technical Report

- Gil Keren, **Maximilian Schmitt**, Thomas Kerenberg, & Björn Schuller: *Weakly Supervised One-Shot Detection with Attention Siamese Networks*, arXiv: 1801.03329, pp. 1–15, arXiv.org, 2018.

Awards

- **Maximilian Schmitt**: Best Reviewer Award, 21^{st} International Society for Music Information Retrieval (ISMIR) Conference, virtual, 11–16 October 2020.
- **Maximilian Schmitt**: Outstanding Reviewer Award, Physiological Measurement, IOP Publishing, 2020.

Contents

Part I

INTRODUCTION

1

Introduction

The field of *computer audition* (CA) or *machine listening* has received a lot of attention during the last two decades, resulting in a huge impact on both research and user communities. Automatic speech recognition (ASR), i. e., speech-to-text transcription, is nowadays a meaningful feature of every computer system or smartphone, while more sophisticated applications such as emotion recognition in speech are paving the way for a fully-naturalistic human-machine interaction [1, 2]. As another example, modules for acoustic scene classification (ASC) [3] can control the sound profile of a smartphone, muting the ringtone when recognising that the current location is a library or a church and increasing the sound level when walking next to a street. One major scope of CA is also the field of health care and medical diagnosis, where research has been conducted on tasks such as depression recognition [4], detection of respiratory tract infection [5], and classification of snoring types [6]. Finally, music information retrieval (MIR) is a further domain that is, to a large extent, based on the analysis of audio signals with applications such as audio fingerprinting [7], genre classification [8], and automatic music transcription [9].

Before the motivation and contributions of this work are defined, a general overview of CA and the corresponding processing chain are presented. The introduction closes with the outline of this thesis.

1.1 Computer Audition

From a technical point of view, most approaches in CA rely on a combination of *signal processing* and *machine learning* (ML) techniques [1]. Recent advancements in hard- and software for both domains and a multitude of novel methods in especially the ML field have boosted the performance of many applications and made them trainable and deployable on any kind of computer or smart device.

The two basic categories in ML are *unsupervised* and *supervised* learning. While in unsupervised learning, latent structures are discovered, e. g., using a clustering

method, in supervised ML, a model is trained on *paired* inputs and outputs, in order to predict an output from only the input. In the case of CA, the input is usually the audio signal and the output is an attribute corresponding to the given task.

In order to employ a method of **supervised** ML, a meaningful number of audio examples are required to *train* the model. The corresponding audio dataset, also referred to as **corpus**, is required to have a **label (target)** for each audio clip, according to the given audio recognition task. The label can be either a discrete *category* or *class*, or a *continuous number* (numeric label), describing a certain quality of the audio, such as the emotional state or the age of a subject speaking, the acoustic scene, or the genre of a given piece of music. In the case of categorical labels, a *classifier* is trained performing **classification**; in the case of numeric labels, a *regressor* is trained performing **regression**. As shown later, ML approaches for classification and regression are usually very similar. Given that the knowledge which is exploited to create a model originates from the given data samples and not from theoretical considerations, methods of ML are considered as *data-driven* methods.

Each audio clip to be classified is referred to as **data sample** or **instance**. The entirety of the provided labels is called **ground truth**, however, if the given task at hand relates to a *subjective quality*, such as, e. g., emotion, the term **gold standard** is preferred [1]. In this case, labels are normally obtained from a multitude of raters and then fused with an appropriate method taking into account the certainty of each rater, such as the *evaluator weighted estimator* (EWE) [1]. For some tasks, a label needs to be predicted not only for each audio clip, but at *discrete* positions in time throughout the audio. In this case, a given task is referred to as a **sequence-labelling** task, as opposed to the classification or regression on **chunk-level**.

1.2 Audio Processing Chain

A processing chain typically used in CA, based on a dedicated *audio corpus* consisting of a collection of audio signals and corresponding labels, is shown in Figure 1.1. The audio is generally represented by digital signals, as a sampled and quantised version of the sound pressure fluctuations picked up by a microphone [10]. The *sample rate* must be at least twice the maximum frequency present in the signal *(Nyquist rate)* [10], otherwise *aliasing* effects would occur in the sampled signal. A typical sample rate for speech signals is 16 kHz [11, 12, 13], while at least 44.1 kHz are used for music signals [10].

The signals are usually **pre-processed** as a first step (see Chapter 2). Next, **acoustic features** are **extracted** from the pre-processed signals. These are numeric descriptors capturing relevant properties of the audio. The feature extraction step is usually executed in two stages: firstly, **frame-level** features are computed for each time frame of an audio signal, i. e., on short intervals wherein its statistics do

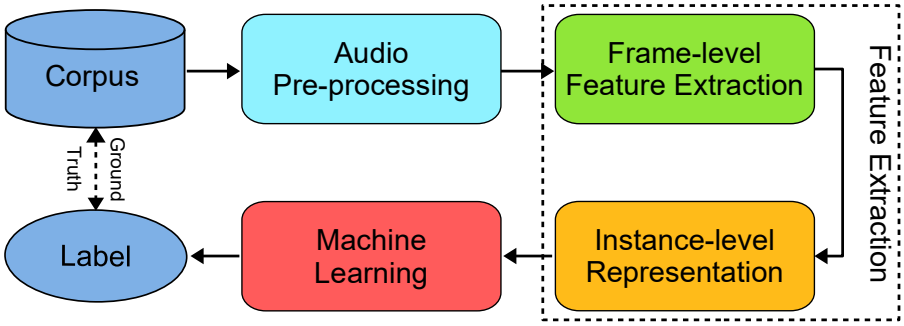

Figure 1.1: A typical processing chain in computer audition.

not change significantly. Secondly, the frame-level features are summarised for each **instance**, forming a single vector of fixed length, independent from the duration of the audio instances. The final instance-level features and the ground truth labels from the corpus are then propagated to a **supervised ML** algorithm, which trains a **model** analysing the relationships between the extracted features and the labels, generalising from the given data. The field of ML has attracted a huge and recently increasing research interest for the last 60 years, resulting in a myriad of approaches, algorithms, and implementations. Finally, given a novel audio sample where the label is unknown, it is possible to **predict** a label, applying the feature extraction steps and the ML model. As the predictions always include a certain error, a proper **evaluation** of the trained model is essential.

During the last few years, a number of benchmark corpora have been made public in order to make research in CA comparable. Very often, they have been published for the first time in the context of challenges (scientific competitions). Examples are the *data science* platform KAGGLE [14], the series of *Computational Paralinguistics ChallengEs* (ComParE), organised annually as special sessions at the INTERSPEECH conference since 2009 [13, 15], and the series of *Audio-Visual Emotion Challenges* (AVEC), organised annually since 2011 [16, 17]. While the different editions of AVEC featured multimodal tasks related to the recognition of *affect* and *emotion* or mental health conditions related to these such as *depression* [18] and *bipolar disorder* [19], ComParE featured a large variety of tasks, such as, e. g., *social signals* [20], *nativeness* [21], *eating condition* [21], *deception* [22], *sincerity* [22], *addressee* [11], *crying* [12], *dialect* [13], *sleepiness* [13], *baby sounds* [13], or *orca activity* [13], to mention just a few.

1.3 Motivation and Research Questions

Despite of all the mentioned and adumbrated achievements, some problems in CA remain challenging. As shown in Figure 1.1, acoustic frame-level features, also called

low-level descriptors (LLDs), need to be summarised over each instance resulting in a feature vector of fixed size. Typically, *statistical functionals* [23] are applied to the LLDs, i. e., certain measures from descriptive statistics, such as *means*, *higher-order moments*, *percentiles*, or *regression lines* are computed. However, this approach implies a 'global pooling' operation, smoothing out local temporal information [24]. In contrast, in this thesis, the **bag-of-audio-words** (BoAW) approach is investigated as an alternative to statistical functionals, representing the LLDs of an audio instance as a **histogram** [25, 26]. For this, a prior **vector quantisation** operation is required, assigning each low-level feature vector to an 'audio word' from a predefined **codebook**. BoAW is originally inspired from the standard 'bag-of-words' vector space model used in the text processing domain [27].

It is generally known that, even for the same type of underlying descriptors, the **feature representation**, i. e., how the information is passed to the ML algorithm, matters [28]. The main goal of this thesis is to advance the state-of-the-art in **unsupervised representation learning**, by promoting BoAW as an alternative and additional component of CA systems. The approach is evaluated on a large variety of audio classification tasks and its pros and cons are carved out.

As pointed out in Chapter 3, many aspects in the BoAW processing chain have not yet been conclusively clarified. Thus, this thesis is developed along the following three research questions:

- **Research question 1 (RQ 1)**:
  Typically, the *codebooks* are generated by clustering the LLDs. Nevertheless, even a simpler sampling has proven to be effective as well. Moreover, data-independent codebooks would be beneficial as they enable the BoAW features to be extracted *on-the-fly*, i. e., without a training or domain-adaptation phase. Given this, one important goal of this thesis is to clarify: **Which is the best codebook generation technique and are data-independent codebooks possible or not?**
- **Research question 2 (RQ 2)**:
  Although BoAW features have already been employed for various audio recognition tasks, their configuration has usually been optimised for a specific task. Thus, the following is investigated: **Is it possible to use the same BoAW-configuration (in terms of selection of LLDs, codebook size, etc.) across various tasks in CA?**
- **Research question 3 (RQ 3)**:
  So far, there has been no systematic comparison or evidence if BoAW features outperform 'classical' statistical functionals and which are the potential benefits of each representation. Especially taking into consideration a large-scale functionals-based acoustic feature set, the following is tried to be answered: **Which representation (*functionals* or *BoAW*) is superior and are they complementary or redundant?**

These research questions are revisited in the conclusion in Chapter 9.

1.4 Contributions

The main contributions of the research compiled in this thesis are listed in the following:

- The open-source *crossmodal bag-of-words* toolkit openXBOW [29] is introduced, providing a systematic generation of bag-of-words representations for both *numeric* (e. g., acoustic or visual) and *symbolic* (e. g., text) low-level features. This pioneering work has been cited more than 100 times so far.
- A large number of experiments and case studies is presented and discussed, focussing on different aspects and variations of the BoAW processing chain, all of them integrated in openXBOW.
- Based on these experiments, it is proven that BoAW is effective for a wide range of applications in the CA field, from *emotion recognition* to *health care*.
- It is shown that ML models trained on BoAW features outperform models trained on classical statistical functionals, without tuning the configuration of openXBOW on a particular dataset or task. In addition to that, some evidence is found that *domain adaptation* works better with BoAW than with functionals.
- It is further shown that BoAW and functionals are *complementary* and a fusion usually leads to better results compared to using only a single representation.
- It is shown that BoAW representations can be used as an input for both *shallow* and *deep* ML approaches.
- Depending on the task and data, evidence is found that recent *deep end-to-end learning* methods are outperformed by BoAW in combination with a *support vector machine* classifier, making the methodology relevant especially for domains of scarce data.
- The BoAW method and the toolkit openXBOW have been established as a baseline method for scientific challenges in the field of CA and *affective computing*, deployed at eight different events so far (4 × ComParE, 3 × AVEC, & the ICMI EAT challenge). In most of these events, the model based on BoAW features gave the best performance across all feature representations and methods. For two sub-challenges of ComParE, the BoAW approach was involved in a fusion model that has not been beaten by any of the participants.
- Several codebook generation techniques are evaluated and compared to each other and it is shown that a pure random sampling of LLDs from the training

set provides optimal results, while also a data-independent codebook generation is feasible.

- The relevance of various types of acoustic LLDs is studied. Moreover, experiments show that splitting LLD vectors into subsets prior to vector quantisation is beneficial for the performance.
- It is proven that *multi-assignment*, i. e., assigning an LLD vector to more than one audio word, improves the results and is required when dealing with large-dimensional codebooks. Furthermore, it is found that a hard quantisation is superior to a soft one in many scenarios, challenging prior works.

1.5 Outline

This thesis is structured in four parts. After the first part, i. e., the Introduction, the Theoretical Background is described, divided into three chapters. In Chapter 2, acoustic frame-level and instance-level features, which are relevant for CA tasks, are introduced, accompanied by some fundamentals of signal processing, such as the *short-time Fourier transform*. At the end of this chapter, two standardised acoustic feature sets are introduced, which are used in several experiments throughout the thesis. Then, the history and background of the BoAW approach is introduced in Chapter 3, including also an overview of related methods. Chapter 4 is dedicated to the fundamentals of ML, with a focus on *support vector machine* and *neural network*-based models, as these are the ones most relevant in the remainder of this work. This chapter also involves a section on the evaluation of supervised ML models.

The third part on the Experiments is subdivided into three chapters as well and starts with a description of the BoAW-toolkit openXBOW in Chapter 5, which has been implemented in the context of this thesis and on which all presented experiments related to BoAW are based. This chapter also includes an overview of all modifications and advancements of the approach that are proposed by the author. Then, in Chapter 6, all experiments on BoAW that have been pre-published in *case studies* are presented, highlighting certain tasks and aspects of the methodology. In Chapter 7, the experiments are complemented by some *systematic evaluations*, in order to find a response to the research questions that have not been solved in the case studies or further literature.

Finally, in the fourth part, which is subdivided into two chapters, a Discussion of the results takes place. Both the *achievements* and the *limitations* of the proposed approaches are demonstrated in Chapter 8, completed by an outline of *ethical considerations*. The last Chapter 9 remains for a *summary* of the thesis and an outlook on possible *future works*, pointing out open research questions.

Part II

THEORETICAL BACKGROUND

2

Acoustic Features

Acoustic features are compact descriptors of the content of an audio signal. They need to capture all relevant information and must be discriminative between target classes. Which information in the audio is relevant highly depends on the given recognition task. Nevertheless, certain *feature types* and *feature sets*, introduced below, have proven to be suitable for a wide range of tasks [20, 30, 31].

As defined in the introduction, it is assumed that the audio is provided as a digital signal of an adequate sampling rate (at least 16 kHz) to capture the relevant frequency range of the signal and a sufficient bit depth (usually 16 bit) to obtain a sufficient *signal-to-noise ratio* (SNR) [32, Chapter 8]. The real-valued[1] discrete signal is then defined as the time series

$$s(n), \quad s \in [-1.0, 1.0], \quad n = 0, \ldots, N-1, \tag{2.1}$$

with the discrete time index n and the length of the signal (duration) N, as the number of samples. The time index corresponds to the time $t = n \cdot T_s$ in the continuous signal, given that it starts at $t = 0$, where T_s is the *sampling period*, i. e., the temporal distance between adjacent samples or the reciprocal of the *sample rate* (sampling frequency) F_s.

For *pre-processing*, in the simplest case, only a waveform normalisation is performed, fixing the maximum amplitude of each signal to the same value [13]. As a usual convention, the largest possible amplitudes w. r. t the data format correspond to -1.0 and $+1.0$, the mean value of the signal is assumed to be approximately 0, as the direct component does not carry any acoustic information [30, 33]. This step ensures that the level of the audio recording, which is usually arbitrary as there is no general conversion factor between a sound pressure and its digital representation, does not vary across the instances of the corpus, which would have an impact on energy-related audio descriptors.

[1] In fact, a computer system always works with quantised data. However, as it is assumed that the bit depth is sufficient, this fact will be neglected in the following, corresponding to the general convention.

More sophisticated pre-processing techniques include a normalisation w. r. t. the *root mean square* (RMS) or the loudness of the signals, *pre-emphasis* of the upper frequency range [30, 2.1.4] (see also Subsection 2.1.5.1) or *speech enhancement* [34, 35]. Furthermore, *downmixing* to reduce the number of channels, e. g., a stereo-to-mono conversion, and *downsampling* are often useful steps to unify the data format and reduce redundant or irrelevant information in order to speed up the training process [36]. In the remainder of this chapter, as defined above (Definition 2.1), it is assumed that mono (single-channel) signals are provided. If more than one channel is given, the data can be downmixed to mono by averaging (stereo) or using only the centre channel (e. g., 5.1 surround sound).

Following the processing chain from Figure 1.1, this chapter specifies the feature extraction process, separated into frame-level features in Section 2.1 and instance-level features in Section 2.2. Two standardised feature sets exploited in this work, the ComParE feature set and eGeMAPS, are described in Section 2.3.

2.1 Frame-level Features

As mentioned, in the first step of the feature extraction, the audio is processed frame-wise, i. e., by subdividing them into small excerpts (*frames* or *blocks*), where the signal can be considered *quasi-stationary* within. This is based on the fact that, the statistics (or parameters) of the signal change considerably slower than the rate of the time-domain samples, given by the sampling frequency [37]. These frame-level features are typically referred to as **low-level descriptors (LLDs)**.

Frames are defined by their length (*frame length*) and the distance between the onsets of adjacent frames, the *hop size* (or *frame step*) [30, 38]. The framed signal is defined as

$$s_f(h,m) = s\left(h \cdot H_s + m - \frac{M}{2}\right), \quad h = 0, \ldots, \left\lceil \frac{N-1}{H_s} \right\rceil, \quad m = 0, \ldots, M-1, \tag{2.2}$$

with the hop size H_s and the frame length M. As can be seen from the boundaries in equation 2.2, the frames capture information from outside the domain of $s(n)$, which needs to be padded with (usually) zeros. Each frame h contains then an excerpt from the signal centred around the point $h \cdot H_s$ in the discrete time domain. Extracting frames with indexes until the $\left\lceil \frac{N-1}{H_s} \right\rceil$ ensures that the last sample of the signal with index $N-1$ does not come after the centre of the last frame. The overall number of frames in a signal with a length of N samples is then $N_h = \left\lceil \frac{N-1}{H_s} \right\rceil + 1$. Figure 2.1 visualises the process of generating frames from a short audio signal. The described padding strategy is appropriate at least for signal *analysis* while other procedures are required for *processing* tasks, where the time-domain signal is restored from overlapping frames [30].

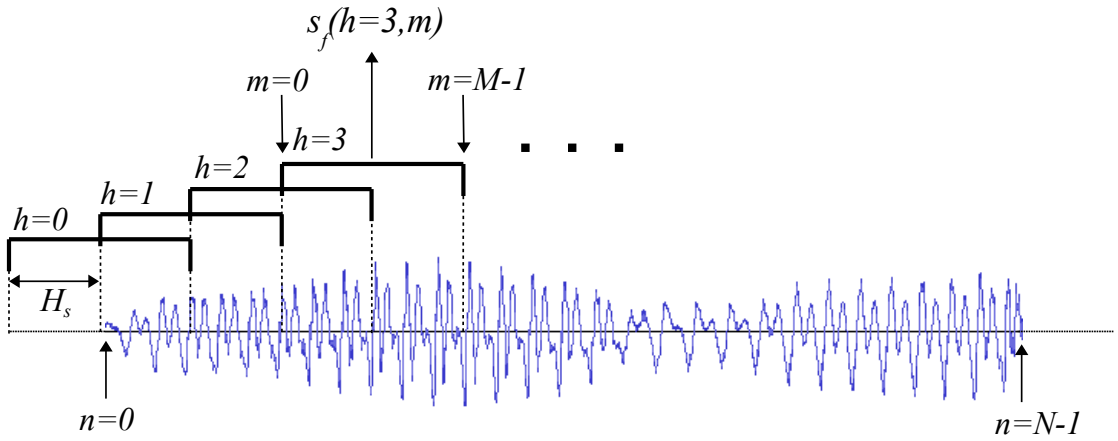

Figure 2.1: Illustration of short-time audio analysis for an audio signal of length N. The boundaries of the first four frames of length M are plotted. The hop size H_s is exactly half of the frame length in this example, resulting in an overlap of 50 %.

Before computing the *short-time* features, the time-domain samples within each frame are usually weighted with a **windowing function**, i. e.:

$$s_w(h,m) = s_f(h,m) \cdot w(m), \quad m = 0,\ldots,M-1, \tag{2.3}$$

with the windowing function $w(m)$ of length M, i. e., the frame length. This is done mainly for two reasons: firstly, as the windows have their largest amplitude at their centre, signal samples with larger distance to the centre of the frame are attenuated, which results in a larger consideration of the samples close to the actual *frame time*

$$t_f(h) = h \cdot H_s \cdot T_s. \tag{2.4}$$

Secondly, when it comes to time-frequency analysis (see Subsection 2.1.2.3), *smearing* and *leakage* effects [39], which deteriorate the precision of the analysis, can be reduced by the choice of an appropriate windowing function.

While for the extraction of LLDs in the time-domain (see Subsection 2.1.1), a *rectangular window*, which—following the notation used here—implies $s_w(h,m) = s_f(h,m)$, is sufficient, this most basic type of windowing function is not suitable for spectral descriptors (see Subsection 2.1.2), due to the mentioned effects [1, 30]. The most common non-rectangular window types [1] are the *Hamming window* $w_{\text{hamming}}(m)$, the *Hann window* $w_{\text{hann}}(m)$, and the *Gaussian window* $w_{\text{Gaussian}}(m)$, which are, based on the given definition that each window goes from $m = 0$ to

$m = M - 1$, defined as follows [30]:

$$w_{\text{hamming}}(m) = 0.54 - 0.46 \cos\left(\frac{2\pi m}{M-1}\right), \tag{2.5}$$

$$w_{\text{hann}}(m) = 0.5\left(1 - \cos\left(\frac{2\pi m}{M-1}\right)\right), \tag{2.6}$$

$$w_{\text{Gaussian}}(m) = \mathrm{e}^{-0.5\left(\frac{m-0.5(M-1)}{0.5\sigma(M-1)}\right)^2}, \tag{2.7}$$

with the *standard deviation* σ for w_{Gaussian}. Their properties depend—without going into details—mainly on their shape in both time and frequency domain (see Subsection 2.1.2). The *Hamming window* is probably the most common one for spectral short-time analysis. The *Hann window* is rather preferred for filtering or processing in the spectral domain with a subsequent back-transform. The *Gaussian window* in spectral analysis has the property that it has the same smooth shape in both domains. More details are found in the corresponding literature [30, 39].

The frame/window lengths must be long enough to compute meaningful features (see Subsection 2.1.2.3) and at the same time short enough so that the quasi-stationarity assumption is not violated [32, 8.2]. Typical frame lengths range from 20 ms to 60 ms, depending on the feature type [30]. The hop size is usually shorter than the frame length, to ensure that the whole signal is represented within the series of LLDs. Therefore, the *overlap* $(1 - \frac{H_s}{M})$ is often given as an alternative to the hop size. Typically, the overlap is chosen between 50 % and 75 % [30, 32]. The resulting frequency of the frames $(\frac{1}{H_s})$ can then be considered as the *parameter sampling frequency* [32, 8.2] or *LLD sampling frequency*. Given a typical hop size of 10 ms, a common LLD frequency in speech or audio processing is 100 Hz or 100 *frames per second*.

In the following, a selection of frame-level feature types, i. e., LLDs, is introduced. The choice depends mainly on their general relevance for audio recognition tasks, with a special focus on speech, and on the relevance for the approaches proposed later in this thesis.

2.1.1 Time-domain descriptors

At first, three simple and conventional LLDs which are extracted from the time-domain of the signal are introduced.

2.1.1.1 Energy, intensity

A very basic, but important descriptor is the **short-time energy** $E(h)$ for each frame h, which is computed as

$$E(h) = \sum_{m=0}^{M-1} s_w^2(h, m), \tag{2.8}$$

i. e., it is the sum of the squared samples (after windowing). A common window type for the measurement of energy is the *Hamming window* [1, 6.2.1.1]. Though the energy could also be extracted from a Fourier representation of the signal (see Subsection 2.1.2.3, it is usually extracted from the time-domain waveform. Optionally, the *normalised energy* can be used instead in order to soften the influence of the frame length [30, 2.2.2]:

$$E_{\text{norm}}(h) = \frac{1}{M} \cdot E(h). \tag{2.9}$$

Very often, the square root is taken from $E_{\text{norm}}(h)$ to obtain the RMS of the frame, a measure that is proportional to the signal amplitudes, and not to the energy:

$$E_{\text{rms}}(h) = \sqrt{E_{\text{norm}}(h)}. \tag{2.10}$$

Another common step is taking the logarithm of the energy, in order to compress the range of values [30, 2.2.2]. As energy and intensity are proportional and the signal does not have a fixed physical reference, the energy is usually also referred to as the **intensity** [30].

The short-time energy or intensity can already be used as a simplistic *voice activity detection* (VAD), in order to chunk speech or audio files at points of silence. Furthermore, it is a fundamental LLD in phonetics and also linguistics, when it comes to analysing speech *prosody* (see Subsection 2.1.4).

2.1.1.2 Zero-crossing rate

Another common time-domain LLD is the *zero-crossing rate* (ZCR) [40]. The ZCR is defined as the signal's number of changes of sign in a certain time interval, e. g., per second [1, 30]:

$$\begin{aligned} \text{ZCR}(h) &= \frac{F_S}{M-1} \cdot \sum_{m=1}^{M-1} s_0(h,m), \\ \text{with} \quad s_0(h,m) &= \begin{cases} 0 & \text{if} \quad \operatorname{sgn}\left(s_f(h,m)\right) = \operatorname{sgn}\left(s_f(h,m-1)\right), \\ 1 & \text{if} \quad \operatorname{sgn}\left(s_f(h,m)\right) \neq \operatorname{sgn}\left(s_f(h,m-1)\right), \end{cases} \end{aligned} \tag{2.11}$$

with the sign function sgn(x). As the window function does not have any effect on the result here, the ZCR can be computed from either $s_f(h,m)$ or $s_w(h,m)$.

The ZCR is a scalar indicator for high-frequency content of a signal. Noise, especially *white noise*, and other 'sharp' sounds typically have much energy in the upper frequency range, compared to harmonic signals, i. e., signals with a prominent *fundamental frequency* [30, 2.2.1]. This is quite intuitive as the number of zero-crossings per second for a pure sine wave is twice its frequency. Thus, the ZCR is suited for the detection of voiced (vowels and voiced consonants, such as 'b')

and unvoiced (unvoiced consonants, such as 'p') parts of speech [41], as well as the distinction between harmonic and percussive musical instruments [42]. This comes with the downside that this LLD is quite prone to noise [30, 2.2.1], which leads to the conclusion that it can just be considered a 'baseline' approach for the mentioned tasks.

2.1.1.3 Autocorrelation function

For the sake of completeness, and, as it is used as a processing step in some of the more complex descriptors, the *autocorrelation function* (ACF) is introduced here as another descriptor computed from a time-series of data. It can be employed as an LLD itself, but in the context of this thesis, it is only used as an intermediate step. The ACF generally describes the self-similarity of signals as a function of the delay [30]. The short-time ACF can be defined as (in modification of [32], *stationary way*)

$$\mathrm{ACF}(h,d) = \sum_{i=0}^{M-1-d} s_w(h,m) \cdot s_w(h,m+d), \quad 0 \leq d \leq M-1, \tag{2.12}$$

with the discrete delay d. In theory, the ACF is a symmetric function defined also for $d < 0$. However, as this information would be redundant, it can be omitted as in Equation 2.12. While the coefficient for a delay of 0, $\mathrm{ACF}(h, d = 0)$, is identical with the short-time energy from Equation 2.8 for each frame step h, the ACF for larger delays provides insights into the *harmonicity* of signals. The largest peak is always the one at $d = 0$ and all other peaks point out periodicities with a frequency close to $\frac{F_s}{d}$. Also a version of the ACF, normalised w. r. t. the frame length, i. e., $\frac{1}{M} \cdot \mathrm{ACF}(h,d)$, or w. r. t. the energy, i. e., $\frac{1}{\mathrm{ACF}(h,0)} \cdot \mathrm{ACF}(h,d)$, are common [1].

2.1.2 Spectral descriptors

Many acoustic features rely on the *short-time Fourier transform* (STFT). The **Fourier transform** (FT) in general is a mathematical tool to convert a function into a representation composed of *sinusoidal* functions [38]. For an audio signal, this means that it is transformed from a function of time into a function of frequency (f), the *spectrum*.

2.1.2.1 Fourier transform

The transform of a time-domain function (or signal) $s_t(t), t \in \mathbb{R}$, i. e., the result of the FT, is generally a complex-valued function, $S_{\mathrm{FT}}(f) \in \mathbb{C}$, and defined as [43]

$$S_{\mathrm{FT}}(f) = \int_{-\infty}^{\infty} s_t(t) \cdot \mathrm{e}^{-j2\pi ft} \mathrm{d}t, \quad f \in \mathbb{R}. \tag{2.13}$$

The complex-valued amplitudes can be represented by a magnitude $M_{\mathrm{FT}}(f) = |S_{\mathrm{FT}}(f)|$ and a phase $\varphi_{\mathrm{FT}}(f) = \arg(S_{\mathrm{FT}}(f))$, with the *argument* function $\arg(x) \in (-\pi, \pi]$, solving the equation $S_{\mathrm{FT}}(f) = M_{\mathrm{FT}}(f) \cdot \mathrm{e}^{j\varphi_{\mathrm{FT}}(f)}$, with the *imaginary unit* j, where $j^2 = -1$. The amplitudes of $S_{\mathrm{FT}}(f)$ can be interpreted as sine waves with amplitude $M_{\mathrm{FT}}(f)$ and a phase (shift) of $\varphi_{\mathrm{FT}}(f)$. Thus, the FT represents the function $s_t(t)$ by a superposition of scaled and shifted sine waves. The *inverse* FT converts $S_{\mathrm{FT}}(f)$ back to the original function, i. e., no information is lost during the (back-)transformation. The domain of the transform is generally unrestricted, however, if the function $s_t(t)$ is real-valued—and this is true for audio signals—the transform on the negative frequency axis ($f < 0$) is the complex-conjugate of the positive axis, i. e., $S_{\mathrm{FT}}(-f) = S^*_{\mathrm{FT}}(f)$ [43].

2.1.2.2 Discrete Fourier transform

The FT can also be computed on time-discrete functions or signals. A signal $s(n)$ that is defined on a **discrete** domain is **periodic** in the frequency domain, with a period of F_s, the sampling frequency. However, in contrast to the signal, the transform is still on a continuous domain [43], which is not suitable when dealing with non-deterministic signals, such as audio signals. In order to achieve a *discrete* frequency domain, the signal must be **periodic**. This requirement is similar to the concept of the *Fourier series* [43], but with the properties of discrete *and* periodic in both domains.

The periodicity in the time domain can easily be achieved: as each recorded signal is finite, the continuation outside of its boundaries can be considered as periodic. This transform is called **discrete Fourier transform** (DFT) and defined as

$$S_{\mathrm{DFT}}(k) = \sum_{n=0}^{N-1} s(n) \cdot \mathrm{e}^{-j\frac{2\pi kn}{N}}, \quad k = 0, \ldots, \left\lfloor \frac{N}{2} \right\rfloor, \tag{2.14}$$

with the discrete frequency index k. In theory, the upper limit of the frequency index is equal to $N - 1$, but as mentioned before, one half of the spectrum is redundant in case of real-valued signals [38]. Given this and the fact that $S_{\mathrm{DFT}}(k)$ is complex-valued, i. e., two numbers[2] need to be stored for each coefficient, the digital representation in the frequency domain will require the same amount of memory as the waveform representation in the time domain. Nevertheless, due to the limited resolution of digital numbers on computers, given by the bit depth, rounding errors will occur in both the forward and the inverse transform, so that a perfect recovery of the original signal is not possible after performing the operations.

[2] Real & imaginary part or magnitude & phase

The maximum frequency, represented by the coefficient with the highest index in equation 2.14, is the half of the sample rate[3]. The frequency corresponding to each k in general is $f_k = k \cdot \frac{F_s}{N}$ [38].

With the **fast Fourier transform** (FFT), a very efficient algorithm exists to compute the DFT in real-time [44]. It reduces the complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(N \log N)$, but requires that the number of samples is a power of two, which can be accomplished by padding. This makes it the probably most important tool in digital signal processing, including, i. a., analysis and filtering operations.

While the Fourier transform (either continuous or discrete) is a powerful tool to analyse the frequencies of sinusoidal waves a signal is composed of, the major drawback is that the information is averaged over the whole signal and temporal information is not stored in the spectrum in a human-readable way [38]. This kind of analysis makes indeed sense for some applications, e. g., if a stationary background noise is investigated or the timbre (acoustic colour) of a musical instrument is studied. However, in most scenarios, where an excerpt of speech, a musical piece, or a non-stationary sound needs to be analysed, the raw FT or DFT are not suitable.

2.1.2.3 Short-time Fourier transform

In order to obtain both temporal and spectral information in a representation that can be both read by humans and processed by machines, the **short-time Fourier transform** (STFT) is a state-of-the-art tool. It was introduced by Gabor in the 1940s [45] and the underlying concept of short-time analysis is the one already introduced above at the beginning of this section. The signal is multiplied by a windowing function with limited support[4] prior to the transform, in order to analyse the spectrum of solely the corresponding time interval. The window is then shifted and the FT is performed, until the spectrum for all intervals has been computed. The STFT is a so-called *linear*[5] **time-frequency transform**, as the information of the signal is broken down into both temporal and spectral domains.

The *discrete* STFT [38] is computed as

$$S_{\mathrm{STFT}}(h,k) = \sum_{m=0}^{M-1} s_w(h,m) \cdot \mathrm{e}^{-j\frac{2\pi km}{M}}, \quad k = 0, \ldots, \left\lfloor \frac{M}{2} \right\rfloor, \tag{2.15}$$

for each hop h within the boundaries of the discrete signal—potentially padded with zeros to complete the frames (cf. the beginning of this Section 2.1). Of course, also for the discrete STFT, the FFT algorithm can be used. For this, it is suitable to choose a frame and window size of a power of two, e. g., 512, 1 024, or 2 048; otherwise, the frames need to be padded with zeros. As in equation 2.15, the 'D'

[3] If the number of samples N is odd, this applies only approximately.

[4] The window is non-zero only for a short time interval, e. g., 20 ms.

[5] Scaling of the input results in an output scaled with the same scaling factor.

for 'discrete' is often omitted in digital signal processing, as the continuous FT or STFT are usually not used in practice.

The choice of the frame length M implies a trade-off between the resolution in time and in frequency. The smaller the frame, the more precise is the resolution w. r. t. time and the less precise is the resolution w. r. t. frequency and vice versa. 'Resolution' in this context means how exactly an instant of time, when a sinusoid of a certain frequency is present, can be recognised, or, how well the frequency of a present tone can be recognised. This means that, e. g., in order to precisely estimate the fundamental frequency of a string, a sufficiently long analysis window is required. This trade-off is called **Gabor limit**, or even *Heisenberg-Gabor limit*, relating to the well-known *Heisenberg's uncertainty principle* in quantum physics [1, 45]. The Gabor limit states for the uncertainty in time, Δt and the uncertainty in frequency Δf, that

$$\Delta t \cdot \Delta f \geq \frac{1}{4\pi}. \tag{2.16}$$

In practice, one can apply zero-padding to a frame in order to receive a higher frequency resolution, however, this will only increase the density of the frequency bins (i. e., the bandwidth of frequency each bin k covers), while not more information is obtained about the signal. Similarly, increasing the overlap, i. e., decreasing the hop size, only returns redundant data and not more information. Nevertheless, an optimisation of frame length and overlap can put the data into a representation that is better suited for further processing or analysis.

Furthermore, the choice of the windowing function is of major importance [38]. It must be noted that the windowing process (multiplication in the time domain) introduces artefacts in the frequency domain, depending on the shape of the window. As discussed before, different common functions exist to weight the signal within a frame, where each function has certain advantages and disadvantages, e. g., if the user wants to exactly analyse a prominent frequency or rather perform signal processing in the time-frequency domain with a subsequent inverse transform. More information on the proper selection of the window function can be found in the given literature [30, 39].

The result of the STFT, $S_{\mathrm{STFT}}(h, k)$, is complex-valued, nevertheless, the phase information is usually omitted for further processing, i. e., the magnitudes $|S_{\mathrm{STFT}}(h, k)|$ are used. The reason for this is that the human sense of hearing is mainly sensitive to the strengths of the partial sinusoids that constitute the signal and less to the (relative) phase. However, this phenomenon, which is referred to as *'Ohm's acoustic law'* is only valid to a certain extent, actually, the human sense of hearing is very complex [46], involving also highly non-linear processes [33]. In any case, if the representation is going to be transformed back into the time domain using an inverse STFT, the phase information is essential.

The visualisation of the squared STFT magnitudes, i. e., the time-dependent spectral energies, is called **spectrogram**. Optionally, the logarithm is taken (with

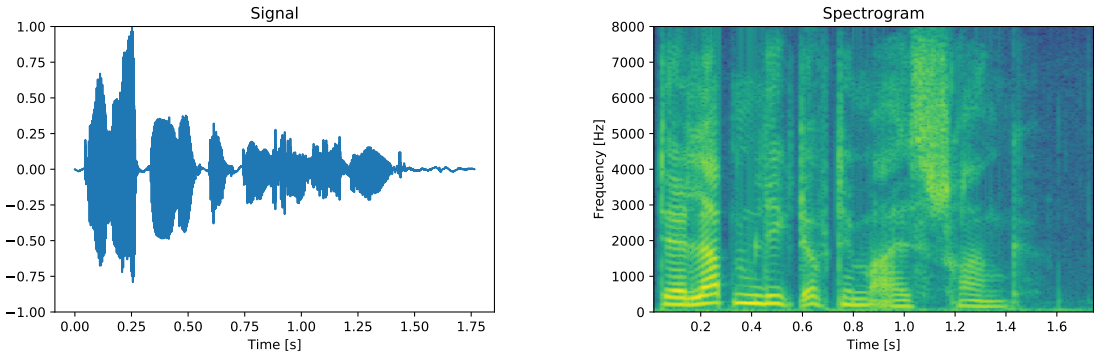

Figure 2.2: Waveform (left) and spectrogram (right) of the German sentence "Der Lappen liegt auf dem Eisschrank." (female speaker, filename: 08a01Na.wav) from Berlin EmoDB [47]. The sample rate is 16 kHz, the STFT is performed with a 'Hamming' window of a length of 512 samples (32 ms), the overlap is 75 % (8 ms, hop size 256). For the visualisation of the spectrogram, the logarithm is taken from the coefficients and the 'viridis' colour map is used (the more yellow, the larger the value).

a bias of 1 to avoid negative values), in order to compress the typically large range of values. This turns the squaring operation into a factor, which is of no relevance for the visualisation. Often, a logarithmic scale is used also for the frequency axis, as most of the energy is contained in the lower frequency bands (see Figure 2.2 on the right) and the human perception of frequencies is approximatively logarithmic [33]. The spectrogram is a very common representation of audio in many subdomains, such as, e. g., phonetics and paralinguistics [32], music information retrieval [38], or computational auditory scene analysis [48]. Nevertheless, the spectrogram does not consider any further stages of psychoacoustic modelling, which are typically used to enhance the analysis methods [48]. Figure 2.2 shows the waveform and the spectrogram of a human speech recording taken from Berlin EmoDB [47]. In the spectrogram, the horizontal axis represents time, while the vertical axis represents the frequency. The figure provides a trained phonetician already with a rough idea of what is said. The pauses between syllables are visible as well as some parallel horizontal lines originating from *voiced* phonemes. Also *formants*, i. e., local emphases of the energy in some frequency bands (see Subsection 2.1.5), can be derived, as well as the *sibilant* 's', manifesting in plenty of energy in the upper frequency bands (Time: 1.0 s to 1.2 s).

The magnitude STFT representation is already a very suitable basis for different kinds of signal processing tasks, such as, e. g., audio source separation or speech denoising [49], however, for audio recognition tasks, further processing steps are usually executed until the data is fed into an ML algorithm. As it is a common

practice for the energy (see Equation 2.9), the STFT is very often normalised to make the strength of the magnitudes independent from the frame length:

$$S_{\text{STFT,norm}}(h,k) = \frac{1}{M} \cdot |S_{\text{STFT}}(h,k)|. \tag{2.17}$$

This step is very often included in the STFT implementation itself, e. g., in the PYTHON library SCIPY[6]. Further processing steps follow in most scenarios where acoustic features are extracted for audio recognition tasks. These steps are mainly based on human perception models [33]. A very common step is to work on **power spectral densities** [30, 2.2.3.2], computed as

$$S_{\text{STFT,P,norm}}(h,k) = \frac{1}{M} \cdot |S_{\text{STFT}}(h,k)|^2. \tag{2.18}$$

From this representation, simple **spectral descriptors** can be derived, by summing up the energies of certain bands (**spectral band energies**, e. g., from 250 Hz to 650 Hz), or by computing **spectral slope**, **spectral flux**, **spectral flatness**, **spectral centroid**, and **spectral moments**. Further suitable descriptors are the **Hammarberg index**, **alpha ratio**, and **spectral roll-off points**, amongst others. Details on their definition are found in the thesis of Eyben [30, 2.2.4].

2.1.2.4 Mel-frequency spectrum

Next, similar to the common practice for spectrograms, the frequency scale, which is generally linear for the STFT output, is often transformed into a non-linear one. This takes account of the non-linear human perception of frequency, which is usually modelled in terms of the **mel scale** [33], approximating the *critical bands*. A popular definition of the mel scale z_{mel} is given by [30]

$$z_{\text{mel}} = 1127 \cdot \ln\left(1 + \frac{f}{700}\right) \tag{2.19}$$

The **mel-frequency spectrum** (sometimes also referred to with the more general term *critical band spectrum*) is then computed by compressing the power spectrum (or power spectral densities) into a certain number of bands, e. g., 26 bands, by using overlapping triangular filters which are equidistant on the mel scale and summing up the powers of the accordingly weighted bins.

The mel-frequency spectrum is used for a set of features called **cepstral coefficients** of **perceptual linear prediction** (PLP), or as an advancement, **relative spectral transform** (RASTA)-PLP, which are used in some of the experiments presented in this thesis. Based on the mel-frequency band spectrum (or another

[6] `https://www.scipy.org/`

critical band spectrum) and some further pre-processing steps, *autocorrelation* coefficients (see Subsection 2.1.1.3) are computed from which a *linear prediction analysis* (see Subsection 2.1.5.1) and finally *cepstral* coefficients are derived. The RASTA extension introduces one more pre-processing step to the critical band spectrum, enhancing modulations around 4 Hz, a frequency, speech is typically modulated with and the human hearing system is especially sensitive to. Moreover, the spectral bands from the spectrum pre-processed by mel-frequency transform and RASTA are also employed as a feature already (without computing the PLP coefficients). For more details on the RASTA, the reader is referred to the thesis of Eyben [30, 2.2.9] and in the original publications of Hermansky [50, 51].

The mel-frequency spectrum and the STFT, respectively, are further relevant for the computation of the so-called *mel-frequency cepstral coefficients*, introduced in Subsection 2.1.6 and the extraction of the fundamental frequency of a human voice or a musical instrument, as described in Subsection 2.1.4.2. As it has been shown, the plain STFT has a constant frequency resolution throughout the domain, i. e., a low frequency bin k represents the same frequency range as a high one. This is, however, not suitable given the previously discussed human perception of the frequency range approximated by the mel scale with a lower selectivity in the higher frequency bands. Thus, besides the mel-frequency spectrum, further alternatives have been proposed, particularly the *wavelet transform*, discussed in the following subsection.

2.1.3 Wavelet descriptors

The wavelet transform (WT) is a linear time-frequency transform, such as the STFT. In contrast to this, the WT is neither restricted to sinusoidal base functions nor are the frame lengths and hop sizes constant. The name 'wavelets' (i. e., 'small waves') describes the shapes of the base functions. In fact, the shape of a wavelet base function is quite arbitrary in case certain requirements are fulfilled. The windowed base functions can be considered as *band-pass filter kernels* with variable centre frequency and support [52]. Compared to the STFT, the wavelet base functions (i. e., the windowed base functions) are compressed and dilated in the time domain, depending on the frequency.

An advantage compared to the STFT is that the time resolution is better for higher frequencies, so that quick frequency changes are seen more clearly. For lower frequencies, the frequency resolution is better, i. e., the exact frequency can be determined more precisely, but the output representation does not change as quickly over time as for the high frequencies. Nevertheless, the Heisenberg-Gabor limit introduced in Subsection 2.1.2.3 applies also for the WT and cannot be challenged.

Also for the WT, a discrete version, the discrete wavelet transform (DWT), exists, which is relevant in digital signal processing. The DWT is usually realised as a cascaded filter bank, decomposing the signal in different levels [52]. For the DWT,

an inverse transform exists and as for the STFT, separate from rounding errors, the operations are lossless, i. e, the full information of the signal is kept with the DTW. Analogously to the FFT, an efficient method called *fast WT* exists, computing the DWT with discrete multi-scale band filters [53].

In some of the experiments described in this thesis, an extension of the DWT, the so-called **wavelet packet transform** (WPT) [54, 55, 56], is employed. While in the DWT, only the *low-pass signals* are further transformed (decomposed) in the cascaded levels, also the *band-pass signals* (corresponding to the higher frequency bands) are split with the same number of levels in the WPT.

As LLDs, measures of the energies of DWT and WPT are used. From the relative energies in each decomposition level, certain statistical measures (*mean*, *variance*, *entropy*, and *waveform length*) are computed for the DWT [55, 57]. For the WPT, the *normalised filter bank energies* [56] are computed. More details about these features are found in the corresponding references. The LLDs introduced in the following are not based on the DWT or WPT.

2.1.4 Prosodic features

In general, *prosody* characterises parameters of natural speech that are not connected to the phonemes (distinguishable sound units in natural speech) [58]. The most commonly used parameters are

1. stress (accentuation),
2. intonation (speech melody), and
3. rhythm & tempo.

Also **voice quality**, e. g., a *harsh* or a *tense* voice, is very often considered a prosodic parameter [32, 59, 60]. Even though prosody is independent from the phoneme, it can be decisive in resolving some lexical ambiguities. Consider the *homographs*[7] 'übersetzen'[8] (German) and 'project'[9] (English), which are pronounced in the same way and differ only by a different stress. On the acoustic side, prosodic parameters have a more or less close correspondence with some LLDs.

2.1.4.1 Stress (accentuation)

Stress (accentuation) is correlated with the contour of the *intensity* over time [61] (see Subsection 2.1.1.1). As an alternative, **loudness** is a much more sophisticated descriptor, modelling human perception by taking into account different sensitivities of the human ear depending on the frequency band and *masking* effects [33].

[7] words of the same spelling

[8] 'übersétzen': to translate vs 'übersetzen': to ferry across the river

[9] 'próject' (noun) vs 'projéct' (verb)

2.1.4.2 Intonation and fundamental frequency (F0)

The **intonation or speech melody** is related to the *pitch* of the human voice [62]. While pitch describes a perceived quality [63], the *fundamental frequency*, usually called *F0* is the technical counterpart and often used as a prosodic feature [64, 65]. For the computation of F0, a large number of algorithms and refinements exists [66]. Many approaches are optimised for specific usage scenarios (speech or singing; background noise or music) and come with certain pros and cons, resulting in different performances [67]. Some approaches are based on only the time domain (usually based on the ACF) [30, p. 63 et seq.], some exploit the frequency or time-frequency domain [66, 68, 69], and more recent ones employ a neural network regressor using the raw time-domain signal as input [70].

In the experiments discussed in this thesis, the F0 LLD is obtained from the *subharmonic summation* (SHS) algorithm. Voiced speech and music signals generally consist of complex tones, consisting of, i. a., an oscillation of a certain frequency (fundamental frequency) and corresponding harmonics (or *overtones*), which are oscillations of integer multiples of the fundamental frequency. This is exemplified in Figure 2.3, where the waveform signal and the corresponding magnitude spectrum from the articulation of the vowel 'a' (German) are visualised. The (quasi)-periodicity of the waveform is obvious and the maximum peak of the spectrum is located around a frequency of 150 Hz, which is the F0 of this speaker. The next peak is at the duplicate of F0, at around 300 Hz. Further harmonics are present at around 450 Hz, 600 Hz, and so on, but with a much smaller magnitude. The (relative) magnitudes of the harmonics are relevant for the perceived timbre of a sound or complex tone and are important for the distinction between phonemes (see Subsection 2.1.5). Besides the intended variations of the fundamental frequency in order to form the intonation, to convey emotion, or a singing melody, F0 is also a voice characteristic of each speaker, where women usually have a higher F0 than men [71].

The sensation of pitch is mainly related to the fundamental frequency and not the harmonics. Moreover, the fundamental frequency can—in many cases—even be recognised if it is not present in the signal, e. g., if a speech signal is transmitted over a band-pass channel (such as telephone). This phenomenon is referred to as *residue pitch* or *virtual pitch* [33] and taken into account by the SHS method. The SHS algorithm introduced by Hermes in 1988 [72] is based on the STFT of the signal. First, the magnitudes $|S_{\mathrm{STFT}}(h, k)|$ are extracted with a frame/window length of 60 ms and a *Gaussian* window. Then, the short-time spectrum for each time step h is pre-processed by enhancing local maxima in setting magnitudes distant from them to 0 and then smoothing with a simple filter over the frequency domain. Next, the frequency scale is transformed into an octave scale and an auditory weighting is applied [30, p. 66]. Then, the pre-processed spectrum is iteratively shifted with a constant factor on the octave scale frequency axis and summed up with a de-

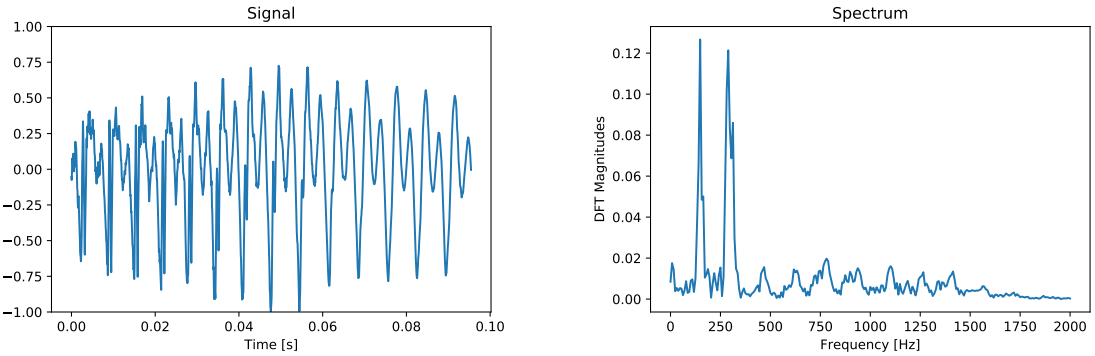

Figure 2.3: Waveform (left) and magnitude spectrum (right) of the vowel 'a' in the German word "Eisschrank" (female speaker, filename: 08a01Na.wav) from Berlin EmoDB [47]. The frequency axis is cropped to 2 kHz for a better visualisation.

creasing weighting factor. The iterative summation of the harmonics ensures that the corresponding fundamental frequency is boosted. Finally, from the *subharmonic sum spectrum*, the F0 can be selected with a *peak picking* algorithm, where different realisations are possible [30, p. 67 et seq.]. Besides F0, also the so-called **voicing probability** [30, p. 69] can be derived from the SHS and is relevant in two respects. Firstly, it says how likely it is that a certain frame of speech does contain a pitch at all. Frames with a voicing probability below a certain threshold can be excluded from the further processing chain. Secondly, the voicing probability can be considered an LLD itself. Finally, a step that is appropriate to improve the LLDs by exploitation of the temporal context, a *temporal smoothing* can be performed (see Subsection 2.1.7).

2.1.4.3 Rhythm

Rhythm and tempo can be expressed by the duration of different speech units [73] or the rate of syllables per second [32, p. 66]. These descriptors are mainly based on the contour of the energy, but also the *voicing probability* can be used to distinguish between *voiced* (F0 detected) and *unvoiced* (F0 not detected) segments, from which a measure for the rate of speech can be derived [74]. Moreover, a descriptor of speech rhythm can be derived as well from the frequency-domain representation of the envelope of band-pass-filtered speech [75].

2.1.4.4 Voice quality

Voice quality is described by the *harmonics-to-noise ratio* (HNR) and the *micro-prosodic* features *jitter* and *shimmer* [1, 6.2]. HNR can be derived from the ACF (see Subsection 2.1.1.3), by calculating the ratio between the energy of the peak

representing F0 and the overall energy [1, p. 61]. As it is the common practice for energy ratios, the logarithm is often taken and the **logHNR** is obtained. Jitter and shimmer describe micro-variations of F0 (jitter) and amplitude (shimmer) between successive fundamental periods, i. e., [30, p. 73 et seqq.]. However, the measure of jitter is not only depending on the voice quality, but is also affected by the age of a subject [76] and her/his heartbeat [77].

Even though, as mentioned above, prosody is also relevant for the linguistic meaning of speech [60], it is generally only of minor importance for ASR systems, but of central significance for *paralinguistics* [64]. Paralinguistics covers all aspects of speech which are not related to 'what is said', but to 'how it is said' [32]. A typical and well-studied paralinguistic task in CA is the recognition of affect and emotion [78, 79]. Cowie et al. studied the relationships between emotion and prosodic features [80]. Ishi et al. showed that also voice quality features prove to be suitable for emotion recognition [64]. Further tasks include the assessment of age and gender [81], personality in terms of the OCEAN big-five dimensions[10] [82], intoxication [83], eating condition [21, 84], physical or mental load [85], sleepiness [13, 83], dialect [13], native language [22], deception and sincerity [22], and health conditions, such as, e. g., Parkinson's disease [21, 86], autism spectrum condition [20, 87, 88], or cold [11].

In the last few decades, *brute-force* modelling, i. e., extracting a large set of descriptors and then let the machine decide on the relevance of certain descriptors, has become a state-of-the-art, however, phoneticians conducted a lot of research in the past, with the goal of analysing and understanding the suitability of features for certain tasks. For example, speech *tempo* has turned out to be a good indicator for *non-nativity* of the speaker [32], the dynamics of pitch over time have proven to be important for the recognition of *arousal* [32]. Arousal is one dimension of a multi-dimensional model of human affect and reflects the level of mental activation associated with the emotional state [89]. Moreover, prosodic descriptors are suitable for the detection of *sarcasm*. Cheang and Pell found that sarcastic speech can be reliably recognised based on F0 and the speech rate [90]. Besides tempo, also intonation is known to be language-specific [62, 91], where language-dependence is a basic aspect which needs to be considered generally, when it comes to building systems analysing both linguistic and paralinguistic parameters, such as ASR and emotion recognition [17, 19].

As indicated before, the best representation of prosody (and other feature types) is not always the raw time-dependent LLD, but also its differentials between successive frames. Consider the case of a subject speaking with high arousal, where usually large dynamics in F0 can be recognised, whereas the absolute or mean values

[10] Openness, Conscientiousness, Extraversion, Agreeableness, Neuroticism

of pitch depend strongly on the gender [92]. Differences between successive frames are also called *deltas* and will be introduced below in Subsection 2.1.7.

2.1.5 Formants

In contrast to the prosodic features, **formants** describe properties of speech that are highly related to the phoneme and the spectral shaping. The same concept is sometimes applied to musical instruments and other types of sound sources. The formants of a sound are the frequency ranges, in which the spectral envelopes, i. e., the contour of the harmonics, are amplified [71, p. 305 et seqq.].

A human speech signal contains several formants, where the first two (called F_1 and F_2, the two formants with the lowest frequencies) determine the vowel spoken, and the third one F_3 or subsequent ones are relevant mainly for the timbre of the voice. As an example, the vowel 'a' typically has its F_1 and F_2 at 900 Hz and 1200 Hz, respectively [71]. The formant frequencies are independent from F0, which leads to the fact that one vowel has the same characteristic sound, no matter which person is speaking or which musical note is vocalised by a singer. Based on the harmonic spectrum with a certain fundamental frequency that is generated by an oscillation of the vocal folds in the human vocal tract, the formants are shaped by the positions of the tongue and the lips during speaking or singing. Analogously, for musical instruments, the harmonic spectrum of a fundamental frequency given by, e. g., the string in the case of a string instrument, is filtered by the material and shape of the resonator, resulting in a characteristic timbre. Nevertheless, these 'formants' are less distinctive than those of the human voice, while the temporal envelope and the transient effects during *attack*, *decay*, *sustain*, and *release* phases of a musical note are quite specific for each musical instrument [71].

Even though the formants are visible as the envelopes of the short-time magnitude spectrum, it is not a suitable approach to derive them directly as the local maxima, due to disturbances by F0 and noise [30, 2.2.8]. A straightforward solution is to smooth the magnitude spectrum with a low-pass filter in order to obtain the spectral envelope [30]. Another method, the one which is used in the experiments in the remainder of this thesis, is based on *linear predictive coding* (LPC) [30].

2.1.5.1 Linear source-filter model

The principle of *linear prediction* is also exploited for the PLP descriptors introduced earlier in this thesis 2.1.2.4, where the coefficients are computed from the power spectrum; according to the *Wiener–Khinchin theorem*, the power spectrum of a signal is the Fourier transform of its ACF. LPC is a common method to model time-discrete signals, especially speech signals. It is used in many sub-domains, such as synthesis, analysis, and coding of speech [93]. In LPC, the signal $s(n)$ at discrete time n is approximated as a weighted sum of P previous samples, i. e., the

approximated signal is given by

$$\hat{s}(n) = -\sum_{i=1}^{P} a_i s(n-i). \tag{2.20}$$

The number of contextual samples considered, P, is also referred to as the *order* of the LPC and a_i are the corresponding *coefficients*. The error term $e(n) = s(n) - \hat{s}(n)$ is then given by

$$e(n) = \sum_{i=0}^{P} a_i s(n-i), \quad \text{with } a_0 = 1. \tag{2.21}$$

The *z-transform* is an elegant tool to represent and analyse time-discrete signals and *linear shift-invariant* filters and it is suitable as such to handle linear predictive modelling. Given the signal from Definition 2.1, the z-transform of $s(n)$ is defined as [43]

$$S(z) = \sum_{n=0}^{N-1} s(n) \cdot z^{-n}. \tag{2.22}$$

Analogously, for arbitrary signals or systems (filters), the summation goes over the whole support. In practice, for the purpose of the computation of formants, the transform is done over a windowed—and possibly pre-emphasised [30]—frame. Pre-emphasis can be realised by a simple 1$^{\text{st}}$-order linear filter executing the operation $s_{\text{emph}}(n) = s(n) - k_{\text{emph}} s(n-1)$ in the time domain, or $S_{\text{emph}}(z) = S(z) - k_{\text{emph}} z^{-1} S(z)$ in the z-domain [43], where the pre-emphasis coefficient k_{emph} is often chosen to be 0.97 [30, 94]. This operation attenuates the (typically prominent) low-frequencies of the signal and thus behaves like a high-pass filter, resulting in a more balanced magnitude spectrum [95]. In order to simplify the readability, $S(z)$ frame indexes and $_{\text{emph}}$ subscripts are omitted until the LLDs are defined at the end of this subsection, but it must be kept in mind, that all corresponding computations need to be executed for each frame.

As a shift in the time domain $s(n - n_0)$ translates to a factor of z^{-n_0} in the z-domain, Equation 2.21 can be expressed as

$$E(z) = \sum_{i=0}^{P} a_i S(z) z^{-i}, \quad \text{with } a_0 = 1. \tag{2.23}$$

The advantage of this representation is that $S(z)$ can be pulled out of the summation and the equation can be divided by the summation and $E(z)$ resulting in

$$H_{\text{voc}}(z) = \frac{S(z)}{E(z)} = \frac{1}{1 + \sum_{i=1}^{P} a_i z^{-i}}. \tag{2.24}$$

The specifications are now made that

1. $E(z)$ represents the *excitation* source, elicited by either the vibrations of the vocal folds in case of voiced phonemes or white[11] noise in case of (most) unvoiced phonemes, and

2. $S(z)$ represents the *speech* signal.

Under these assumptions, $H_{\text{voc}}(z)$ is the z-domain transfer function of the vocal tract, i. e., the filter function of the vocal tract. It must be noted that $H_{\text{voc}}(z)$ is defined only by the coefficients a_i. As the signal is modelled by $S(z) = E(z) \cdot H_{\text{voc}}(z)$, this model of speech is called the **(simplified) linear source-filter model** [1]. It must be pointed out that this model is just an approximation of the filter transfer function of the vocal tract. Firstly, it applies primarily to vowels and some types of consonants; secondly, a recorded speech signal includes also the transfer functions of the radiation from the mouth and the acoustics of the room [30], i. e., these sources of error must be considered when performing LPC analysis.

The coefficients a_i are defined by the resonator characteristics of the vocal tract, thus, mainly by the positions of tongue and lips. As the speech signal is a product of source signal and filter response in the z-domain (and also in the Fourier domain), the peaks of the filter are exactly the resonance frequencies, or, the formants.

To estimate the model parameters a_i, several methods can be applied minimising $e(n)$, such as the estimation from the *autocorrelation* coefficients; an overview is given by Eyben [30, 2.2.7]. The LPC can also be considered an *autoregressive* model as future samples of a signal are a linear combination of their predecessors. The vocal tract filter model can be realised as an *infinite impulse response* filter [43].

2.1.5.2 Formant descriptors

To determine the formant frequencies, the poles of $H_{\text{voc}}(z)$ need to be computed, or, as the dual problem, the roots (zeros) of $H_{\text{voc}}^{-1}(z)$, using numeric methods [1]. Boersma, who developed the voice analysis software Praat [96], described the following method to derive formant frequencies. First, from all poles p_i, those which are outside the unit circle are mapped into it as follows:

$$p_i \leftarrow \frac{1}{p_i^*} \quad \text{if} |p_i| > 1, \tag{2.25}$$

where * denotes the complex-conjugate. This operation does not change neither the corresponding frequency nor the bandwidth. Furthermore, as each pole is present twice with its complex-conjugate pair, only the ones with a positive imaginary part are considered. For each frame h, the **formant frequencies** are given by [30]

$$F_i(h) = \frac{1}{2\pi T_s} \left\| \arctan\left(\frac{\Im\{p_i(h)\}}{\Re\{p_i(h)\}} \right) \right\| \tag{2.26}$$

[11] White noise is a stochastic signal with a constant power density spectrum across the whole frequency range.

and the corresponding **bandwidths** by

$$F_{\mathrm{B},i}(h) = \log\left(\frac{|p_i(h)|}{2\pi T_s}\right). \tag{2.27}$$

Usually, not more than the first three formants, (F_1, F_2, and F_3), are employed as acoustic features [31]. Further restrictions can be applied (minimum frequency and maximum bandwidth) to ensure that the estimated formants are plausible and the method has not failed due to disturbances or absence of a pronounced formant [97]. In addition to the bandwidth—or alternatively—the **STFT energies** or **magnitudes** of corresponding frequencies can be used as a feature. However, it should be considered that energies might be at a local minimum in frequency ranges that are not multiples of F0. Thus, a smoothing of the magnitude or energy short-time spectrum is advisable.

2.1.6 Mel-frequency cepstral coefficients

It has already been shown that the LLDs of speech can be—loosely speaking—subdivided into prosodic ones, which are closely related to the speaking style, and the ones that contain information about the phonemes, e. g., the formants, which contain more or less low-level linguistic information. In the following, a major LLD in machine speech analysis is introduced, which belongs to the second group, and has been the most widely used one in ASR: the **mel-frequency cepstral coefficients** (MFCCs) [98, 99].

The mel scale has already been introduced in Section 2.1.2.4, to non-linearly transform the frequency axis into a scale mimicking properties of the human sense of hearing. Before presenting the steps to compute MFCCs, another important concept needs to be introduced, the **cepstrum**. Though the original definition, introduced by Bogert et al. in 1963 [100], was the *power cepstrum* and also further specifications exists, such as the *complex cepstrum*, the following definition for the short-time cepstrum is used in this thesis [30, 101]:

$$\mathrm{CEP}(h,q) = \mathrm{DFT}^{-1}\{\ln S_{\mathrm{STFT,P,norm}}(h,k)\}, \tag{2.28}$$

with the inverse DFT (DFT^{-1}) of a short-time spectrum $X(h,k)$ given by

$$\mathrm{DFT}^{-1}\{X(h,k)\} = \sum_{k=0}^{M-1} X(h,k)\cdot \mathrm{e}^{j\frac{2\pi kq}{M}}, \quad q = 0,...,M-1. \tag{2.29}$$

It must be noted that the complex exponent does not have a negative sign for the inverse DFT, in contrast to the DFT, and, that the normalisation of a coefficients needs to be performed if it has not been done as part of the DFT or STFT. Furthermore, the upper half of the short-time spectrum $X(h,k)$ must be added (mirrored and complex-conjugated) in order to be able to perform the summation up to $M-1$.

The result of the inverse transform, which is the cepstrum when using the logarithm of the power spectrum, is a function of the *quefrency*[12] q [100]. As the input of the analysis is a time-domain signal, also the quefrency has a dimensionality of time. Nevertheless, it is not similar to the original signal, as the STFT has undergone the operations of squaring and logarithm. Thus, it is closely related to the ACF, which is the result of the inverse transform of the power spectrum (see also Section 2.1.5). The only difference of the cepstrum is that the logarithm of the power spectrum is subject to the inverse transform.

The advantage of the logarithm in the frequency domain is that products turn into sums[13]. As mentioned in Section 2.1.5.1, filtering operations, such as the vocal tract transfer of an harmonic excitation signal, are multiplications of the signal spectrum and the frequency response of the filter in the frequency domain. Converting this into sums makes it easier to separate excitation signal and vocal filter parts in subsequent processing steps. Moreover, harmonic signal parts with low amplitudes can be recognised more easily. The inverse DFT is (as the DFT) a linear operation and preserves the additive characteristic. Consequently, the cepstrum is, i. a., well suited for the estimation of either the fundamental frequency or the vocal tract transfer function (formants), as the effects of the other can be suppressed more easily.

Nevertheless, with the MFCCs, a more compact representation than the raw cepstrum has been established [102]. Although there are variations between different implementations [99], the computation steps are typically similar to what is described in the following [30]:

1. The (normalised) short-time magnitude spectrum (see Equation 2.17) or power spectrum (see Equation 2.18) is computed. Sometimes, the signal is pre-emphasised before (see Section 2.1.5.1).

2. The mel-frequency (power) spectrum is obtained as described in Section 2.1.2.4, with $B = 26$ bands, ranging from 20 Hz to 8 000 Hz [30].

3. The logarithm is applied, except for very small magnitudes/powers, which are below a certain threshold, in order to avoid large negative numbers [30].

4. Finally, the discrete cosine transform (DCT, more specifically, DCT type-II) [103] is applied. Assuming that the result of step 3 is $S_{\text{logMel}}(b, h)$ for each frame h, with the mel-band b, the MFCCs are computed as

$$\text{MFCC}^*{}_{k_c}(h) = \sqrt{\frac{2}{B}} \cdot \sum_{b=0}^{B-1} S_{\text{logMel}}(b, h) \cos\left(\frac{\pi k_c}{B}\left(b + \frac{1}{2}\right)\right). \tag{2.30}$$

[12] The terms 'cepstrum' and 'quefrency' were derived from 'spectrum' and 'frequency' by Bogart et al. in order to point out their origin from the backwards transform.

[13] $\log xy = \log x + \log y$

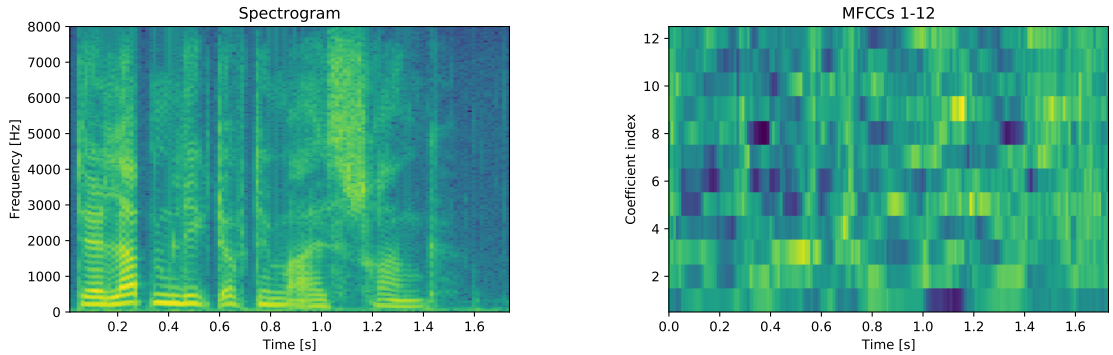

Figure 2.4: Spectrogram (left) and MFCCs 1 to 12 (right) of the German sentence "Der Lappen liegt auf dem Eisschrank." (female speaker, filename: 08a01Na.wav) from Berlin EmoDB [47]. The sample rate is 16 kHz, a pre-emphasis with $k_{\text{emph}} = 0.97$ is done, the STFTs for both spectrogram and MFCCs are performed with a 'Hamming' window of a length of 400 samples (25 ms), the hop size is 160 samples (10 ms, i. e., the overlap is 60 %). Liftering is performed with a coefficient of 22. For the visualisation of the spectrogram, the logarithm is taken from the coefficients. For the MFCCs, the energy coefficient (MFCC 0) is not shown as its range is larger than the range of the other coefficients. For both, the 'viridis' colour map is used (the more yellow, the larger the value).

> From this result of the DCT, typically, only the first 12 to 16 coefficients with corresponding index k_c are used [30]. As it can be seen in Equation 2.30, the 0-th coefficient is computed as the unweighted sum of the log-mel spectrum and is therefore closely related to the energy of the frame.

The DCT has a similar effect as the inverse DFT, which is used to generate the cepstrum. The advantage of the DCT is that it is more suited to represent magnitude spectrum-like information, i. e., data that exhibits most of its energy in the lower frequency range (cf. Figure 2.3), in just a low number of coefficients [104]. The DCT implicitly assumes a symmetric continuation of the function to be transformed, while the (inverse) DFT assumes a periodic one and therefore rather faces jumps at the boundaries. As a post-processing step, *liftering* can be applied to emphasise the—usually more important—lower coefficients with a coefficient-specific weight [30]:

$$\mathrm{MFCC}_{k_c}(h) = \mathrm{MFCC}^*{}_{k_c}(h) \left(1 + \frac{L_c}{2} \sin \frac{\pi c_k}{L_c}\right), \tag{2.31}$$

where L_c is the *liftering coefficient*.

Figure 2.4 compares the spectrogram representation of the speech signal shown before with its MFCC representation. It can be seen that the MFCC representation is less structured than the spectrogram and cannot be interpreted easily by humans.

Nevertheless, it is a compact descriptor of a speech signal with little redundancy and well-suited as an input for ML. As mentioned, MFCCs are a set of well-established LLDs in CA and can be used for a great variety of tasks, ranging from ASR [102] to ASC [105] and MIR [106].

2.1.7 Smoothing and deltas

All LLDs introduced in the previous subsections have in common that they have been computed from single frames. This comes with two problems: first, the short-time analysis involves some artefacts, as the quasi-stationarity assumption is not completely fulfilled in practice [30]. This issue is addressed by applying a **smoothing** operation, where a simple *moving average* filter over the LLD values of a group of adjacent frames has proven to be suitable [107]. I. e., for an arbitrary LLD $D(h)$, the smoothed LLD contour is given by

$$D_{\text{sma}}(h) = \frac{1}{W_{\text{sma}}} \sum_{i=-\frac{1}{2}(W_{\text{sma}}-1)}^{\frac{1}{2}(W_{\text{sma}}-1)} D(h+i), \tag{2.32}$$

with the odd-numbered length of the smoothing average window W_{sma}[14], typically set to 3 [30].

The second problem is that although the employed frame lengths can vary, no contextual information, i. e., information from neighbouring frames, is taken into account and captured, so far. However, as it will also be shown later in this thesis (see Chapter 6), the differential information between frames of certain LLDs is sometimes more relevant than the descriptor itself. Consider the scalar LLDs of intensity or F0, which usually contain information about the overall level of a recording and the gender of a person, respectively, or, in case of music signals, information about the pitch of the current note. In case, e. g., the emotion is to be classified, the mentioned parameters are not of interest as information about the affective states of a person or the mood of a song is rather encoded in the **dynamics** of these descriptors [108]. As an example, the *dominance* of a speaker has been found to be correlated with the rise and falls of intensity and F0 ranges [109].

The simplest way to define the differential information is by computing the **differences** between the LLDs of adjacent frames, i. e., for the smoothed LLDs, the differences $\Delta_{\text{diff}} D_{\text{sma}}(h)$ are

$$\Delta_{\text{diff}} D_{\text{sma}}(h) = \begin{cases} D_{\text{sma}}(h) - D_{\text{sma}}(h-1), & \text{if} \quad 1 \leq h \leq N_h - 1 \quad \text{and} \\ 0 & \text{if} \quad h = 0. \end{cases} \tag{2.33}$$

Another method to capture dynamic information of the LLDs is the so-called **delta regression**, as proposed by Young et al. [102]. It captures the differences over more

[14] sma: smoothed with moving average

than just one frame step and implies a smoothing operation. The delta regression $\Delta D_{\text{sma}}(h)$ is the preferred method used in the experiments in the remainder of this thesis and defined as follows:

$$\Delta D_{\text{sma}}(h) = \begin{cases} \frac{\sum_{i=1}^{W_\Delta} i \cdot (D_{\text{sma}}(h+i) - D_{\text{sma}}(h-i))}{2 \sum_{i=1}^{W_\Delta} i^2}, & \text{if} \quad W_\Delta \leq h \leq N_h - 1 - W_\Delta \quad \text{and} \\ 0 & \text{otherwise.} \end{cases} \tag{2.34}$$

The size of the context window W_Δ is often chosen as 2 [30]. The larger the window, the more the long term slopes and dynamics are captured by the delta regression coefficients (deltas). Additionally, the **double delta**, or **acceleration**, coefficients $\Delta\Delta D_{\text{sma}}(h)$ are commonly used [102], defined by simply executing the operation in Equation 2.34 to the deltas.

Finally, it must be noted that for *jitter*, in addition to the 'default' *local* coefficients, the 'difference of differences of periods' (DDP) [96] is used as a standard descriptor [30], without consideration of the delta regression and computed as the absolute value of the difference (i. e., the absolute value of the result of Equation 2.33).

2.2 Instance-level Features

All acoustic features defined so far in this thesis are frame-level features, or LLDs, describing only very local properties of the audio. The whole signal is thus represented as a sequence of LLD vectors, with one LLD vector for each frame. For many paralinguistic CA tasks, such as emotion recognition or music genre classification, a machine requires an input of a duration from a few seconds up to a minute in order to be enabled to make a decision [30]. The necessary amount of data highly depends on the task, if a personal trait (such as, e. g., the age or the gender of a person) or a rapidly changing state (such as, e. g., emotion) is to be classified. This is in contrast to classical ASR models, where a first classification is made on phoneme level ($\ll 1\,s$), followed by decisions on word and sentence level [110].

Usually, paralinguistic datasets for ML are provided in a suitable format, either as a chunk-labelling task, with one label per chunk, where the chunks usually have a variable duration (e. g., in ComParE 2019 [13]), or, as a *sequence labelling* task, with one label for each time stamp on a uniformly sampled grid (e. g., in AVEC 2019, Cross-cultural Affect task [17]). Many ML models in the field of CA are so-called *static* approaches (see Chapter 4), i. e., they require one feature vector of a constant size for each label to be predicted, the so-called **instance-level** feature vector (see Figure 1.1). Generally, an instance-level feature vector can be created by *stacking* (concatenating) all LLDs into one large vector of a dimensionality of N_h times the number of LLDs [30]. In case one chunk is the instance to be classified, all sequences would need to be padded to match with the chunk of the longest

duration. However, this approach is not very suitable, as the feature vectors are not *shift-invariant*. This means, if two audio recordings have a very similar (or even the same) content with a slight temporal shift between them, the feature vectors look very different, comparing them 'index-wise'. Such kind of shifts are very difficult to learn for most ML approaches as one single instance-level feature is supposed to have the same meaning in all instances. Moreover, the overall vector would be quite large for an audio file of several seconds duration.

For these reasons, it has been established to 'summarise' the information of one instance as **supra-segmental** features [32] in order to create the instance-level feature vector. For this, several methods have been proposed. Probably the most established one is the use of **functionals**, introduced below in Subsection 2.2.1. An alternative is the **bag-of-audio-words** (BoAW) approach, which is the central theme of this thesis and introduced in Chapter 3.

2.2.1 Functionals

A functional maps a time series of a value of arbitrary length (the contour of a certain LLD) into a scalar [32]. Functionals can also be considered as *statistics* or descriptors of the properties of one individual LLD contour $D(h)$. In the following, a non-exhaustive list of functionals $F_{\text{Func}}(D)$ is given; for further information, the reader is referred to the thesis of Eyben [30]:

- Moments:
  - Arithmetic mean: $F_{\mu}(D) = \frac{1}{N_h} \sum_{h=0}^{N_h-1} D(h)$ (1^{st} order statistical moment),
  - Variance: $F_{\sigma^2}(D) = \frac{1}{N_h} \sum_{h=0}^{N_h-1} (D(h)-\mu)^2$ (2^{nd} order statistical moment),
  - Standard deviation: $F_{\sigma}(D) = \sqrt{F_{\sigma^2}(D)}$,
  - Coefficient of variation: $F_{\bar{\sigma}}(D) = \frac{F_{\sigma}(D)}{F_{\mu}(D)}$,
  - Skewness: $F_{\text{skew}}(D) = \frac{1}{N_h F_{\sigma}^3(D)} \sum_{h=0}^{N_h-1} (D(h)-\mu)^3$ (standardised 3^{rd} order statistical moment),
  - Kurtosis: $F_{\text{kurt}}(D) = \frac{1}{N_h F_{\sigma}^4(D)} \sum_{h=0}^{N_h-1} (D(h)-\mu)^4$ (standardised 4^{th} order statistical moment).
- Extreme values:
  - Absolute maximum: $F_{\max}(D) = \max_h D(h)$,
  - Absolute minimum: $F_{\min}(D) = \min_h D(h)$,
  - Index of the absolute maximum: $F_{\text{posmax}}(D) = \arg\max_h D(h)$,
  - Index of the absolute minimum: $F_{\text{posmin}}(D) = \arg\min_h D(h)$,

  - Range: $F_{\text{range}}(D) = F_{\max}(D) - F_{\min}(D)$.

- Percentiles $F_{\text{p}}(D)$, i. e., the value below which a certain percentage p of all values in $D(h)$ lies. Percentiles are less sensitive than extreme values under presence of outliers and can replace them. Typical choices are the 1^{st} or 5^{th} percentile and the 99^{th} or 95^{th} percentile to get a more robust alternative for absolute minimum and maximum, respectively, and the functionals related to them.

- Linear regression coefficients (slope and offset: $F_{\text{slope}}(D)$ & $F_{\text{offset}}(D)$), similar to the spectral slope (see Section 2.1.2.3) and the corresponding regression error $F_{\text{error}}(D)$.

- Up- and down-level times, $F_{\text{up-level}}(D)$ & $F_{\text{down-level}}(D)$, i. e., the absolute or relative number of frames, the signal is above/below a certain threshold. Typical thresholds are: 0.25, 0.5, 0.75, and 0.9.

- Rise and fall times $F_{\text{rise}}(D)$ & $F_{\text{fall}}(D)$, i. e., the absolute or relative number of frames, the signal is rising/falling, corresponding to the positive/negative result of $D(h) - D(h-1)$.

- Peaks and valleys, i. e., local maximum/minimum values, with a large variety of derived functionals, such as, e. g., (the exhaustive list is found in [30]):

  - Number of peaks/valleys,
  - Arithmetic mean of peak/valley amplitudes and their difference to $F_{\mu}(D)$,
  - Peak/valley amplitude range,
  - Slopes between peaks/valleys.

- Onsets and offsets (see [30])

As already mentioned, depending on if the recognition task is a chunk- or a sequence-labelling task, the instance to predict a label for can either be a full recording or each time stamp from a recording. In the first case, the functionals are—obviously—computed over the whole chunk [111], in the second case, they are computed over a certain segment [112], sometimes also referred to as *window* or *block*, not to be confused with the windowing functions from the short-time processing defined at the beginning of this section. The functionals of F0 contours and voice quality descriptors are usually computed only based on those frames where the voicing probability is above a certain threshold [31]. The process of generating supra-segmental instance-level features using functionals is illustrated in Figure 2.5.

As a final note, it must be pointed out that instance-level features are not essential for all types of ML models for audio recognition. For example, *dynamic* approaches, such as *hidden Markov models* (HMMs) [113, 114] and several types of

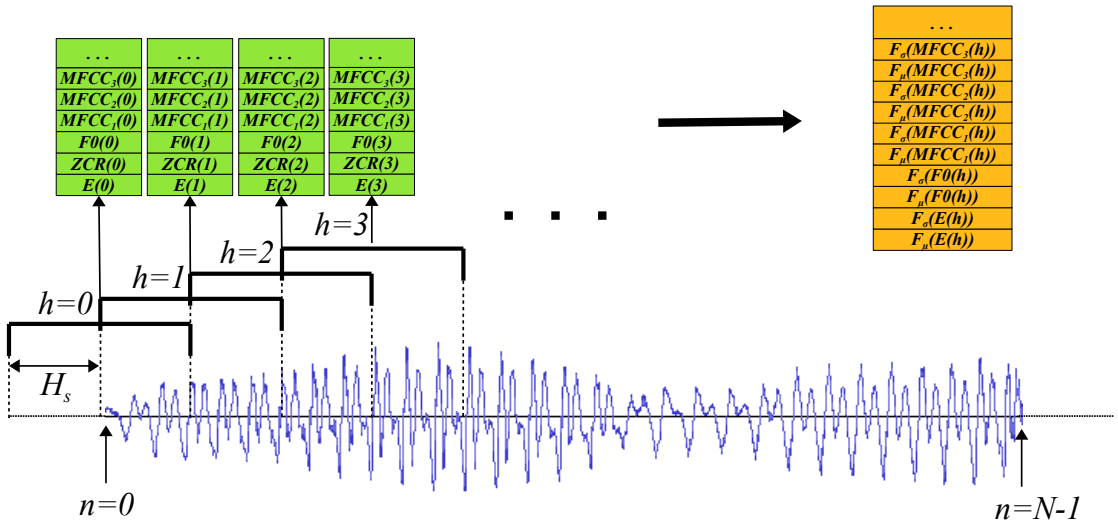

Figure 2.5: Illustration of frame-level feature (orange) generation from acoustic LLDs (green) using functionals. Only a small selection of potential LLDs (*energy*, *ZCR*, *F0*, & *MFCCs 1–3*) and functionals (*arithmetic mean* & *standard deviation*) are shown.

neural networks are able to handle time-series input of variable length in a robust way. Nevertheless, functionals over short segments of audio are often considered also when using these models, in order to reduce the amount of input data, to increase the robustness, or to synchronise with the hop size of the target labels in case of a sequence labelling task [115].

2.3 Feature Sets

To conclude with this chapter, two well-established and public acoustic feature sets with functionals will be introduced. They have established in the CA community and proven to be suitable for many kinds of tasks. For this reason, they are used throughout this thesis and have been used in related publications as a **baseline**, i. e., a comparative approach to benchmark novel methods against. The definitions of the two sets follow completely different paradigms.

2.3.1 ComParE

The ComParE feature set [20, 30], introduced at the Computational Paralinguistics ChallengE in 2013, is a *large scale* feature set, consisting of **6 373 instance-level features**, originating from **65 LLDs** with corresponding deltas and up to 54 functionals applied to them [30]. The set is referred to as a (semi) *brute-force* set as

single features have not been hand-picked, motivated by phonetic or acoustic considerations [32, 109], but rather by aggregating established LLDs and functionals that have proven their usefulness in previous research and form a final feature vector of large dimensionality. Moreover, the ComParE feature set is not a task-specific one, but due to the large variety of its components, it has proven to be meaningful for a wide range of speech and audio tasks, such as, e. g., the recognition of emotion [12, 20, 116], autism spectrum condition [20, 111], Parkinson's disease [21], snoring [11, 117], and sleepiness [13]. Good recognition performances have been achieved throughout in combination with a classifier that is able to handle large input vectors, such as a *support vector machine* (SVM) with a linear kernel (see Chapter 4).

ComParE contains all the LLDs listed in the left column of Table 2.1 [30]. LLDs are extracted with a hop size of 10 ms; a frame length of 60 ms and a Gaussian window are used for *F0*, the corresponding *voicing probability*, *jitter*, *shimmer*, and *logHNR*, a frame length of 20 ms and a Hamming window are used for all other LLDs, except for the *RMS* and the *ZCR*, which are extracted with rectangular windows from 20 ms and 60 ms frames, respectively. A pre-emphasis is not applied. For all LLDs, the corresponding *deltas* are included, but no *double deltas*.

For the functionals, the reader is referred to the thesis of Eyben [30, p. 129]. Different functionals are applied depending on the feature group. For *F0*, *voicing probability*, *logHNR*, *jitter*, and *shimmer*, the same 39 functionals are applied for both the descriptors and corresponding deltas. For all other LLDs, 54 functionals are applied to the descriptors, and 46 to the deltas. In brief, *moments*, *percentiles* (with corresponding ranges), *regression coefficients*, *up-level times*, and *rise times* are computed for all LLDs, while functionals describing *peaks and valleys* of the contours are only considered for the latter group. The set is complemented with five global descriptors, namely, the *percentage of voiced frames*, and the *minimum*, *maximum*, *mean*, and *standard deviation* of voiced segments lengths, totalling up to 6 373 instance-level features.

2.3.2 eGeMAPS

eGeMAPS (extended Geneva Minimalistic Acoustic Parameter Set) is, in contrast to ComParE, a reduced feature set, designed in a knowledge-based and hand-selected approach [31]. It has been originally engineered with a special focus on affective speech [118, 119, 120], but it has proven its suitability also for further paralinguistic and voice analysis tasks, such as, e. g., the related task of recognising autism spectrum conditions from speech [111]. It consists of a set of **88 instance-level features**.

eGeMAPS contains all the LLDs listed in the right column of Table 2.1 [30]. The most significant difference to the ComParE set is that *formants* are included (frequencies, energies, and bandwidths from the first three formants), but there is

ComParE	**eGeMAPS**
RMS (energy) ZCR	
Spectral energy 250—650 Hz Spectral energy 1—4 kHz Spectral roll-off points (4) Spectral flux, entropy, & variance Spectral skewness & kurtosis Spectral centroid & slope Spectral harmonicity & sharpness	Spectral flux Spectral slopes within 0–500 Hz & 500–1 500 Hz Alpha ratio Hammarberg index
RASTA auditory band energies (1–26)	
MFCCs (1–14)	MFCCs (1–4)
	Frequency of formants F_1, F_2, F_3 Relative energies of formants F_1, F_2, F_3 Bandwidth of formant F_1
Loudness Modulation loudness [30]	Loudness
F0 Voicing probability	F0 Harmonic differences (H1–H2 & H1–A3) [31]
logHNR Jitter (local & DDP) Shimmer	logHNR Jitter (local) Shimmer
Sum: 65	**Sum: 23**

Table 2.1: List of LLDs included in the ComParE and eGeMAPS acoustic feature sets. The descriptors are grouped according to the sections where they have been introduced together.

only a *limited selection of spectral features* and the first four *MFCCs*. The *voice quality parameters* are more or less the same. A pre-emphasis is not applied either. In accordance with the ComParE set, the **23 LLDs** are extracted with a hop size of 10 ms. A frame length of 60 ms and a Gaussian window are used for *F0*, *harmonic differences*, *jitter*, *shimmer*, and *logHNR*; a frame length of 20 ms and a Hamming window are used for all other LLDs, also for the *formants*. In contrast to the ComParE set, *no deltas* are included.

As functionals, the *arithmetic mean* and the *coefficient of variation* are computed for all 23 LLDs. In addition to that, from *F0* and *loudness*, the 20^{th}, 50^{th}, and 80^{th} *percentiles*, the *range* from the 20^{th} to the 80^{th} percentile, and the *mean* and the *standard deviation* of the *slope of rising and falling segments* in the signal are considered [31]. These, in total, 66 features are computed over only the *voiced* frames, except for *spectral flux*, *MFCCs*, and *loudness*, for which the values from

all frames are taken into account. For *alpha ratio*, *Hammarberg index*, *spectral flux* and the two *spectral slopes*, the *arithmetic means* computed over **only** the *unvoiced* segments are added to the set. Likewise, the *arithmetic mean* and the *coefficient of variation* are added for the *spectral flux* and *MFCCs* from *voiced* frames. The set is completed with six temporal features, describing the durations of voiced and unvoiced segments, and a feature describing the 'equivalent sound level' [31], totalling up to 88 acoustic instance-level features in eGeMAPS.

Both feature sets are provided by the open-source toolkit openSMILE [23, 121, 122], which has been used to compute most of the LLDs and functionals employed in the experiments throughout this thesis. The main characteristic of openSMILE is that it supports both incremental and off-line processing at a very high speed. openSMILE is implemented in C++, without any third-party dependencies for the feature extraction, and runs on multiple platforms, including Windows and Linux. As an example, for a 16 kHz speech signal, *energy*, *MFCCs*, and *PLP* frame-level features together with their *deltas* and *double deltas* are extracted at a *real-time factor* of 0.012 [121][15], i. e., more than 80 times faster than the duration of the corresponding signal.

From the ComParE feature set and eGeMAPS, in some experiments in this thesis, only the LLDs are used—especially as an input for the BoAW features—, whereas in other experiments the original sets with functionals are employed as a baseline, as described in Chapters 6 and 7.

[15]The reported evaluation has been done on a single core of an AMD Phenom 64 bit CPU at 2.2 GHz and 4 GB DDR2 RAM.

3

Bag-of-Audio-Words

In this chapter, the *bag-of-audio-words* (BoAW) method is introduced as an alternative method to transform acoustic LLDs, i. e., sequences of features extracted from short-time frames, into instance-level representations. The BoAW approach is the central topic of this thesis. While the theoretical foundations and related works are discussed in this chapter, the implementation of the method in the toolkit openXBOW and enhancements are presented in Chapter 5; experimental results are presented in Chapters 6 and 7 and discussed in Chapter 8.

In this chapter, first, the *bag-of-words* (BoW) approach in text processing, where the BoAW approach is inspired from, and its common enhancements are introduced in Section 3.1. Then, in Section 3.2, it is explained how the concept can be adapted to numeric features, such as visual features and acoustic LLDs. This section includes a detailed outline of the academic history of BoAW in Subsection 3.2.2. Afterwards, an overview of the BoAW approach is given in Section 3.3 and the most essential steps are formalised, with a focus on the codebook generation (Subsection 3.3.1) and assignment of audio words (Subsection 3.3.2). This section concludes with a characterisation of the properties of the representation (Subsection 3.3.3). Finally, in Section 3.4, relationships of BoAW with similar methods are described.

3.1 Bag-of-Words

The BoW method is a standard approach in *natural language processing* (NLP) to represent texts as a numeric feature vector for ML [123].

3.1.1 Methodology

The fundamental idea is quite simple: based on a certain **dictionary**, the frequencies (numbers of occurrences) of all words from a text document are counted, creating a **histogram**. This histogram is then interpreted as a feature vector having a

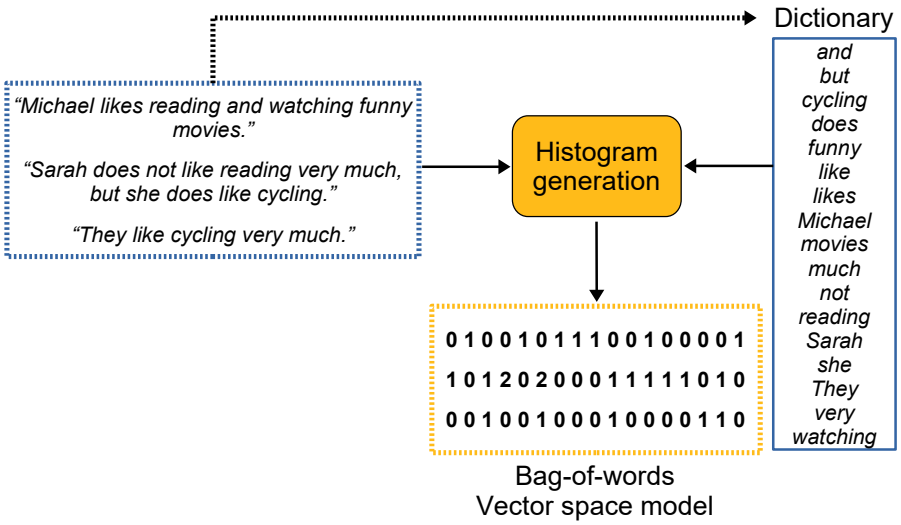

Figure 3.1: An example of the bag-of-words approach. Three sentences are turned into their bag-of-words/vector space model representation. The dictionary is learnt from the given sentences (and displayed in alphabetic order).

fixed length—the size of the dictionary—independent from the length of the text document. The principle is visualised in Figure 3.1. It is apparent that the order of the words does not have any effect on the BoW representation, which is in accordance with findings from information retrieval, stating that word order is of only minor relevance for many recognition tasks in the NLP domain [27].

Strictly speaking, the concept of BoW denominates just an unordered list of the words from a document, while the feature vector representation generated from the BoW is called a **vector space model** (VSM) [124, 125]. In the VSM, the value of each of its elements is not necessarily the raw word count in the document, but it can as well be a boolean (word present / word not present) or a value based on another weighting scheme (see below). As for the purpose of ML, usually only the VSM is of relevance, the terms BoW (or BoAW) and VSM are used equivalently to denominate the feature vector.

The dictionary is usually derived from all words (more generally: terms) forming the *training set* [123], i. e., the subset of a corpus utilised to train the recognition system (see also Chapter 4). Using a larger, possibly generic dictionary would not help to improve the performance as no dependencies could be learnt by the ML algorithm from words with zero-frequency in the training data. Moreover, a context-specific dictionary is smaller in size and therefore, in terms of storage.

Formally, based on a suitable dictionary $D_W = \{d_1, d_2, \ldots, d_{L_D}\}$, the word sequence $S_W = \{w_1, w_2, \ldots, w_{L_S}\}$ is summarised in a BoW feature vector F_{BoW} as

$$F_{\mathrm{BoW},i} = \sum_{j=1}^{L_S} \delta(d_i, w_j), \quad \text{for} \quad i = 1, \ldots, L_D, \tag{3.1}$$

$$\text{with} \quad \delta(d_i, w_j) = 1 \quad \text{if} \quad d_i = w_j \quad \text{and} \quad \delta(d_i, w_j) = 0 \quad \text{otherwise.} \tag{3.2}$$

L_D and L_S are the lengths of the dictionary (and the resulting BoW feature vector) and the sequence, respectively; d_i and w_j are the words of the dictionary and sequence, respectively, with the corresponding index. The word sequence can be any kind of written document, from a short message to a text book.

In other words, a **term frequency histogram** is created from the words present in document S_W, based on a given dictionary D_W. The VSM F_{BoW} is a feature vector, where each single feature describes the *term frequency* of a certain word. In many cases, depending on the type of document, the VSM representation is *sparse* [27], i. e., most elements in the vector are zero.

The BoW approach can, in principle, be employed for any imaginable application of *document classification*, or more generally, *text classification* [126] of all kinds of written documents, such as, e. g.,

- information retrieval in *digital humanities* [127],
- authorship attribution of books [123, 128],
- e-mail spam filtering [129],
- web (content) mining [130],
- search for research papers [131], and
- emotion recognition and sentiment analysis in news headlines [132] or short messages in social networks, such as *Twitter* [133].

3.1.2 History

The BoW concept has been mentioned for the first time yet in 1954 by Harris [134]. The author established *distributionalism* as a linguistic theory, with the central hypothesis ('distributional hypothesis') which suggests: linguistic items, such as words, that are used in similar contexts/distributions also have similar meanings [135]. The theory is also supposed to describe the way how children learn a language and the meaning of words. However, Harris clearly states that "language is not merely a bag of words" [134]. Nevertheless, the basic BoW/VSM approach introduced above can be considered as a feature vector summarising the topic of a certain document. This document must, usually, consist of several words in order to have the BoW provide a meaningful description. Thus, BoW does not directly implement the distributional

hypothesis, though it is related and motivated by the idea that some higher order semantics can be recognised from a representation of a certain linguistic context.

The VSM has been employed in order to perform document similarity ranking, computing the *cosine similarity* of the feature vectors from the documents in a corpus and a query document [125]. Moreover, they have already been studied as an input representation for different kinds of ML approaches for decades [125, 136, 137]. In 1965, Rocchio and Salton introduced their *relevance feedback* method for document retrieval [138], as one part of the SMART[1] Information Retrieval System project, originally started in 1961 [139, 140]. As opposed to keyword-based search strategies, a 'full text retrieval' system has been implemented as (in principle) the full text is used to create a document-specific feature vector [131]. In 1973, Salton and Yang described properties of term frequency assignment schemes given a collection of documents for information retrieval [141]. The topic of approaches for weighting the term frequencies occurring in documents has been a topic of research for many years [131, 142]. However, the term 'bag-of-words' became widely used not before the late 1990s [136].

At this point, it needs to be mentioned that very novel approaches closely derived from distributional theory exist as well. As an approach closely linked to the distributional hypothesis can be considered a VSM of a word represented by the BoW of its context [135]. Most importantly, the WORD2VEC model has been proposed by Mikolov et al. [143, 144]. Based on a neural network architecture, *word embeddings* are trained in a way that they reconstruct the context (i. e., the surrounding words) of a given word. The resulting model, the embedding space, is a continuous vector space of a dimensionality much lower than the size of the underlying dictionary, with the property that similar words, especially synonyms, have a very close distance from each other. In the embedding space, a whole text can be represented, e. g., by the mean of the embeddings of all words occurring in the document [126].

3.1.3 Enhancement

The BoW approach comes with a few *problems*, having an impact on the employed approach for document search, or more specifically, the ML approach that is using the VSM from the BoW as an input. An unclosed list of the most important issues is presented in the following, where some problems can be tackled by the modifications to the BoW approach introduced in the subsections right after this list.

- The relevance of *function words*, such as articles and conjunctions, and of *personal names* is questionable. On the one hand, one might argue that words like 'the' or 'and' can be omitted as they imply no meaning. On the other hand, it has been shown that, e. g., the frequency of function works is a quite reliable feature for the task of authorship attribution [123, 128].

[1] SMART: System for the Mechanical Analysis and Retrieval of Text

- The raw BoW approach differentiates between words in *upper case* and *lower case* [123]. A regular conversion from upper case to lower case (or the other way round) is questionable as capitalisation may have a meaning, e. g., in social media posts [145], in case of person or organisation names [125], or in some languages, such as German[2]. However, a strict conversion might make sense in order to handle, e. g., capitalisation at the beginning of sentences, non-standardised capitalisation, and typos in social media posts.

- Words of the same *word stem* with diverse endings are mapped to different terms in the dictionary (e. g., *likes* vs *like*), leading to a lower robustness and compactness of the feature vector.

- The same goes for the relevance of *punctuation* (dots, commas, etc.), which might not provide any further information for the task at hand, but can as well imply information on authorship or the intensity of social media posts [145].

- *Synonyms* are especially difficult to be handled as they could be mapped to the same entry in the dictionary, but additional knowledge about the mappings or word meaning is required.

- The *word order* or *context* and *grammar* are not taken into account at all, but excerpts such as, e. g., 'I like' and 'I do not like' have opposite meaning, where also the second excerpt [144] accounts for the term 'like' in the dictionary.

- The VSM has a *high dimensionality* and is usually *sparse* [144], where the level of sparseness depends on the lengths of the documents to be classified.

Most of the problems introduced can be resolved or at least attenuated by one or more of the modifications of the BoW approach, as introduced in the following subsections.

3.1.3.1 Stop words, stemming, and external knowledge databases

The problem of the presence of irrelevant *function words* can be tackled by defining a list of **stop words**, i. e., words that are excluded from the dictionary and not considered during processing [123]. The most common criterion to identify this set is based on the *relevance* of the word with respect to the given query or classification task, i. e., whether a word is more likely to appear in particular query classes than in others [146]. Generic lists may work for a variety of use cases, but also automated systematic approaches have been proposed. For example, Wilbur and Sirotkin define a stop word list by selecting terms whose likelihood to contribute to similar feature vectors (w. r. t. cosine similarity) between documents of the same

[2]Consider, e. g., the German words 'Weg' (noun, engl.: *path*) and 'weg' (adverb, engl.: *away*)

classes is below chance level [146]. They found that—targetting a document query system for technological documents—the majority of the occurring words can be included in the stop list and thus be excluded from the dictionary. In fact, Yang and Wilbur claim that task-specific dictionaries, where up to 87 % of the terms are removed, lead to a higher precision in document classification [147].

Furthermore, the exclusion of very rare words can make sense as well. Rare words often do not occur frequently enough to be meaningful for ML and they contribute massively to the large dimensionality of the feature vector. This can be implemented simply by introducing a *minimum term frequency* for dictionary generation [123].

Methods to reduce inflected words to their stem are called **stemming** [123]. A basic stemming step has already been included in the early SMART system since the beginning of the 1960s [139]. A popular algorithm has been introduced by Lovins in 1968 ('Lovins Stemmer') [148]. In 1980, Porter published an algorithm for stemming of the English language, which is still being used nowadays ('Porter Stemmer') [149]. The core of the method is that each word is considered as a sequence of vowels and consonants, where, English suffixes (e. g., '-ed' or '-ation') are either removed or replaced, based on predefined lists in a defined order and based on other conditions, such as the length of vowel-consonant sequences. The method has proven to be capable of increasing the accuracy of information retrieval systems, but it is important to note that the Porter Stemmer is language-specific. Even if the approach itself can be adapted to some other languages, the replacement rules need to be defined and optimised for each particular language.

Advanced methods of stemming are also used, called **lemmatisation**, taking into account also the context and the *part of speech* of the word [150]. Moreover, improved morphological tables are provided in order to reduce words to stems that are not obvious from the letters, such as, e. g., 'better' reduced to 'good'.

Further refinements have been introduced, e. g., Gabrilovich and Markovitch propose a method to augment the BoW with features representing *concepts* from **external knowledge databases** [151], also referred to as *bag-of-concepts* [152]. *Concept* in this context describes the terms in an *ontology* that relates the terms and their properties to each other using relations. The external data resources can be lexical databases such as WordNet [153]. However, it has been found that WordNet ontologies to generate concepts require a lot of manual efforts to be maintained [152] and that the knowledge included about word meaning is limited [154]. It has been shown that bag-of-concepts generated from public encyclopedias such as Wikipedia are more promising [154].

3.1.3.2 Modified term frequency weighting

In general, each bin in the VSM $F_{\mathrm{BoW},i}$ can be any positive function of the corresponding term frequency. As mentioned, in the simplest case, the mapping is the pure term count (word count) $F_{\mathrm{BoW},i}$ for the i^{th} term in the dictionary (see Equa-

tion 3.1). Another option would be just a boolean indicating whether the term is present in the corresponding document or not. However, it has been found that more sophisticated mapping functions usually lead to a better performance of the information retrieval system [142]. Many term frequency weighting schemes have been proposed in the past; the ones most relevant for the presented works are introduced in the following.

Logarithmic term frequency
A common way to compress a range of descriptors is applying the *logarithm* [123]. In order to avoid negative numbers, a bias of 1 is usually added to each term frequency before applying the logarithm. Given the pure word count $F_{\mathrm{BoW},i}$, the VSM with a modified term frequency weight is then

$$F_{\mathrm{BoW,log},i} = \log(F_{\mathrm{BoW},i} + 1), \quad \text{for} \quad i = 1, \ldots, L_D. \tag{3.3}$$

As a strictly monotonous function for all positive numbers, the logarithm is invertible, i. e., no information is lost during conversion, while at the same time, counts of very frequent words (such as, e. g., articles if they are not in the stop list) do not result in a very large weight. Depending on the further processing or ML, such outliers could deteriorate the robustness of the conversion.

Term frequency–inverse document frequency
So far, each term in the dictionary is given the same importance a priori. However, one could argue that terms that are present only in a few instances (documents) may be more discriminative or relevant w. r. t. captioning the content than terms that are present throughout and differ only by their exact count [125]. This is modelled by the **inverse document frequency** (IDF) [123, 141]. The idea was first introduced by Spärck Jones already in 1972 [155], who defines 'specificity' of a term as a function that is inverse to the number of documents where the term occurs. Given a corpus of documents (a collection of word sequences) $\mathcal{C} = \{S_{W,1}, S_{W,2}, \ldots\}$, the IDF of the term with index i from dictionary D_W is defined as

$$\mathrm{IDF}_i = \log \frac{|\mathcal{C}|}{|\{S_W \in \mathcal{C} | d_i \in S_W\}|}, \quad \text{for} \quad i = 1, \ldots, L_D. \tag{3.4}$$

In other words, the IDF of dictionary term d_i is the logarithm of the ratio of the number of documents in the corpus and the number of documents where d_i appears at least once.

For the resulting VSM, the term frequency–inverse document frequency (TF-IDF) is usually employed, combining the term frequency count with the IDF as a product:

$$F_{\mathrm{BoW,TFIDF},i} = F_{\mathrm{BoW},i} \cdot \mathrm{IDF}_i, \quad \text{for} \quad i = 1, \ldots, L_D. \tag{3.5}$$

Intuitively, for the TF-IDF, it is evident that words or terms occurring in only a few documents are considered more specific and receive an above average weight in the model. Correspondingly, also the logarithmic term frequency from Equation 3.3 can be used instead of $F_{\mathrm{BoW},i}$.

Vector normalisation
As a final step (independent of the usage of the previous weighting methods) a **normalisation** of the feature vector can be taken into account [131]. This is relevant especially in cases where the lengths of the documents vary throughout the corpus, but also the influence of other non-linear operations such as the logarithmic or IDF weightings on the norm of the vector is removed. Normalisation is typically done by dividing each of its elements by the *Euclidean norm* (L^2-norm) of the feature vector [125], i. e.,

$$F_{\mathrm{BoW},L^2,i} = \frac{F_{\mathrm{BoW},i}}{\sqrt{\sum_{j=1}^{L_D} F_{\mathrm{BoW},j}^2}}, \quad \text{for} \quad i = 1, \ldots, L_D, \tag{3.6}$$

in case no other weighting is applied. The same applies accordingly to the feature vectors from Equations 3.3 or 3.5. Following Equation 3.6, the resulting feature vector will have an Euclidean length of 1, independent from the length of the input document. The *inner product* (i. e., *dot product* as the vectors are in an Euclidean space) is then equivalent to the *cosine similarity* [131], which provides a robust (length-independent) metric for document similarity. However, in some use cases, a normalisation on the average document length (average number of words) could be preferred due to numerical reasons.

Alternatively, also the L^1-norm can be employed, which appears to be more common in BoW of multimedia data [156, 157, 26] (see Section 3.2). If there has been no other prior weighting of the term frequencies, the output of the L^1-normalisation are the *relative term frequencies* (i. e., the sum of all features is equal to 1).

3.1.3.3 Context-sensitive approaches

So far, except for the mentioned *lemmatisation*, the context of the words is not taken into account by the described methods and enhancements. Nevertheless, numerous approaches have been proposed to improve the model semantically by taking into account the interrelation with the surrounding words in a document. This is somehow challenging the basic principle of BoW, which represents the document as an *unordered* multiset. As an example, Cohen and Singer proposed several popular algorithms taking into account word contexts [158]. However, most of them are not closely related to the BoW paradigm anymore, such as RIPPER, where sets of rules, implemented as conjunctions of occurring terms, are generated using *information gain* as a criterion. Erk and Padó introduced a *structured* VSM or *semantic space*

model [159] in order to augment the BoW model with the meaning of the word in the given context. This is made by combining VSM feature vectors with so-called *preference vectors*, which are encoding, i. a., the subjects and objects of a given phrase. However, significant improvements can be achieved as well by methods that are still highly related and originating from the BoW model. In the following two paragraphs, the context-sensitive approaches most relevant for this thesis are quickly introduced.

N-grams
The concept of *n-grams* is found in many sub-fields of computer science and computational linguistics. N-gram modelling means that sequences of linguistic entities—these can be either *phonemes*, *letter*, *syllables*, or *words*—are represented as overlapping sub-sequences of n items [160]. For the task of document classification, n-grams are often generated on word level. As an example, given the phrase 'they like cycling very much', the extracted *bigrams* ($n = 2$, 2-grams) are 'they like', 'like cycling', 'cycling very', and 'very much'. The *trigrams* ($n = 3$, 3-grams) are 'they like cycling', 'like cycling very', and 'cycling very much'. For word-level n-grams, grams consisting of more than 3 words are usually not taken into account.

Typically, e. g., in the ML toolkit Weka [161], also the shorter grams are used and added to the dictionary. This means that for $n = 3$, all trigrams, bigrams, and unigrams (the standard BoW dictionary terms) are extracted as terms and a VSM of the corresponding term frequencies is generated.

Character n-grams
Instead of using word-level n-grams, building a BoW-like representation from sequences of characters (letters) is more common [125, 160, 162]. Also the word boundaries are usually encoded as a separate character [160]. In the example from the previous paragraph, the *character trigrams* ($n = 3$) would be '_the', 'the', 'hey', 'ey_', 'y_l', '_li', 'lik', and so on (the character '_' encodes the word boundaries in this example). The document representation based on a character-level n-gram dictionary is accordingly referred to as *bag-of-(character)-n-grams*.

3.2 Adaptation for Numerical Data

Inspired and motivated by the success of BoW and VSM, other communities adopted the main idea of the approach, namely describing a document by a histogram of 'words', to their fields. First, the computer vision community introduced *bag-of-visual-words* (BoVW) as global image descriptors and then, the *bag-of-audio-words* (BoAW) approach became established in CA. The main difference between the NLP domain and the audio or visual domains is that NLP data consists of terms from a

limited vocabulary while data in the other two domains are represented by, in principle, continuously-valued signals or continuously-valued features (see Section 2.1). In other words, the BoW approach requires **symbolic** input data whereas audio and video/images are usually represented as **numeric** LLDs or *(local) video/image descriptors* [52], respectively.

This gap is closed by introducing a step of **vector quantisation** (VQ). In VQ, a low-level feature vector is assigned to an 'audio word' or 'visual word' by finding the most similar *template* feature vector from a dictionary of pre-defined templates in the corresponding domain, usually determined using the *Euclidean distance*. The dictionary in the context of BoAW and BoVW is also referred to as **codebook**. A 'word' can be considered as a *vector quantised* low-level feature vector, as given in the codebook.

3.2.1 Bag-of-visual-words

The idea of BoVW got published first—to the best knowledge of the author—by Idris and Panchanathan in 1995 [163, 164]. Even though their early work is only recognised rarely and their approach is not inspired from BoW in text classification, they have first proposed the concept of *VQ* in combination with a *histogram* to capture the content of an image, while the idea of using histograms (e. g., of colours) as image descriptors has been common already before [165]. Although they use the term 'histogram', the frequency of occurrences are not counted and only a boolean is used to indicate whether a template is present or not. In contrast to more recent references, the focus of their work is on *image indexing*, i. e., generating compact descriptors for image search systems, and they quantise the raw pixel information from blocks. Nevertheless, the authors introduce the term 'codebook' for the set of templates for the image blocks and generate it by using a *clustering* method (see Subsection 3.3.1). A similar approach with some adaptations was published by Lu and Teng in 1999 [166].

One widely noticed contribution was presented by Leung and Malik in 2001 [167], focussing on the task of *texture classification*, though they still have not introduced the term 'BoVW'. The authors employ the concept of 'textons', a terminology that actually has been proposed already 20 years earlier by Julesz [168], who established a theory of human texture perception in images. Leung and Malik use a set of filters to compute *appearance vectors* as local image features, describing small patches of a texture. They employ *k-means clustering* on the local features from the training set in order to define a *codebook* of textons. These textons can then be considered the 'words', each appearance vector is assigned to during VQ. As for BoW, the locations of the textons are not taken into account and they are aggregated into a histogram representation. A *chi-square significance test* is employed to measure the similarity between histograms of different images.

Zhu et al. also propose feature-based clustering to create codebooks in 2002 [169], probably working on it in parallel with Leung and Malik [167] without being aware of each others progress. They also use VQ and, amongst others, a histogram representation for the images. Zhu et al. call the visual words 'keyblocks', as opposed to the 'keywords' (words in the dictionary) from BoW. As a novelty, the codebooks are generated at different resolutions and levels of the images.

The term 'BoVW' was first introduced by Sivic and Zisserman in 2003 [170]. The pursued application was the development of a very fast and efficient video search engine, finding given objects. They used continuously-valued *viewpoint invariant image region descriptors* as an input and performed a VQ to map them to 'template' visual words. The dictionary of templates ("vocabulary") is constructed by a *k-means clustering*, though the authors mention other methods that could be employed. In order to save computational efforts, clustering is performed only on a subset of the videos. LLD feature vectors have a dimensionality of 128; between 6 000 and 10 000 cluster centroids are used. The authors say that the clustering process has been performed several times and the set of centroids with the best performance is used. The authors also adopt the ideas of *stop words*, i. e., they exclude (visual) words appearing in (almost) all video frames, and TF-IDF. Cosine similarity can be used to find relevant documents in a database, given a query object to be found in the videos. In 2004, Csurka et al. propose a similar method combined with ML methods for image classification and call it 'bag-of-keypoints' [171]; also the terminologies 'bag-of-visterms' [172] and, more generally, 'bag-of-features' are utilised [173].

BoVW were then established during the following years, e. g., in 2006, it has been applied to *flower classification* [174] and *visual scene classification* [175]. The authors of both publications employ a *nearest neighbour classifier*, which assigns an instance to the class of the training sample closest in terms of a certain distance measure. Dedicated codebooks need to be learnt for each sub-domain using a clustering method. Moreover, some widely noticed extensions have been proposed, such as *spatial pyramid matching* to generate and combine the BoVW in different levels at sub-regions of the images [176]. Also in the video domain, BoVW has established itself, e. g., for the task of *action recognition* [177]. Recent advancements include the usage of local image features learnt by a *deep neural network* (DNN) (see Section 4.4) instead of classical 'hand-crafted' features (analogously to the acoustic low-level features discussed in Section 2.1) [178] and novel codebook generation techniques [179].

3.2.2 A survey on bag-of-audio-words

Early works

The BoAW approach has been employed for the first time—to the best knowledge of the author—by Jonathan Foote in 1997 [180]. Based on MFCCs as low-level features

and a *tree-based* VQ, a histogram is created to represent an audio document for the purpose of similarity search, applied to both general sound and music data. Foote motivates his approach by the success of such representation schemes in the text retrieval domain, but does not introduce the term BoAW or 'audio words' for the quantised frame-level features. Pye showed that a *Gaussian mixture model* (GMM) outperforms the tree-based VQ for the same task [181].

The first multimodal approach was presented by Leopold et al. in 2003 [182]. In this work, the BoW, BoVW, and BoAW approaches are combined—similar to the work presented later in this thesis—and an SVM classifier (see Chapter 4) is trained for the task of topic mining in broadcast data. For the audio domain, *spectral flatness* and *envelope* for a number of frequency bands are used, extracted from frames of 30 ms duration. However, the authors state that the performance of BoAW is much lower than that of BoVW or BoW, except for the topic class of 'commercials'. They hypothesise that this is likely due to a high general sound level of this class.

BoAW becomes popular in MIR

The straight BoAW method has become successful in MIR first, starting in 2005. Vignoli and Pauws [183] propose a BoAW-like system to recognise *music similarity*, based on a number of *timbre*-related features, which are clustered by a *self-organising map* [184]. Histograms ('probability distributions') are computed based on the 256 derived clusters and compared using the *Kullback-Leibler divergence* [185], a measure for the dissimilarity of probability distributions. The authors argue that *self-organising maps* are well-suited for the presented use case, due to their structure-preserving property. Nevertheless, the authors avoid the analogy with VSMs and text retrieval and the terminology of 'BoAW'.

The latter was introduced in the MIR-domain by Riley et al. in 2008 [25]. An *audio fingerprinting* system is proposed based on *Chroma*-features [1]. These are 12-dimensional descriptors of the relative presence of the musical notes of the Western scale, independent from the octave they are played in. The system is able to match also studio and live versions of the same song, even though distortions, tempo variations, and extended guitar solos might be present. In analogy to some of the mentioned BoVW approaches, clusters, i. e., dense regions of feature vectors, are derived via clustering applied to a subset of the training set and VQ is applied to find the nearest correspondence in the audio words space for each Chroma-feature vector in each song. A TF-IDF weighting is used and the generated histograms between songs are then compared in terms of different similarity measures. The authors' findings include that a *k-means clustering* outperforms more complex approaches such as, e. g., *expectation-maximisation* (EM) or *hierarchical agglomerative clustering*. They experiment with assigning not only the closest but several audio words to each feature vector, i. e., the N_A closest ones and find that a number of 3 works best, while it degraded the performance if the corresponding histograms

bins were not incremented by 1, but with a variable increment, decreasing with a larger distance to the cluster centroid. This last result is contrary to findings for BoVW [186] and nevertheless, approaches closely related to BoAW, but with a soft assignment, came up at around the same time.

Soft assignment

Barrington et al. proposed to use 'bags-of-feature-vectors' for audio information retrieval in 2007 [187]. They employed EM-clustering and a GMM of MFCCs instead of the 'hard' VQ executed to generate the 'typical' BoAW representation, and thus end up with a soft encoding. The same approach for the task of music classification was studied later by Wang et al. [188]. The parameters of a GMM (prior probabilities, mean vector, covariance matrix) are trained in an unsupervised manner on a selection of LLDs (MFCCs, spectral, and tonal descriptors) from the training set. The soft assignment is done by computing posterior probabilities of the mixture components and the assignments are averaged over the whole instance (song). The authors find that the soft assignment works better than a hard VQ based on a codebook derived from k-means clustering. They emphasise on the diversity of the templates (GMM components) and also find that considering uniformly distributed prior probabilities (mixture weights) for the GMM outperforms the learnt priors.

Variations of codebook generation and word assignment

Su et al. compare the effectiveness of several BoAW variants and related methods, mainly *sparse coding* and *deep belief networks* codebook learning approaches, in the MIR field [189]. They summarise the works under the concept of 'bag-of-frames', a term which is also used for a different set of methods (see Section 3.4). Sparse coding and deep belief networks do not imply a VQ step and are therefore not considered as representatives of BoAW. Su et al. discuss a two-level bag-structure, in other words, two 'levels of abstraction', where the lower layer is called—in analogy to the linguistic domain—'bag-of-audio-alphabet' and the second layer is the BoAW, taking into account the histogram-like output from the first layer as LLDs. A similar approach had been proposed before by Yeh et al. for the task of music genre classification [190]. Su et al. conclude that two levels improve the recognition abilities in, especially, high-level MIR tasks, such as predominant instrument and genre classification. Moreover, the authors conclude that the TF-IDF weighting scheme and energy/power normalisation are generally beneficial. Nevertheless, McFee et al. found that TF-IDF weighting, in contrary, degrades the performance of their music similarity-driven recommendation system, based on MFCCs [191].

Yeh and Yang compared different codebook generation and LLD-encoding techniques with each other for the task of *music genre classification* [24]. The authors found that there is no meaningful difference in performance between a *random sampling* and *k-means clustering*, but both are outperformed by the *online dictionary*

learning method. For the latter, they received some evidence that a *supervised* dictionary learning, i. e., balancing the number of *codewords* for each class, improves the performance. Moreover, they observed that mel-spectrogram LLDs are superior to MFCCs for the given task.

Sundaram and Narayanan extended the concept and converted the term frequency (histogram) feature space to a latent space by *singular value decomposition* in 2008 [192]. The feature vectors in the latent space can be used for *query-by-example* audio retrieval. Also ML-based audio retrieval has been proposed at the same time, e. g., by Chechik et al. ('bag-of-acoustic-words') using an SVM [193]. Both approaches employ MFCCs, while the first one by Sundaram and Narayanan employed an additional larger set of acoustic LLDs.

Using the magnitude STFT, transformed with a logarithmic frequency axis, as LLDs in a BoAW approach for the application to measure music similarity was proposed by Seyerlehner et al. [194]. They also suggested to use a multi-level clustering, where k-means is performed first on the instance-level and then on the centroids from all instances, which also reduces the computational complexity.

Marques et al. experimented with BoAW models for the task of *music genre classification* [195]. In their work, a codebook is constructed for low-level feature vectors of MFCCs, spectral descriptors, and ZCR. Histogram-based genre models are then learnt with different ML algorithms. They find that a *random sampling* instead of clustering is sufficient to generate a representative codebook of audio words, even if the sampling is done only from the feature vectors of one class (one music genre). The authors emphasise on the fact that the collection or distribution of the LLDs carries more information w. r. t. the task than the individual LLDs. Furthermore, they find that a *Markov* model, using the *sequence* of the audio words as an input, outperforms models learning from the histogram, i. e., the BoAW. This provides some evidence that the order of the LLDs is actually relevant, at least for some audio recognition tasks.

Learning across modalities

Some works have been published using the audio modality of videos to learn video categories [196] or detect video copies [197]. Zeng et al. [196] use a large number of LLDs and compare the performance of three different classifiers, where SVM performs best, outperforming a nearest neighbour classifier. Liu et al. [197] employ MFCCs and RASTA-PLP as LLDs and a non-ML detector. They introduce a technique called 'coherency vocabulary' to speed-up the VQ process by generating a different codebook for each low-level feature. Moreover, the authors claim that this variant improves also the robustness of BoAW. A similar result of improved performance using multiple codebooks instead of only one for the task of music classification was found by Fu et al. [198].

Multimedia event detection and n-grams
Pancoast and Akbacak use the audio component of videos for the task of *multimedia event detection* based on BoAW and an SVM [26]. In their works, they employ MFCCs 1–12, logarithmic energy, and corresponding deltas as LLDs to form a 39-dimensional frame-level feature vector. K-means clustering is used throughout the experiments to generate codebooks of variable sizes. They find that a codebook size of 500 or 1 000 is more suitable than one of 2 000. Concerning the SVM, the authors find that a *histogram intersection kernel* outperforms the standard *linear kernel* (see Chapter 4) and that a L^1-normalisation is not required. They conclude that the usefulness of L^1-normalisation depends on the given task, on the type of audio word assignment, and if the length of the video is useful as a feature itself [199]. Pancoast and Akbacak extend their approach with *audio word n-grams*, by forming codewords with n-tuples of audio words (cluster centroids) [200]. As the number of codewords increases drastically by doing so, they introduce a term selection method based on a minimum term frequency of the tuples. They find that, even though the performance is generally improved by *audio word n-grams*—the original 'unigram' codewords need to be kept as well, adding n-grams with $n > 2$ does not improve the performance any further. It has been shown that 2-grams ('bigrams') in most of the cases simply consist of a doubled 'unigram'. The same authors also proposed two methods to create a histogram with a *soft* audio word assignment instead of the default increment by one [199]. In contrast to the findings by Riley et al. [25], they found evidence that a *Gaussian encoding*, i. e., an increment depending on the distance between LLD and codeword, can actually be beneficial. Nevertheless, a *soft encoding*, where the BoAW feature space is doubled and, depending on the distance, the histogram bin in either of both halves is incremented by the difference between the distance and a threshold, outperforms both a hard and a Gaussian encoding.

As it is already evident from this overview, BoAW offers many variants and extensions, and requires input features, parameters, and hyperparameters to be optimised. This is also pointed out by Rawat et al. [201], who conducted experiments on the same task of *multimedia event detection*. As mentioned before, they found as well that a random sampling to generate the codebook performs as good as k-means clustering. Their reasoning is that using clustering, many audio words are compressed into dense regions of the feature space, which are not informative w. r. t. the classification task. Based on only MFCCs 1–20 and deltas, the classification performance gets better with an increasing codebook size, up to a size of 16 000. Nevertheless, the authors find that using the dimensionality reduction methods *latent Dirichlet allocation* and *agglomerative clustering*, the codebook size can be reduced drastically, without loss in accuracy. The performance can be further improved by fusing BoAW generated from the mentioned MFCCs with BoAW generated from temporally stacked MFCCs and large-scale acoustic features with temporal pooling (similar to the ComParE feature set, see Chapter 2). Finally, the authors show that a *soft encoding* technique by selecting the N_A closest audio

words and increment the histogram counters by the inverse of their rank reduces the error rate by up to 5 % for a large codebook size of 16 000 codewords, while the improvement is lower for smaller codebook sizes.

Acoustic event detection

Plinge, Grzeszick, and Fink made research on BoAW applied to *acoustic event detection* (AED) [202], a task sharing some similarities with *multimedia event detection* and where both, the class and the time stamp of acoustic events (such as, e. g., 'laughter' or 'phone ringing'), need to be recognised in an audio file of a duration much longer than the event. Thus, BoAW-based histograms are generated w. r. t. a sliding window of 0.6 s, which is shifted over the frames. The authors use the EM-algorithm for clustering, obtaining a codebook of Gaussian means and variances, instead of only means (centroids), similar to the mentioned approach by Barrington et al. [187]. However, the codebook is not trained on the whole training data, but a separate codebook is trained for each class and the class-specific codebooks are then fused. The codeword assignment process is then a *soft* one, where each bin in the VSM is incremented by the likelihoods of the corresponding Gaussian. They call their approach *bag-of-features*, or *bag-of-super-features*, as the codebooks are created in a *supervised* way, i. e., taking the known class labels of the training data into account. Nevertheless, it can be considered as a representative of the BoAW method. Besides, inspired by the 'spatial pyramid' from the visual domain [176], a 'temporal pyramid' scheme is introduced, where two (or more) sub-histograms are created, one for each tile within the analysis window; in addition, one histogram with a maximum-pooling of all sub-histograms is concatenated. As LLDs, they use a concatenation of MFCCs and cepstral coefficients that are computed in a way very similar to MFCCs, but using a *gammatone* scale instead of a *mel* scale. A *maximum likelihood* classifier is trained, where one explicit class for the background noise is trained as well.

The authors find that both the supervised codebook training and the temporal pyramid improve the results and that 11 clusters per class (121 for 11 classes) are optimum. In later work, it is shown that a 'temporal augmentation' instead of the temporal pyramid gives slightly better results [203]. Here, the low-level feature vector is augmented with a discrete integer time index, depending on the position within the window instance, which is subdivided into tiles, each carrying the same index. However, the superiority of one over the other approach is only low and depending on the number of tiles [204]. The authors claim that *bag-of-features* methods produce state-of-the-art results on *acoustic event detection* benchmark datasets, but have a comparably low computational cost [204].

Lim et al. propose to use a combination of *local binary pattern* and *histogram of oriented gradient* as LLDs for AED [205]. These local descriptors originate from the image classification domain and can be extracted from the spectrograms. It

is shown that BoAW based on these features outperform MFCC-based BoAW for outdoor environments and low SNRs. Image descriptors applied to spectrograms as input for BoAW were proven to be effective also for *music genre classification* [206].

Computational paralinguistics
Mangin et al. introduce BoAW for the recognition of speech invariants in an incremental framework [207]. They call it *bag-of-features*, in order not to confuse it with the bag-of-*linguistic*-words that are both the output and potential intermediate representations of an ASR model. Pokorny et al. introduced BoAW for emotion recognition in speech in 2015 [208]. They used MFCCs, energy, F0, HNR, ZCR, and the deltas of these LLDs and applied some functionals to them over a window of 200 ms. The resulting feature vector of a dimensionality of 384—the features from ComParE 2009 [81]—is then split into several, equally sized, sub-vectors. The (sub-)codebook generation and VQ are then first executed only on the features from each sub-vector ('split vector quantisation'). Then, the resulting feature vector of word indexes is subject to another VQ, based on a dedicated codebook.

Crossmodal BoW
Crossmodal (multimodal) bag-of-words have been employed for *multimedia event detection* [209] and *depression monitoring* [210]. As acoustic LLDs, also here, MFCCs are used throughout, while for the depression task, also the prosodic features loudness, intensity, and F0 have proven suitable. In both works, k-means clustering was employed to generate the codebooks, which were learnt separately for each modality. A crossmodal system using the idea of two cascaded layers of histograms [189, 190] has been proposed by Bhatia et al., where the second layer is used to fuse the first-level BoW representations from each single modality [211].

Finally, besides text, video, and audio domains, the BoW concept has also established in other fields. For example, Baydogan et al. have adopted the idea for time series classification and shown that a *bag-of-features* is able to handle time warping to match time series of different durations [212].

Summary
In summary, it has been shown that the BoAW method has established in many sub-fields of CA. It is striking that authors refer to the—more or less—same idea and approach but naming their algorithms variously as 'bag-of-feature-vectors', 'bag-of-features', 'bag-of-acoustic-words', or 'bag-of-frames', while the latter is also used for substantially different methods (see Section 3.4). Even though, based on the review of the given literature, the term 'bag-of-audio-words' is used rather for systems with a hard VQ and 'bag-of-features' is used rather for systems with EM-based clustering and soft VQ, the cited authors do not seem to systematically

differentiate between them, but seem to use the terminology they prefer personally.

The following findings can be drawn based on the literature from this section:

- Many different **LLDs** have been proposed to be used within a BoAW framework, but MFCCs are the only ones used in (almost) all approaches, even for non-speech/non-music tasks [204]. Nevertheless, also spectrogram-based features seem to be promising [205], fusion of different LLD-types usually improves the performance [204], and for musical tasks depending on keys and notes, Chroma features are required [25].
- The **size** of the **codebooks** (number of audio words or cluster centroids) seems to be a hyperparameter of the model that needs to be optimised for each scenario (task, data, and LLDs).
- There is also some evidence that using **multiple codebooks**, fused at some stages of the approach, and **supervised codebook** learning improve the results by trend.

However, there are still many questions partially or completely unclear:

- There is no final conclusion about the optimal **codebook generation** procedure. Some authors observe no improvement of EM over k-means [25], while others do [188]. In several works, a *random sampling* of templates was shown to be superior to, or at least competitive with, k-means clustering [24, 195, 201].
- A **soft encoding** has proven to be beneficial in some works [187, 199, 201], but seemed to be detrimental in others [25].
- There is also no common finding concerning the **term frequency weighting**, e. g., on the general usefulness of TF-IDF ([189] vs [191]). The same applies for the histogram **normalisation**.
- Concerning the **classifier**, which makes the decision based on the BoAW representation, there is also no distinct insight on the optimum choice. As it is a general experience in ML, there is no unique algorithm that always works best. Even considering only SVM, it is not clear which kernel type is best ([26] vs [190]).

It is obvious that findings vary fundamentally, depending on the task and also on the corpus the research is based on, the codebook generation, the weighting scheme, etc. All this calls for an open-source **toolkit** to standardise research on Bo(A)W, letting researchers optimise all hyperparameters in a **reproducible** way and further increase the insights into the method.

3.3 Overview of the Bag-of-Audio-Words Approach

In the this section, just the basic BoAW approach is defined. All enhancements, options, and details are further explained in Chapter 5 on the toolkit.

As pointed out earlier, the general idea is quite simple and illustrated in Figure 3.2. The sequence of LLDs is often normalised first. **Normalisation** of the co-domain (ranges of values) of descriptors is especially required if different LLD types, e. g., MFCCs and F0, with typically different orders of magnitudes are fused on LLD-level. Otherwise, the distance measures used in the VQ step would suffer from the higher weight of the expected larger differences of, e. g., F0 compared to an MFCC. Normalisation can be done using the *z-score* [203, 213], also referred to as *standard score*. Therefore, this type of normalisation is also called *standardisation*. Given an LLD vector D_j, the z-score normalised version is computed as

$$D_{\mathrm{z},j} = \frac{D_j - \bar{D}}{\tilde{D}}, \tag{3.7}$$

with the *sample mean* $\bar{D}$ and *sample standard deviation* $\tilde{D}$ as parameters. In ML practice, these parameters are estimated from the LLDs in the training set $\mathcal{L}$ only (see also Chapter 4) as

$$\bar{D} = \frac{1}{|D \in \mathcal{L}|} \sum_{D_j \in \mathcal{L}} D_j \tag{3.8}$$

$$\tilde{D} = \sqrt{\frac{1}{|D \in \mathcal{L}| - 1} \sum_{D_j \in \mathcal{L}} (D_j - \bar{D})^2} \tag{3.9}$$

using the corrected estimator for the standard deviation, with the *number of frames* in the training set $|D \in \mathcal{L}|$. The dashed lines in Figure 3.2 indicate that the corresponding parameters are estimated only from the training set and applied to all other instances processed.

The same is valid for the **codebook generation** which is typically done by either a clustering or just a **random sampling** from the (normalised) LLDs in the training set. This is more elaborated on in the following subsection. With a codebook given, the **assignment** of the (normalised) LLDs is done, usually in terms of a simple VQ (see Subsection 3.3.2). From the output of the assignment step, a **histogram** or—depending on the type of assignment—a probability distribution vector can be generated for each data instance. This vector is then object to a **weighting** scheme and/or a **histogram normalisation** method, exactly as pointed out already in Section 3.1 on BoW. Also here, parameters for, e. g., TF-IDF are estimated from the training set only. The output from all these processing steps is then a set of

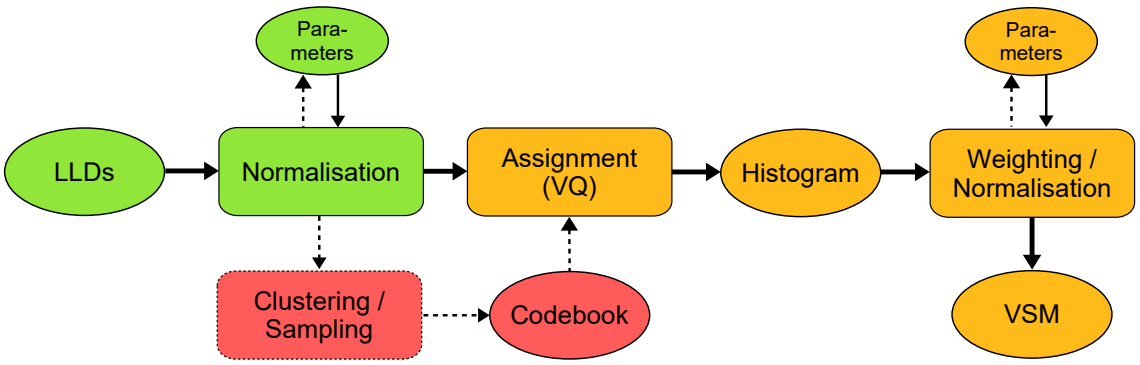

Figure 3.2: Overview of the BoAW approach. The dashed lines represent data flows that are only present for the estimation of model parameters during training. The normalisation and weighting blocks are optional.

instance-level feature vectors in a **VSM**, which can be used as an input for ML (see Chapter 4).

The following subsections will formalise the codebook generation and audio word assignment procedures, and specify some properties of the BoAW representation.

3.3.1 Codebook generation

As mentioned previously, the codebook used for BoAW is typically generated via a clustering approach or—even simpler—via random sampling of the LLDs in the training set. **Clustering**, in general, is a representative of **unsupervised learning**, i. e., a type of ML algorithms that do not employ any class label of the data. The goal of clustering is to identify *groups* of data samples with similar properties, usually measured in terms of a certain *distance metric* [123]. The output of the process is then a set of **clusters**, usually defined given the cluster **centroids**, i. e., the means of the clusters, and—depending on the clustering approach—further quantities. The clustering can be performed in a **hard** way, i. e., a data instance is either part of a certain cluster or not, or in a **soft** way, i. e., a data instance is part of all clusters, with a given ratio for each cluster.

Different approaches for clustering exist and they are usually implemented as an *iterative* optimisation algorithm. The most fundamental hyperparameter is always the **number of clusters**, which has to be manually defined for all common approaches. Obviously, the number of clusters is then exactly the number of **codewords** or templates used for the BoAW.

Probably the two most common clustering approaches are **k-means** clustering and **EM** clustering. While the first produces a set of means (centroids) only, the latter produces a set of means and covariances, implying a soft clustering. However, also k-means can be interpreted as a special case of **EM** clustering. As most of the

experiments in this thesis are based on codebooks generated by k-means clustering, this will be the focus in the remainder of this subsection.

It must be mentioned that, even though clustering itself is an unsupervised method, the codebook generation process can be performed in a **supervised** way [24, 202], by performing clustering (or random sampling) separately on the features from each class and then concatenating the class-specific codewords.

Many further clustering types exist, which might be suitable in the given context, but they are not discussed here. For example, Passalis and Tefas found that, for BoVW, *spectral clustering* and a corresponding assignment outperforms k-means clustering [213].

3.3.1.1 K-means clustering

The idea of k-means was presented first by Steinhaus in 1956 [214]. Given the set of LLD vectors D_j[3] with $j \in \mathcal{L}$, the following term is **minimised**, to derive the k cluster centroids b_i of codebook $\mathcal{B} = \{b_i\}_{i=1}^{L_B}$:

$$\sum_{i=1}^{k} \sum_{D_j \mapsto \mathcal{B}_i} ||D_j - b_i||_2^2, \tag{3.10}$$

where $||D_j - b_i||_2^2$ is the squared Euclidean distance between LLD and codeword vectors and $D_j \mapsto \mathcal{B}_i$ is the set of LLDs for which the i$^{\text{th}}$ centroid is the closest one. This means that the goal is to find a set of centroids for which the sum of the squared distances between LLD samples and centroids is minimum, in other words, the *variance* is minimised. Thus, k-means is also a representative of the *least squares* methods.

As a direct solution of the minimisation problem is not efficient (*NP-hard*), different optimisation algorithms exist, where **Lloyd's algorithm** is probably the most popular one. It has been published originally not to solve the k-means problem, but with the focus on an optimised quantisation scheme for *pulse-code modulation*, a standard digitisation method as described in the introduction of this thesis [215].

To perform Lloyd's algorithm, after selecting k, the centroids b_i must first be **initialised**, by (randomly) choosing k samples from the LLDs in the training set. The sampling from the LLDs must be performed *without replacement*, i. e., the same LLD cannot be drawn more than once. Then, the iterative optimisation starts, consisting of the following two steps, related to *EM*:

1. **Expectation:** Each sample (LLD) D_j in $\mathcal{L}$ is assigned to the cluster $\mathcal{B}_i$ with the closest centroid b_i w. r. t. the Euclidean distance:

$$\mathcal{B}_i \leftarrow \{D_j : ||b_i - D_j||_2^2 \leq ||b_i^* - D_j||_2^2\}, \quad i^* = 1, \ldots, k \tag{3.11}$$

[3] Usually, here and in the following, the z-score normalised version of the LLDs is chosen, i. e., $D_{\text{z},j}$, but for simplicity, the subscript z is dropped.

This ensures that the cluster assignment with the lowest variance is selected.

2. **Maximisation:** The centroids are updated, using the mean of all samples currently assigned to the corresponding cluster:

$$b_i \leftarrow \frac{1}{|D_j \mapsto \mathcal{B}_i|} \sum_{D_j \mapsto \mathcal{B}_i} D_j \tag{3.12}$$

These two steps are repeated until the assignment in the expectation step has not changed from the last iteration. It is important to note that Lloyd's algorithm **does not ensure that the global minimum** of the term 3.10 is reached and that the result depends on the chosen initialisation. In practice, a maximum number of iterations is usually defined to reduce computational efforts as the changes of the centroids are usually decreasing over time.

3.3.1.2 K-means++ clustering

Enhancements of the algorithm have been proposed, most importantly, **k-means++** [216], which is mainly speeding-up the convergence of the algorithm. K-means++ is only modifying the *initialisation* step by selecting the initial centroids in a way that their mutual distance is maximised and thus, the centroids are distributed in a better way over the feature space. The first centroid b_1 is selected randomly. To select each further centroid, the squared Euclidean distance to the *closest* centroid (from the already selected ones) is computed first for each sample $D_j \in \mathcal{L}$. Then, the *probability* of choosing a sample as a further centroid is proportional to the squared Euclidean distance. After the initial centroids have been fixed, the optimisation process is the same as with standard k-means.

3.3.1.3 Random sampling

Random sampling [195, 24, 201] can be interpreted as performing just the initialisation step of k-means (or k-means++) clustering *without* performing the optimisation steps (expectations & maximisation) to update the centroids.

3.3.2 Audio word assignment

The audio word assignment is the central step of the BoAW approach, sometimes also referred to as the **encoding** step, where the LLDs are mapped to the codewords. The available methods for the assignment are closely connected to the type of codebook generation. If k-means clustering or random sampling have been employed, a **hard** VQ is the default method [26], but a *soft encoding* can be introduced by exploiting the distance between LLD and codeword [25, 199]. In contrast, when EM-clustering has been used, a **soft** VQ based on the likelihoods of the Gaussian distributions is

straightforward [202]. For codebooks constructed with techniques based on other assumptions, e. g., *sparse coding*, different encoding methods might be required [190].

As the variants of VQ based on k-means clustering are the ones most relevant for the remainder of this thesis, the assignment step is defined only for this case. With the LLDs from one audio instance $\mathcal{I}$ (entity to be classified), the BoAW feature vector is given by

$$F_{\text{BoAW},i} = \sum_{D_j \in \mathcal{I}} a(b_i, D_j), \quad \text{for} \quad i = 1, \ldots, L_B, \tag{3.13}$$

$$\text{with} \quad a(b_i, D_j) = 1 \quad \text{if} \quad ||b_i - D_j||_2^2 \leq ||b_i^* - D_j||_2^2, \quad i^* = 1, \ldots, L_B \tag{3.14}$$

$$\text{i. e.,} \quad \text{if} \quad i = \arg\min_{i^*} ||b_i^* - D_j||_2^2, \quad i^* = 1, \ldots, L_B \tag{3.15}$$

$$\text{and} \quad a(b_i, D_j) = 0 \quad \text{otherwise.} \tag{3.16}$$

$a(b_i, D_j)$ is the assignment function and $L_B = |\mathcal{B}|$ the length of the codebook. A soft variant can be achieved by modifying the assignment function with a *Gaussian encoding* [25, 199]:

$$a(b_i, D_j) = e^{\frac{-||b_i - D_j||_2^2}{2\sigma^2}} \quad \text{if} \quad ||b_i - D_j||_2^2 \leq ||b_i^* - D_j||_2^2, \quad i^* = 1, \ldots, L_B \tag{3.17}$$

$$\text{and} \quad a(b_i, D_j) = 0 \quad \text{otherwise,} \tag{3.18}$$

with a hyperparameter σ to be chosen manually. In case of **multiple assignments** per LLD, an assignment $a(b_i, D_j) > 0$ in Equations 3.14 and 3.17 is made if $||b_i - D_j||_2^2$ is less or equal to the N_A smallest distances $||b_i^* - D_j||_2^2$.

3.3.3 Properties

The BoAW representation exhibits some commonalities with the BoW-based VSM, where it is inspired from. Except for some special enhancements proposed, a BoAW feature vector is invariant w. r. t. the **order** of the LLDs in the instance, as a BoW does not take into account the order of the words. Moreover, a fixed length feature vector is constructed for each instance, a property that is shared with the *functionals* representation (see Section 2.2). As mentioned before, exactly the same types of term frequency or TF-IDF **weighting**, and **normalisation** techniques can be employed.

Some further correspondences exist, most importantly, the VQ step can be considered as an adoption of **stemming** [25], given that different but similar LLDs—assuming that they have the same 'meaning'—are mapped to the same codeword. VQ can also be imagined as a step for noise removal, however, this largely depends on the type and energy of the disturbances. Concerning **noise robustness**, it is supposed that low levels of noise can be overcome by hard VQ, so that only large

levels of noise remain problematic and hard VQ is not suitable [217]. In such cases, a soft assignment might be preferable. Nevertheless, in the best knowledge of the author, no systematic evaluation of how (low-level) noise impacts on the VQ and audio word assignment exists.

The stop words typically used in BoW do not have a direct correspondence with any of the BoAW variants introduced before. Nevertheless, one can imagine removing 'common' audio words from the dictionary or defining *out-of-codebook* audio words, i. e., not assigning any audio word if the distance of an LLD is above a defined threshold. Besides, another idea to adopt 'stopping' is to remove LLDs from the input where the short-time *energy* of the signal is below a threshold.

The **sparseness** typically found in BoW-VSMs is not always present in BoAW feature vectors. To be specific, the sparseness depends on certain hyperparameters, namely the codebook size and the number of assignments. The larger the codebook size and the smaller the number of assignments, the larger is the level of sparsity. Moreover, a soft VQ in combination with EM-clustering avoids sparseness completely. Generally, it is evident that the **codebook size** is a very important parameter of the approach. Choosing a low number of codewords increases the *generality* of the representation, while a large number improves the *selectivity* [26]. Thus, as depending on the data, the LLDs, and the task, optimisation of the codebook size is imperative.

In contrast to a representation based on *functionals*, the BoAW method does **not smooth out outliers** of LLDs—**no temporal pooling** is done [24]. This means that each LLD, in principle, has the same effect on the final feature vector. A BoAW-based VSM captures the **global distribution** of an audio instance, in other words, the "long-range characteristics" [205] or long-term statistics of the LLDs.

One major **benefit** of BoAW is that the VSM is **scalable** by modifying the number of codewords (and/or number of assignments). This can also be done in an automated process, analogously to the tuning of hyperparameters of a ML approach (see Chapter 4). In contrast, when using *functionals*, the scaling of the feature vector is more complicated as suitable statistics need to be identified and implemented.

3.4 Relationships with Other Techniques

The BoAW method (just as the BoVW method), as introduced, have some commonalities with other popular approaches used in CA (or computer vision, respectively), even though the resulting features and classifiers have only little in common.

For example, Perronnin and Dance describe *Fisher kernels* as an extension to the BoVW model [218]. Similar to the mentioned approach by Barrington et al. [187], a GMM is learnt, modelling the general distribution of the low-level features. In contrast, however, not the (soft) mixture weights of the GMM components are used, but the *Fisher kernels* are computed, describing the derivatives of the 0^{th}- to 2^{nd}-

order statistics of the GMM. Thus, the distribution (posterior probabilities) of the components is not (directly) part of the computed feature vector. *Fisher kernels* have been successfully applied to CA tasks as well [219].

Furthermore, more or less similar approaches to audio recognition have established that may not be confused with the BoAW method investigated in this thesis. Popular methods have been employed called *bag-of-frames*, as proposed, e. g., by West and Cox [220] and by Schuller and Rigoll [221]. In these works, a classifier is trained on local features, i. e., LLDs, and the classification decision on instance level is done by either a majority voting over the frame-level decisions or a hard decision, returning a 'positive' output if at least one frame belongs to the 'positive' class. In contrast to BoAW, the overall distribution is not directly taken into account when training the model—a property which can be advantageous if the target, such as, e. g., an emotional state, is not present throughout the whole chunk to be analysed but only in either the majority [220] or only very few frames [221].

Another method referred to as *bag-of-frames* has been used by Aucouturier et al. [222] and Lagrange et al. [223]. The parameters of a GMM modelling the LLD distribution are learnt for each instance and their distributions are compared. In contrast, for a BoAW model, a codebook is trained for the general distribution of an audio corpus and only the *mixture weights* (speaking in the terminology of the GMM, i. e., histogram frequencies) for each instance are taken into account, not the distribution of the feature space. Nevertheless, the instance-level distribution of LLDs is modelled in a certain way. The authors conclude that their approach is—but only to a certain extent—suitable for classification of *urban soundscapes*, but not for classification of music.

Finally, it must be noted that some authors employing BoAW in their work use the term *bag-of-frames* as well [188, 189, 190, 194, 195]. This is just to sensitise readers to pay attention on the actual methods when performing literature research on this topic, as the terminology varies.

4

Machine Learning

Machine learning (ML) [224] is a (highly topical) discipline in computer science, concerning itself with *learning from examples*. *Pattern recognition* is a term often used as a synonym for ML, originating from the engineering domain [224]. In contrast, *artificial intelligence* (AI) is a superordinate concept of principles to teach machines adopting tasks humans can accomplish (weak AI [225]) or even behaving like equipped with a mind (strong AI). The crucial point of (statistical) ML, to distinguish it from other sub-fields of AI, is that in ML, all learnt knowledge originates from a given set of data. In other words, the decisions made by an ML model (e. g., classifying a recorded speaker as *happy*) are not programmed explicitly, but based on **generalisations** from training samples (i. e., recordings from different subjects speaking in a *happy* and further moods).

While *unsupervised learning*, more specifically *clustering*, has been introduced in Chapter 3 for BoAW codebook generation, this chapter is dedicated to **supervised** learning, i. e., learning a mapping between an input (usually, a feature vector) and a target label (e. g., a class). For this, a fundamental requirement in supervised ML is the availability of a *ground truth* annotation of labels, to both *train* and *evaluate* a model. Referring to the example of the emotions in speech, unsupervised learning might, under the assumption that suitable features are used, be able to identify that different speech recordings belong to different categories of emotions, but only supervised learning provides tools to derive a model (classifier) to map from the features to the label in a robust way, even if *redundant* or *irrelevant* features are fed into the classifier.

In the first section, an overview of the fundamental concepts and the generic workflow in supervised ML are given. Then, in Section 4.2, the most important preprocessing steps are quickly introduced. Afterwards, the two ML approaches most relevant for this thesis are explained: *support vector machine* and *neural networks*, in Sections 4.3 and 4.4, respectively. Finally in Section 4.5, the principles and metrics that are used for the evaluation of ML models are introduced, with a focus on the metrics used in the experiments presented in this thesis.

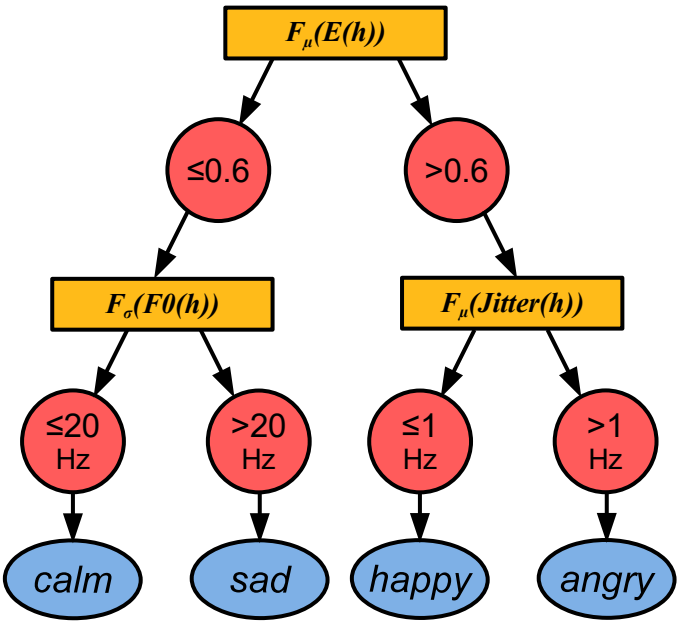

Figure 4.1: A visualisation of a decision tree classifier for emotion recognition in speech, based on three functionals of acoustic LLDs (mean energy, standard deviation of F0, and mean jitter).

4.1 Overview of Supervised Machine Learning

A very simple example of an ML-based model is a **decision tree** classifier as shown in Figure 4.1. A decision tree is easy to visualise in a tree graph. The given example shows a model to predict *emotion* from a speech signal for which three acoustic features (LLD with a specific functional) have been extracted: the *mean energy* $F_\mu(E(h))$, the *standard deviation* of *F0* $F_\sigma(F0(h))$, and the *mean jitter* $F_\mu(Jitter(h))$ (see Chapter 2). The model evaluates $F_\mu(E(h))$ first and based on a given threshold of 0.6, either the left or the right branch is taken. Then, according to the tree, either $F_\sigma(F0(h))$ or $F_\sigma(Jitter(h))$ is evaluated in the second step. Then, as there are no further nodes in this example and a leaf of the tree is reached in any case, the model predicts the corresponding emotional label as the one most likely [1, 123].

It must be mentioned that this is just an example of a model given to illustrate how emotions can be modelled based on acoustic features. Decision trees in practice are much more complex, with more levels, branches, and thresholds, where certain features can be queried several times. The point is that techniques of supervised ML construct and optimise such model automatically, given just the acoustic *features* from the instances of a training set and the corresponding *labels*. The learning algorithm itself works without any theoretical background of the task or prior knowledge of the features. It exploits just statistical dependencies (between features and la-

bels) to generate a model that is able to make decisions based on the values of the feature vector. In the case of decision trees, the learning algorithm to make decisions about the pivotal features (and corresponding thresholds) at each node is based on *information theory* [1].

Concerning decision trees, the method is not further amplified here as it is not competitive with other approaches used in CA. Nevertheless, following the concept of *random forests*, the robustness of decision trees can be improved by a *fusion* of several trees, trained on randomised subsets of the features [123].

Another very simple example of supervised ML is the already mentioned *nearest neighbour*, or more generally, the *k-nearest neighbour* classifier[1]. The model is, in principle, the whole training set and given a new instance to be classified, the class of the training sample with the lowest (e. g., Euclidean) distance is predicted [123, 224]. In case a k larger than 1 is chosen, the *majority* of the classes of the k training instances with the lowest distance is predicted. As no training phase is required, *k-nearest neighbour* is a representative of a 'lazy learner'. However, common ML approaches require an explicit (and often computationally expensive) training phase.

The common workflow applied to create a supervised ML model is illustrated in Figure 4.2. As mentioned, a general concept is to subdivide the workflow into a training phase and a test phase. In the **training** phase, the parameters of the chosen model are optimised exploiting only a subset of the available data, the **training set**. After the extraction of features from the training instances, as detailed in the previous two chapters, a supervised ML algorithm, such as a *decision tree*, *support vector machine*, or *neural network* learning algorithm generates a **model**, optimising a function to map from the features to the given labels (targets). The type of function may differ fundamentally, based on the chosen approach. In the **evaluation** phase, or testing phase, the predictions, i. e., the model outputs for the **test set**, are obtained. Exactly the same feature types as for training are extracted from the instances of the test set and fed into the model. The ground truth labels are not required to obtain the predictions, but are then compared to the model outputs (predictions). For the evaluation, different metrics are in use (see Section 4.5).

It is of fundamental importance to split the data before model training and keep a test partition aside. Evaluation on the training set itself cannot lead to a reliable estimate of the performance as each ML algorithm tends to **overfitting**, i. e., tends to adapt its model parameters to the training instances too much, losing the ability to generalise. Overfitting occurs with any supervised ML algorithm. Thus, suitable measures have to be taken to avoid or reduce it, besides a proper evaluation protocol.

In most current supervised ML approaches however, it is not sufficient to split the corpus into only two disjunct partitions, as these approaches rely on one or more **hyperparameters** that need to be optimised during model engineering. Hyperparameters are options to configure the training process of the learning algorithm,

[1]Not to be confused with the unsupervised *k-means* approach.

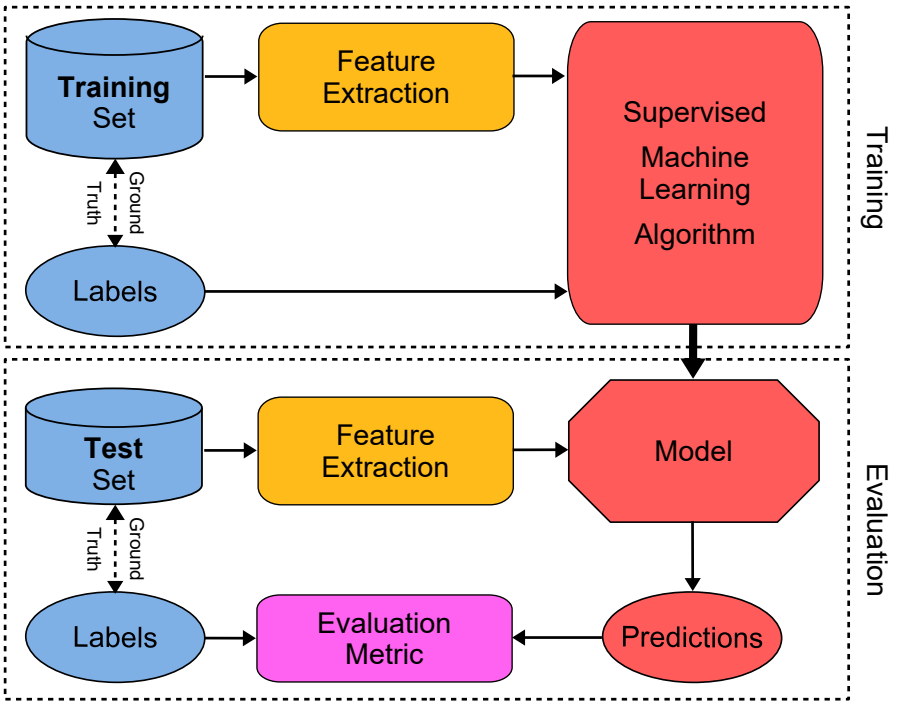

Figure 4.2: Overview of the general workflow applied for supervised machine learning.

where meaningful improvements can be achieved. If models trained with different hyperparameter settings were evaluated on the *test set*, there is the risk that the configuration is adapted on the test set and the results are again not reliable. Thus, it is common to select another set, usually referred to as **development set** or **validation set**, which is employed to evaluate the model performance during hyperparameter optimisation. Once the optimum setting is found, the final model is evaluated on the *test set*, to receive a more reliable performance estimate. For the training of the final model, it is legitimate to fuse training and development sets. More details on data partitioning and performance metrics are found in Section 4.5.

To describe supervised ML approaches, in the remainder of this thesis, the following notation is used:

- $x \in \mathbb{R}^I$: a (numeric) feature vector (of dimensionality I), e. g., a BoAW-feature vector (see Equation 3.13).
- y is the label/target, either a continuous number (regression, $y \in \mathbb{R}$) or a discrete class (classification, e. g., $y \in \{-1, +1\}$)
- $\mathcal{L}$, $\mathcal{T}$: the training set, the development (validation) or test set,
- L: the number of instances in the training set ($L = |\mathcal{L}|$),

- $x^{(l)}$, $y^{(l)}$: a feature vector/label of an instance in the training set[2],
- $x^{(t)}$, $y^{(t)}$: a feature vector/label of an instance in the development (validation) or test set[3],
- $\hat{y}^{(l)}$, $\hat{y}^{(t)}$: the label predicted by a model (for a training or development/test instance, respectively).

As it will be seen in the following, the definition of a classification task as a mapping to $\{-1, +1\}$, i. e., a binary decision, makes the derivation of the ML algorithms easier, but it can be easily extended to multiple classes. Moreover, models for classification and regression are usually quite similar and the focus of the following descriptions will be exemplified based on the binary classification task.

Supervised ML approaches can be roughly categorised into two paradigms, concerning their statistical modelling: **generative** and **discriminative** models. Though no consistent definition exists [226, 227], it is often considered that approaches modelling the joint probability of observations (feature vectors) and targets (labels), or, the conditional probabilities of the observations given the target, are *generative* models. In contrast, whenever the conditional probability of a certain target, given the observation, is (directly) modelled, it is considered a *discriminative* classifier, a term that is also often used for models where no probabilistic modelling is employed. Representatives of generative models are the already mentioned *GMMs*, while examples for discriminative classifiers are *k-nearest neighbour*, *support vector machine*, and most types of *neural networks*.

Practically more important is the distinction between **static** and **dynamic** models: *static* models predict (one or several) unique target(s) for a given instance, while *dynamic* models predict a 'sequence' of targets given sequential input data. As mentioned in Section 2.2, instance-level feature representations, such as functionals or BoAW, are designed for static ML models. When employed for a dynamic 'task', a unique vector is computed for each interval—relating to the intervals of the target—and each interval is considered as a separate instance. In contrast, dynamic models of supervised ML might directly use the sequence of frame-level features as an input. Finally, it should be noted that more recent systems in CA employ also *end-to-end* (E2E) approaches, where the whole processing chain, usually in form of a neural network, is learnt as one model, from the raw signal (audio waveform) to the target [228, 229]. This paradigm can be exploited for both static and dynamic tasks.

Generally, in (supervised) ML, it is a basic knowledge that there is no universally 'best' modelling approach for all tasks. In other words, the optimal type of classifier must be determined for each task, where a certain approach might outperform approaches that are superior for different tasks. This is also referred to as

[2] l: *l*earning
[3] t: *t*esting

the *no-free-lunch* theorem [230]. The fact that also the choice of input modalities, pre-processing techniques, and the selection of features are degrees of freedom and several models can be combined in multiple ways, makes CA such a dynamic and large-scale research topic.

4.2 Pre-processing

Given the (numeric) feature vector x, as defined in the previous section, a few pre-processing steps are usually undertaken before the actual learning process takes place.

4.2.1 Balancing & upsampling

The classes present in a given corpus might not be equally distributed. This can result in a bias of the model towards the majority class(es)—depending on the chosen learning algorithm. As in most usage scenarios, a similar recognition performance is desired for all classes, this needs to be handled when designing a system.

For the *training* set, a simple but powerful method is an *upsampling* of the instances of the minority classes [116], so that a *balancing* between classes is reached. Also more sophisticated techniques have been proposed, such as, e. g., *SMOTE* ('synthetic minority over-sampling technique') [231], where a combination of upsampling (of less represented classes) and downsampling (of more represented classes) and *generation* of synthetic instances is used. For some ML methods, *cost-sensitive learning* [123] is appropriate as an alternative, e. g., by assigning different *weights* to the instances when training a *neural network* [232].

For the evaluation on *development* and *test* sets, in the case of imbalanced classes, it is meaningful to take into account suitable ('unweighted') metrics (see Section 4.5).

4.2.2 Feature normalisation

Besides a normalisation of the *waveform* (see Chapter 2) and a normalisation of the *LLDs* when computing BoAW representations (see Chapter 3), also a normalisation of the ML input needs to be considered [123]. As discussed in Section 3.3 for the LLDs, also here, normalisation can be done using the *z-score* (*standardisation*), where *outliers* have only a small impact on the process. Alternatively, a *min-max* normalisation, i. e., an affine transform setting the minimum/maximum of each feature type in a set to 0/1, or -1/+1, is often used.

In case no normalisation was used, features with a larger range would implicitly have a larger weight in some approaches (e. g., in k-nearest neighbour classification). For many optimisation algorithms, also the speed of convergence is much faster after normalisation [233]. Feature normalisation is usually done **'on-line'**, i. e.,

that the corresponding parameters (e. g., *sample mean* and *standard deviation* for z-score normalisation) are estimated from the training set only and stored to be applied also to the development/test set [234]. If the whole corpus was considered for parameter estimation, information on the distribution of the test data would already be included in the model, and the performance estimate would be biased.

In some specific cases, also **'off-line'** normalisation can be taken into account, e. g., if a model needs to be *transferred* into another domain, with samples having a distribution different from the training data. Then, parameters are estimated from the test domain itself. However, in most applications, a model is required to work directly without the opportunity to learn a new distribution. This is why 'on-line' normalisation is usually preferred.

4.2.3 Feature selection

In feature selection, a subset of features is identified, where

- *irrelevant* features, i. e., features that do not contain any information w. r. t. the given task and
- *redundant* features, i. e., features whose information is covered by other features

have been removed. As for ML itself, many approaches have been proposed [123, 235, 236], but no best practice exists. In the context of this thesis, the topic of feature selection is of little importance as the mainly used methods have proven to cope well with large-scale feature spaces [30, 116].

4.3 Support Vector Machine

One of the most employed supervised ML methods in general, and particularly in this thesis, is the *support vector machine* (SVM) method. The idea of SVM has been developed mainly by Vapnik [237, 238] and is actually related to *artificial neural networks* (see Section 4.4).

4.3.1 Problem statement and solution

The basic problem to be solved by an SVM is a binary classification task, i. e., only two classes are assumed. During training, the SVM optimisation algorithm fits a **hyperplane** into the feature space, separating these two classes with the *largest margin* between training instances of either class and the hyperplane. For this, the classes must be *linearly separable* in the feature space, as it can be seen in Figure 4.3 for a two-dimensional feature space ($x = [x_1, x_2]^T$) and classes $\oplus$ and $\ominus$. In the two-dimensional case, the *hyperplane* is given as a *straight line*.

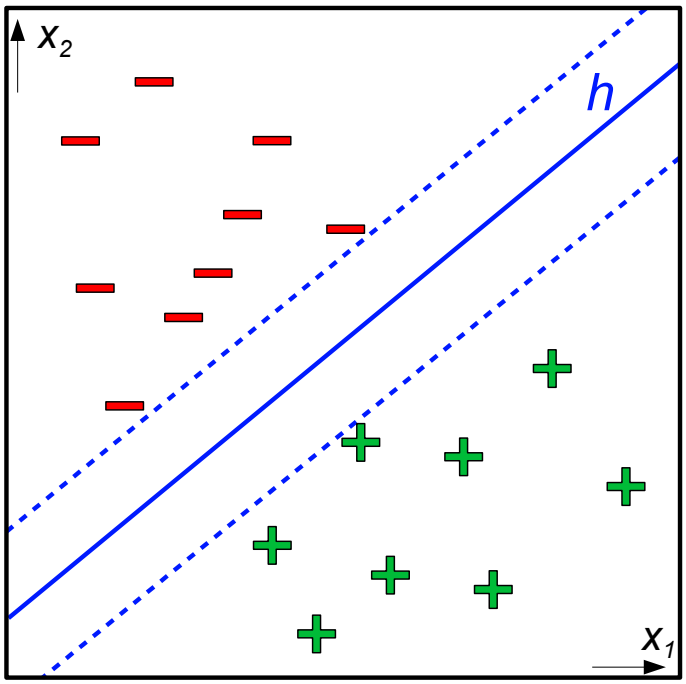

Figure 4.3: A linearly separable binary decision problem solved by an SVM (h: hyperplane).

In general, a hyperplane is given by the equation

$$w^T x + b = 0. \tag{4.1}$$

with the *weight* vector w and the scalar *bias* b [224]. These are the *parameters* of the model, which are learnt during the training process. For the training instances, the hyperplane must fulfil the two conditions

$$w^T x^{(l)} + b \geq +1, \quad \text{for } x^{(l)} \text{ where } y^{(l)} = +1, \quad \text{and} \tag{4.2}$$

$$w^T x^{(l)} + b \leq -1, \quad \text{for } x^{(l)} \text{ where } y^{(l)} = -1. \tag{4.3}$$

The terms ± 1 in the inequations means that a *channel* without any instances around the hyperplane is pursued. Multiplying both sides of the inequations with $y^{(l)}$ shows that the two condition are actually the same:

$$y^{(l)}(w^T x^{(l)} + b) \geq 1. \tag{4.4}$$

Maximising the minimum distance of a training instance from the hyperplane (see [1, 224]) can also be formulated as a dual problem [224], minimising the term

$$\frac{1}{2} w^T w. \tag{4.5}$$

Under the linear side condition given by Inequation 4.3, this can be done using *Lagrange multipliers*. More information about this solution can be found in the corresponding literature [224, 239].

It is obvious that the precondition of linearly separability will not be valid for many real datasets. Even in cases where the feature space seems to be linearly separable on the first view, *outliers* in the training set can lead to a non-separable problem, or at least, they have an excessive influence on the fitting of the hyperplane. This latter aspect usually results in the previously mentioned overfitting. To resolve the problem of outliers, so-called *slack variables* have been introduced. The slack variables $\xi^{(l)}$ model somehow a term of 'allowed error' for each training instance. By this, the side conditions in Inequations 4.2 and 4.3 or 4.4, respectively, are adapted as:

$$x^{(l)}(w^T x^{(l)} + b) \geq 1 - \xi^{(l)}. \tag{4.6}$$

Nevertheless, the optimisation problem can be solved analogously to the previous one as

$$\frac{1}{2} w^T w + C \cdot \sum_{\mathcal{L}} \xi^{(l)}. \tag{4.7}$$

with a free hyperparameter C, sometimes referred to as the *complexity* or *cost*. C regulates the 'allowed' error, i. e., the amount of instances that can be misclassified in the training set in order to avoid overfitting. A large value of the complexity implies that outliers have a large influence on the error term on the right of Term 4.7, while a low complexity allows a larger amount of error. Thus, optimising C, the overfitting (large value of C) vs underfitting (low value of C) can be controlled. In practice, the optimisation of the complexity is usually done evaluating on the development set or using *cross-validation* (see Section 4.5) [240].

In order to obtain the parameters of the hyperplane, the following term is *maximised* [1, 224]:

$$\sum_{l=1}^{L} a_l - \frac{1}{2} \sum_{k=1}^{L} \sum_{l=1}^{L} a_k a_l y^{(k)} y^{(l)} (x^{(k)T} x^{(l)}). \tag{4.8}$$

For the Lagrange multipliers a_l, the following two conditions ('box constraints') apply:

$$0 \leq a_l \leq C \quad l = 1, \ldots, L \quad \text{and} \tag{4.9}$$

$$\sum_{l=1}^{L} a_l y^{(l)} = 0. \tag{4.10}$$

The first condition determines that all coefficients a_l are non-negative and not larger than hyperparameter C, the second condition ensures that the coefficient weights associated with positive and negative samples are balanced. The maximisation problem itself is a **quadratic optimisation problem** and can be solved with different quadratic programming algorithms [241]. Moreover, further solver types and also further formulations of the problem have been proposed, such as the ν-SVM by Schölkopf et al. [242].

Having learnt the optimal coefficients, the parameters of the hyperplane are given by [1]

$$w = \sum_{l=1}^{L} a_l y^{(l)} x^{(l)} \quad \text{and} \tag{4.11}$$

$$b = y^{(l^*)}(1 - \xi^{(l^*)}) - w^{(l^*)T} x^{(l^*)}. \tag{4.12}$$

It is evident that w, the normal vector of the hyperplane, is a weighted sum over the training instance feature vectors $x^{(l)}$ multiplied by the sign of the label $y^{(l)}$. Depending on hyperparameter C, most of the learnt weighting coefficients a_l are actually 0. The feature vectors corresponding to non-zero coefficients a_l are defining the SVM hyperplane and are called 'support vectors'. In Equation 4.12, instance l^* is the one with the largest a_l.

The prediction $\hat{y}^{(t)}$ for a test instance $x^{(t)}$ is then given by

$$\hat{y}^{(t)} = \text{sgn}(w^T x^{(t)} + b), \tag{4.13}$$

with the *sign*-function sgn() returning +1 for a positive input and −1 for a negative one. Instead of a binary classification problem, also regression problems can be solved analogously. For more information on the derivation of **support vector regression** (SVR), which is also employed in the third part of this thesis, the reader is referred to the literature [1].

So far, for support vector *classification*, the assumption has been that only two classes are present in the data. Nevertheless, more than two classes are often present in CA tasks [13]. To tackle **multi-class** problems with an SVM, a 'workaround' needs to be employed, training an *ensemble* of SVMs. The most popular schemes for this are *one-vs-one* (pairwise) and *one-vs-all* [243]. In the first scheme, for each pair in the set of classes, an SVM is trained and predictions are combined. In the second one, a single SVM is trained for each class, discriminating class instances from instances of all other classes; then, the SVM with the maximum output of the prediction function (see Equation 4.13 without the *sign*-function) is considered the most likely class.

4.3.2 Kernels

Finally, also classification (or regression) problems, where the feature space is inherently **not linearly separable** need to be taken into account. To tackle this, **kernels** are employed, a concept which is used also for ML approaches other than SVM, such as, e. g., *k-nearest neighbour*.

The idea is that the feature space is *implicitly* transformed into a higher-dimensional space, in which it is linearly separable again. Putting w from Equation 4.11 into Equation 4.13, it is shown that the *dot product* of $x^{(l)}$ and $x^{(t)}$ is required for the prediction function, but not the feature vectors themselves. The same is valid for Equation 4.8 [224]. This means that it is not required to (explicitly) transform the feature vectors into another space, but it is sufficient to know the result of the *dot product* between the transformed versions of $x^{(l)}$ and $x^{(t)}$, or all pairs of $x^{(l)}$ for training, respectively. This is computationally much more efficient and known as the 'kernel trick' [1].

Given a feature space transformation Φ, the corresponding kernel K^{Φ}, given two feature vectors x and x', is defined as

$$K^{\Phi}(x, x') = \Phi(x)^T \Phi(x'). \tag{4.14}$$

Valid kernels need to fulfil certain conditions [1] and need to be optimised for a given task. Besides the *linear* kernel, which is implicitly given by just using the (linear) *dot product* as in Subsection 4.3.1, the most popular kernel is probably the *Gaussian* or *radial basis function* kernel [240], given by

$$K^{\Phi}(x, x')_{\sigma} = \exp^{\frac{||x-x'||^2}{2\sigma^2}}, \tag{4.15}$$

with a tunable hyperparameter σ. It has been shown that a Gaussian kernel can be tuned to have a behaviour similar to linear SVM [244], but it is able to separate feature spaces with a highly non-linearly separable distribution [240].

Suitable kernels for histogram feature representations have been proposed with the *histogram intersection* kernel (HIK) [245] and with the *generalised histogram intersection* kernel (GHIK) [246]. The HIK is defined as

$$K^{\Phi}_{\mathrm{HI}}(x, x') = \sum_{i=1}^{n} \min\{x_i, x'_i\} \tag{4.16}$$

and the GHIK as

$$K^{\Phi}_{\mathrm{GHI}}(x, x') = \sum_{i=1}^{n} \min\{|x_i|^{\beta}, |x'_i|^{\beta}\}, \tag{4.17}$$

with a hyperparameter β.

In the context of BoAW, it has been shown that a HIK is superior to the linear kernel [26, 190]. However, the authors of both works state that the performance increase is not big in case of an L1-([26]) or L2([190])-normalisation of the BoAW histograms and they conclude that a linear SVM is preferable when working with large datasets and/or codebook sizes as more efficient optimisers can be employed [190]. Furthermore, it is not clear whether the complexity was optimised in the authors' work.

4.4 Neural Networks

Artificial neural networks (ANNs) have promoted the success of ML and AI during the last decade in a meaningful way. Though early works date from the 1940s (McCulloch and Pitts [247]) and 1950s (Rosenblatt [248]), a major breakthrough has not happened until the recent decade, where **deep learning** (DL) [249] started to dominate the research on ML [250]. The huge impact and success of deep learning has been triggered by *three* circumstances [251]: first, the technical progress in computer hardware, most importantly *graphics processing units*, which allow for a large degree of parallelisation (vectorisation), corresponding interfaces, and software frameworks, such as TensorFlow [252] and PyTorch [253]. Second, large-scale data could be stored and distributed easily, at minimum costs. Third, novel ANN architectures and learning techniques [249] have improved recognition performances in many ML tasks in computer vision, NLP, CA, and other fields.

While ANNs have clearly been inspired by the mechanisms in the human brain and nervous system, they are not (yet) as capable as the human brain throughout many tasks [254]. More specifically, ANNs require larger amounts of data to be trained compared to humans. Moreover, it is assumed that humans perform *unsupervised* or *reinforcement* learning [255], while many ML tasks, such as, e. g., emotion recognition in speech, are often solved by *supervised* learning [20].

This section gives a quick overview on the basics of ANNs and DL and the model types or architectures that are used in the experiments of this thesis. A comprehensive textbook on DL has been published by Goodfellow et al. [249].

4.4.1 Artificial neurons

All types of ANNs are typically built from basic units, the so-called **neurons**. Common definitions are similar to the model proposed by McCulloch and Pitts [247]; in the following, the definition given in Figure 4.4 is assumed.

The neuron model consists of $I + 1$ inputs x_i, where I is the number of features. The extra input (with index 0) is a commonly used *bias*, which has a constant value of 1. Each input is weighted with a specific **weight** w_i. Then, all weighted inputs are **summed up** ($\sum$) and mapped by a **non-linear activation function** $f()$. The

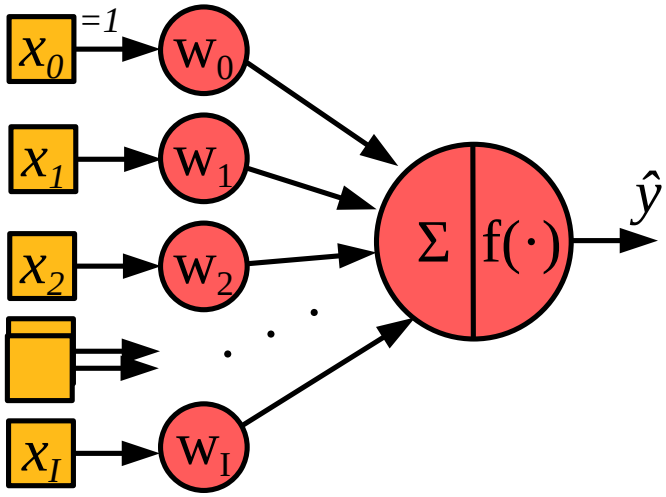

Figure 4.4: The neuron model.

output $\hat{y}$ is then given by

$$\hat{y}(x) = f\left(\sum_{i=0}^{I} w_i x_i\right) = f(w^T x), \quad \text{with} \quad x_0 = 1. \tag{4.18}$$

This equation is quite similar to the output prediction of an SVM with a linear kernel (cf. Section 4.3, Equation 4.13).

4.4.2 Multi-layer perceptron

As it has been shown already for SVM that only few classification (or regression) problems can be solved with this type of 'linear' model, several neuron units are usually used in parallel and concatenated to layers. The neuron outputs are then propagated and used as an input for all neurons in the succeeding layer. This type of model is shown in Figure 4.5 and usually called (standard) **feedforward neural network** or **multi-layer perceptron** (MLP) [249]. The activation function is of fundamental importance when building neural networks from several layers of neurons. Without a non-linearity, the final output of the network $\hat{y}$ could be represented equivalently by a simple linear model (as given in Equation 4.18). Yet, with a non-linear activation function, also non-linearly separable problems can be solved [249].

As illustrated in Figure 4.5, the first layer is called **input layer** and does typically not contain any weights, but their outputs are just identical with the input features. For this, the number of neuron units in the input layer is always equal to the number of (input) features. The succeeding layers are called **hidden layers** as they contain an 'internal' representation and the neuron outputs ('activations') are

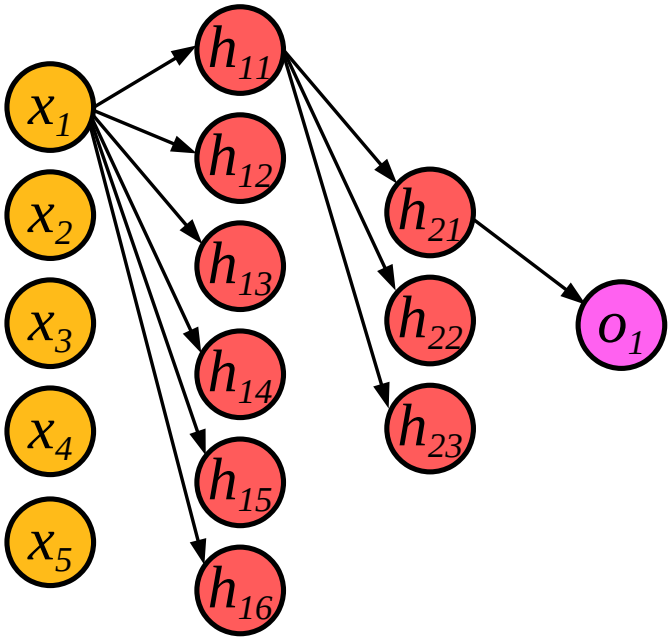

Figure 4.5: Multi-layer perceptron with two hidden layers *(h)*. The input dimensionality (feature space) is 5. The 1^{st} hidden layer has 6 nodes (neurons), the 2^{nd} hidden layer has 3 nodes, and the output layer has one unit. Each neuron of a layer is connected to each neuron of the subsequent layer, scaled with a weight. Only a few connections are shown in this illustration. The biases are not shown either.

neither the model input nor output, i. e., they are not observable from outside, when considering the neural network as a *black-box model*. The final layer is called **output layer** and the number of output neurons is equal to the number of outputs. More specifically, this means that for a regression task or a binary classification problem, the model has usually only one output neuron, where the co-domain of the output activation function must be in the range of the expected outputs (regression), or, either in the range of $[-1, +1]$ or $[0, 1]$ for binary classification, respectively. Of course, also several regression targets or binary classification tasks can be learnt at the same time. For multi-class tasks, however, the number of outputs is typically equal to the number of classes. The targets are then represented in terms of a **one-hot encoding**, i. e., classes A, B, & C would be converted into $(1, 0, 0), (0, 1, 0), (0, 0, 1)$, as an example. This ensures that the Euclidean distances between output representations of two different classes are equal and that there is no 'order' of classes. During *inference* (when using the model to make predictions), the class with the maximum output is considered as the prediction of the model.

Each node (neuron) in a layer is connected with each node in the subsequent layer, where the corresponding activation is scaled with a weight at the input of each node and a bias is added (not shown). As recent neural network architectures

usually incorporate at least two but usually (much) more hidden layers, the terminology 'deep neural network' (DNN) and 'deep learning' is commonly used for corresponding models. It has been shown theoretically that, in principle, any function can be approximated by an MLP with corresponding weights and activation functions ('universal approximation theorem') [256]. The exact *architecture* of the network and *hyperparameters*, i. e., the configuration of the training algorithm, need to be designed and/or optimised depending on the task and data.

4.4.2.1 Activation functions

As discussed above, a non-linear activation function is normally required at the output of each neuron node. The most common activation functions with corresponding output ranges (co-domains) are listed in the following [257].

1. Sigmoid:

$$f(x) = \frac{1}{1 + e^{-x}}, \quad \text{Range: } (0, 1) \tag{4.19}$$

2. Hyperbolic tangent:

$$f(x) = \frac{e^{2x} - 1}{e^{2x} + 1}, \quad \text{Range: } (-1, 1) \tag{4.20}$$

3. Rectified linear unit (ReLU):

$$f(x) = \max(0, x), \quad \text{Range: } [0, \infty) \tag{4.21}$$

4. Softmax:

$$f(x) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}, \quad \text{Range: } (0, 1) \tag{4.22}$$

The *softmax* activation function is usually (but not only) used in output layers in multi-class classification tasks, where only one class is correct for one instance. It implies a normalisation of the activations (after the non-linear exponential function) and thus, the output activations sum up to 1. All activation functions share the property of *monotony*.

4.4.2.2 Training

During the training process, the optimal *parameters* of the ANN, i. e., the *weights*, are learnt. As a mathematical term to decide how 'good' the learnt parameter set

is, a **loss function**, also called **objective function**, is defined to measure the *error* between the model output (given the current weights) and the ground truth. The weights of the ANN are then usually optimised using a **gradient descent** approach. In this class of optimisation methods, the weights are initialised with (pseudo-)random numbers and then updated iteratively in the direction of the *negative gradient* (derivative) of the loss function.

With the parameters of the ANN θ (weights, including biases), the loss function $J(\theta)$ can be the **mean squared error** (MSE) [249], given as

$$J_{\text{MSE}}(\theta) = \frac{1}{2L} \sum_{l \in \mathcal{L}} \left(y^{(l)} - \hat{y}^{(l)}(x^{(l)}; \theta) \right)^2 . \tag{4.23}$$

MSE loss is the 'default' loss for regression problems but it can be applied in classification problems (one-hot encoding) as well. Nevertheless, the most commonly used loss for classification is **categorical cross-entropy** (CCE) [249], defined as

$$J_{\text{CCE}}(\theta) = -\frac{1}{L} \sum_{l \in \mathcal{L}} \sum_{k=1}^{K} y_{\text{OH},k}^{(l)} \log \hat{y}_{\text{OH},k}^{(l)}(x^{(l)}; \theta), \tag{4.24}$$

where $y_{\text{OH},k}^{(l)}$ is the *one-hot encoded* ground truth label and $\hat{y}_{\text{OH},k}^{(l)}$ the corresponding model output. Usually, the *softmax* activation function is used in the output layer, but any other activation function returning an output in the range of $[0, 1]$ could be applied. When dealing with only two classes and one output neuron, the **binary cross-entropy** (BCE) is more appropriate:

$$J_{\text{BCE}}(\theta) = -\frac{1}{L} \sum_{l \in \mathcal{L}} \left(y_{\text{OH},k}^{(l)} \log \hat{y}_{\text{OH},k}^{(l)}(x^{(l)}; \theta) + (1 - y_{\text{OH},k}^{(l)}) \log(1 - \hat{y}_{\text{OH},k}^{(l)}(x^{(l)}; \theta)) \right), \tag{4.25}$$

BCE can also be used for *multi-label* tasks, i. e., when one instances may belong to (none or) multiple classes, usually with a *sigmoid* activation function in the output layer. However, **mean absolute error** (MAE) loss has been shown to be a better alternative for classification when facing datasets with *noisy* labels [258].

Once the loss is defined, the **gradient** of the loss $\nabla J(\theta)$ can be computed and the weights are updated iteratively. For this, the gradients need to be computed for each training instance, i. e., $\nabla J^{(l)}(\theta)$, as defined before but without the averaging. In each iteration, the following operation to update the model parameters is performed:

$$\theta \leftarrow \theta - \frac{\eta}{L} \sum_{l=1}^{L} \nabla J^{(l)}(\theta). \tag{4.26}$$

The hyperparameter η is called the **learning rate**. It controls how fast the weights are updated in each iteration and is usually chosen $\ll 1$. Usually, in the updates,

a **momentum** term is added, which means that the parameter updates from the previous iteration are added scaled with a certain factor [249]. This prevents that the learning algorithm converges into local minima.

It is quite straightforward that the gradients of the loss can be directly utilised to update the weights in the output layer. However, for the hidden layers, the loss 'before' the output layer would be required. This is commonly done using **backpropagation**. For this, the employed activation functions must be (at least piecewise) differentiable. The equations are not derived here, but the reader is referred to the corresponding literature [249, 259]. Very efficient implementations on graphics processing units have been published as part of the previously mentioned toolkits. For this, researchers and engineers are usually not required to implement the training algorithm itself, but they need to take care only of feature design and the optimisation of the ANN architecture and hyperparameters.

The update algorithm in Equation 4.26 is called *batch gradient descent* as the neural network is updated at once based on the (averaged) gradients from the whole training set ('batch'). However, it is often beneficial to update the network already using the gradients from each (randomly picked) instance. This procedure is called **stochastic gradient descent** [260]. Nevertheless, the most common method is to use so-called **mini-batches** of data in each update cycle, i. e., a randomly selected subset of the whole training date. The learning algorithm is then called **mini-batch gradient descent** [249].

Once all instances from the training set have been used for updates, this is called one **epoch**. After this, the training cycle restarts using new mini-batches from $\mathcal{L}$, where each instance is used only once in each epoch. Though the number of epochs is usually predefined as a hyperparameter, the ANNs tend to overfit after training for a while. This is why **early stopping** is commonly employed as part of the training procedure, which means that the training process is finished once the loss (or another metric) does not improve anymore on the development set. As an alternative, the network weights at the epoch where the minimum loss was achieved can be restored.

In practice, more sophisticated training algorithms are applied, referred to as **optimisers**, which improve and adapt the *learning rate*, such as **RMSprop** ('root mean square propagation'), or both *learning rate* and *momentum*, such as **Adam** ('adaptive moment estimation') [261]. In fact, the learning rate is a very critical hyperparameter and the mentioned optimisers make them adaptive w. r. t. both iterations and individual network weights, by this improving the robustness of the training process. Thus, only an *initial learning rate* is typically set, together with the *(mini-)batch size* and the *(maximum) number of epochs*.

4.4.2.3 Regularisation techniques

As ANNs tend to overfit, training is usually enhanced by so-called *regularisation* techniques. These are approaches with the goal to improve generalisation, i. e., reduce the error on the test set, without taking into consideration the (potentially larger) error on the training set [249]. A large number of methods have been proposed in this context. In the following, the ones most relevant for this thesis are quickly introduced.

Weight decay A penalty term is added to the loss function, penalising large weights in the ANN [249].

Dropout When using *dropout*, a certain percentage of outputs is set to 0 randomly ('dropped') in corresponding layers [262]. To ensure the sum over the input to the next layer remains at approximately the same magnitude, the remaining activations are scaled. Dropout helps to prevent 'co-adaptation' of neuron weights in the same layer and thus it can be considered as generating an *ensemble* of models [263]. When the model is trained, during inference, dropout is not applied and activations remain with their original scale.

Batch normalisation A very popular technique applied to the outputs of ANN layers is *batch normalisation*. It is quite similar to (z-score) feature normalisation as discussed in Section 4.2.2, but it is applied to the internal representations of the ANN. During training, the parameters for normalisation (*mean* and *standard deviation*) are estimated from the data in each mini-batch. During inference, the averaged parameters learnt during training are used for normalisation. Besides improved generalisation, batch normalisation mainly speeds-up the convergence of the training process [264].

Data augmentation As datasets used in ML are usually *relatively* scarce, an artificial augmentation of the training data can increase the robustness of the model. A very generic way to do this is by simply *adding noise* to the features at the input of the neural network [265]. When working with audio signals, domain-specific augmentation techniques can also be applied directly to signals, prior to a potential feature extraction step. This includes, e. g., *time stretching* [266], *pitch shifting* [267], and *random frequency filtering* [268].

4.4.3 Convolutional neural networks

In recent years—mainly pushing 'deep learning' forward—novel architectures have been proposed, which are much more efficient than the discussed MLPs, most importantly, the so-called **convolutional neural networks** (CNNs) [249].

In these types of networks, operations similar to sample-domain filtering (i. e., convolutions [43]) are executed. The specific characteristic of CNNs is **weight sharing**, i. e., that overlapping patches in the input are processed with the same weights, while the output remains with the same geometric shape (potentially sub-sampled). Though most popular in image classification tasks [269], they have proven to be very useful also in the audio domain, either using the spectrogram [270] or directly the audio waveform [271] as their input.

The property of sharing weights between different regions of the input also provides a certain degree of *shift invariance*. This is meaningful because objects (in images) or events (in audio) are commonly not present at always the same spot. CNNs are composed of a series of **convolutional** layers and **pooling** layers, often followed by *batch normalisation*. Convolutional layers consist of a predefined number of *kernels*, where each kernel is shifted over the input with a certain *stride* (step size). As an example, for a *2D-convolutional* layer, a spectrogram given as a 3D-*tensor* of size $(n_{\text{frames}} \times n_{\text{bins}} \times 1)$ is transformed into a tensor of size $(\frac{n_{\text{frames}}}{\text{stride}} \times \frac{n_{\text{bins}}}{\text{stride}} \times n_{\text{kernels}})$. For each kernel ('filter'), a *feature map* is generated. In subsequent layers, each kernel uses the full set of feature maps as input, thus, resulting in a 3D-convolution. Consequently, a *1D-convolutional* layer maps from a two-dimensional input to a two-dimensional output, where the first dimension is representing *time* (samples/frames) and the second one representing the *features*. Convolutional layers are usually followed by a ReLU activation function [249].

After each convolutional layer (and potential batch normalisation and activation operations), **pooling** is applied [249]. Pooling is usually realised as *max pooling*, where only the *maximum* activation over a certain neighbourhood is propagated to the next layer. As an example, when working with 2D-convolutional layers, they are typically followed by a max pooling operation with a neighbourhood of size $(2, 2)$ and a stride (step size) of the same size, resulting in a (non-linear) sub-sampling by a factor of 2. Max pooling is usually done independently for each feature (map). Pooling generally increases the shift-invariance of the model, while at the same time compressing the information propagated through the neural network.

CNNs in the context of audio are usually employed for *feature extraction*, based either on the raw signal or the spectrogram as a mid-level representation. CNNs as a 'feature extraction' front-end are often combined with MLPs as a 'back-end' [270]. Generally, the same regularisation techniques, learning algorithms and optimisers as discussed before can also be applied for CNNs.

It has been shown that *pre-training* of CNNs for feature extraction, i. e., training them on data different from the task they are applied to in the end, can improve the results of the model. Employing pre-trained CNNs can even be done across domains ('transfer learning'), e. g., from the image classification domain to CA, where either only the neural back-end is re-trained for the given task or a different classifier (e. g., an SVM) is trained based on the outputs of a certain layer of the network [270, 272].

4.4.4 Recurrent neural networks

While MLPs and CNNs are types of *feedforward neural networks*, another major class of ANNs are the so-called *recurrent neural networks* (RNNs) [249]. The main difference is that in RNNs, units hold an internal state which is propagated to units of the subsequent, previous, and/or the same layer ('feedback connections'). For this, the information in RNNs is processed in terms of 'time' steps, which typically requires a 'sequential' nature of the input data. For this, RNNs are suitable to cope with *time series*, such as, e. g., *financial data* [273], *natural language* [274], or *audio* [275].

Though many different types of RNNs, such as *echo state networks* [249] or *Hopfield networks* [276] exist, the ones most commonly used in CA and speech recognition are the so-called **gated RNNs** [249], where the most famous representative is the **Long short-term memory** (LSTM)-RNN.

4.4.4.1 Long short-term memory

LSTM-RNNs have been proposed by Hochreiter and Schmidhuber in the 1990s [277]. Instead of using standard *neurons* (as introduced in Subsection 4.4.1), they are replaced by LSTM *cells* [249]. Each cell is composed of a **cell memory**, an **internal cycle** ('self-loop'), and four neurons, where one neuron processes the input and the other three are **gates** controlling the flow of information within the cell, as illustrated in Figure 4.6. All gates normally have a *sigmoid* activation as their output, which is then multiplied with the corresponding activations in the cell. There are the following three gates:

1. The **input** gate, to control the amount of the input activation that is propagated into the cell memory.

2. The **forget** gate, to control the amount of activation in the cell memory that is propagated to the next time step and added to the input of the next time step.

3. The **output** gate, to control the amount of activation in the cell memory that is present at the output of the cell.

LSTM-RNNs are one method to solve the **vanishing gradient problem** [249], which is generally present in RNNs, resulting in gradients converging to zero (or diverging, depending on the activation functions) when propagating the error back through a large number of time steps (and layers) during training. In principle, the same regularisation techniques and optimisers as introduced before can be used also in the context of LSTM-RNNs. However, the training process is usually much more complex than for feedforward neural networks [278].

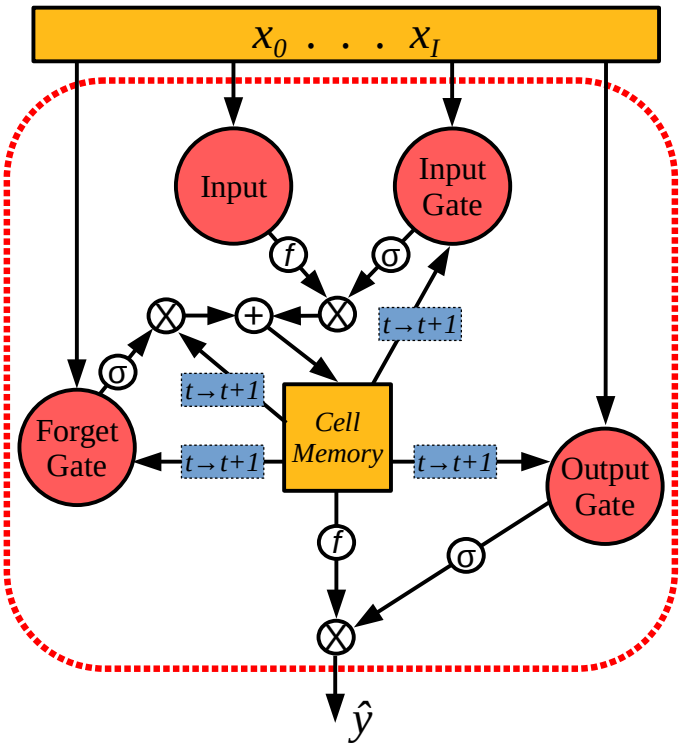

Figure 4.6: Long short-term memory cell. The arrows show the propagation of information. The cell memory usually contains only a scalar, but also variants with vectors have been proposed. $t \rightarrow t+1$ represents a time step (delay). $\times$ stands for a multiplication ('gate'), $+$ for a summation, f stands for the input/output activation function, e. g., *tanh*, and σ is the *sigmoid* activation function of the gates. The input (vector) is x and the output (usually a scalar) is $\hat{y}$. The weights of the neurons are not displayed explicitly.

In scenarios where both past and future data from a time series is available, **bidirectional** LSTM-RNNs [279] (BLSTM-RNNs) can be employed, where two RNNs are combined—one network propagating the data in the correct temporal order and a second one processing it in the reverse temporal order.

While LSTM-RNNs generally output a sequence of the same rate as the input, in the final LSTM layer, depending on the given ML task, it might be preferred to output only the *last output* of the sequence, to end up either in a single feature vector that can be further processed with an MLP or directly in the output. Otherwise, a *pooling* across the time dimension is required after the LSTM-RNN.

Gated recurrent units (GRUs) [280] are a derivative of LSTM units, where the *output gate* is dropped, leading to fewer parameters while keeping a similar modelling ability.

4.4.5 Final remarks

This section has just focussed on the essential fundamentals of ANNs and deep learning that are required for the models employed in this thesis. Research in the field has developed a very high dynamics during the last few years, with very successful **generative approaches**, mainly *generative adversarial nets* [281] and *variational autoencoders* [282]. Furthermore, refinements in CNN-based models, such as, e. g., *residual neural networks* [283] or *capsule neural networks* [284], networks pre-trained on large amounts of data [285], and large numbers of (annotated) online resources [286] have provided researchers and developers with an almost infinite set of tools to create models and systems.

4.5 Evaluation of Models

In this section, the methods of evaluation that are used in this thesis are discussed. In general, evaluation is done to optimise hyperparameters and to obtain a reliable estimate of the performance of a finalised model.

4.5.1 Data partitioning

As mentioned before, prior to model training, the available data needs to be partitioned into (at least) a training and test set, and usually also a development set. While the partitioning can be done in a random way for many tasks, for speech-related tasks—or other kinds of subject-related tasks—, **speaker-independence** (subject-independence) is an important aspect [1]. This means that all instances from a subject must be assigned to the same partition. Otherwise, the performance estimates would likely be overestimated as the model has knowledge about specific personal specificities of the subjects' voices. Disjunct splits are of particular importance in cases where human *traits* are recognised, i. e., targets that are constant for each subject. Another important aspect of partitioning is **stratification**, which means that the class (or continuous) label distribution should be similar across partitions.

An interesting alternative to using fixed partitions, especially in cases of little data, is **cross-validation** (CV) [123]. In *(stratified) k-fold CV*, a given dataset is first split into k different (disjunct) *subsets*. Then, k models are trained on k different combinations of the data, where one subset is withhold for evaluation and the remaining $k - 1$ subsets are used for training. Finally, predictions on the 'evaluation subsets' are (concatenated and) evaluated. In case of hyperparameter tuning, the CV can be used either for tuning only and a fixed test set is kept aside, or more sophisticated techniques such as *nested* CV [287] can be utilised. To train the final model which is evaluated on the test partition, all training and development

	A	B	C
A	$e_{1,1}$	$e_{1,2}$	$e_{1,3}$
B	$e_{2,1}$	$e_{2,2}$	$e_{2,3}$
C	$e_{3,1}$	$e_{3,2}$	$e_{3,3}$

Table 4.1: A confusion matrix (3 classes).

splits can be fused—if the optimum hyperparameter setting is expected to be similar for the fused data.

Also when employing CV, speaker-independence should be respected. For this, *leave-one-speaker-out (LOSO) CV* is often used, where each split consists of all samples from one subject—consequently, each model is trained on the data from all subjects but one and the performance estimate is representative for 'unseen' subjects. In general, in order to preserve comparability between ML experiments, always the same partitioning should be used for a given dataset.

In the following, different metrics are discussed to assess the 'correctness' of predictions, based on the available ground truth labels.

4.5.2 Metrics for classification

In the context of the evaluation of classifiers, the **confusion matrix** is the fundamental representation [123]. Given K classes, the confusion matrix has the size $K \times K$, where rows and columns are assigned to each class. Though different conventions for the assignment of rows and columns exist, in this thesis, it is defined that the **rows** contain the **true** instances while the **columns** contain the **predictions**. An example with three classes (A, B, & C) is shown in Table 4.1.

The entries $e_{i,j}$ (row i, column j) specify the number of instances from class i that have been classified as j. It is evident that the correctly classified instances are those on the main diagonal ($e_{i,i}$). To compress the information into a scalar metric, the simplest option is to use the **accuracy**. It is defined as the sum of the entries on the main diagonal (*trace*) divided by the number of instances in the respective test set:

$$\mathrm{Acc} = \frac{\sum_{i=1}^{K} e_{i,i}}{\sum_{i=1}^{K} \sum_{j=1}^{K} e_{i,j}}. \tag{4.27}$$

The accuracy is a number between 0 and 1, usually expressed in percent. An accuracy of 100 % means perfect classification, while an accuracy of 0 % means that all instances have been misclassified.

As the accuracy does not provide any information on the classification performance of individual classes or the type of error, further measures are usually provided [1]. The most important ones are **recall** and **precision**. The *recall* is defined

as

$$\mathrm{Rec}_i = \frac{e_{i,i}}{\sum_{j=1}^{K} e_{i,j}} \tag{4.28}$$

and the *precision* is defined as

$$\mathrm{Prec}_j = \frac{e_{j,j}}{\sum_{i=1}^{K} e_{i,j}}. \tag{4.29}$$

In other words, the recall expresses the ratio of instances from a class that has been classified correctly, while the precision specifies the ratio of the instances predicted as a class that actually belongs to this class. To combine recall and precision, the *harmonic mean* of precision and recall is often reported, called **F-measure**:

$$\mathrm{F}_{1j} = \frac{2 \cdot \mathrm{Rec}_i \cdot \mathrm{Prec}_i}{\mathrm{Rec}_i + \mathrm{Prec}_i}. \tag{4.30}$$

Recall, precision, and F-measure are *class-specific measures*, i. e., the respective measure can be computed for all classes.

To summarise the class-specific metrics into a single one, the *unweighted average* over all classes is often reported, most importantly, the **unweighted average recall** (UAR). It is defined as

$$\mathrm{UAR} = \frac{\sum_{i=1}^{K} \mathrm{Rec}_i}{K}, \tag{4.31}$$

i. e., the arithmetic mean of the class-specific recalls. For this, the UAR is also referred to as **macro-average recall** or **unweighted accuracy**. The UAR is usually reported as the only metric, as a low precision in a certain class always results in a low recall in another class and the information would be somehow redundant to a certain extent. Furthermore, it must be noted that the precision is not defined in cases where a class is never predicted by a model. The confusion matrix is also very often presented in terms of percent, where each entry is divided by the *row sum*. In this representation, the recalls are explicitly given as the entries on the main diagonal and the UAR is their arithmetic mean.

The UAR is suitable for datasets with imbalanced classes as the recall of each class has an equal weight ('unweighted'), compared to the *accuracy*, where larger classes have a larger influence on the result. The *accuracy* can also be computed as the sum of the recalls weighted by the corresponding ratio of instances in each class and is therefore sometimes referred to as **weighted average recall**.

For the UAR, the **chance level** of a classifier is $\frac{1}{K}$, i. e., a classifier predicting always the same class (or making random decisions) will always lead to this UAR. This is considered as the lowest *baseline* a classifier has to surpass in order to prove that anything meaningful can be learnt at all.

For **binary decision problems**, the same overall-metrics can be used as for multi-class problems. However, when dealing with **detection** tasks ('positive' vs 'negative'), the recall, precision, or F-measure are commonly defined as the one for the *positive* class [1]. One reason for this is that the negative class usually has much more instances and a prediction of 'negative' is considered the default and not relevant to be reported. In this context, the correctly classified instances from the positive class are also referred to as *true positives*, the ones misclassified as negative as *false negatives*, while the negatives ones misclassified as positive are the *false positives*. The *recall* is sometimes called *sensitivity* in this context.

Another representation relevant for binary decision tasks is the **receiver operating characteristic** (ROC) curve [123]. It exploits the *confidences* of a classifier (probability estimate of 'positive') and not the binary predictions. For different thresholds to make the binary decision based on the confidences, the *true positive-rate* (=recall) is plotted over the *false positive-rate* (false positives divided by the sum of 'negative' instances). As a scalar metric between 0 and 1, the **area under the curve** (AUC) can be reported.

4.5.3 Metrics for regression

To evaluate regressors, the similarity between test labels and predictions is measured. While the **mean absolute error** [1] or the **mean squared error** [123] can be suitable measures for this, for the experiments in this thesis, correlation metrics are utilised.

The simplest metric is the (linear) correlation [239], also called **Pearson's correlation coefficient** (PCC) [1]. With $\overline{y}$ and $\overline{\hat{y}}$ as the means (cf. Section 3.3, Equation 3.8) of y and $\hat{y}$ over all test samples and the number of test samples $|\mathcal{T}|$, the (empirical) *covariance* is given by

$$\mathrm{s}_{\mathrm{y},\hat{\mathrm{y}}} = \frac{1}{|\mathcal{T}|-1}\sum_{t=1}^{|\mathcal{T}|}(y^{(t)}-\overline{y})(\hat{y}^{(t)}-\overline{\hat{y}}), \tag{4.32}$$

and the variances by

$$s_y^2 = \frac{1}{|\mathcal{T}|-1}\sum_{t=1}^{|\mathcal{T}|}(y^{(t)}-\overline{y}) \quad \text{and} \tag{4.33}$$

$$s_{\hat{y}}^2 = \frac{1}{|\mathcal{T}|-1}\sum_{t=1}^{|\mathcal{T}|}(\hat{y}^{(t)}-\overline{\hat{y}}). \tag{4.34}$$

With these definitions, the PCC is defined as

$$\rho_{\mathrm{PCC}} = \frac{s_{y,\hat{y}}}{s_y s_{\hat{y}}}. \tag{4.35}$$

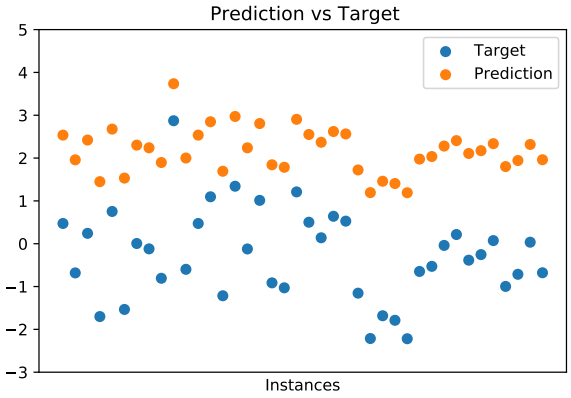

Figure 4.7: Visualisation of a continuously valued target and corresponding predictions. The predictions are a scaled and shifted version of the target. The PCC is 1.00 and the CCC is 0.15.

The PCC is a value in the interval of $[-1, 1]$, where 1 is maximum correlation between prediction and target and 0 stands for orthogonality ('chance level'); negative values imply a negative correlation. It must be noted that the PCC is a *scale- and shift-invariant* metric, which means that it is also maximum ($= 1$) if the prediction is a scaled and shifted version of the target (cf. Figure 4.7).

However, it might be required that the predictions directly meet with the targets. Therefore, as a 'combination' of correlation and error, the **concordance correlation coefficient** (CCC) has been proposed [288]. It can be defined as [289]

$$\rho_{\mathrm{CCC}} = \frac{2 s_{y,\hat{y}}}{s_y^2 + s_{\hat{y}}^2 + (\overline{y} - \overline{\hat{y}})^2}. \tag{4.36}$$

As the PCC, the CCC ranges from -1 to 1, but it is not scale- and shift-invariant and therefore reaches the maximum only if predictions and targets are identical (cf. Figure 4.7). The CCC has established itself as the main evaluation metric for *emotion recognition* in continuous annotation spaces [18].

Finally, for evaluations where the numeric range, scale, and proportionality are not relevant, a **rank correlation** metric, such as **Spearman's** ρ is suitable [290]. It is also maximum ($= 1$) in cases where the predictions and corresponding labels have a *monotonous* relationship.

4.5.4 Statistical significance testing

To complete this section on evaluation, the concept of *statistical significance* should be mentioned [291]. In statistics, hypothesis testing is an omnipresent, yet controversial, topic [292]. Loosely speaking, significance testing tells if the difference

between two results (e. g., the classification performance of two models) is based on coincidence or not.

More precisely, a *null hypothesis* is made presuming that the samples (i. e., the results provided by two different machine learning approaches) are actually drawn from the same population. This means, the null hypothesis assumes that there is no difference in performance. Then, a significance test is performed computing a ***p*-value**, which tells the probability of receiving the actual experiment result (or a result even more different from the baseline) under the assumption that the null hypothesis is *true* [293]. In case the p-value is below a certain (predefined) **level of significance**, the null hypothesis is rejected and the difference between two distributions is considered *statistically significant*. The *level of significance* is fixed usually as either 0.05 or 0.01.

In order to conduct significance testing, an ML experiment with one model (configuration) needs to be repeated a certain number of times to simulate a population of results. In case of BoAW or neural network models, this is quite easy setting a different *random seed* in each run to generate new sequences of (pseudo-)random numbers for codebook generation or weight initialisation, respectively.

As a significance test, under the assumption that the experimental results are normally distributed and their variances are equal, the two-sample **student's t-test** can be used [294]. If the variances are different, the **Welch's t-test** can be employed [295]. The **z-test** is an alternative to student's t-test if the population variance is known [296]. In the experiments presented in Chapter 7, an *omnibus test* by D'Agostino and Pearson is considered to check for the normality of the obtained results [297].

Part III

EXPERIMENTS

5

The openXBOW Toolkit

This chapter introduces and describes in detail the openXBOW toolkit for multimodal BoW processing. openXBOW[1](cf. Figure 5.1) stands for ***open**-source **X**-modal ('crossmodal') **B**ag-**O**f-**W**ords toolkit* and has been developed in the context of the research work of the author. As the name implies, the main purpose is to evolve and disseminate a software tool to:

1. **Unify** approaches to generate BoW representations from both **numeric** and **symbolic** input data, **across** modalities.

2. **Integrate** all different **variants** of the BoW processing chain, as introduced in Chapter 3, into a single software.

3. **Standardise** a **toolchain** for BoW processing, promoting **transparent** and **reproducible research**.

4. Provide a **simple-to-use** software to generate **additional feature representations** (in addition to, e. g., functionals) and advance the state-of-the-art of standardised feature representations.

OpenXBOW is written in the Java programming language [298] and published as a repository on GitHub[2] under the *GNU General Public License v3* [3]. An accompanying article has been published in the *Journal of Machine Learning Research* in 2017 [29].

This chapter first introduces the main paradigms, the software architecture, and the computational performance of openXBOW in Section 5.1. Then, an overview of all features provided by the toolkit is given in Section 5.2. Finally, in Section 5.3, the impact of openXBOW is presented.

[1] 'open crossbow'

[2] `https://github.com/openXBOW/openXBOW`

[3] `https://www.gnu.org/licenses/gpl-3.0.html`

Figure 5.1: The logo of openXBOW.

5.1 Paradigms and Architecture

When developing openXBOW, the design decisions have been motivated by several paradigms, as introduced in the following.

5.1.1 Simplicity

All configurations of openXBOW, including the definition of input and output filenames and all options of the BoW approaches, are done through a **command-line interface**. This has the advantage that no additional configuration files are required and settings can be shared easily via e-mail, text messages, or websites. Moreover, only a few options are needed, in fact, only the name of the input and output files for the default configuration. A **tutorial**, explaining six different use cases is provided on the website of the repository[4].

As both the source code and an executable JAR (Java archive) file are available in the repository, no installation or compilation is required to use openXBOW. This is even possible across **multiple platforms**, such as *Linux*, *Microsoft Windows*, or *macOS*, i. e, on each platform, where a Java *runtime environment* is available.

Furthermore, **no external libraries** are linked to openXBOW and it uses solely the Java *standard library*.

5.1.2 Flexibility

As several standards for file formats have established in the ML research community, users of openXBOW should have the option to use them with openXBOW right away, without the requirement of converting files into a specific format. Therefore, openXBOW supports the following file formats:

- **ARFF** (attribute-relation file format), as used by the ML toolkit Weka [161].
- **CSV** (comma-separated values), as a very common and simple format. Besides the *comma*, also the *semicolon* is supported as a separator of the columns in the feature files (as used, e. g., by openSMILE [23]). The type of the separator can be inferred automatically for the input and chosen for the output.

[4] `https://github.com/openXBOW/openXBOW`

- **Libsvm** file format, as required by the open-source SVM toolkits LIBSVM [299] and LIBLINEAR [300]. This format is only supported for the output files as it is typically used by ML toolkits, not by feature extractors.

Furthermore, *labels* can be either included in the feature files or not and may be optionally propagated to the *crossmodal bag-of-words* (XBOW) output files. Also *pre-* and *post-processing* of the data can be done directly by the toolkit on request. Moreover, it is even possible to provide openXBOW with a LIBLINEAR model file and write the SVM predictions directly into a JSON (JAVASCRIPT object notation) file. This avoids the requirement of providing the corresponding software and documentation when deploying models based on openXBOW features.

5.1.3 Reproducibility

All routines where *random* numbers are used (mainly the initialisation of the clustering) are **seeded**, i. e., the output of these (pseudo-)random processes is always the same, when the program is run on the same files with the same command line options; this even applies when the toolkit is run on different platforms. Moreover, the **seed** that is used to generate the random numbers is adjustable. By this, it is possible to run openXBOW several times with the same BoW configuration to study the influence of random decisions for the model.

In order to ensure reproducibility across different **versions** of openXBOW, a **testing module** has been published as part of the tool, using the JUNIT framework[5] [301]. When running the automated test, it is checked whether certain test files provided in different formats through the repository are read correctly by the corresponding module of openXBOW. Then, in a *regression test*, the BoW-representations are generated for some default configurations and it is assessed whether the output is the same as expected and provided by previous versions of the toolkit. Finally, for a variety of further options, the processing chain is run and it is checked whether no errors are thrown by any module. Using this test, researchers modifying the source code can quickly receive some evidence if their code changes the behaviour of openXBOW or crashes the code.

5.1.4 Software architecture

An overview of the software architecture of openXBOW is shown in Figure 5.2. The classes that are most important to understand the basic workflow and their relations are illustrated in the figure, while minor components (*helper classes*) and *subclasses* of the *Codebook* class are not explicitly illustrated in order to improve the readability.

[5] `https://junit.org/junit5/`

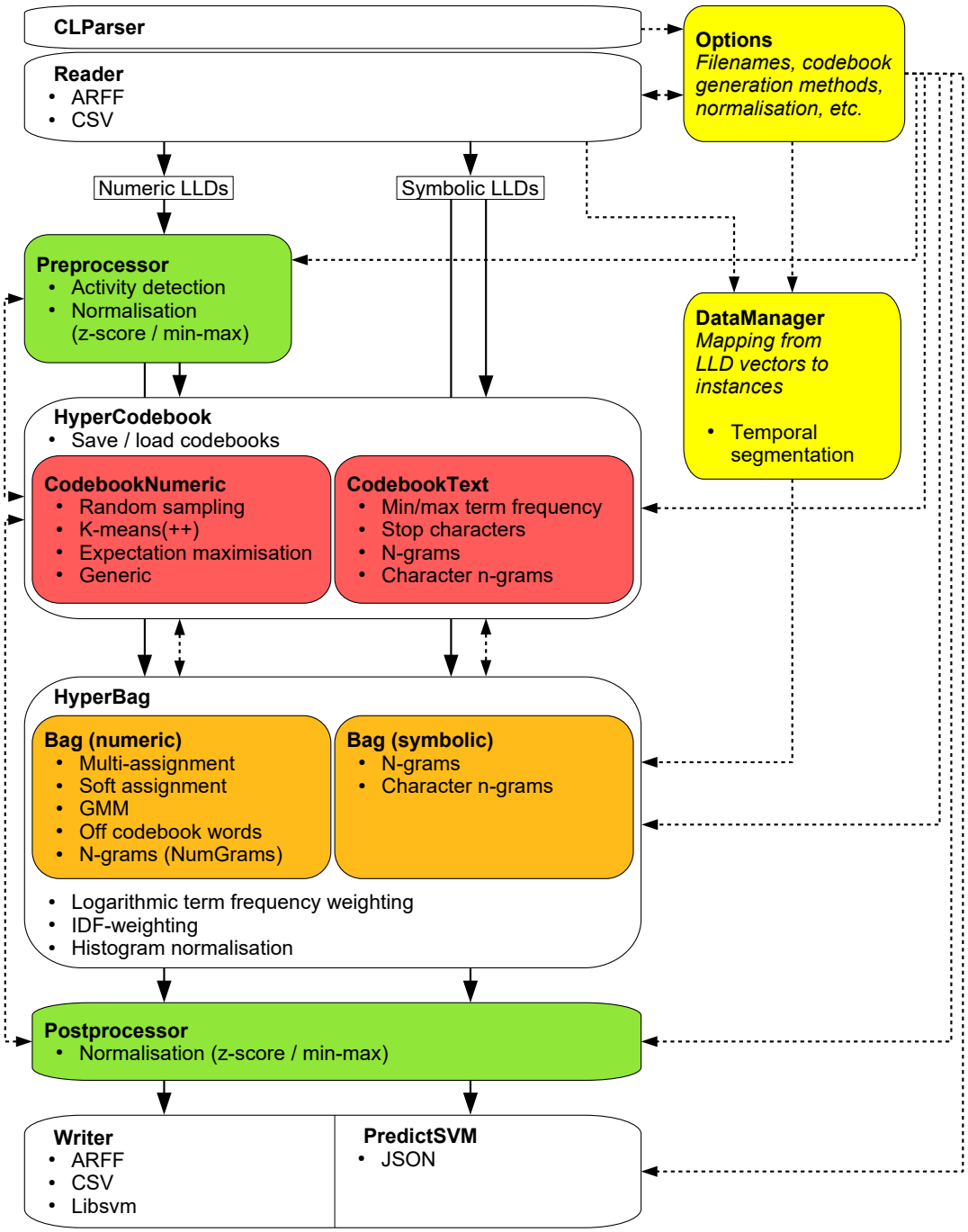

Figure 5.2: The software architecture of openXBOW. Solid lines represent the propagation of data through the components. Dashed lines represent exchange of information on configurations, normalisation parameters, or codewords. Arrows on two sides mean that information can be exchanged in both directions, depending on if an existing codebook is loaded or stored, respectively. Helper classes and subclasses of the Codebook class are not explicitly displayed.

A *command-line parser (CLParser)* interprets the user-specified command line arguments, adds default parameters and stores them in an *Options* object. The *Reader* component then loads the data file (and labels) and updates the *Options* with information that is evident from the data and not necessarily specified in the command line, e. g., the number, type (numeric/symbolic) and index of input features.

In the next step, a *DataManager* creates a mapping between the index of the LLD feature vectors and the instances (e. g., one audio file—as specified by the first column of the input file). In case a temporal segmentation is requested ('sequence labelling', cf. Section 2.2.1), the mapping from each LLD feature vector to each temporal block (over which a BoW-representation is generated) is already initialised. This has the advantage that LLDs that contribute to several output instances (one analysis block is one instance in the case of a sequence labelling task) need to be vector quantised only once—a process that is computationally intensive.

As a first optional processing step of the numeric LLDs, a *normalisation* (z-score 'standardisation'[6] or min-max normalisation) and/or an *activity detection* are performed. The latter tags all LLDs—based on either energy thresholding or VAD information from the LLDs—where the signal of interest (e. g., speech) is not present and which should not be taken into account for the BoW-representation. This is implemented in the *Preprocessor* class.

Next, a so-called *HyperCodebook* is initialised, which is a wrapper of all codebooks, i. e., a *symbolic* codebook and an arbitrary number of *numeric* codebooks (for different LLD subsets), as defined by the user. This class also provides methods to load and store all codebooks, together with the parameters for *pre-* and *post-processing*, the type of *assignment* (see below) and information on IDF weighting. This functionality is quite important as openXBOW needs to be called on test sets without re-training codebooks or re-estimating normalisation parameters ('online normalisation'), obviously. Details on the implemented methods for *codebook generation* are given in Section 5.2 in this chapter.

The class *HyperBag* wraps all methods for *numeric* and *symbolic word assignment*. Furthermore, the class is responsible for the (optional) *histogram normalisation*, *logarithmic term frequency weighting*, and *IDF weighting*. For the latter, the *document frequency* as a model parameter is stored/loaded using the *HyperCodebook* class. Also here, more details are given in Section 5.2.

A *Postprocessor* class implements two options to normalise the output XBOW feature vectors, analogously to the *Preprocessor*. Finally, with the *Writer* class, the representations can be stored in one of the three mentioned file formats, with some specific formatting options, depending on the file type (e. g., header line, instance names, user-defined separator; see Appendix A). Instead (or in addition to) outputting the features, they can directly be decoded by openXBOW and the

[6] as used by, e. g., Grzeszick et al. [203] ('whitening')

predictions (class labels and confidences) are written to JSON files through the *PredictSVM* class. For this, an SVM model file trained using the toolkit LIBLINEAR [300] must be provided.

To conclude this section on the software architecture, a short example is given to exemplify how to work with openXBOW. Consider a CSV file in the format shown in the following excerpt:

```
instance_name,nfeat_1,nfeat_2,nfeat_3,nfeat_4,symfeat,label
'instance_01',-0.5615,-2.0867,-0.5095,-1.8624,'',pos
'instance_01',-0.0661,-2.1741,-0.4108,-1.7599,'This',pos
'instance_01',-0.2243,-1.2169,-0.2017,-1.6096,'is',pos
'instance_01',-0.0797,-2.3057,-0.6106,-1.9149,'is',pos
'instance_01',-0.0319,-2.0945,-0.3948,-1.6333,'good',pos
'instance_01',-0.1467,-2.1875,-0.4088,-1.7196,'',pos
'instance_02', 0.5563,-0.6476, 0.0919,-0.6839,'I',pos
'instance_02', 0.7485,-0.7040, 0.1001,-0.5700,'really',pos
'instance_02', 0.7070,-0.8274,-0.1066,-0.5037,'like',pos
'instance_02', 1.5221,-0.9025,-0.0244,-0.3836,'that',pos
'instance_03', 0.1581,-0.7246, 0.4406, 0.7134,'Oh',neg
'instance_03', 0.3474,-0.5006, 0.5103, 0.5582,'no',neg
'instance_03', 0.2366,-0.6574, 0.4297, 1.0157,'',neg
'instance_04', 2.1883,-0.3156,-2.1238,-0.7530,'This',pos
'instance_04', 1.9796,-0.1564,-2.1679,-0.7507,'is',pos
...
```

The given file has 7 columns, where the first one specifies the name of the instance, the corresponding line (input feature vector) belongs to. For each unique instance name, a XBOW representation (VSM) is created. The 2nd to 5th column contain a numeric LLD vector, the 6th column natural language (text, i. e., symbolic features) and the last column contains the class label (either *pos* or *neg*, which must be unique for all rows of the same instance).

With the following call of openXBOW, a XBOW representation can be generated:

```
java -jar openXBOW.jar -i example.csv -o output.csv
```

This 'default' configuration generates a *numeric codebook* of 500 codewords by *random sampling* (see Section 5.2) and a *symbolic codebook* (dictionary) of all words found in the text. For the given file, the meaning of the columns is inferred automatically; in the case of, e. g., multiple labels or if the numeric LLDs should be split into subsets with dedicated codebooks, this can be specified with the `-attributes` command-line parameter.

If a different *codebook size* is requested, this can be set using the `-size` option, as shown in the following:

```
java -jar openXBOW.jar -i example.csv -o output.csv -size 3
-c kmeans++ -B codebook.txt -writeName
```

This call creates the following output (`output.csv`):

```
'instance_01';0.0;0.0;0.0;0.0;1.0;0.0;0.0;2.0;1.0;6.0;0.0;0.0;pos
'instance_02';1.0;0.0;1.0;1.0;0.0;1.0;0.0;0.0;0.0;0.0;4.0;0.0;pos
'instance_03';0.0;1.0;0.0;0.0;0.0;0.0;1.0;0.0;0.0;0.0;3.0;0.0;neg
'instance_04';0.0;0.0;0.0;0.0;1.0;0.0;0.0;1.0;0.0;0.0;0.0;2.0;pos
...
```

In order to ensure the readability of the output, a relatively small *codebook size* of 3 (`-size 3`) has been chosen[7]. Instead of the default *random sampling*, a *k-means++* clustering is chosen (`-c kmeans++`). The codebook is written to a file as well (`-B codebook.txt`); for creating the VSM on a *test set*, this codebook can be loaded using the option `-b codebook.txt` (lowercase 'b'). Moreover, the instance name is added to the output (`-writeName`); the labels are added by default (if present in the input).

This example gives only a rough impression of the usage of openXBOW and does not reflect many more advanced options. For a more comprehensive *tutorial*, the reader is referred to the repository[8], where six different usage scenarios are explained.

A full list of all command-line parameters is given in the manual of openXBOW, which is found in the Appendix A. Section 5.2 highlights some of the most important features.

5.1.5 Computational performance

The computational performance of openXBOW is respectable. The code was optimised to speed-up the XBOW generation process as much as possible. Nevertheless, in case of handling large data files, the platform is required to provide sufficient memory. However, except for the clustering process (see Subsection 5.2.1), data files can also be processed **incrementally** (option `-append`).

In the following, the processing speed is exemplified for two usage scenarios: *crossmodal time-continuous emotion recognition* and *sentiment analysis of tweets*. The experiments were conducted on a workstation with an **Intel Core i7-4770 (3.4 Ghz)** CPU, **16 GB** RAM, and **Windows 10** operating system with **Java Version 8, Update 121**.

For the first task, the *SEWA ('Automatic Sentiment Analysis in the Wild')* database was exploited [302]. The partitioning and evaluation setup from the *affect analysis task* of AVEC 2017 [18] was used. Detailed information on this challenge task is given in Section 6.5 in the next chapter. In short, the data consists of audio-visual recordings from 64 German subjects discussing an advertising spot through a video chat. The data, with a total duration of approximately 89 minutes, has been partitioned into a training (34 subjects), a development (14 subjects), and a test set

[7] This applies only to the *numeric* codebook, the size of the *symbolic* codebook is configured differently, see Appendix A

[8] `https://github.com/openXBOW/openXBOW`

(16 subjects). Besides the audio and video modalities, manual transcriptions of the speech are provided.

As numeric low-level features, the 65 acoustic LLDs from the ComParE feature set (see Section 2.3.1) and 49 *facial landmarks*[9] extracted from the videos using the Chehra Face Tracker [303] were employed. The numeric LLDs from acoustic and visual domains were fused (cf. LLD-level fusion in Figure 5.3) at a step size of 10 ms[10]. As symbolic input, the textual transcription was used. A numeric BoAW/BoVW representation with a codebook size of 1 000 was created using the default `random++` sampling (see Section 5.2); for the text domain, a symbolic codebook (dictionary) of 346 words was created based on the text data available in the training set. A XBOW output using overlapping blocks of 8 s, with a hop size of 100 ms, was computed, resulting in approximately 53 000 instance-level feature vectors of size 1 346 each.

The whole process on the mentioned machine took **263 s** for the training set (including the codebook generation) and **67 s** for the development and the test set altogether. Generating only a single XBOW feature vector for a chunk of 8 s duration took 0.57 s.

For the second usage scenario, the runtime of the 4^{th} *tutorial* in the openXBOW repository[11] was measured ('Sentiment analysis in tweets using Bag-of-Words'). Tweets are short messages of up to 280 characters shared by the users of the microblogging service *Twitter*[12]. Sentiment analysis [304] on Twitter is one of the most investigated tasks in NLP [305]. In the given example, a BoW representation is created for each of the 1 578 627 tweets in the *Twitter Sentiment Analysis Training Corpus* [306]. The dictionary is generated from the first 1 000 000 instances in the corpus, where a *minimum term frequency* of 2 000 is taken into account, which results in a dictionary of 1 875 terms. Generating the BoW CSV file for this whole set (1 M instances × 1 875 features) took only **118 s**.

5.2 Features

In the following, the techniques for *codebook generation* and *word assignment (encoding)* of *numeric* LLDs that have been implemented in openXBOW are explicated. This section is focussed on these functionalities as these are considered the most relevant ones for the topic of this thesis. Thus, all options concerning *symbolic* features and NLP are not further discussed at this point.

[9] Coordinates from defined points of reference in the face.
[10] The facial landmarks needed to be upsampled by a factor of 2.
[11] `https://github.com/openXBOW/openXBOW`
[12] `https://twitter.com/`

5.2.1 Codebook generation

Generally, either a single or several numeric codebooks can be configured in openXBOW. In case of several codebooks, the XBOW feature extraction process is run independently for each subset of LLDs. This is exemplified in Figure 5.3. In the top of the figure, the 'standard' processing chain is executed, where a single codebook is generated for the full LLD feature space ('LLD-level fusion'). In contrast, at the bottom of the figure, the LLD space is first split into a defined number of subsets, where codebooks and VSMs are generated for independently. These VSMs are then concatenated to form the final VSM ('bag-level fusion'). In the given example, for three acoustic feature groups (prosodic, MFCCs, formant frequencies), a separate codebook/VSM is built. However, it is also possible to realise the split between different modalities (e. g., acoustic vs visual), to perform VQ on a single LLD, or to use overlapping subsets.

For codebook generation, the techniques given in the following are available in openXBOW. It must be pointed out that all methods are using an **adjustable random seed**, so that the generation process can be both reproduced and randomised.

Random sampling The simplest method to generate a codebook is a random sampling (cf. Chapter 3, option `-c random`). From all LLDs in the given dataset (usually, the training set), a number (specified by `-size`, 500 by default) of LLDs is selected to form the codebook. It is ensured that there are not two equal codewords (Euclidean distance is 0), but there are no further conditions.

To increase the *diversity* and avoid selecting codewords that are too similar in terms of the Euclidean distance, an advanced version is integrated with *random sampling ++* (`-c random++`). This method reflects the cluster initialisation of the *k-means++* algorithm [216], as already introduced in Section 3.3.1.

K-means As a very common clustering method, both the *k-means* (`-c kmeans`) and the *k-means++* (`-c kmeans++`) algorithm (Lloyd's algorithm) are available, as defined in Section 3.3.1.

EM EM clustering (cf. Section 3.3.1) is available as an alternative for clustering, where the feature space is modelled as a GMM [224] (`-c em` / `-c em++` / `-c em-kmeans` / `-c em-kmeans++`). More specifically, the feature space is modelled as a probability distribution defined as a mixture of k Gaussian distributions, i. e.,

$$p(x) = \sum_{i=1}^{k} \pi_i \mathcal{N}(x|\mu_i, \Sigma_i), \tag{5.1}$$

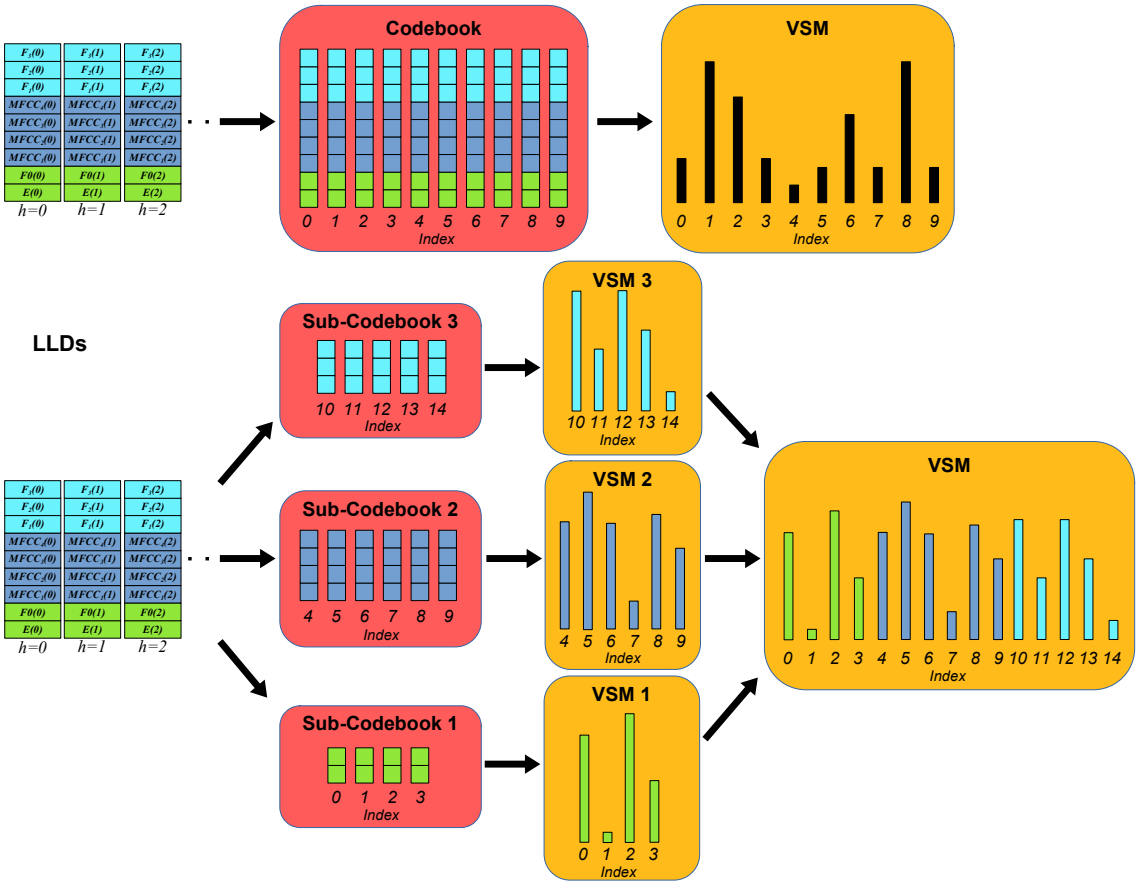

Figure 5.3: Visualisation of the LLD-level fusion (top) compared to the bag-level (VSM-level/histogram-level) fusion (bottom). In **LLD-level fusion**, a unique codebook is generated for the whole LLD feature space. In **bag-level fusion**, a codebook is generated for each defined subset of the LLD feature space first ('sub-codebooks'). The VQ is performed individually for each subset and VSMs, i. e., BoW representations, are created separately for each sub-codebook. In this example, the acoustic LLDs are grouped into three classes: *prosodic (green)*, *MFCCs (blue)*, and *formants (cyan)*. Different codebook sizes are employed for each group. The resulting VSMs are concatenated to form the final VSM.

where π_i are the *mixture weights* and $\mathcal{N}(x|\mu_i, \Sigma_i)$ is the multivariate Gaussian density [224] with *mean vector* μ_i and *covariance matrix* Σ_i.

Along the lines of Section 3.3.1, the centroids can be initialised with either *random sampling* or *k-means* (either with or without the '++' initialisation). The mixture model is then updated iteratively with a series of *expectation* and *maximisation* steps, according to Bishop [224]. In order to speed-up the clustering process and as it is a common way in speech processing [307], the covariance matrices are assumed to be *diagonal*. As the EM algorithm returns not only the cluster centroids

but also mixture weights and covariances, more advanced *assignment* techniques can be employed (see Subsection 5.2.2).

Random sampling from a Gaussian probability density function As a first method to create **data-independent codebooks**, a random sampling from a *univariate Gaussian* probability density function (PDF) with mean 0 and variance 1, i. e., $\mathcal{N}(x|0,1)$, as used by Pandit et al. [308], is available (`-c pdf`). Each component of a codeword is selected independently from the others. To use this method in a meaningful way, a *z-score normalisation* ('standardisation') of the LLDs is strongly advised, in order to fit the distribution of each input feature to the given Gaussian PDF.

Generic Finally, a *'generic'* method to create codebooks is implemented (`-c generic`). Also this method does not take into account the actual distribution of the data and is thus assumed to generate codebooks that are data-independent. Given the number of LLDs n, a codebook of the fixed size 2^n is generated, where the entry j of the codeword with index i is given by

$$b_{i,j} = (-1)^{\lfloor \frac{i-1}{2^{n-j}} \rfloor + 1}, \quad i = 1, 2, \ldots, 2^n, \quad j = 1, 2, \ldots, n. \tag{5.2}$$

As an example, for 3 LLDs, the following codebook is created:

$$\begin{array}{ccc} -1 & -1 & -1 \\ -1 & -1 & +1 \\ -1 & +1 & -1 \\ -1 & +1 & +1 \\ +1 & -1 & -1 \\ +1 & -1 & +1 \\ +1 & +1 & -1 \\ +1 & +1 & +1 \end{array}$$

In analogy with the previous method, a z-score normalisation of the input is appropriate (`-standardizeInput`). Instead of the fixed offset of 1, a configurable number can be chosen (`-gen`). In a generic codebook, the codewords are uniformly distributed over the feature space and each codeword has n nearest neighbours with the same distance.

For each of the *random sampling* and *clustering*-based methods—i. e., all codebook generation methods except for the *data-independent* ones, the following three refinement options are available.

Supervised If openXBOW is provided with class labels, the codebook generation can be done in a supervised way, by creating codebooks from the LLDs

of each class separately, first, and then concatenating the class-specific codebooks (`-supervised`). This approach has been employed by Yeh and Yang [24] and Plinge et al. [202].

Pre-selection In order to speed-up the clustering process—especially on large data and/or feature sets—it can be performed not on the whole training corpus, but only on a subset of the input LLDs (`-numTrain`). This kind of a 'random downsampling' has been used before, e. g., by Sivic and Zisserman [170].

Reduction Finally, it might be beneficial in terms of the robustness to reduce the set of codewords by fusing similar ones. This can be done with the option `-reduce`, where similar codewords are identified by evaluating the pairwise PCC (see Section 4.5) and then fusing them by taking their mean.

5.2.2 Encoding

In the following, the openXBOW options for word assignment are presented.

Vector quantisation (hard assignment) The 'standard method' for assignment is the VQ of the LLD vectors, i. e., a *nearest neighbour* assignment, as discussed in Section 3.3.2 (Equation 3.13). As an enhancement, **multi-word assignment** is available, where each LLD vector can be assigned to the set of N_A closest codewords (`-a` N_A) [25].

Furthermore, a method to exclude **off-LLDs** has been implemented. With this option (`-off` with a threshold to be specified) LLDs where the closest codeword has an Euclidean distance above a threshold are discarded from the VSM.

Soft assignment In contrast to the so far described *hard assignment* methods, where the histogram counter is only incremented by 1, the following two *soft assignment* techniques can be used in openXBOW.

Firstly, a *Gaussian encoding* as employed by Pancoast and Akbacak [199] and defined in Equation 3.17 is available (`-gaussian`).

Secondly, a *GMM-like encoding* has been implemented. This method requires that the codebook is generated using an *EM-clustering* (see Subsection 5.2.1) as not only the cluster *centroids* are necessary but also the *mixture weights* and *covariance matrices*. Referring to Equation 3.13, the *increment* for each codeword i is the *a posteriori* probability

$$a(b_i, D_j) = \frac{\pi_i \mathcal{N}(D_j|\mu_i, \Sigma_i)}{\sum_{i^*=1}^{L_B} \pi_i^* \mathcal{N}(D_j|\mu_i^*, \Sigma_i^*)}. \tag{5.3}$$

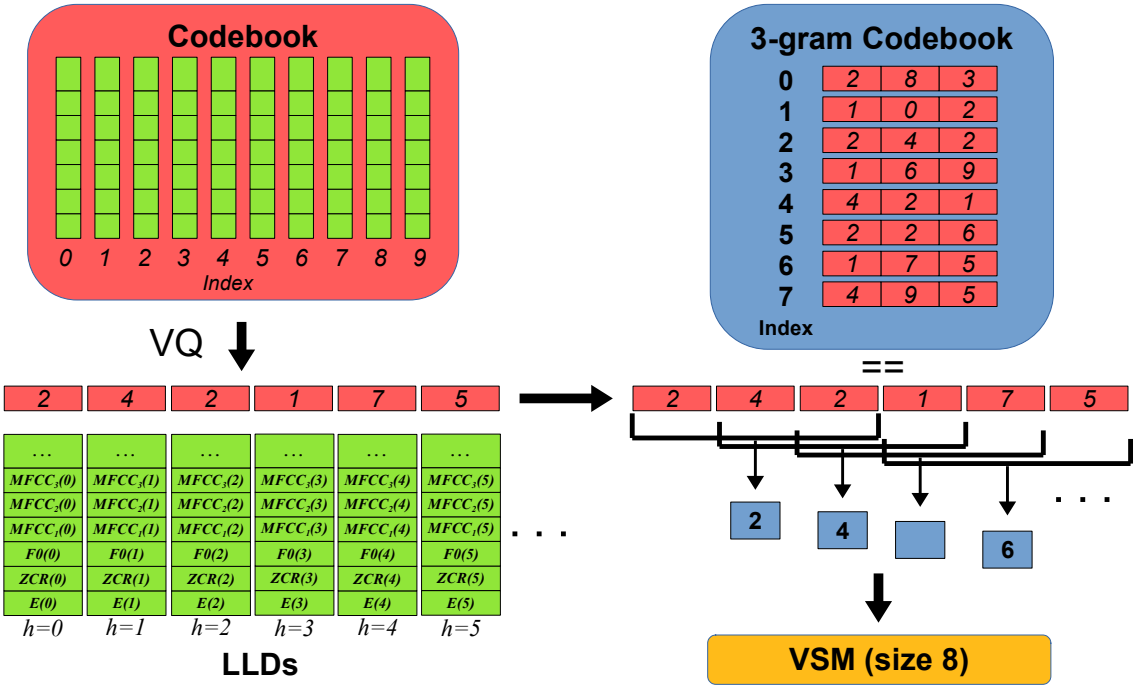

Figure 5.4: Visualisation of the *numeric n-gram* option of openXBOW. A codebook of size 10 and a 3-gram codebook of size 8 are used. A standard VQ is performed for the LLDs. Then, the n-gram sub-sequences of codebook indexes are matched with the templates in the numeric n-gram codebook.

Thus, each LLD vector increases the counter of each codeword. The GMM-like encoding can be done by using either uniformly distributed mixture weights (`-gmm 1`) or the 'correct' ones obtained from the EM-clustering (`-gmm 2`).

Numeric n-grams As a further alternative encoding method, the *n-gram* approach, well-known for symbolic features and introduced in Section 3.1.3.3, has been adopted for numeric features following the work by Pancoast and Akbacak [200]. Figure 5.4 illustrates the process of the numeric n-gram generation in openXBOW. In the given example, *3-grams* (`-trigram`) are generated, but also *2-grams* are available (`-bigram`), as well as combinations of these and the 'standard' XBOW features. As a first step, a *codebook* is generated from the training set, using one of the previously introduced methods. In this example, a codebook of size 10, indexed from 0 to 9, is employed. Then, a hard VQ is performed in the 'usual' way. Now, a dedicated **n-gram codebook** is trained, by counting the frequencies of n-gram ($n = 3$ in the example) sequences of codebook indexes over the whole training set. A user-defined number of n-grams with the highest number of occurrences in the training set is considered as the n-gram codebook. In the example from Figure 5.4, eight

3-grams (indexed from 0 to 7) are taken into account. Finally, a matching (==) between n-grams in each instance and the n-gram codewords is performed. Only exact matches are taken into account. In the given example, the 3-grams (2,4,2), (4,2,1), and (1,7,5) occur also in the codebook and are counted in the final histogram. The 3-gram (2,1,7), however, does not occur and is disregarded in the histogram. This corresponds to the *off-LLDs* introduced before, but is used by default for numeric n-grams.

Also the *multi-word assignment* can be used for the VQ prior to the numeric n-gram method, but not the *soft encoding* methods. Exactly the same *weighting* and *post-processing* techniques as for the standard XBOW approach can be applied to the numeric n-gram histograms.

Weighting For *weighting* of the XBOW-histograms, three common methods have been implemented that are performed *prior* to a potential (z-score or min-max) normalisation of the output feature vectors and can be employed either in combination or exclusively:

- Logarithmic term frequency weighting (`-log`)
- Term frequency-inverse document frequency weighting (`-idf`)
- A histogram normalisation (`-norm`) in one of the three following variants:

  1. Divide the term frequencies by the number of LLDs in the instance (`-norm 1`),
  2. L1-normalisation (absolute length equals 1, `-norm 2`), or
  3. L2-normalisation (Euclidean length equals 1, `-norm 3`).

  For numerical reasons, in each of the normalisation options, the resulting term frequencies are multiplied by the codebook size.

Details on these methods are found in Section 3.1.3.2. openXBOW stores the options `-log` and `-idf`, along with the corresponding *document frequencies* of the terms in the *codebook file*, while the normalisation options can also be used for either training or testing only, to create more flexibility.

5.3 Impact

To conclude this chapter on the openXBOW toolkit, its impact in the research community is presented, with reference to some key statistics and usage examples.

5.3.1 Key statistics

By the beginning of May 2021[13], the openXBOW repository[14]

- has received **71 stars**, and
- has been **forked 16 times**.

The paper on the toolkit [29], published in 2017, has received **111 citations**, according to Google Scholar[15].

The social network ResearchGate[16] counts **116 citations**, **360 reads** on the platform, including **100 full-text reads**. A **research interest score** of **66.1** is achieved, which is higher than the score of **99 %** of the 'research items' published in 2017 and registered on ResearchGate.

5.3.2 Usage examples

In addition to the research work presented in this thesis, mainly in Chapters 6 and 7, some further examples of research works, where openXBOW has been utilised by the authors, are presented in the following. In some of the works, interesting combinations and advancements are proposed, partially with the involvement of the author of this thesis.

Han et al. proposed to use 'bags in bag' or 'bag-of-context-aware-words' on the emotion dataset RECOLA, outperforming the 'basic' BoAW approach in Section 6.1 [309, 310]. This approach is similar to the hierarchical two-level BoAW, as employed by Su et al. [189] and Yeh et al. [190].

Amiriparian et al. introduced 'bag-of-deep-features' [311], using intermediate layer outputs from pre-trained neural networks for image classification as input for BoAW generation. These features are referred to as DeepSpectrum features [272, 312], which are found to be robust to noise present in the audio. In a similar way, Gosztolya et al. [313, 314] extended BoAW using *neural network posteriors*, where the neural network is trained on, e. g., a dataset for phoneme modelling [314].

Qian et al. employ wavelet features (see Section 2.1.3) to obtain so-called 'bag-of-wavelet-features' for audio classification [57, 315]. Moreover, 'bag-of-behaviour-words' have been proposed by the same first author for the task of depression monitoring, using LLDs from activity sensors [316]. Pandit et al. proposed the mentioned way to generate data-independent codebooks by sampling from a Gaussian PDF to be used for *cross-corpus* emotion recognition in speech [308].

[13] Statistics retrieved on May 4, 2021.
[14] `https://github.com/openXBOW/openXBOW`
[15] `https://scholar.google.de/citations?user=bTLclOYAAAAJ`
[16] `https://www.researchgate.net/`

Elbarougy et al. applied feature selection on features generated with openXBOW and an LSTM-RNN for audio-visual emotion recognition[317]. Syed et al. recently showed that BoAW representations of both hand-crafted and deep CNN-based LLDs outperform the *Fisher vector* representation [318].

Due to the competitive results, openXBOW has established as a toolkit to generate *baseline* acoustic feature sets for various tasks: Amiriparian et al. employed it as a baseline for the classification of heart sounds [319] and for various further paralinguistic tasks [320]. Moreover, it was used as a baseline for classification of children's vocalisations by Zhang et al., outperforming functionals (ComParE and eGeMAPS) in terms of the UAR [321].

BoAW features were used to create a baseline for a pain database by Ren et al. [322], and by Cummins et al. for emotion recognition in speech [323] and detection of upper respiratory tract infections from the subjects' speech [5]. BoAW has shown to outperform traditional acoustic features (i. e., functionals) for some sub-tasks in a database of emotional dog barks [324], namely, *aggression*, *despair*, and *fear*. Recently, openXBOW was employed to generate baseline features for the task of detecting *gibbons' calls* by Tzirakis et al. [325] and classifying *sleep quality*, *fatigue*, and *anxiety* in the speech of patients suffering from *COVID-19* by Qian et al. [326].

Furthermore, openXBOW is an integral part of the MixedEmotions open-source toolbox for multimodal emotion analysis [327]. It is also part of the *automatic emotion recognition* module of EmotAsS [328], a speech-driven system that assists cognitively impaired individuals at their workplace.

Most importantly, the demonstrated success of openXBOW has been largely triggered by its usage for the *baseline approaches* of the following scientific **challenges**:

- the ComParE editions from 2017 to 2020 ([11, 12, 13, 116]),
- the AVEC editions from 2017 to 2019 ([17, 18, 19]), and
- the ICMI EAT challenge in 2018 [84].

All of these events have had a big impact on the research in the field of CA and/or *affective computing*. For the ComParE challenge, **BoAW** has been **three times** (i. e., for three sub-challenges) the **best approach** out of three or more approaches provided to the participants. In two of these cases, the official challenge **baseline**, which was given by a late fusion of a BoAW-based model and two or three other models, was not surpassed by any of the challenge participants (see Section 6.4). For the AVEC challenge baseline in 2018, **BoAW features** with an LSTM-RNN-based regressor **outperformed the two other acoustic representations** across all tasks and partitions. In 2019, BoAW was the best out of three acoustic representations in **7 out of 12 evaluations**—across two dimensions (*arousal*/*valence*),

three cultures (languages), and two partitions (see Section 6.5). In the baseline of the ICMI EAT challenge, features from openXBOW with an SVM **outperformed the E2E-baseline** across audio and video modalities and for all three sub-tasks (see Section 6.7).

The features utilised for training of the baseline models, i. e., BoAW (and BoVW) computed by openXBOW, have been provided to the participants, to enable them to re-use them and work on ML models only. Nevertheless, as the scripts for feature extraction were issued as well, it was also possible to optimise the configuration and/or input features of openXBOW only. As an example, in the context of the EAT challenge, Guo et al. found that BoAW features outperform functionals, but a fusion of both improves the performance; the results were even better when fused with deep features extracted from *cochleagrams* with pre-trained deep image networks [329]. Gosztolya made the similar discovery that BoAW features are complementary to other feature types and can improve the robustness of the model [330]. Vlasenko et al. emphasised on the importance of codebook training and proposed to train it on all partitions [331]. Concerning the usage of codebooks across corpora, Vetráb and Gosztolya found that a codebook trained on emotional speech of a language other than the target one can even improve the results [332]. Features from openXBOW have been re-used by various contributions to ComParE and AVEC [333, 334, 335].

6

Case Studies

As announced in the introduction, the experimental evaluation is subdivided into two chapters: This chapter reports on *case studies* that have already been published by the author and focus on certain tasks and datasets in CA, where the BoAW approach can be employed, and on certain aspects or enhancements of the approach. The second part, which is presented in Chapter 7, offers a systematic cross-corpus analysis of the BoAW method, focussing on research questions where no definite conclusion could be made, based on the case studies.

This chapter on case studies is subdivided into the following sections, each one highlighting a certain publication or a related series thereof: In the first section, the author's work on BoAW for prediction of emotion in speech is reported. This publication has been the first one employing the approach on a time-continuous emotion recognition task. Then, in Section 6.2, the openXBOW toolkit is applied to a preliminary version of the *Munich-Passau snore sound corpus* (MPSSC), a public dataset for classification of human *snore sounds* w. r. t. their location of vibration in the throat. In this work, several LLD groups and fusion types are compared to each other. In Section 6.3, the BoAW method is applied to the task of classifying audio effects present in short recorded chunks. This work focusses on the relevance of acoustic LLDs for this task and the suitability of normalisation strategies. Sections 6.4, 6.5, and 6.7 summarise the work on the challenge series of ComParE and AVEC, and the EAT challenge, respectively. In all of these events, where the author has been involved and which partially have been co-organised by the author, openXBOW has been utilised as a toolkit to generate feature sets used in the baseline provided to the participants. In Section 6.6, a supplementary work by the author on the emotion recognition task of AVEC is presented, with a focus on the suitability of RNN input representations.

6.1 Time-continuous Emotion Recognition in Speech

The first peer-reviewed work by the author employing openXBOW was published at the *Interspeech* conference in 2016: 'At the Border of Acoustics and Linguistics: Bag-of-Audio-Words for the Recognition of Emotions in Speech' [112]. After Pokorny et al. [208] had showcased their implementation of BoAW for a binary emotion classification task (*negative* vs *positive*), the motivation for this work was to apply the method to an emotion recognition task that is *continuous* in *time* and *value*, i. e., time-continuous *regression*.

The multimodal database **RECOLA** *(REmote COLlaborative and Affective interactions)* was employed throughout the experiments [336]. RECOLA consists of audio, video, and physiological signals recorded during a dyadic conversation of French subjects. A *gold standard* for the emotion is provided in terms of the two-continuously valued dimensions **arousal** and **valence** following the *circumplex model* by Russell [89]. *Arousal* describes the level of the 'activation' of the human nervous system, while *valence* describes whether the emotion is a positive or a negative one. Each dimension is typically valued in the range of $[-1, +1]$. For example, *happiness* is an emotion of high arousal and positive valence, *anger* has a high arousal, but a negative valence, and *sadness* has both negative arousal and valence. The gold standard for each dimension was generated by the authors of the database by fusing the time-continuous ratings from six annotators using an *evaluator weighted estimator* (EWE) [1]. Labels are provided for each time step of 40 ms (25 Hz). RECOLA was used as a benchmark for the AVEC editions in 2015 [337], 2016 [338], and 2018 [19]. For the discussed work, however, only the audio modality is taken into account. In total, recordings from 46 subjects were given, with a duration of 5 min per subject, i e., a total duration of 3 h and 50 min.

6.1.1 Acoustic LLDs

As implied in Section 2.1, MFCCs are not particularly suited to capture affective (or more generally, paralinguistic) information, given that prosody (e. g., F0) is partially suppressed. Nevertheless, in practice they have proven to be even more meaningful for emotion recognition in spontaneous speech than F0 or microprosody [339]. Moreover, as MFCCs have established as LLDs to be employed in the context of GMMs, they closely follow a normal distribution [340, 341], and they have a low dimensionality, there is some justification to consider them also suitable for VQ using clustered or sampled codebooks.

Motivated by this, only **MFCCs 1–12 and the logarithmic short-time energy** are used as LLDs in this work. They have been extracted by openSMILE,

with a *Hamming* window, a frame length of 25 ms, and a hop size of 10 ms, after performing a pre-emphasis with $k = 0.97$.

6.1.2 BoAW

LLD vectors are **z-score normalised** in an *on-line* strategy, i. e., estimating normalisation parameters from the respective training partition (see below) only. Codebooks are generated using the `random++` option of openXBOW, as `kmeans++` provided only a small improvement in previous experiments. Both the **codebook size** (C_S) and the **number of assignments** (N_A) are optimised in the experiments. A **logarithmic term frequency weighting** is applied to the resulting term frequency histograms.

As instance-level features are required for each time step of 40 ms ('sequence labelling'), a temporal segmentation is done by openXBOW providing a 'time-continuous' BoAW representation. The amount of **context**, i. e., the **block size** (B_S) or the amount of frames over which the bag is created, is optimised considering the following range: [4 s, 6 s, 8 s, 10 s, 12 s].

6.1.3 Regressor

In initial experiments, the performances of SVM regressors with different kernels were compared to each other. As found also by Pancoast and Akbacak [26], *Gaussian* and *polynomial* kernels did not outperform *linear* kernels. In contrast to their work, also an *histogram intersection kernel* [245] did not perform any better.

Thus, an SVM regressor with a **linear kernel** is trained for each BoAW configuration using the toolkit Liblinear [300]. The *default* solver type for regression tasks is chosen, i. e., an L2-regularised L2-loss solving the dual formation of the SVM's quadratic optimisation problem; a bias of 1 is added to the feature space. The **complexity hyperparameter** C is optimised in the range of $[1e{-}5, 2e{-}5, 5e{-}5, 1e{-}4, \ldots, 1e0]$, i. e., with a step size close to a logarithmic scale.

6.1.4 Experimental setup

The RECOLA corpus is partitioned into 3 disjoint (speaker-independent) and gender-balanced partitions: training set (16 subjects), development set (15 subjects), and test set (15 subjects). It must be noted that, due to legal reasons, this experimental dataset is larger than the dataset released for AVEC [337], but benchmarks using exactly the same set exist as well, e. g., [228, 309]. The CCC (see Section 4.5.3 is used as the metric for evaluation throughout.

To speed-up the training process, BoAW instances are created only for each step of 800 ms on the training set, which seemed not to deteriorate the performance due to the high correlation between subsequent time steps. As the gold standard—due to

C_S	N_A 1	2	5	10	20	50	100	200	500	1 000
10	.750|.358	.716|.332	.715|.222							
20	.751|.355	.750|.353	.744|.314	.739|.319						
50	.776|.447	.773|.463	.775|.410	.782|.393	.765|.425					
100	.771|.469	.777|.477	.786|.477	.784|.440	.784|.422	.768|.382				
200	**.766|.474**	.774|.502	.779|.491	.785|.458	.786|.431	.782|.388	.769|.399			
500	.761|.480	.760|.477	.779|.519	.787|.518	.790|.512	.788|.466	.789|.442	.784|.383		
1 k	.763|.444	.760|.471	.777|.501	.783|.522	**.789|.539**	.789|.509	.787|.490	.788|.462	.785|.402	
2 k	.746|.459	.752|.459	.770|.494	.779|.505	.783|.528	.787|.541	.790|.530	.790|.515	.788|.449	.789|.406
5 k	.742|.423	.746|.423	.760|.482	.768|.493	.772|.504	.785|.525	.791|.540	**.793|.543**	.792|.514	.791|.491
10 k	.747|.373	.750|.373	.761|.484	.761|.484	.764|.494	.780|.515	.787|.522	.790|.532	.791|.520	.791|.509

Table 6.1: CCC (arousal|valence) for the given codebook size (C_S) and number of assignments (N_A) on the development set of the RECOLA database. The SVM complexity C and the block size B_S are optimised as denoted in the text. A constant delay D of 3.2 s is taken into account to compensate the reaction time of the raters. Three pairs of C_S and N_A are selected for further optimisation (printed in **bold**).

the reaction time of the human raters—is not perfectly aligned with the affect present in the recordings, a **delay compensation** needs to be taken into account [342]. This is especially necessary for *static* ML approaches such as, e. g., SVM, in contrast to *dynamic* approaches such as, e. g., RNNs, which are capable of modelling sequences. The delay compensation is realised by shifting each gold standard sequence backward in time before model training and shifting model predictions forward in time for evaluation on development and test sets. The optimisation is performed on a grid between 0 s and 8 s, with a step size of 0.4 s. Evaluation is done at the original sample rate of 25 Hz (40 ms step size).

The hyperparameters (D, B_S, C_S, N_A, & C) are optimised to obtain the best performance (in terms of the CCC) on the development set. For the final evaluation on the test set, the SVM regressor is trained again on fused training and development sets. The same (supervised) *post-processing* chain as in the work by Trigeorgis et al. [228] is applied to the predictions, in order to account for a potential mismatch in terms of scale and bias.

6.1.5 Results

Given the 5-dimensional hyperparameter space, the optimisation is done iteratively in two steps. Furthermore, results obtained for a fusion of BoAW and functionals are reported.

6.1.5.1 Optimisation of codebook size and number of assignments

Initial experiments have shown that a delay $D = 3.2\,s$ and a block size between $B_S = 6\,s$ and $B_S = 8\,s$ provide the highest CCC on the development set. Optimising only the block size between these two values and the complexity C as defined above, an exhaustive evaluation for different configurations of codebook size and number of assignments is performed as presented in Table 6.1.

Dimension	N_A	C_S	D [s]	B_S [s]	C	CCC Devel	CCC Test
Arousal	1	200	4.0	8.0	10^{-3}	.768	.716
	20	1 000	3.6	8.0	10^{-4}	.789	.738
	200	5 000	3.2	6.0	10^{-5}	**.793**	**.753**
Valence	1	200	4.8	12.0	10^{-2}	.490	.417
	20	1 000	4.4	10.0	10^{-3}	.550	**.430**
	200	5 000	5.2	12.0	10^{-1}	**.558**	.378

Table 6.2: Optimised hyperparameters and results (in terms of CCC) for the development set (Devel) and the test set (Test) for preselected combinations of codebook size (C_S) and number of assignments (N_A). Delay (D), block size (B_S) and SVM complexity (C) are optimised on the development set for each of the six experiments.

On the first view, there is some evidence that there is a dependency between the optimum settings for C_S and N_A, respectively, while at the same time, an improved performance can be observed for larger values of C_S and N_A (provided a suitable ratio); this applies especially for *valence*. For the next optimisation steps, three combinations of C_S and N_A are selected and further evaluated:

1. $C_S = 200$, $N_A = 1$, as a suitable single-assignment configuration,
2. $C_S = 1\,000$, $N_A = 20$, as a configuration reaching almost the best results with a comparably low computational complexity, and
3. $C_S = 5\,000$, $N_A = 200$, as the best overall configuration.

6.1.5.2 Optimisation of delay and block size

Table 6.2 shows the results for the three selected combinations and each affective dimension after optimisation of delay D, block size B_S, and SVM complexity C.

The results show that the prediction of *valence* benefits from a larger block size (10 s–12 s) compared to *arousal* (6 s–8 s). This can be seen as some evidence that the subjects' *arousal* (or at least its annotation) is changing more rapidly over time than their *valence*. The larger optimum complexity for *valence* can be regarded as an indicator that this dimension is the more complex one to train a model for. This hypothesis is further supported by the fact that a larger delay compensation seems to be beneficial, representing a more challenging decision process by the raters.

Figure 6.1 exhibits more insights into the optimal ratios of *delay* and *block size*. This figure shows the CCCs obtained on the development set for five different block sizes over the optimisation range of the delay; a fixed combination of $C_S = 1\,000$ and $N_A = 20$ was selected. The plots indicate that a proper configuration of delay and block size is more critical for *valence* than for *arousal*.

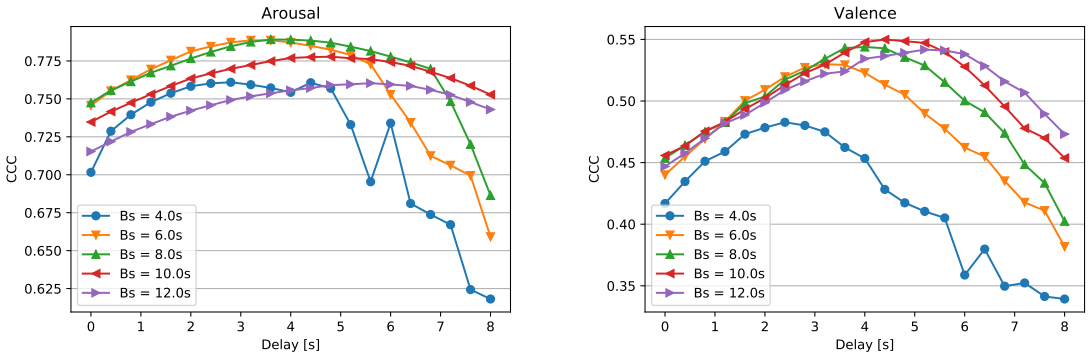

Figure 6.1: Performances for arousal (left) and valence (right) on the development set of RECOLA with different delays and block sizes (B_S); $N_A = 20$, $C_S = 1\,000$, C optimised for each configuration.

Dimension	D [s]	B_S [s]	CCC Devel	CCC Test
Arousal	4.0	8.0	.790	.720
Valence	4.0	10.0	.459	.402

Table 6.3: Emotion recognition results on RECOLA using functionals (mean + standard deviation). Reported are the CCCs for arousal and valence on the development set (Devel) and the test set (Test) with delays (D) and block sizes (B_S) optimised on Devel; C is also optimised on Devel for each configuration.

6.1.5.3 Fusion with functionals

Finally, an evaluation for a fusion of BoAW and functionals is performed. As functionals, **mean** and **standard deviation** of the 13 LLDs are computed. First, to report a *baseline*, the results using only the functionals are reported. For delay, block size, and SVM complexity, exactly the same optimisation as for the BoAW is performed; Table 6.3 shows the corresponding results.

While the results in terms of CCC for functionals are slightly worse than the results with BoAW, an improvement can be reached for some configurations by an early fusion of them, as shown in Table 6.4. As the combination ($N_A = 200, C_S = 5\,000$) results in a feature vector of a quite large dimensionality (i. e., 5 000) compared to the 26-dimensional functionals-based feature vector and the performance is not much better, only the two preselected BoAW configurations of lower dimensionality are taken into account for the fusion. For delay and block size, the optima from Table 6.3 are considered. To mitigate the effects of potential deviations of feature ranges, results are computed with and without a *z-score normalisation* of the fused feature vectors in Table 6.4.

Dimension	N_A	C_S	B_S [s]	z-score norm.	CCC Devel	CCC Test
Arousal	1	200	8.0	no	**.799**	**.738**
	20	1000	8.0	no	.677	.511
	1	200	8.0	yes	.796	.728
	20	1000	8.0	yes	.535	.384
Valence	1	200	10.0	no	.518	.457
	20	1000	10.0	no	.309	.234
	1	200	10.0	yes	**.521**	**.465**
	20	1000	10.0	yes	.245	.196

Table 6.4: Emotion recognition results on RECOLA using a early fusion of functionals and BoAW. Reported are the CCCs for arousal and valence on the development set (Devel) and the test set (Test). The optimum delay $D = 4.0\,s$ from the results with functionals and the optimum block size (B_S) for each dimension are considered; C is optimised on Devel for each configuration. Results are reported with and without z-score normalisation of the fused feature vector.

However, the normalisation of the fusion leads to an improvement mainly for *valence* on the test set. Surprisingly, fusing the BoAW with the smallest codebook size (and single-assignment, i. e., $N_A = 1$, $C_S = 100$) improves the model, but fusing the larger feature vector ($N_A = 20, C_S = 1\,000$) impairs the performance drastically, independent from the normalisation. A possible reason for this might be the meaningful deviation of the feature space dimensions between functionals and BoAW for this configuration.

6.1.5.4 Summary

In summary, it was found that **BoAW representations outperform functionals**, that they are partially **complementary**, and that **multi-assignment**, by trend, **improves** the model (especially for *valence*). More precisely, the configurations for **codebook size** and **number of assignments** need to **follow a certain ratio**; **small values** for both hyperparameters are required when **fusing with functionals**. Generally, it can be observed that hyperparameters need to be tuned more carefully for *valence* than for *arousal*.

Statistical significance ($p < 0.05$) w. r. t. *Fisher's z-transformation* [343] (a hypothesis test for correlation coefficients), between functionals and BoAW is obtained for all results of **valence**, except for the test set result with the largest codebook size. In contrast, a *significant* improvement for *arousal* is only observed for the BoAW features with the largest codebook size.

Table 6.5 shows a comparison of the presented BoAW models and the fusion models performing best for each dimension with other approaches evaluated on RECOLA. The audio-only models by the winners of AVEC 2015 [344] are signifi-

Model	Reference	CCC			
		Arousal		Valence	
		Devel	Test	Devel	Test
Large acoustic LLD set, BLSTM-RNN	[344]	**.800**		.398	
CNN + BLSTM-RNN (end-to-end)	[228]	.741	.686	.325	.261
BoAW, SVM	Table 6.2	.793	**.753**	.550	.430
BoAW + functionals, SVM	Table 6.4	.799	.738	.521	**.465**
Bag-of-context-aware-words, SVM	[309]	**.800**	.750	**.603**	**.465**

Table 6.5: Comparison of the proposed model for emotion recognition on RECOLA with other approaches. Reported are the CCCs for arousal and valence on the development set (Devel) and the test set (Test).

cantly ($p < 0.05$) outperformed for *valence* by the BoAW-based models[1]. However, it must be noted that He et al. [344] were working on the reduced challenge version of the dataset, featuring fewer subjects. The *end-to-end* approach by Trigeorgis et al. [228], concatenating a CNN with a BLSTM-RNN trained on the raw waveform, is significantly ($p < 0.05$) outperformed in both dimensions of emotion.

This makes the proposed BoAW-based approach—up to this point—the best published model for speech-based emotion recognition on the RECOLA database. The reported metrics are achieved using only MFCCs (and log-energy) as acoustic LLDs and a static regressor. In 2018, the previously mentioned *bag-of-context-aware-words* by Han et al. [309], which is also using openXBOW and is an extension of the proposed approach, outperforms the set milestone.

6.2 Snore Sound Excitation Localisation

In the work entitled "A Bag-of-Audio-Words Approach for Snore Sounds' Excitation Localisation" [345], openXBOW was applied to a chunk-level audio classification task, namely, the classification of snore sounds. This task from the health care domain is highly relevant as *habitual snoring* is prevalent in society [346] and can lead to *obstructive sleep apnea*, a syndrome where the airflow during sleep is partially or fully blocked [347]. *Cardiovascular diseases* or *stroke* can result from this [348]. Moreover, the *sleep quality* of the snorer's *bed partner* can be affected [349]. As the exact medical treatment of habitual snoring depends on the location of the vibration or excitation in the upper airways, a *drug induced sleep endoscopy* is the typical means of diagnosis [350]. However, as this kind of intervention is stressful for the patient, an **automatic localisation during natural sleep** is preferable.

One typical classification scheme for excitation localisations is **'VOTE'**, discriminating between the following locations in the upper airways:

[1]The authors did not evaluate their audio-only model on test as there was a limited number of trials during the challenge.

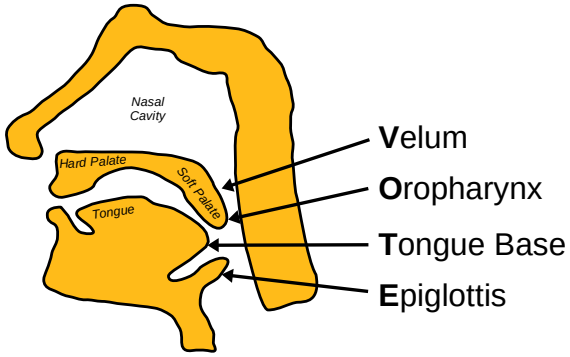

Figure 6.2: Sketch of the sagittal plane (longitudinal plane) of the human head with the four locations of excitation considered in the VOTE scheme.

Class	V	O	T	E	Total
Subjects	14	4	2	5	24
Events	66	20	10	21	117

Table 6.6: Number of subjects and snore events per class. The numbers of subjects do not add up to 24 as one subject showed both E-type and V-type snoring events.

1. **V**: *velum* or *soft palate* level,
2. **O**: *oropharyngeal* level,
3. **T**: *tongue base* level,
4. **E**: *epiglottis* level.

An anatomic illustration of these locations is provided in Figure 6.2. The dataset used in this study is an early version of MPSSC [117], with less samples than in the released corpus. All data has been collected during *drug induced sleep endoscopy* and manually labelled by an expert.

From each of the 24 subjects' recordings (sample rate 16 kHz, bit depth: 16 bit), up to 5 snoring *events* were selected. This process resulted in 117 events. The distribution of the subjects and events per class is shown in Table 6.6. Their durations range from 0.31 s to 2.17 s, with a mean duration of 1.24 s.

In order to perform a CV, the patients were split into two partitions of 12 subjects each, with their corresponding snore events. As the given data was considered too scarce to train and evaluate a classifier on, the *events* were *segmented* into instances of a fixed length of 200 ms, with 50 % overlap between adjacent segments. This leads to the distribution of instances shown in Table 6.7.

6.2.1 Acoustic LLDs

The following LLD types have been taken into account in the experiments:

Class	V	O	T	E	Total
Partition 1	376	132	18	125	651
Partition 2	434	111	46	141	732
Total	810	243	64	266	1 383

Table 6.7: Number of snore instances per partition and class.

1. **MFCCs** 1–12 + **logarithmic energy** extracted with openSMILE, employing exactly the same configuration as used in Section 6.1.
2. **Formants** F_1, F_2, & F_3, using their **frequencies** and **amplitudes** in the short-time spectrum as LLDs. The implementation by Qian et al. written in MATLAB[2] is used [351].
3. **Wavelet** descriptors based on the WPT (see Section 2.1.3) employing the *multiscale wavelet transform features* by Khushaba [352], which consist of *energy*, *variance*, *waveform length*, and *entropy* of the decomposed signal. For decomposition, wavelets from the 'symlets' family are used. This results in 28 frame-level descriptors.

This selection has been motivated mainly from the work by Qian et al. [353], who found for the same dataset that these three feature types were suited for classification, whereas energy-based and F0-based features performed worse. In contrast to the work discussed in the following, Qian et al. used functionals instead of BoAW for supra-segmental representation.

6.2.2 BoAW

All LLDs are *on-line* **min-max normalised** by openXBOW, which prove to perform better than z-score normalisation in initial experiments. The `random++` option is used for codebook generation. The codebook is reduced (see Section 5.2.1) by combining pairs of codewords with a correlation above a threshold T_C to be optimised. Also the number of assignments (N_A) and the codebook size (C_S) are optimised in an exhaustive search considering all values shown in Table 6.8. It must be noted that the codebook size defines only the **initial number of codewords** and the actual size might be lower as a result of the reduction. *Supervised* codebook generation and *Gaussian encoding* were tried in initial experiments but they did not improve the results.

BoAW features are computed with and without **logarithmic term frequency weighting** (log-TF weighting). Finally, the **histogram normalisation** option of openXBOW normalising by the number of LLDs (`-norm 1`) is applied for numerical reasons only, as due to the fixed-length instances the normalisation factors are constant.

[2] `https://de.mathworks.com/products/matlab.html`

Hyperparameter	Values
N_A	1, 2, 5, 10, 20
C_S	100, 200, 500, 1 000, 2 000
T_C	0.8, 0.85, 0.9, 0.95, 1.0
C	1e−11, 1e−10, ..., 1e0

Table 6.8: Considered values for the exhaustive search through BoAW configurations and hyperparameters. Besides, the usage of a logarithmic term frequency weighting is optimised.

6.2.3 Classifier

An SVM classifier with a *linear kernel* is employed, as the *histogram intersection kernel* performed worse. In analogy with the work presented in the previous section, LIBLINEAR [300] is used with the default solver and a complexity hyperparameter (C) optimised according to Table 6.8. Prior to SVM training, the BoAW feature vectors are **z-score normalised**.

6.2.4 Experimental setup

The experiments are run for each feature type separately, for a fusion of each pair of feature types, and a fusion of all three feature types. This results in the following 7 LLD sets:

1. MFCCs
2. Formants
3. Wavelet
4. MFCC + Formants
5. MFCC + Wavelet
6. Formants + Wavelet
7. MFCC + Formants + Wavelet

For each of the sets consisting of different features types, both the **LLD-level fusion** and the **bag-level fusion** are evaluated, resulting in 11 feature setups in total.

Experiments are run in a 2-fold CV setup, with the two partitions defined in Table 6.7. All four hyperparameters from Table 6.8 and the log-TF weighting are optimised for each feature setup (LLD set + fusion option) and partition, maximising the UAR.

LLD set	Fusion	log	N_A	C_S	T_C	C	UAR [%]	Acc [%]
MFCCs		yes	10	500	0.8	1e−3	72.5 ± 6.6	75.4 ± 10.6
Formants (F_{123})		no	2	500	0.95	1e−5	76.4 ± 2.2	78.0 ± 11.6
Wavelet (Wt)		yes	5	500	-	1e−5	73.7 ± 0.2	75.5 ± 7.5
MFCCs + F_{123}	bag	yes	5	500	-	1e−11	75.3 ± 6.0	75.6 ± 12.1
MFCCs + F_{123}	LLD	yes	10	1 000	0.9	1e−3	78.3 ± 9.2	78.9 ± 11.5
MFCCs + Wt	bag	yes	1	200	-	1e−5	77.3 ± 0.3	77.5 ± 8.0
MFCCs + Wt	LLD	yes	10	1 000	-	1e−5	78.8 ± 4.4	78.2 ± 11.1
F_{123} + Wt	bag	yes	5	500	-	1e−5	78.1 ± 4.3	77.4 ± 12.8
F_{123} + Wt	LLD	yes	5	2 000	-	1e−11	78.3 ± 1.0	78.7 ± 9.0
MFCCs + F_{123} + Wt	bag	yes	10	2 000	-	1e−6	77.9 ± 5.4	77.5 ± 12.9
MFCCs + F_{123} + Wt	LLD	yes	5	1 000	0.95	1e−5	$\mathbf{79.5 \pm 1.2}$	79.7 ± 9.3

Table 6.9: Results for the task of snore sound classification using BoAW with different LLD sets and fusion techniques. The mean and the standard deviation of UAR and Accuracy (Acc) over both partitions are given. All hyperparameters are optimised w. r. t. the mean UAR.

6.2.5 Results

The mean and standard deviation of UAR and accuracy (Acc) over both partitions are presented for each feature setup in Table 6.9. Generally, it is evident that a **fusion** of all three LLD sets performs **best** on average. In correspondence with the results by Qian et al. [353], **formant**-based and **wavelet**-based features seem to be more meaningful for the task, given that MFCCs achieve the lowest mean UAR with the highest standard deviation of all single LLD set experiments.

Except for the 'Formants'-only LLD set, **log-TF** weighting ('log') improves the results throughout. Except for the 'MFCCs+Wavelet' feature set with bag-level fusion, **multi-assignment** and a **codebook size** of at least 500 are optimal. Moreover, there is the tendency that a larger number of LLDs requires a larger codebook size, which is not surprising. The same is valid also for the LLD-level fusion (except for the fusion of all LLD types), which requires a larger codebook size, while for the bag-level fusion, the given codebook size applies for each LLD set. In this context, it must be noted that the codebook sizes have not been optimised for each LLD set separately in the case of bag-level fusion, however, the results with only one LLD type show that the optima are quite similar. Generally, the results show that **LLD-level fusion outperforms bag-level fusion**, i. e., based on these experiments, splitting the LLD sets into separate codebooks degrades the performance.

A **codebook reduction**[3] is found to be **beneficial only in few cases**; for the best result, a threshold of 0.95 is determined, which results in only few codewords to be merged.

[3] $T_C = -$ stands for $T_C = 1.0$, i. e., the codebook is not reduced.

	V	O	T	E	Recall
V	699	86	3	22	86.3 %
O	117	104	0	22	42.8 %
T	0	0	64	0	100.0 %
E	12	14	0	240	90.2 %

Table 6.10: Confusion matrix for the best model in terms of the UAR from Table 6.9. The instances of both partitions are summed up in the matrix.

The baseline set by Qian et al. [353], UAR=71.2 %, Acc=78.2 % is clearly surpassed, with the best result given in the last row of Table 6.9 representing an LLD-level fusion of all three LLD sets: UAR= 79.5 %, Acc=79.7 %. This improvement in terms of the UAR is **statistically significant** (*one-sided z-test*, $p < 0.001$).

Table 6.10 shows the confusion matrix of the optimum model, summing up the predictions on both partitions. It might be surprising that the recall for class 'T' is 100 %, even though this class is the one least represented in the data. In comparison, there is much confusion between 'V' and 'O' types, even though there are more samples available for these classes. A possible explanation for this is that the excitation locations of these two snoring types are relatively close in the upper airways and thus, they might result in a similar acoustic spectrum. Moreover, there is the limitation that the data for class 'T' originate from only two patients (one patient per partition) and the *reliability* of this result needs further proof.

6.3 Audio Effect Classification

In the publication "Recognising Guitar Effects – Which Acoustic Features Really Matter?" [354], the importance of acoustic descriptors for the recognition of *audio effects* present in recordings of *electric guitar* and *bass guitar* is studied. *Functionals* and *BoAW* as two types of instance-level representations are directly compared to each other.

Audio effects [355], generally speaking, are applied to raw recordings of musical instruments to enhance the sound, e. g., to produce a certain mood or to simulate a certain environment. Especially for electric guitar or bass guitar, dedicated effects are part of the common recording and playing setup. The task of audio effect *classification* is relevant in the context of other MIR tasks, such as, e. g., augmented *automatic music transcription* [356], *music indexing* [357], or *genre classification* [340].

The experiments are run on the IDMT-SMT-Audio-Effects database [358]. These data comprise *monophonic* and *polyphonic guitar* recordings and *monophonic* recordings of *electric bass guitar*, provided in chunks with isolated *tones* or *chords* (in the polyphonic case) present. The recordings originate from the following **four instruments**, each of them recorded in **two different configurations**:

1. Yamaha BB604 (bass guitar)
2. Warwick Corvette (bass guitar)
3. Schecter Diamond C-1 Classic (guitar)
4. Chester Stratocaster (guitar)

In the *monophonic* case, **each note** between the 0^{th} and the 12^{th} fret was recorded for **each string**[4] using **two (guitar) or three (bass) different plucking styles**[5]. In the *polyphonic* case (only for guitar), a number of chords (2 to 6 notes each) were recorded (using *plectrum*). The recordings have an approximate **duration of 2 s** each. All chunks have been processed in a *digital audio workstation*, applying **10 different audio effects**:

- 5 modulating effects: `tremolo`, `vibrato`, `chorus`, `flanger`, `phaser`
- 3 ambience effects: `reverb`, `slapback delay`, `feedback delay`
- 2 nonlinear distortion effects: `overdrive`, `distortion`

Each chunk is processed with each audio effect in **3 different settings**. Details on the characteristics and differences of these effects are given in the book by Zölzer [355] and in the original publication [354]. In addition to that, the clean, unprocessed chunks (`noFX`) are present in the database, augmented with **two amplifier simulations**, resulting in perfectly **balanced** data across all **11 classes**.

By this, the IDMT-SMT-Audio-Effects database consists of **55 044 chunks**, including 10 296 monophonic instances for each of the four instruments and 6 930 polyphonic instances for each of the two guitars.

6.3.1 Acoustic LLDs

The ComParE feature set, as introduced in Section 2.3.1, is employed throughout the experiments. All 65 LLDs and the corresponding 65 deltas are taken into account. The features are grouped into the categories of *10 feature types* according to Table 6.11 and evaluated separately and combinedly. This categorisation follows the division shown in Table 2.1, but with all *spectral* features grouped together and *RASTA* and *loudness + modulation loudness* features grouped together.

The instance-level features (functionals) present in Table 6.11 are those defined in the ComParE feature set and used as a *baseline* approach to compare the BoAW features to. For BoAW, only the frame-level features (LLDs) are relevant.

[4] bass: 4 strings, guitar: 6 strings

[5] bass: *finger plucked soft*, *finger plucked hard*, and *plectrum*, guitar: *finger* and *plectrum*

Feature type	# Frame-level features (LLDs)	# Instance-level features (from LLDs)	# Instance-level features (from deltas)	# Instance-level features (sum)
RMS (energy)	1	54	46	100
ZCR	1	54	46	100
Spectral features	15	810	690	1 500
RASTA + Loudness	28	1 512	1 288	2 800
MFCCs (1–14)	14	756	644	1 400
F0	1	44	39	83
logHNR	1	39	39	78
Voicing probability	1	39	39	78
Jitter (local + DDP)	2	78	78	156
Shimmer (local)	1	39	39	78
Total	**65**	**3 425**	**2 948**	**6 373**

Table 6.11: Features present in the ComParE feature set, grouped into 10 feature types with corresponding numbers of frame-level features and instance-level features (subdivided into instance-level features originating from the raw LLDs only, those derived from deltas, and their sum).

6.3.2 BoAW

All LLDs get *on-line* **z-score normalised**. Aligned with the previous case studies, codebooks are generated by `random++` sampling. The **codebook size** is **optimised** in initial experiments (see below), in terms of multiples of the size of the baseline feature set with functionals.

The *number of assignments* is 1. A **logarithmic term frequency weighting** is applied to the histograms.

6.3.3 Classifier

As in previous experiments shown, an SVM classifier with a *linear* kernel is employed. Training is done using the Liblinear toolkit with the `L2-regularized L2-loss (primal)` solver and adding a *bias* of 1 to the feature space. The complexity hyperparameter is optimised in the range of [1e−6, 1e−6, . . . , 1e0] in a 2-fold CV setup as explained in the following subsection.

Concerning **feature normalisation**, in this particular case study, **on-line** and **off-line** z-score normalisation are compared to each other for the original ComParE feature set with **functionals**. The reason for this is that, as shown below, the experimental setup following the work by Stein at al. [358] includes an evaluation across conditions, such as, e. g., training on *monophonic* data and testing on *polyphonic* data. In order to align the feature distributions between such different training and test conditions, it is evaluated whether *off-line* normalisation, i. e., es-

ID	Acronym [358]	Train	Test
A1	BS-MO	Yamaha BB604	Warwick Corvette
A2		Warwick Corvette	Yamaha BB604
B1	GIT-MO	Schecter Diamond	Chester Stratocaster
B2		Chester Stratocaster	Schecter Diamond
C1	BS-GIT	Yamaha BB604 + Warwick Corv.	Schecter Diamond + Chester Strat.
C2		Schecter Diamond + Chester Strat.	Yamaha BB604 + Warwick Corv.
D1	GIT-MP	Schecter + Chester (monophonic)	Schecter + Chester (polyphonic)
D2		Schecter + Chester (polyphonic)	Schecter + Chester (monophonic)

Table 6.12: Overview of the experiments conducted on the IDMT-SMT-Audio-Effects database.

timating the normalisation parameters on the test data, might be more appropriate for this scenario.

BoAW features are not normalised throughout the experiments.

6.3.4 Experimental setup

Referring to the work by Stein et al. [358], the setup follows their experiment 4, which means that only the 11-class ML task is taken into account. Table 6.12 shows the 8 experiments that are conducted, where each *letter* (A,B,C,D) denotes one out of 4 selections of instruments or condition (monophonic/polyphonic) and the *digit* (1,2) denotes the permutation, i. e, that each split is used for training or testing, respectively, and vice versa.

Concerning the *complexity optimisation*, a 2-fold CV is performed on the respective training set; for experiments A and B, each split consists of one particular *instrument configuration* (as mentioned above), for experiments C and D, each split consists of all samples from *one instrument*. The classifier is then re-trained on the whole training set with the optimum complexity.

All results reported are on the corresponding test set in terms of the UAR. This evaluation is aligned with the results reported for the 'contextual CV' by Stein et al. [358].

6.3.5 Results

First, the original ComParE features with functionals are evaluated as a *baseline*. Results are displayed in Table 6.13 with **on-line** normalisation and in Table 6.14 with **off-line** normalisation. In each table, the UARs obtained for each feature group alone, for all features ('ALL'), for all features but without those derived from deltas ('ALL, raw LLDs-only'), and for all features derived from deltas only ('ALL, deltas-only') are shown.

Feature type	A1	A2	B1	B2	C1	C2	D1	D2
RMS (energy)	64.1	62.0	74.0	77.3	59.3	53.8	9.2	**23.9**
ZCR	29.0	29.9	33.7	37.5	24.6	22.8	9.9	9.3
Spectral features	72.9	70.7	84.7	88.1	**65.2**	52.2	9.1	17.4
RASTA + Loudness	61.1	59.0	81.4	82.2	50.4	49.6	10.6	9.3
MFCCs (1–14)	43.6	50.0	58.6	65.1	36.0	37.5	**17.1**	21.4
F0	41.2	36.6	58.8	56.7	36.2	35.8	14.6	21.6
logHNR	55.7	53.8	66.7	67.9	45.3	49.2	4.7	10.4
Voicing probability	44.4	43.5	64.9	64.9	45.3	34.9	13.6	11.3
Jitter (local + DDP)	37.7	35.0	61.8	58.2	33.5	29.7	13.9	15.3
Shimmer (local)	48.9	43.7	63.3	62.5	45.1	46.5	12.9	16.0
ALL	72.5	**77.0**	**89.3**	94.8	61.5	**56.5**	9.1	13.6
ALL, raw LLDs-only	67.5	70.9	86.3	90.5	56.0	52.4	9.4	12.2
ALL, deltas-only	**73.5**	74.5	87.9	**95.3**	60.3	55.7	9.1	12.9

Table 6.13: Results for audio effect classification (UAR [%]) based on ComParE (with functionals) and **on-line** z-score normalisation.

While for experiments A and B (i. e., 'bass vs bass' and 'guitar vs guitar'), the combination of ALL features achieves usually the highest UAR or at least approximately, for experiments C and D (i. e., 'bass vs guitar' and 'monophonic vs polyphonic'), better results are achieved using only a particular feature group (except for C2, and also D2 for off-line normalisation). Concerning on-line normalisation, most interestingly, the features derived from the **deltas only** seem to be **more meaningful** than the LLDs itself, achieving always the higher UAR in experiments A and B and even the highest overall UAR in A1 and B2. Nevertheless, the performance for experiments D is not much above chance level (= 9.1 %) for on-line normalisation, while it is above 50 % for off-line normalisation, which also achieves the higher UARs throughout.

Particularly in the experimental setup D, the advantages of off-line normalisation are evident, based on the previously described differences in training vs evaluation conditions ('monophonic' vs 'polyphonic'). However, off-line normalisation requires an adaptation of the normalisation parameters on the test domain.

Looking at particular feature groups, it seems that **spectral features** are most discriminative for audio effect classification whereas ZCR is least. This is expected, as ZCR is only a single LLD and highly depends on the 'pitch' of the played note. This finding is valid for both normalisation types. However, it is quite surprising that the RMS energy alone achieves already a UAR of 77.3 % for experiment B2 with on-line normalisation (and similar results with off-line normalisation), which is higher than the UAR achieved by MFCCs (65.1 %). This might be explained by the laboratory settings under which the database was produced, where the application of audio effects resulted in similar 'energy patterns' over time. This hypothesis is confirmed by the comparably low results in setup D.

Feature type	A1	A2	B1	B2	C1	C2	D1	D2
RMS (energy)	64.2	62.8	76.1	77.0	57.4	44.4	25.5	36.5
ZCR	30.5	30.8	34.5	38.9	29.8	24.3	1.6	11.3
Spectral features	75.7	73.4	92.0	93.3	**73.1**	54.9	40.9	45.4
RASTA + Loudness	71.2	66.2	85.6	87.7	57.3	56.1	**66.0**	36.5
MFCCs (1–14)	52.5	50.8	65.8	67.7	37.2	40.0	31.2	25.4
F0	32.4	33.8	55.1	55.1	33.0	27.8	15.3	21.9
logHNR	52.1	51.0	63.1	62.7	42.3	28.3	13.7	11.3
Voicing probability	43.4	43.0	63.1	63.7	42.7	34.6	18.1	30.2
Jitter (local + DDP)	35.4	32.9	57.8	54.2	30.2	20.1	12.3	15.8
Shimmer (local)	46.7	43.1	63.3	62.3	36.7	31.9	15.9	16.0
ALL	**83.3**	**82.0**	**96.7**	**97.8**	70.5	**63.7**	59.8	50.7
ALL, raw LLDs-only	79.4	78.0	95.1	96.7	68.7	58.4	56.4	46.0
ALL, deltas-only	79.5	78.3	94.9	96.3	71.9	63.2	63.5	**51.1**

Table 6.14: Results for audio effect classification (UAR [%]) based on ComParE (with functionals) and **off-line** z-score normalisation.

For a **BoAW**-based model, the **codebook size** is one of the key hyperparameters. Initial experiments showed that the results improve if the BoAW feature vector size is larger than the size of the ComParE feature set (6 373). The question is how to make a fair comparison between the two studied feature representations. On the one side, SVM classifiers (with a linear kernel) usually perform better in a larger feature space. On the other side, the codebook selection is a randomised and scalable process that can be tuned automatically, increasing 'only' the computational complexity, whereas functionals need to be defined manually. Thus, results with BoAW features of the same dimensionality and larger dimensionalities, in comparison with the baseline set, are reported in the first BoAW experiment.

Table 6.15 shows the UARs obtained with BoAW features in experiment B, where the UAR measures of the two permutations (B1 & B2) are averaged. Results are reported, as before, for all feature groups and a fusion of the LLDs. In addition to that, results for *raw LLDs-only* and *deltas-only* are provided now for each feature group. For each of these 'LLD-sets', 3 different **codebook sizes** are selected, as specified, as **multiples** of the corresponding functionals-based feature vectors, as displayed before in Table 6.11. The factors 1, 2, and 4, are used. For the fusion of feature types ('ALL'), a **bag-level fusion** is performed, i. e., the BoAW representations from each feature vector are concatenated.

It is evident that overall, the largest BoAW dimensionality (x4) performs best. Similar findings as for the functionals-based experiments with on-line normalisation can be made: overall, *deltas* of the LLDs seem to be more meaningful than the LLDs themselves and *spectral descriptors* are most discriminative, based on these results. Also here, *energy* alone achieves relatively good results, while ZCR, F0, and jitter perform worst in the case of BoAW.

Feature types	**BoAW** C_S factor			**BoAW raw LLDs-only** C_S factor			**BoAW deltas only** C_S factor		
	x1	x2	x4	x1	x2	x4	x1	x2	x4
RMS (energy)	59.5	62.1	62.9	34.8	34.2	33.5	50.9	51.6	51.6
ZCR	33.1	32.8	33.5	27.7	27.7	27.7	26.5	27.2	27.1
Spectral features	58.2	59.0	61.6	41.8	41.7	43.7	61.9	63.6	67.9
RASTA + Loudness	53.7	56.9	58.1	42.4	45.2	48.1	51.8	54.3	56.7
MFCCs (1–14)	48.5	50.7	51.8	34.3	34.4	35.1	50.3	52.2	53.8
F0	23.3	26.9	29.9	21.0	24.5	25.8	19.4	34.4	38.1
logHNR	50.8	56.9	60.5	36.3	37.4	37.2	45.2	52.7	53.6
Voicing probability	43.4	48.5	50.9	30.3	29.9	29.0	39.2	46.1	48.0
Jitter (local + DDP)	28.7	28.6	33.0	34.4	37.3	39.3	12.9	21.5	29.4
Shimmer (local)	45.2	48.7	50.5	34.9	35.2	35.1	37.6	40.2	40.3
ALL	83.8	86.2	87.9	70.5	72.0	75.6	85.9	90.5	**90.8**

Table 6.15: Results for audio effect classification (mean UAR [%] over experiments B1 & B2) based on ComParE-LLDs with BoAW. The codebook size (C_S) is specified as a multiple (with given factor) of the functionals feature space.

Feature type	**A1**	**A2**	**B1**	**B2**	**C1**	**C2**	**D1**	**D2**
ALL	72.0	69.4	87.2	88.0	64.9	55.1	**41.1**	45.7
ALL, raw LLDs-only	64.0	58.9	73.6	76.8	49.4	41.8	22.4	40.1
ALL, deltas-only	**76.5**	**72.4**	**90.9**	**92.8**	**67.7**	**58.2**	35.8	**62.1**

Table 6.16: Results for audio effect classification (UAR [%]) based on ComParE-LLDs with BoAW. A fixed codebook size of 2 000 is employed for each of the 10 feature types (bag-level fusion).

Results for all experiments A–D are shown in Table 6.16. Here, a **fixed codebook size** of 2 000 was chosen for **each feature type**. The BoAW representations of all of the 10 feature types are then concatenated (bag-level fusion). The measures show that this configuration provides even better results than the configuration with variable codebook sizes depending on the number of LLDs and functionals for each feature type.

It is evident that throughout the experiments, **BoAW based on deltas provides better results** than BoAW based on only the raw LLDs. With the exception of experiment D1, deltas even perform better alone than together with the LLDs. This finding shows that by the application of audio effects, the short-term evolution of the signal is more affected than its static properties.

Overall, it can be concluded that the **BoAW** approach, which only employs an *on-line normalisation of the frame-level input features*, **outperforms functionals with on-line normalisation** for most of the experiments. This is especially true for experiments D, where BoAW outperforms functionals by a large margin (D1: 24.0 %, D2: 38.2 %).

For **off-line normalisation**, the functionals-based approach performs better, except for experiment D2 (training: *polyphonic*, test: *monophonic*), where the UAR with BoAW is more than 10 % higher. This is remarkable as the comparison is not really 'fair', because off-line normalisation always requires enough data in the *test domain* to tune the normalisation parameters on. The superior performance of BoAW can be explained by the fact that the *energy* or *level* of a signal, which differs between the polyphonic and monophonic cases, has a *more direct impact on the functionals* than on the BoAW, where only the word assignment step is affected and not the 'energy' of the BoAW representation. This might also be the reason for the better performance using deltas as input, which are not as much level-dependent as the raw LLDs. While off-line normalisation of functionals provided better results, also an off-line normalisation of the frame-level features was tried for BoAW, but the performance was slightly lower.

Generally, it must be noted than **experiments C and D are the most challenging ones**. For experiment C (*bass* vs *guitar*), different effect settings were applied for each instrument type by the authors of the database. For experiment D, in addition to the expected level differences, some audio effects have a different effect on polyphonic samples, especially `overdrive` and `distortion`, where *harmonics* are added to the signal. Thus, these particular experiments, where the BoAW approach performs comparably well, can be considered the most realistic ones.

The achieved measures are lower than those obtained by Stein et al. [358], however, they employed a pre-processing step of the samples, where only the *sustain part* [38] of the samples is analysed; a pre-processing step, which might introduce further error when deployed in a realistic setting with non-isolated tones or chords. Moreover, the BoAW approach in general and openXBOW in particular offer some room for improvement, especially the number of assignments and codebook sizes could be adapted specifically for each feature type.

The original publication of the author [354] offers more results in terms of the class-specific recalls and feature types, i. e., which audio effect is recognised how well and which feature type is most discriminative. An analysis shows that the audio effects `distortion` and `tremolo` achieve the highest recalls, while `phaser` is recognised the worst. However, the classification of the similar effects `chorus` and `flanger` shows only a low level of confusion.

6.4 The ComParE Series

The series of the Computational Paralinguistics ChallengE (ComParE) promoted the usage of BoAW features and the toolkit openXBOW by employing them for the corresponding baseline approaches provided to the participants. BoAW features and scripts to extract them with openXBOW were provided, so far, in all recent

editions of ComParE from 2017 to 2020, throughout all sub-challenges launched in these years.

ComParE[6] has been organised as an annual event, starting in 2009. Prior to the *Interspeech* conference in the corresponding year, usually in January (8–9 months before the conference), research teams (academic and industrial) may start to register for participation and get access to 3 or 4 packages (one for each sub-challenge), containing *data* in terms of audio files, *labels* for training (and development) set, *acoustic feature* sets (see below), and *scripts* to both (re-)extract acoustic features and run an ML experiment to reproduce the *baseline* results. Participating researchers are free to work on features, ML models, algorithms, etc. in order to improve the measure (depending on the task) on the test set, where only data is provided for but not the labels. Participants may submit predictions of up to 5 models (per sub-challenge) and their best result is counted. In addition to that, they need to submit a regular paper to a *special session* at the conference, dedicated to ComParE. All submissions need to pass the regular reviewing process of the conference in order to be presented at the event. For each sub-challenge, the team achieving the best result receives a prize—given that their model outperforms also the baseline result.

During the recent years, classification, regression, and detection tasks were organised. In detail, in the years 2017—2020, the following sub-challenges (tasks) were launched:

1. **Addressee** [11] (10 886 instances, 2 classes: parents speaking directed to their *child / between themselves*)
2. **Cold** [11] (28 652 instances, 2 classes: *infection* in the upper respiratory tract / *no infection*)
3. **Snoring** [11] (828 instances, 4 classes: $V/O/T/E$ (see Section 6.2), introducing MPSSC [117])
4. **Atypical Affect** [12] (10 627 instances, 4 classes: *anger*/*happiness*/*sadness*/*neutral* for mentally or physically disabled subjects)
5. **Self-assessed Affect** [12] (2 313 instances, 3 classes *low*/*medium*/*high* valence, assessed by the subjects themselves)
6. **Crying** [12] (5 587 instances, 3 classes: *neutral*/*fussing*/*crying* vocalisations of babies)
7. **Heart Beats** [12] (845 instances, 3 classes: *normal*/*mild*/*moderate-severe* health condition of the heart)
8. **Styrian Dialects** [13] (9 732 instances, 3 classes: *northern Styrian*/*urban Styrian*/*eastern Styrian*)

[6] `http://www.compare.openaudio.eu/`

9. **Continuous Sleepiness** [13] (16 462 instances, 9 classes: according to the *Karolinska Sleepiness Scale*[359])
10. **Baby Sounds** [13] (11 304 instances, 5 classes: *canonical/crying/junk/laughing/non-canonical*)
11. **Orca Activity** [13] (13 409 instances, 2 classes: orca *present* or *not*)
12. **Elderly Emotion** [116] (261 instances, 2 × 3 classes: *low/medium/high arousal/valence*)
13. **Breathing** [116] (49 speakers, time-continuous regression task)
14. **Masks** [116] (36 554 instances, 2 classes: *mask/no mask*)

6.4.1 BoAW

For all the listed sub-challenges, a unique BoAW configuration was chosen. Only the *codebook* size was optimised for each task as a hyperparameter. As input features, the 65 acoustic LLDs from the official ComParE feature set (see Section 2.3.1) have been used throughout, in combination with the 65 deltas of the LLDs. The frame-level features are **z-score normalised**, using the corresponding option in openXBOW.

One BoAW representation was learnt for the raw LLDs and another one for the deltas, **fused on bag-level**. For both representations, the same codebook sizes were employed, optimised in the range of 125 and 8 000. Throughout all of the challenges, the **number of assignments** was fixed as **10** and a **logarithmic term frequency weighting** was applied.

6.4.2 Other feature sets

Besides BoAW features generated by openXBOW, the following further feature sets were taken into account for the baselines presented in the accompanying publications [11, 12, 13, 116]:

ComParE The official ComParE feature set, extracted with openSMILE [23], consisting of 6 373 acoustic features, originating from 65 LLDs and corresponding deltas. Statistical functionals are applied to LLDs and deltas to generate the fixed-length instance-level feature vectors. The ComParE feature set was first used in ComParE 2013 [20]. A detailed description is given in Section 2.3.1 in this thesis.

auDeep Since 2018, auDeep features were added, extracted with the homonymous toolkit [360, 361]. They are learnt—as BoAW features—in an **unsupervised** way, using a *recurrent sequence-to-sequence autoencoder*. This is a special type of recurrent neural network (see Section 4.4.4), mapping the input—in this case, the

mel-spectrogram of the audio signal—to itself, learning a lower-dimensional representation of fixed length, independent of the duration of the signal. *Clipping* at 4 different levels [dB] is applied to the mel-spectrograms, resulting in a 1 024-dimensional feature vector for each clipping level. In addition, also a fusion of these is evaluated.

DeepSpectrum In the 2020 edition of ComParE, DeepSpectrum features [272] were considered for the **Elderly Emotion** and **Masks** sub-challenges. Mel-spectrograms are extracted as a first step and then propagated through ResNet50 [283], a pre-trained DNN from the image recognition domain. The 2 048-dimensional internal representations from the 'avg_pool' layer are used as a fixed-length feature vector.

LiFE For the **Elderly Emotion** sub-challenge in 2020, where also transcriptions were provided for, **linguistic features** were taken into account additionally. A state-of-the-art *LinguistIc Feature Extractor* (LiFE) pipeline was implemented, employing a so-called *Transformer* model (BERT)[362] to extract context embeddings for the words of each recording. The 768-dimensional context embedding sequence is then compressed in two ways in order to obtain a fixed-length feature vector: 1) a global *maximum pooling* and 2) a *bidirectional LSTM-RNN* with an *attention* mechanism and two feedforward layers resulting in a 512-dimensional output; in addition, a part-of-speech embedding was trained.

E2E An End-to-end system, learning audio representations as internal representations of a DNN, consisting of a CNN and an LSTM-RNN, with an optimised number of layers, similar to the pioneering work by Trigeorgis et al.[228]. The whole network is trained in a **supervised** way, using the audio signal as input and the provided labels as a target. Thus, the features are not used explicitly. In the 2018 edition of ComParE, the toolkit End2You [363] was employed for the end-to-end approach. In the 2019 and 2020 editions, an E2E system was not provided, except for the Breathing sub-challenge, which is a sequence labelling task.

6.4.3 Machine learning

All extracted feature types—except for the internal representations of the E2E approach—are fed into an **SVM** classifier or regressor (depending on the task). In 2017 and 2018, the toolkit Weka [161] was employed, using the function 'SMO' (sequential minimal optimisation). From 2019 onwards, the organisers switched to the Python library Scikit-learn [364], with the functions LinearSVC and LinearSVR, for classification or regression, respectively. These functions are based on Liblinear [300], which is used also in other experiments presented in this thesis.

Given this, nothing but a **linear kernel** has been used throughout. This decision was based on the compelling experience made in previous editions of ComParE [20, 21, 22, 85] with linear kernels in large-scale feature spaces.

For most of the sub-challenges, the available data was split into disjunct training, development, and test sets. Only for the **Crying** sub-challenge in 2018, a LOSO CV was preferred, given the relatively low number of subjects. In the ComParE editions of 2017 and 2018, a **z-score normalisation** was applied; in the editions of 2019 and 2020, a **min-max normalisation** was applied. There were no obvious reasons for this change, as there was no clear superiority of one or the other method, but it was connected to the switching from WEKA and BASH scripts to SCIKIT-LEARN and PYTHON. Furthermore, a *min-max normalisation* seemed more appropriate at least for BoAW features, as the log-term frequency histograms are typically not normal distributed, but biased towards *zero* term frequencies ('sparse'). The normalisation was always performed on-line, i. e., normalising the features of all partitions based on the parameters derived from the training set (for evaluation on the development set and LOSO CV) or derived from training+development sets (in the case of three partitions given).

The **complexity** hyperparameter was optimised in the range of [1e−6, 1e−5, 1e−4, 1e−3, 1e−2, 1e−1, 1e0] to obtain the best performance on the development set (or in LOSO CV). The final model for prediction of the test set labels is trained on fused training and development sets (on the training set in case of LOSO CV) using the optimised complexity.

An **upsampling** of instances of the minority classes, in order to balance class distributions, was used in the case of *imbalanced* datasets. A detailed overview over class distributions and the key statistics of the data is found in the original publications [11, 12, 13, 116].

6.4.4 Results

An overview of the results from all sub-challenges and feature types is given in the following. The metrics for all classification tasks is the **UAR**, taking into account the sometimes imbalanced class distribution in the test sets. For the **Sleepiness** sub-challenge, **Spearman's correlation coefficient** is the official metric. For the **Orca Activity** sub-challenge, which is a (binary) detection task, the **ROC-AUC** is noted. Finally, for the **Breathing** sub-challenge in 2020, **PCC** is employed. All these metrics are introduced in Section 4.5.

As mentioned, besides the SVM *complexity*, only the *codebook size* was adapted for all tasks. The main reason for this procedure was that, performing too much optimisation, the baseline results would have been too competitive for the participants of ComParE. Also a **late fusion** of the 'n-best' models was provided. From each approach, the model performing best on the test set was selected for the fusion. The results of a **majority voting** has always been reported and in years 2017 and

TASK	**Addressee**	**Cold**	**Snoring**	**Atypical**	**Self-assessed Affect**	**Crying**	**Heart Beats**
Classes	2	2	4	4	3	3	3
Metric	UAR [%]	UAR [%]	UAR [%]	UAR [%]	UAR [%]	UAR [%]	UAR [%]
Partition	Dev / Test	Dev / Test	Dev / Test	Dev / Test	Dev / Test	Lo / Test	Dev / Test
C_S	**BoAW**						
125, 125	63.2 / 67.5	55.9 / 62.8	43.8 / 48.7	38.7 / 36.4	56.5 / 61.7	75.6 / 73.2	43.1 / 43.4
250, 250	61.4 / 66.6	62.8 / 66.5	46.6 / 49.9	40.5 / 36.5	55.8 / 59.1	75.9 / 71.8	42.3 / 47.2
500, 500	62.4 / 68.2	63.9 / 66.7	44.2 / 51.2	39.8 / 38.1	52.5 / 63.2	76.9 / 67.7	43.7 / 41.0
1 k, 1 k	62.2 / 67.2	64.2 / 67.3	42.8 / 50.0	38.1 / 41.3	51.1 / 62.2	75.5 / 69.8	42.6 / 52.3
2 k, 2 k	63.4 / 67.7	64.1 / 67.3	41.0 / 48.3	37.9 / 39.2	56.7 / 60.9	75.4 / 68.8	39.9 / 43.3
4 k, 4 k	63.4 / 68.2	63.8 / 67.2	39.8 / 48.2				
8 k, 8 k	63.3 / 68.3	64.0 / 69.7	36.6 / 47.8				
	Other approaches						
ComParE	60.5 / 67.7	64.0 / 70.2	40.6 / **58.5**	37.8 / 43.1	56.5 / 65.2	75.6 / 71.9	50.3 / 46.4
E2E	59.8 / 60.1	59.1 / 60.0	40.3 / 40.3	41.8 / 28.0	48.2 / 46.6	— / 63.5	41.2 / 37.7
auDeep				40.4 / 35.6	49.9 / 57.3	73.2 / 71.1	38.6 / 47.9
	Late fusion (Majority voting)						
2-best				— / 42.9	— / 65.4	— / 70.4	— / **56.2**
3-best	64.0 / 68.0	65.2 / **71.0**	43.4 / 55.6	— / 42.0	— / **66.0**	— / **74.6**	— / 51.1
4-best				— / 41.0	— / 62.2	— / 71.3	— / 53.0
	Late fusion (Confidence-based)						
2-best	62.8 / 68.7	64.2 / 70.1	42.1 / 52.4	— / **43.4**	— / 64.6	— / 73.1	— / 49.3
3-best	66.4 / **70.2**	66.1 / 70.7	43.5 / 53.0	— / 42.0	— / 64.7	— / 73.9	— / 53.6
4-best				— / 42.0	— / 64.7	— / 73.9	— / 53.6

Table 6.17: Results of ComParE 2017 & 2018 with different feature sets / ML approaches. Dev: Development. C_S: Codebook size of Bag-of-Audio-Words (BoAW), splitting the input into two codebooks (ComParE-LLDs,ComParE-LLD-Deltas), with 10 assignments per frame. E2E: end-to-end learning (CNN + LSTM). All approaches (except for E2E) use an SVM classifier/regressor, where the complexity hyperparameter is optimised on the development set/through LOSO CV (Lo). Figures of empty cells were not computed for the original publications [11, 12].

2018, also a fusion based on the confidence outputs was given additionally. For a majority vote of the two best approaches (2-best), the output is the label that is predicted most often for the whole partition in case of contradictory predictions of the two underlying models. For the **Continuous Sleepiness** and the **Breathing** task, the **means** of the predictions were used for the fusion.

Table 6.17 shows the results for all sub-challenges of ComParE editions 2017 & 2018 and Table 6.18 shows the corresponding results for ComParE editions 2019 & 2020. Except for BoAW, where the evaluation measures are given for different codebook sizes, only the configuration achieving the highest measure on the test set for each approach is given. The **official baselines**, i. e., the approach obtaining the best result on the test set for each sub-challenge, are **highlighted in bold**. For most of the sub-challenges, a **late fusion**, combining the predictions of two or more approaches achieves the best result.

For three tasks (**Addressee**, **Crying**, **Heart Beats**), the best **BoAW** configuration **outperforms all other approaches** on the test set. For the 12 other

TASK	**Styrian Dialects**	**Cont. Sleepiness**	**Baby Sounds**	**Orca Activity**	**Elderly Emotion**	**Breath-ing**	**Masks**
Classes	3	9	5	2	3 A | 3 V		2
Metric	UAR [%]	Spearman	UAR [%]	ROC-AUC	UAR [%]	PCC	UAR [%]
Partition	Dev / Test	Dev / Test	Dev / Test	Dev / Test	Dev / Test	Dev / Test	Dev / Test
C_S	**BoAW**						
125, 125	38.2 / 31.9	.240 / .291	51.5 / 52.7	.772 / .815	42.0|38.9/40.6|37.7	.185 / .357	59.8 / 58.7
250, 250	38.2 / 32.4	.236 / .268	51.0 / 54.3	.763 / .822	40.5|33.3/49.1|31.5	.201 / .349	61.5 / 62.7
500, 500	38.2 / 31.2	.250 / .304	51.2 / 53.7	.762 / .831	41.0|38.9/46.6|31.7	.209 / .367	63.1 / 65.0
1 k, 1 k	37.4 / 32.2	.265 / .286	51.1 / 54.3	.770 / .823	39.0|38.7/42.2|32.4	.226 / .366	63.6 / 66.1
2 k, 2 k	38.0 / 32.0	.269 / .260	51.0 / 54.9	.771 / .836	39.7|40.6/42.2|33.8	.215 / .355	64.2 / 67.7
	Other approaches						
ComParE	38.0 / 36.0	.251 / .314	54.0 / 57.7	.810/.866	39.1|40.4/47.9|41.7	.244 / .442	62.3 / 67.8
E2E						.507 / **.731**	
auDeep	44.4 / **47.0**	.243 / .325	51.6 / 48.1	.740 / .798	34.9|36.7/44.3|33.8		64.4 / 66.6
DS					35.0|31.6/**50.4**|40.3		63.4 / 70.8
LiFE					40.6|44.0/49.2|**49.0**		
	Late fusion (Majority voting)						
3-best	— / 40.0	— / **.343**	— / **58.7**	— / **.866**		— / .621	
4-best							— / **71.8**
5-best					— /47.9|39.8		

Table 6.18: Results of ComParE 2019 & 2020 with different feature sets / ML approaches. A: Arousal. V: Valence. Dev: Development. C_S: Codebook size of Bag-of-Audio-Words (BoAW), splitting the input into two codebooks (ComParE-LLDs,ComParE-LLD-Deltas), with 10 assignments per frame. E2E: end-to-end learning (CNN + LSTM). DS: DeepSpectrum. All approaches (except for E2E) use an SVM classifier/regressor, where the complexity hyperparameter is optimised on the development set/through LOSO CV (Lo). Figures of empty cells were not computed for the original publications [13, 116].

tasks, BoAW is the second best approach (on the test set) in 6 cases. In 10 out of 15 tasks, a *late fusion* of models outperforms the single models. This is an indication of the **complementarity** of different feature representations, which has already been observed in Section 6.1 for an *early fusion* of functionals and BoAW features.

For the **Addressee** and the **Heart Beats** sub-challenges in 2017 and 2018, respectively, the baseline was *not* surpassed by any of the participants of the challenge. Here, a late fusion of all single models (**Addressee**) and a confidence-based fusion of BoAW and auDeep features (**Heart Beats**) provided the best performance, i. e., the **BoAW approach was involved in all unbeaten systems**.

To underpin this and as already mentioned at the end of Chapter 5, where examples for openXBOW usage by other researchers were given, further improvement on the **Continuous Sleepiness**, **Baby Sounds**, and **Orca Activity** tasks (but not on the **Styrian Dialect** task) were achieved by a late fusion of ComParE, BoAW, and the previously introduced *Fisher vectors* [330].

The results for BoAW features show further that the codebook size needs to be tuned for each dataset, even though exactly the same set of LLDs are used throughout. It is also evident that the performance on the development set is not

a robust indicator of the performance on the test set, i. e., a codebook size that is optimal for a model trained on the training set only might not be the best choice for a model trained on fused training and development sets. However, looking at the complete results tables ([11, 12, 13, 116]), a mismatch between development and test set results can be observed also for the other approaches.

After the challenge, some of the baselines have been outperformed by a large margin using BoAW representations based on other frame-level feature types. For the **Snoring** task (MPSSC), BoAW computed from *Wavelet* descriptors achieve a UAR of 69.4 % on the test set [315], which is considerably higher than the baseline of 58.5 %. Nevertheless, also a refined *end-to-end* model [229], as published by the author of this thesis, reaches a competitive UAR of 67.0 %.

6.5 The AVEC Series

The **Audio/Visual Emotion Challenge and Workshop** (AVEC) is, just like ComParE (see Section 6.4), an annual scientific competition in the field of *affective computing*, ML, CA, and computer vision. In contrast to ComParE, however, AVEC is a **multimodal** challenge and the focus of the tasks is on such with an **emotional context**. Typical modalities are—as the name suggests—audio and video, but also *physiological signals* [337] and *manual transcriptions* [18] were provided for some sub-challenges in the past. The main task has usually been the recognition of emotion in terms of the continuously-valued dimensions *arousal* and *valence* [337], as introduced in Section 6.1; but also the detection or monitoring of *depression* [338]—as a health condition affecting mood—was studied as a challenge task. For this, more than one sub-challenge based on several datasets were offered in some editions. The main event of AVEC was normally held as a *satellite workshop*[7] at the *ACM Multimedia* conference. Similar to ComParE, participants were provided with the data and scripts to reproduce baseline results a few months prior to the paper submission deadline.

BoAW features, generated by openXBOW, have been employed for the baseline approach in the AVEC editions of 2017 [18], 2018 [19], and 2019 [17]. In these years, an annual sub-challenge was **time-continuous emotion recognition** on the **SEWA ('Automatic Sentiment Analysis in the Wild')** database [302], which was already mentioned in Section 5.1.5 on the computational performance of openXBOW. In total, **audio-visual recordings from 6 cultures**[8] are present in the corpus, balanced across age groups from 18 to 60+. In the 'video chat' subset of the database, which was taken into account for AVEC, **pairs of subjects are discussing a commercial**, recorded by consumer webcam and microphones at the subjects' workplaces or homes. A video chat of a duration of up to 3 minutes is

[7] `https://sites.google.com/view/avec2019/home`
[8] British, Chinese, German, Greek, Hungarian, & Serbian

Culture	Years	Partition	Number of Subjects	Duration [h:min:s]
German	2017 + 2018 + 2019	Training	34	1:33:12
	2017 + 2018 + 2019	Development	14	0:37:46
	2017 + 2018 + 2019	Test	16	0:46:38
Hungarian	2018 / 2019	Test / Training	34	1:08:24
	2018 / 2019	Test / Development	14	0:28:42
	2018 / 2019	Test / Test	18	0:36:06
Chinese	2019	Test	70	3:17:52
All			**200**	**8:28:50**

Table 6.19: Overview of the key statistics of the SEWA corpus as used in AVEC 2017–2019 [17, 18, 19]. The German subset is used with the same partitioning in each year. The Hungarian subset is used entirely as a Test set in AVEC 2018 and split into Training, Development, and Test partitions in the 2019 challenge. The Chinese subset is only used in AVEC 2019 as a Test set.

available for each pair. As the recording equipment and conditions (background noise, etc.) varies across recordings, SEWA is considered an 'in the wild' database. More details are found in the article introducing the corpus [302].

Several **incremental subsets** of the full SEWA database were introduced during the years, adding a new culture/language each year and different training and testing conditions. In the 2017 challenge, only *German* subjects were included and used for *training*, *development*, and *testing*. In 2018 and 2019, **cross-cultural** emotion recognition tasks were set, with recordings of *Hungarian* subjects forming the test set in 2018 and recordings of *Chinese* subjects for testing in 2019, while training on the first two cultures.

Table 6.19 shows the key statistics of the SEWA data used in AVEC 2017–2019, with the numbers of subjects and durations of each partition. The German subset of SEWA was always used with the same partitioning into training, development, and test set. The Hungarian subset was used only for testing in the 2018 challenge and with the split into three partitions (training, development, & test) in the 2019 challenge. The Chinese subset was only used as a test set in its completeness in 2019. It must be mentioned, however, that the official ranking of the participants was depending only from the measures obtained on one culture in each year (2017: German, 2018: Hungarian, 2019: Chinese).

The SEWA corpus was annotated by 5 (Hungarian) or 6 (Chinese, German) annotators from the corresponding subjects' culture, respectively, in terms of the emotional dimensions *arousal* (high/low), *valence* (positive/negative), and *liking* as a third dimension. While *arousal* and *valence* are affective measures [89] that are commonly used in the field of *affective computing*, e. g., for the annotations of the RECOLA corpus used in the experiments presented in Section 6.1, the dimension of *liking* is a measure specifically used in SEWA. It describes whether a subject seems to 'like' or 'dislike' either the product advertised in the commercial or the commercial itself. All three dimensions are measured on a scale from -1 to $+1$,

separately and independently by each of the annotators, using the vertical axis of a *joystick*. To generate a unique *gold standard* for each dimension and each recording (i. e., subject), an EWE is utilised, similar to the approach applied for RECOLA (see Section 6.1) and described in detail in the publication on AVEC [18].

While in AVEC 2017 [18], openXBOW features were employed exclusively in the baseline scripts of the **Emotion** sub-challenge, i. e., the task using the SEWA database for benchmarking, in the subsequent years, BoW representations across modalities were taken into account for all tasks. These were, in 2018 [19], the

- **Bipolar Disorder** (3 classes) sub-challenge, using the *Turkish Audio-Visual Bipolar Disorder Corpus* [365], and the
- **Gold-standard Emotion** sub-challenge, where participants were asked to propose novel methods of computing the *gold standard* (time-continuous arousal & valence), based on the annotations from six individuals. For this task, the RECOLA corpus, as introduced in Section 6.1, was exploited.

In 2019 [17], besides the **Cross-cultural Emotion** sub-challenge, the other tasks were the

- **State-of-Mind** sub-challenge (regression) with the *Ulm State-of-Mind* corpus [366], which had already been featured in the 2018 ComParE [12], and the
- **Detecting Depression** sub-challenge, using the *Extended Distress Analysis Interview Corpus* [367], with a regression task on a scale of 8 scores.

All corpora were partitioned into reasonable sets for *training*, *development*, and *test*, where the labels for the test partition have not been disclosed.

However, given that a systematic analysis of BoAW features and their comparison with functionals is only available for the **Cross-cultural Emotion** tasks, the results for the other sub-challenges are not reported in this thesis. As the focus of this thesis is on BoAW, i. e., on the **audio modality**, visual/facial features and corresponding results using BoVW are not presented in the following, but the reader is referred to the above mentioned publications on AVEC, where also the key statistics of all corpora are given.

6.5.1 BoAW and other acoustic features

AVEC 2017 For the baseline of AVEC 2017 [18], only the LLDs from eGeMAPS (see Section 2.3.2) are used and a **BoAW** representation is generated with a **fixed codebook size of 1 000**, after **z-score normalisation** of the LLDs. Also a **logarithmic term frequency weighting** is applied; all other options of openXBOW are the default ones. One BoAW feature vector is computed over a **block of 6 s length**, with a hop size of 100 ms to align with the labels.

AVEC 2018 In AVEC 2018 [19], a larger variety of acoustic descriptors and instance-level features are taken into account and evaluated to create the baseline. Again, the LLDs from eGeMAPS are considered and additionally, **MFCCs 1–13**, their **deltas**, and **double-deltas**. The **BoAW** features are computed using exactly the same configuration as in AVEC 2017, but with a **codebook size of 100** and a **block length of 4 s** for the **Cross-cultural Emotion** sub-challenge.

Besides BoAW representations, the performance is also evaluated for the 'official' eGeMAPS set with **functionals** (88 features in total) and the **mean** and **standard deviation** of the **MFCCs**, generating instance-level features. They are computed over the same fixed block size as the corresponding BoAW features.

Furthermore, DeepSpectrum features, as already introduced in Section 6.4.2, are considered for the AVEC 2018 baseline.

AVEC 2019 In AVEC 2019 [17], very similar to AVEC 2018, eGeMAPS and MFCCs with deltas and double-deltas are employed. Analogously, a sliding window of 4 s length, with a hop size of 100 ms, is used to segment the input into overlapping blocks. **BoAW** features are generated with a **fixed codebook size of 100**, one assignment per LLD frame, and also the other options as in AVEC 2018. Besides, for the **MFCCs**, an alternative representation is provided by computing the **functionals mean** and **standard deviation**. Finally, DeepSpectrum features are considered again, this time using a larger variety of pre-trained DNN architectures.

6.5.2 Experimental setup

The baseline approach(es) were implemented using exclusively open-source feature extraction and ML toolkits, to enable participants to reproduce the full processing pipeline.

AVEC 2017 In AVEC 2017, an **SVM regression** model was utilised for the baseline approach. All code was provided in **Python**, using the toolkit Scikit-learn [364], whose SVM implementation is based on Liblinear [300]. To take account of the inherent delay of annotations obtained through a real-time rating (cf. Section 6.1), a **delay compensation** is done. For this, the LLD sequence is shifted towards the back for each recording, while simply padding the missing frames at the front with the first LLD vector. A **delay** in the **interval of [0 s, 5 s]** with a **step size** of **0.2 s** is considered and tuned on the development set, together with the SVM **complexity**, which is optimised in the range of $[2^{-15}, 2^{-14}, \ldots, 2^{0}]$ (following a logarithmic scale). The optimisation is done independently for each dimension (*arousal*, *valence*, & *liking*). All further hyperparameters are the default ones of the LinearSVR ML model in Scikit-learn. For the final model, which

is evaluated on the test partition, the model is re-trained on the fusion of training and development sets, using the optimised delay and complexity.

AVEC 2018 The baseline scripts for AVEC 2018 were published as a GitHub repository[9]. A **2-layer LSTM-RNN** regressor is employed for the given sequence labelling task. The Keras framework[10] is used for the implementation and training of the model. The number of LSTM units is chosen as 64 for the 1st layer and as 32 for the 2nd, with **10 % dropout** in each RNN layer, followed by an output layer with one node for each target. All three targets (*arousal*, *valence*, & *liking*) are learnt together ('multi-task learning'), optimising a **CCC-based loss function**. The neural network is trained for a **maximum of 50 epochs**, where—independently for each target—the network weights are restored for the epoch achieving the highest CCC measure on the development set. In analogy with the findings for RECOLA (see Section 6.1 and SEWA in AVEC 2017, a **delay compensation** by time-shifting the labels was found to improve the results, even though the RNN model as a dynamic regressor has access to all past input. Nevertheless, the optimum of 2 s seems to be a lower value than the optimum for the static SVM regressor employed in AVEC 2017.

AVEC 2019 Also the baseline scripts for AVEC 2019 were published as a GitHub repository[11]. As in AVEC 2018, a **2-layer LSTM-RNN** (64 units & 32 units) is employed for the **Cross-cultural Emotion** task. The network architecture and hyperparameters for model training are exactly the same, however, the model for AVEC 2019 is trained and optimised on both the **German** and the **Hungarian** training and development sets. In contrast to AVEC 2018, all input features are *z-score normalised*.

6.5.3 Results

For the emotion recognition sub-challenges of AVEC, using the SEWA database, the **CCC**, as defined in Section 4.5.3, is taken into account for the ranking of the participants, while also PCC and MSE were computed during the evaluation of the submissions.

It must be noted that the results shown in the following are **only** those from the **models trained on acoustic features**. The official baselines for all sub-challenges of AVEC take into account all the modalities present in the data, i. e., also video and, for AVEC 2017, additionally transcriptions. Different fusion techniques have

[9] `https://github.com/AudioVisualEmotionChallenge/AVEC2018`
[10] `https://keras.io/`
[11] `https://github.com/AudioVisualEmotionChallenge/AVEC2019`

Dimension	**Culture**	**Partition**	BoAW eGeMAPS
Arousal	German	Development	.344
	German	Test	.225
Valence	German	Development	.351
	German	Test	.244

Table 6.20: Results of the AVEC 2017 affect recognition sub-challenge (SEWA database, German) for the acoustic modality. BoAW representations of eGeMAPS are used together with an SVM regressor. All results are in terms of CCC.

Dimension	**Culture**	**Partition**	Functionals		BoAW		Deep
			MFCCs	eGeMAPS	MFCCs	eGeMAPS	Spectrum
Arousal	German	Development	.279	.124	.282	**.421**	.332
	German	Test	.216	.112	.246	**.247**	.101
	Hungarian	Test	.218	.189	.185	**.226**	.238
Valence	German	Development	.253	.112	.306	**.398**	.276
	German	Test	.259	.111	.229	**.268**	.106
	Hungarian	Test	.080	.128	.098	**.166**	.040

Table 6.21: Results of the AVEC 2018 cross-cultural emotion sub-challenge (SEWA database) for the acoustic modality. A 2-layer LSTM-RNN is used for regression throughout. All results are in terms of CCC.

been employed for the sub-challenges across the years and for some sub-tasks, the acoustic modality is more meaningful than for others.

As shown in the AVEC 2017 baseline paper [18], the dimension of *liking* is not recognised at all through the acoustic modality alone. In fact, only the *linguistic domain*, based on the manual transcriptions of the dialogues, provides meaningful results across partitions. For this reason, in order to ensure clarity in the results tables, the performance measures achieved for *liking* are not shown in the following.

The results for the sub-challenges on the SEWA database are shown in Tables 6.20 to 6.22. For AVEC 2018, it can be seen that BoAW features based on eGeMAPS perform best consistently across all partitions and both German and Hungarian speech. Comparing the measures achieved for AVEC 2018 (Table 6.21) with the ones for AVEC 2017 (Table 6.20), there is some evidence that an LSTM-RNN outperforms an SVM regressor across both emotional dimensions.

In contrast to this, for *arousal* in AVEC 2019 (Table 6.22), the eGeMAPS-based BoAW is the best representation only for the German development set and the fused German and Hungarian development sets. For all test partitions, the functionals representation outperforms BoAW for this model trained on fused German and Hungarian training data. However, the main reason for this may be the feature normalisation, as described in the previous section. Nevertheless, for *valence*, the BoAW features based on eGeMAPS-LLDs are again those achieving the highest CCC, except for the Hungarian test partition, where DeepSpectrum features perform better.

Dimension	Culture	Partition	Functionals		BoAW		Deep
			MFCCs	eGeMAPS	MFCCs	eGeMAPS	Spectrum
Arousal	Ger. + Hung.	Development	.326	.371	.298	**.398**	.312
	German	Development	.389	.396	.323	**.434**	.380
	Hungarian	Development	**.326**	.305	.237	.291	.156
	German	Test	**.296**	.293	.164	.276	.133
	Hungarian	Test	.160	**.272**	.233	.250	.257
	Chinese	Test	**.145**	.100	.082	.107	.035
Valence	Ger. + Hung.	Development	.187	.286	.134	**.352**	.233
	German	Development	.344	.405	.190	**.455**	.317
	Hungarian	Development	.017	.073	.042	**.135**	.084
	German	Test	.288	.309	.066	**.325**	.105
	Hungarian	Test	-.019	.166	.083	.151	**.173**
	Chinese	Test	.166	.267	.160	**.281**	.024

Table 6.22: Results of the AVEC 2019 cross-cultural emotion sub-challenge (SEWA database) for the acoustic modality. A 2-layer LSTM-RNN is used for regression throughout. All results are in terms of CCC.

Concerning the *cross-cultural* emotion recognition, for *valence*, it might be surprising that the results for the Chinese culture are better than those obtained for the Hungarian culture, even though no Chinese data is used for training. However, based on this study alone, there is no clarity whether this can be explained rather by the complexity of the emotions typical for the one or the other culture, or by the diversity of emotional states present in the research corpus.

Regarding the *delay compensation*, for AVEC 2017, it was shown that for a static SVM-based regressor, the optimum shift is between 2 and 3 seconds for both arousal and valence [18], which is slightly less than the delay that was found optimum for RECOLA (see Section 6.1).

6.6 Input Representations for Emotion Recognition with LSTM

A follow-up work on time-continuous emotion recognition in speech, using BoAW features and the SEWA corpus [302] in the version exploited in AVEC 2017 [18], was published by the author under the title "Deep Recurrent Neural Networks for Emotion Recognition in Speech" [368]. The motivation for this research was the question—based on the success of eGeMAPS-LLDs based BoAW (see Section 6.5)—which would be the **optimal input representation** for a **deep LSTM-RNN**. Unless *raw* LLDs are fed into a DNN, besides the *type* (*functionals* or *BoAW*) and the corresponding *degrees of freedom* (e. g., the *codebook size* for BoAW), also the *block length/window size*, over which the summarisation is made, is a critical factor. In the extreme case of a block covering only a single frame (LLD), the corresponding BoAW feature vector is a *one-hot* encoded representation (see Sec-

tion 4.4)—in case of a single audio word assignment per frame. The block size can be controlled with the option `-t` in openXBOW.

In contrast to static ML models, such as SVM [18, 112], LSTM-RNNs are inherently suited for sequence labelling tasks, as they learn the required amount of context by themselves and have—in theory—access to all past and (in the bidirectional case) future information. The superiority of this architecture over SVM for the given task was—besides the insights from the AVEC baseline presented in the previous section—also found by the winners of AVEC 2017 [369] and earlier by Eyben et al. [370]. The authors of these works also made the observation that *multi-task learning* outperforms *single-task learning*, fitting a network to predict up to *five* emotional dimensions at the same time. The reasoning for this is that *cross-dependencies* between the targets can be expoloited. Furthermore, Eyben et al. arrive at the conclusion that summarising LLDs in a supra-segmental approach is better than using raw LLDs as input for the LSTM-RNN [370]. This is the justification for the experimental setup described in the following [368].

6.6.1 Supra-segmental features

Three types of supra-segmental features, i. e., approaches to summarise LLDs in overlapping blocks, with a hop size aligned with the target sequences (*arousal* & *valence*), are compared to each other:

1. The official 88-dimensional eGeMAPS feature set, with the functionals as described in Section 2.3.2.
2. The functionals *mean* and *standard deviation* applied to all 23 LLDs from eGeMAPS, resulting in a 46-dimensional feature vector for each block.
3. BoAW representations with codebooks generated by `random++` sampling on the training partition, logarithmic term frequency weighting, and the following four configurations for *codebook size* (C_S) and *number of assignments* (N_A):
   (a) C_S: 100, N_A: 1
   (b) C_S: 100, N_A: 10
   (c) C_S: 1 000, N_A: 1
   (d) C_S: 1 000, N_A: 10

Each feature space is augmented by a an additional uni-dimensional feature, based on the manual speaker turn annotations, indicating if the *target speaker* is audible or not (1 or 0). The reason for this is that in the SEWA database, the speech of both speakers is present in the audio and encoding target speaker information is meaningful. This is also referred to as 'mark method' by Huang et al. [371]. The supra-segmental features are computed for each time step of 100 ms, in order to match with the step size of the targets.

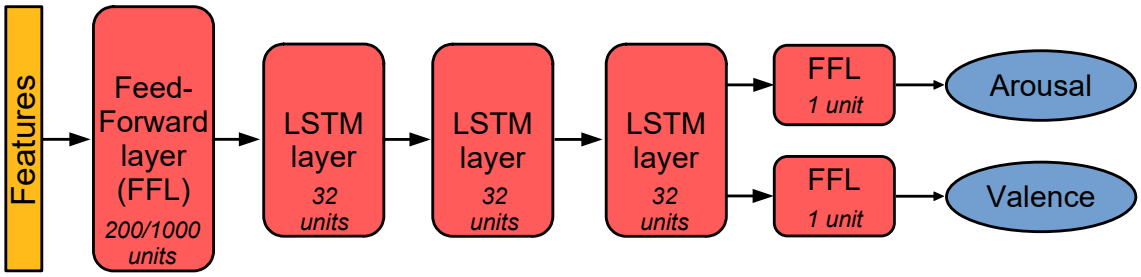

Figure 6.3: Recurrent neural network architecture for time-continuous emotion recognition on the SEWA corpus. The step size of features and labels is 100 ms in each layer of the network. Activation functions are *tanh*, except for the output layers, where one feedforward unit with a *linear* activation is used. In the first layer, the number of units is optimised (either 1 000 or 200).

6.6.2 Neural network

Experiments are based on the neural network architecture shown in Figure 6.3. The network consists of four layers, where the first one is a (time-distributed) feedforward layer (fully-connected) with *tanh* activation and an optimised number of neurons. 1 000 and 200 are taken into account, respecting the differences in the dimensionality of the input feature space. Next, three LSTM layers with *tanh* activation, which is default for LSTM, are employed, each layer with a constant amount of 32 LSTM units. For each target (*arousal* & *valence*), an output layer is present in the form of a feedforward layer with a single unit and a *linear* activation. A dropout with a rate of 10 % is employed in each layer.

Initial experiments have shown that the performance is approximately the same when utilising an LSTM layer instead of the first feedforward layer or using more units in the LSTM layers, but at the cost of a higher complexity and a larger number of parameters. Moreover, all LSTM layers are *unidirectional*, as the quality of the predictions has proven to be similar; a finding that has already been made by Trigeorgis et al. [228].

6.6.3 Experimental setup

As done for the AVEC baseline [18], a **delay compensation** is taken into account. In initial experiments, it was found that the optimum time interval for the shift can be approximated by the *half of the block size* for the functionals and BoAW computation. This block size-dependent delay is used throughout the experiments.

The neural network is trained by optimising a **CCC-based loss function**, more precisely, $1 - \rho_{\mathrm{CCC}}$ is minimised, where the loss is averaged across all sequences from the training set (full batch). Training is run for a maximum of *300 epochs* with *RMSprop* optimiser, considering **early stopping** with a *patience* of 10 epochs, but

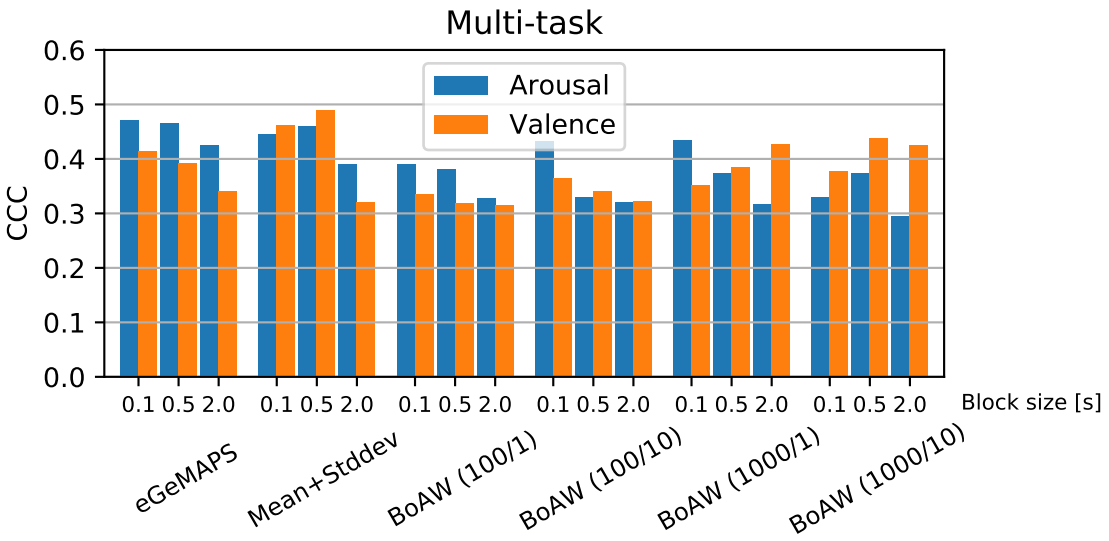

Figure 6.4: Results for emotion recognition on the SEWA corpus (test partition of AVEC 2017), using an LSTM-RNN architecture and **multi-task** learning. All measures are CCC. Results are shown for six different supra-segmental feature representations of the eGeMAPS-LLDs and three block sizes; two of the feature representations are based on functionals (original eGeMAPS functionals / *mean* and *standard deviation* [Mean+Stddev] for each LLD), four of them are BoAW representations with different codebook sizes (C_S) and numbers of assignments (N_A) [BoAW (C_S/N_A)].

weights from the epoch achieving the highest CCC on the development set are restored. Besides this, also the *learning rate* is optimised on the development set [range between 2^{-4} and 2^{-3}] and, as mentioned above, the *number of neurons* in the first layer [200 or 1 000].

6.6.4 Results

The whole training and hyperparameter optimisation process, as described in the previous section, is conducted for all **six feature representations** (defined in Subsection 6.6.1) and **three block sizes** for the summarisation: 0.1 *s*, 0.5 *s*, & 2.0 *s*. Results on the **test partition** with **multi-task learning** are shown in Figure 6.4. It is observable that a short block size of 0.1 s or 0.5 s leads to better results, except for the BoAW features with a codebook size of 1 000 for the prediction of *valence*. A potential reason for this behaviour might be the large degree of *sparsity* when using a large codebook size; an effect that is attenuated when summarising over a longer block. Similar to the findings in Section 6.1 with the RECOLA corpus, by trend, *valence* seems to require a larger codebook size than *arousal*. Nevertheless, the best

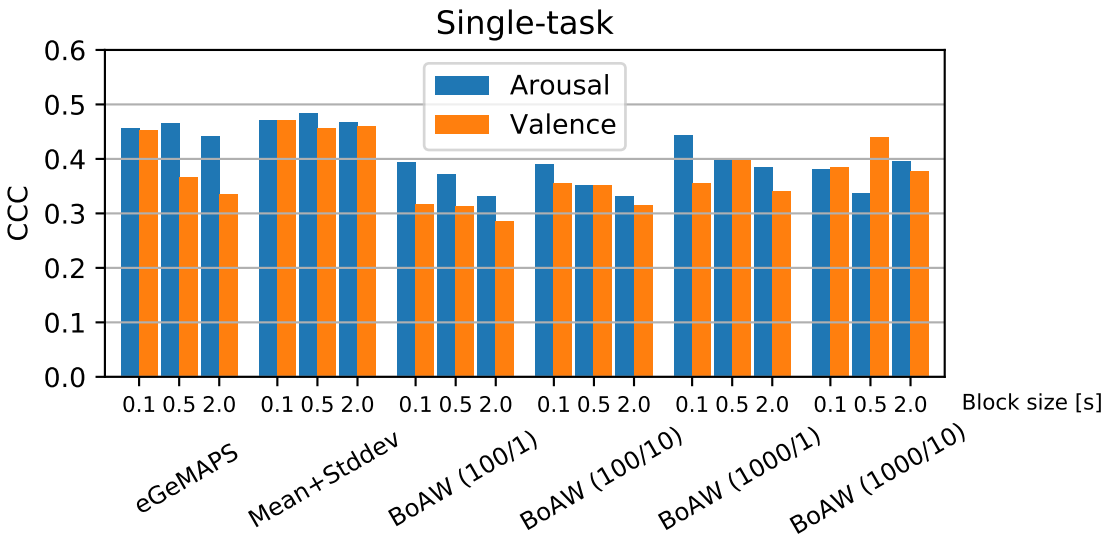

Figure 6.5: Results for emotion recognition on the SEWA corpus (test partition of AVEC 2017), using an LSTM-RNN architecture and **single-task** learning. All measures are CCC. Results are shown for six different supra-segmental feature representations of the eGeMAPS-LLDs and three block sizes; two of the feature representations are based on functionals (original eGeMAPS functionals / *mean* and *standard deviation* [Mean+Stddev] for each LLD), four of them are BoAW representations with different codebook sizes (C_S) and numbers of assignments (N_A) [BoAW (C_S/N_A)].

results are obtained without BoAW features, for *arousal* by using all eGeMAPS functionals and for *valence* using only the *mean* and the *standard deviation*. As described in Section 2.3.2, the main difference is that in the full set, the *coefficient of variation* is used instead of the *standard deviation*, and some more functionals (e. g., percentiles) are computed for *F0* and *loudness*.

The results with **single-task learning** in Figure 6.5 show a similar picture overall. The superiority of employing *mean + standard deviation only* is more evident here. On average, the results with single-task learning are even better than those achieved by multi-task learning, which is contrary to the findings in related works [115, 370].

Table 6.23 shows the results for both development (Dev) and test partition. The results for the proposed LSTM-based model are reported for both multi-task and single-task learning and have been selected by strictly choosing the combination of feature representation and block length that is optimum on the development set. The optimum feature representation is *mean + standard deviation* in all cases except for *valence* with single-task learning, where the original eGeMAPS set performs best.

Method	Ref.	Arousal		Valence	
		Dev	Test	Dev	Test
AVEC 2017 Baseline – audio-only	[18]	.344	.225	.351	.244
AVEC 2017 Baseline – multimodal	[18]	.373	.375	.507	.466
AVEC 2017 Winner (Chen et al.) – audio-only	[369]	—	.437	—	.494
AVEC 2017 Winner (Chen et al.) – multimodal	[369]	.823	.672	.796	.756
Proposed LSTM model, multi-task – audio-only	[368]	.586	.460	.506	.490
Proposed LSTM model, single-task – audio-only	[368]	.607	.467	.549	.452
BLSTM model with uncertainty modelling (Han et al.) – audio-only	[115]	.356	.275	.396	.292
BLSTM model with uncertainty modelling (Han et al.) – multimodal	[115]	.559	.450	.575	.515
CNN model – audio-only	[372]	.564	.536	.547	.479

Table 6.23: Results for emotion recognition on the SEWA corpus, comparison with AVEC baseline, AVEC winners, and two further approaches with involvement of the author (Dev: Development). All measures are CCC.

The measures achieved with the proposed model clearly **outperform the multimodal AVEC 2017 baseline** and the **audio-only results of the AVEC 2017 winners** for *arousal* and are competitive with their results for *valence*. They used a combination of hand-crafted acoustic features and SoundNet, which are deep features learnt frow the audio waveform in a kind of *transfer learning* approach on multimedia data [285, 369].

The proposed model performs in the same range as the *multimodal* 2-layer BLSTM-RNN model by Han et al. [115], while outperforming the *audio-only* one, using ComParE-LLDs, by a large margin. Also their approach employs multi-task learning, taking into account not only the emotional dimensions as targets but also the *uncertainty*, i. e., the standard deviation between the ratings of the single annotators.

An extension of the proposed approach was published by the author in the work entitled "Continuous Emotion Recognition in Speech—Do We Need Recurrence?" at the Interspeech conference in 2019 [372]. It is challenged that the *recurrence* property of LSTM is actually not of fundamental importance for time-continuous emotion recognition and that a **fully convolutional** neural network performs even better, based on the experiments. As acoustic features, also here, the 23 eGeMAPS-LLDs were employed and summarised in blocks of 100 ms length by computing their *mean + standard deviation*. Also the additional feature marking the *target speaker* is concatenated. A 4-layer CNN is employed as a model, with increasing filter lengths (5–50 steps, corresponding to 0.5 s–5 s) and $(200, 64, 32, 32)$ filters in the corresponding layers. The results in Table 6.23 show that the LSTM model is outperformed for both *arousal* and *valence* on the test partition.

6.7 The EAT Challenge

Finally, openXBOW was providing baseline features for the Eating Analysis and Tracking (EAT) challenge [84], a one-time event organised as part of the

International Conference on Multimodal Interaction in 2018[12]. As the name of the conference suggest, the used database, iHEARu-EAT [373], contains audio-visual recordings of subjects *speaking while eating certain types of food*. The *audio* part of the database had already been exploited as the **Eating Condition** task of ComParE 2015 [21]. Three sub-challenges were offered in the EAT challenge:

1. **Food Type**, a 7-class task, to recognise the following food types: *Apple*, *Banana*, *Biscuit*, *Crisps*, *Gummibear*, *Nectarine*, and a *No Food* class. This task had already been set for ComParE 2015.

2. **Likability**, a binary classification task: *Neutral* or *Like*.

3. **Difficulty**, a classification or regression task on a 5-point *Likert* scale, where subjects rated the effort of speaking while eating.

In total, **30** German speaking **subjects** (15 female, 15 male; mean age of 26 years) were recruited and recorded in a low reverberant office room. The subjects were allowed to leave out food types they did not want to eat for any reason. Chunks of both *read* and *spontaneous* speech are included in the recordings; in total, **2 h 53 min of audio**. More details on the experiments are found in the corresponding references [84, 373]. After each experiment, the subject themself rated the **Likability** and **Difficulty**. The speaker-disjunct partitioning of ComParE was reused and is shown in Table 7.1 in Chapter 7, where more results are shown, denoting the numbers of instances for the 'main task' of recognising **Food Type**. No development set is provided for the data but subject IDs for the training set (20 speakers) as hyperparameter optimisation is supposed to be done using LOSO CV. As before, the presentation of models and results in this thesis is focussed on the audio domain.

6.7.1 BoAW

The BoAW representation is based on the 130 ComParE-LLDs (including deltas) (see Section 2.3.1). They are **z-score normalised** and—different from ComParE (see Section 6.4)—split into **8 feature groups**. For each group, a BoAW feature vector is generated and then fused on histogram-level (bag-level fusion):

1. RMS energy, ZCR, F0, jitter, shimmer, voicing probability
2. RASTA
3. Spectral descriptors
4. MFCCs

[12] `https://icmi-eat.ihearu-play.eu/`

	Food Type UAR [%]		Likability UAR [%]		Difficulty CCC	
C_S	Lo	Test	Lo	Test	Lo	Test
200	58.6	65.6	64.8	51.7	.439	.506
400	63.1	66.8	64.0	50.9	.432	.482
600	61.5	68.3	64.6	51.5	.455	.530
800	63.1	66.7	64.5	50.6	.446	.500
1 000	63.7	67.1	64.7	49.8	.440	.503
1 200	**64.3**	**67.2**	**66.5**	**54.2**	.451	.515
1 400	63.5	68.2	65.7	52.2	**.470**	**.506**
1 600	63.5	67.1	65.7	53.5	.466	.505
1 800	63.8	66.7	65.4	53.0	.466	.510
2 000	63.1	68.1	65.3	53.4	.469	.505

Table 6.24: Results for the EAT challenge, optimising the BoAW codebook size. The model performing best in LOSO CV (Lo) is chosen for each sub-challenge (in **bold**).

Groups 5–8 consist of the **deltas** of feature groups 1–4. The **number of assignments** is fixed at **1**, the **codebook size is optimised** in the range between 200 and 2 000, with a step size of 200. It should be pointed out that the codebook size applies to *each feature group* and the final feature vector is 8-times larger than the employed codebook size. As usually, the term frequencies are transformed applying a **logarithmic weighting**.

6.7.2 Experimental setup

The baseline scripts for the EAT challenge are provided as Python scripts. The ComParE feature set is extracted using openSMILE [23], the BoAW features are generated with openXBOW, and Scikit-learn [364] is utilised for ML model training. An SVM classifier is learnt for each task, using LinearSVC with default options. The **complexity hyperparameter** is optimised using LOSO CV in the range [1e−5, 1e−4, 1e−3, 1e−2, 1e−1, 1e0].

Besides the BoAW+SVM approach, also an E2E system was provided by the organisers. The E2E model takes the raw signals (audio waveform and/or video) as input and consists of a stack of convolutional layers and LSTM layers. The toolkit End2You [363] is employed; more details on the domain-specific pre-processing and the architecture are found in the paper on the challenge [84].

6.7.3 Results

The models for **Food Type** and **Likability** are evaluated in terms of the UAR metric and the models for **Difficulty** are evaluated in terms of the CCC. Table 6.24 shows the results for all tasks and the codebook sizes in the considered range. The optimum codebook size is between 1 200 and 1 400 for all three tasks (1 400 for the

Modality	openXBOW		End2You	
	Lo	Test	Dev	Test
Food Type				
Audio-only	64.3 %	**67.2 %**	35.2 %	32.8 %
Video-only	28.0 %	27.0 %	27.1 %	24.5 %
Audio+Video	63.9 %	67.0 %	34.8 %	33.6 %
Likability				
Audio-only	66.5 %	54.2 %	55.1 %	53.7 %
Video-only	55.9 %	**58.3 %**	52.9 %	50.9 %
Audio+Video	65.5 %	51.8 %	55.1 %	54.2 %
Difficulty				
Audio-only	.470	**.506**	.342	.323
Video-only	.246	.252	.264	.220
Audio+Video	.481	.501	.345	.311

Table 6.25: Results for the EAT challenge, comparing the BoAW/BoVW approach with E2E, considering single modalities (audio/video) and their fusion. Models based on openXBOW are optimised with LOSO CV (Lo), E2E models are optimised on a development split consisting of 5 subjects (Dev). Metrics: UAR for Food Type and Likability, CCC for Difficulty. The official baseline result for each task is highlighted in **bold**.

Difficulty task), however, the performance is quite similar over the whole considered range and drops only with the lowest size of 200 for the Food Type task.

Table 6.25 shows the results for both approaches; for the sake of completeness, also the results achieved with the **visual modality** and an **early fusion** of both are shown here. The BoAW approach outperforms E2E learning in all three tasks. While for the Food Type and Difficulty tasks, the acoustic domain provides the best results on the test set, for the Likability task, the visual domain in terms of BoVW features (based on *facial landmarks* as LLDs) is best; however, the general 'accuracy' of the predictions is worse than for the Food Type task, given that it is a binary classification problem. A fusion of BoAW and BoVW does not improve the results.

In the EAT challenge, baseline results were **surpassed** only for the **Food Type** task. The best result was achieved by a research group with involvement of one of the organisers, by Sertolli et al. [374]. They extracted features generated by a CNN pre-trained for the task of ASR and employed *compact bilinear pooling* [375] to combine information from different layers. With a (unidirectional) 2-layer LSTM-RNN, a UAR of **73.3 %** on the test set is obtained for the Food Type task. The official **winner** of the challenge, Pir [376], reached a UAR of **69.1 %** using a novel 'functional-based acoustic group feature selection' method.

Furthermore, Guo et al. [329] showed that the provided BoAW features are indeed complementary with both *functionals*-based representations and *deep representations* and that an **early fusion** of all three results in the highest UAR. In a work outside of the context of the EAT challenge, Gosztolya proves that also a **late fusion** of ComParE, BoAW computed on different MFCCs, and BoAW computed

from DNN posteriors improves the Food Type recognition performance on the EAT database [314].

7

Systematic Evaluation

Based on the findings in the survey on BoAW in Section 3.2.2 and the results from the case studies in Chapter 6, some fundamental aspects of the BoAW method still remain unclear. This is the motivation for a more systematic evaluation of BoAW configurations and hyperparameters, across **four different datasets**.

The following section sets out the open research questions. In Section 7.2, the considered datasets and their key statistics are introduced. Section 7.3 describes the experimental setup that applies to all experiments. Then, in Section 7.4, the results of a number of experiments on certain aspects are presented and shortly discussed.

7.1 Open Research Questions

A central step of the BoAW approach is the **codebook generation method**. In the experiments presented in the previous chapter, the `random++` method of openXBOW was used throughout, i. e., the codebooks were constructed from the centroids resulting from the *initialisation step* of k-means++ clustering [216]. Nevertheless, as mentioned in Section 3.2.2, there is not enough evidence that this method is superior to either a pure random sampling from the training set or to clustering. Moreover, previous works obtained contradictory results on the usefulness of EM clustering (cf. [25] and [188]).

Generally, the suitability of one or another method might also depend on the *codebook size*, the *LLD set*, or the *data* itself. For this reason, a comparison of **codebook generation techniques** evaluated across datasets, input features, and configurations is performed in the **1st experiment**, in Section 7.4.1. This corresponds to **RQ 1** defined in the introduction (Section 1.3).

Concerning the **SVM kernel**, as mentioned in Section 4.3.2, only a *non-significant* performance gain was recognised with HIK or GHIK, compared to the standard *linear* kernel, yet depending on the type of histogram normalisation. No strong conclusion has been drawn so far. In the **2nd experiment** in Section 7.4.2,

evaluations comparing **linear kernel** and **GHIK** [246] are offered, employing similar configurations as in the 1st experiment.

In the **3rd experiment**, presented in Section 7.4.3, it is investigated if the finding from Section 6.3 that the **deltas** of LLDs are more meaningful than the LLDs themselves generalises also to the datasets and tasks taken into account in this chapter.

The method of **encoding** is investigated in Section 7.4.4 in the **4th experiment**, by comparing *hard* (single- & multi-assignment) and *soft* assignment with each other. In related works (Section 3.2.2), some authors have found that a GMM [187] or other methods of soft encoding [199, 201] provide better representations than a hard VQ. Nevertheless, there are also works concluding that a soft quantisation gives worse results [25]. Moreover, the influence of a *logarithmic term frequency* (log-TF) weighting is examined.

In the **5th experiment**, presented in Section 7.4.5, BoAW generated from different LLD sets and two ways of splitting the large-scale ComParE feature set into subsets, is evaluated.

Finally, in the **6th experiment** in Section 7.4.6, the BoAW method is compared to the classical functionals-based instance-level features. Moreover, results achieved for a fusion of both are reported and compared to further approaches. This experiment is a basis for answering **RQ 2** and **RQ 3** defined in the introduction (Section 1.3).

7.2 Description of the Datasets

The experiments are performed across the following four corpora:

1. iHEARu-EAT [21, 84, 373]
2. Berlin EmoDB [47]
3. MPSSC: The *Munich-Passau Snore Sound Corpus* [11, 117]
4. MASC: The *Mask Augsburg Speech Corpus* [116]

The **key statistics** of all datasets are shown in Tables 7.1 to 7.4. The following criteria were taken into account in the selection of these datasets:

- All tasks are **classification tasks** and measured in terms of the **UAR**, which means that the **same training and evaluation protocol** can be applied.
- The selected corpora are **well-studied** and **public**, with various publications using them.
- The corpora show a **large variety** in terms of **number of speakers/subjects, instances, classes, and chunk durations**.

Food Type	Training	Test	Total
Apple	140	56	196
Banana	140	70	210
Biscuit	133	70	203
Crisps	140	70	210
Gummibear	119	70	189
Nectarine	133	63	196
No Food	140	70	210
Total	945	469	1 414
Number of subjects	20	10	30
Total duration (h:mm:ss)	1:52:22	1:00:39	2:53:01

Table 7.1: Key statistics of the iHEARu-EAT database. The number of instances of the training and test partition are given for each Food Type class.

Emotion	Training	Test	Total
Anger	78	49	127
Boredom	54	27	81
Disgust	36	10	46
Fear	50	19	69
Happiness	45	26	71
Neutral	45	34	79
Sadness	39	23	62
Total	347	188	535
Number of subjects	6	4	10
Total duration (h:mm:ss)	0:16:10	0:08:36	0:24:46

Table 7.2: Key statistics of Berlin EmoDB. The partitioning is done by the author, using subjects with IDs 03, 08, 09, & 10 for testing and the remaining ones for training and optimisation (LOSO CV).

- Different **types of audio data** are present in the corpora: Three of them contain **speech** (spontaneous and/or read), one (MPSSC) contains **sounds** produced in the vocal tract.

The iHEARu-EAT [84] corpus has already been introduced in Section 6.7 in the context of the EAT challenge, where it was used for as a benchmark database. Only the 7-class **Food Type** task is taken into account in this chapter.

In Berlin EmoDB [47], speech in six 'basic' emotional states (*anger*, *boredom*, *disgust*, *fear*, *happiness*, *sadness*) and a *neutral* reference are present, i. e., seven classes. Classification on this corpus is a relatively simply task and high accuracies are expected as the emotions are *acted* (not *natural* as, e. g., in SEWA [302]) and recorded in a laboratory setting. The speech is *scripted*, i. e., the chunks contain utterances from a pool of 10 sentences. Berlin EmoDB does not come with a partitioning, thus, speakers with IDs 03, 08, 09, & 10 form the *test set* while the other six speakers are used for training and optimisation (LOSO CV).

MASC is a relatively large corpus and was first used in ComParE 2020 [116] (see Section 6.4). In total, 32 subjects (16 female, 16 male) were recorded *with* and *without* wearing a *surgical mask*. A variety of free speech and scripted utterances were collected in both scenarios and segmented into chunks of 1 s each, totalling up

Class	Training	Development	Test	Total
No mask	5 353	6 666	5 553	17 572
Mask	5 542	7 981	5 459	18 982
Total	10 895	14 647	11 012	36 554
Number of subjects	12	10	10	32
Total duration (h:mm:ss)	3:01:35	4:04:07	3:03:32	10:09:14

Table 7.3: Key statistics of MASC.

Type	Training	Development	Test	Total
V	168	161	155	484
O	76	75	65	216
T	8	15	16	39
E	30	32	27	89
Total	282	283	263	828
Number of subjects				219
Total duration (h:mm:ss)	0:07:02	0:07:15	0:06:32	0:20:49

Table 7.4: Key statistics of MPSSC. Four types of snoring are discriminated: V, O, T, E; see Section 6.2.

to 36 554 instances. The chunks were split into subject-disjunct and gender-balanced partitions, as shown in Table 7.3. MASC is the only one of the four datasets in this chapter with a fixed instance length (1 s).

MPSSC was first used in ComParE 2017 [11] (see Section 6.4). Similar to the corpus discussed in Section 6.2, it contains recordings from subjects *snoring during drug-induced sleep*, annotated by experts according to the location of excitation in the throat. Analogously, four levels or classes are distinguished: *V*, *O*, *T*, & *E* (see Figure 6.2).

All audio data is given **uncompressed** as **mono** files, with a **bit depth** of **16 bit** and a **sample rate** of **16 kHz**, except for the samples from iHEARu-EAT, which are given in **44.1 kHz**. More details on the corpora, recording protocols, and subjects present are found in the corresponding publications mentioned above.

7.3 Experimental Setup

The BoAW are based on the LLDs from ComParE and eGeMAPS feature sets as introduced in Section 2.3, extracted with openSMILE [23]. Scikit-learn [364] is used to train and evaluate the ML models. As classifier, an **SVM** is employed throughout; the **kernel** is, except otherwise noted, a **linear** one, i. e., LinearSVC (which is based on Liblinear [300]), with the default configuration: the *dual* problem is solved, using *squared hinge loss* and *L2* penalisation. A **one-vs-all** scheme is applied to handle the given multi-class classification problems. The reasoning for choosing only SVMs in the following experiments is as follows:

- As already mentioned by Joachims [27], SVM is well suited for BoW features, and more generally, for classification in *high-dimensional*, and potentially *sparse* feature spaces.
- Most of the related works listed in Section 3.2.2 utilise SVM.
- SVMs, especially with a linear kernel, a relatively fast to train while still achieving competitive results. The only critical *hyperparameter* to be optimised is the *complexity*; thus, much less tuning and training is required, most importantly, in comparison with DNN models.
- A fast and robust classification method is necessary as the focus of this chapter is the evaluation and comparison of BoAW features, suppressing the effect of the classifier.

As a *pre-processing*, all instances are first **balanced**, by duplicating the instances of all minority classes in the given order until they exactly match the number of instances of the majority class. Then, an *on-line* **min-max normalisation** (between 0.0 and 1.0) is performed, by estimating the parameters from the training set only (for the databases where LOSO CV is used and for the optimisation process on the development set), or, from fused training and development sets (for the final model for the databases with three partitions), respectively.

The **complexity** hyperparameter is optimised in the following range: [1e−5, 1e−4, 1e−3, 1e−2, 1e−1, 1e0]. As mentioned, for the experiments on iHEARu-EAT and Berlin EmoDB, which expose a relatively low number of subjects and samples at the same time, a LOSO CV is taken into account for the optimisation. For MASC and MPSSC, the complexity is optimised on the development set.

In previous experiments, it has been found that the performance of ML models based on BoAW features depend on the **random seed** used for codebook generation, i. e., a meaningful variance is observed when re-running the same experiment with different seeds (option `-seed` in openXBOW). Thus, in the following experiments, for each BoAW configuration, the process is run **10 times**, employing random seeds 10 (default) to 19 in openXBOW. The mean and the standard deviation of the UARs for each configuration are reported.

7.4 Experiments and Results

Six experiments that seek to answer the open questions posed are presented in the following.

7.4.1 Codebook generation method

In the **1st experiment**, the influence of the codebook generation method is studied. Two sets of acoustic LLDs are taken into account:

- MFCCs 1–14 + RMS energy (15 LLDs)
- eGeMAPS [31] (23 LLDs)

These LLD sets have already proven to achieve suitable representations for various tasks, as shown in Chapter 6. For each LLD set, based on experience gained in previous experiments [112], two configurations of codebook size and number of assignments are considered:

⟨1⟩ $C_S = 100$, $N_A = 1$

⟨2⟩ $C_S = 1\,000$, $N_A = 10$

The corresponding codebooks are generated comparing **8 different methods**, as explained in Section 5.2.1:

1. Sampling from a Gaussian PDF (`pdf`).
2. Random sampling from the training set (`random`).
3. Random++ sampling from the training set (`random++`), considering two *sub-samplings* (`numTrain`) of the training LLDs: (a) 50 000 and (b) 10 000.
4. K-means++ clustering on the training set (`kmeans++`), considering two *sub-samplings* (`numTrain`) of the training LLDs: (a) 50 000 and (b) 10 000.
5. EM clustering on the training set with k-means++ initialisation (`em-kmeans++`), considering two *sub-samplings* (`numTrain`) of the training LLDs: (a) 50 000 and (b) 10 000.

A **z-score normalisation** ('standardisation') of the input (LLDs) and a **log-TF** weighting of the BoAW features are performed throughout. The motivation for considering two types of sub-samplings for 'random++' and the two clustering methods is that sub-sampling increases the codebook training time significantly and a negligible influence would be beneficial when dealing with large-scale datasets. The results in terms of the UAR, averaged over 10 random seeds, are reported in Figure 7.1.

Overall, a pure **random sampling** (option `random` in openXBOW, without the initialisation from k-means++) from the training LLDs is the **best option**. Only for the **MFCCs**, an **EM or k-means++ clustering** might be preferred, which is especially evident for configuration ⟨1⟩ ($C_S = 100$, $N_A = 1$), i. e., a small codebook size. Concerning **EM clustering**, it is evident that it works relatively **well for MFCCs**, but **worse for eGeMAPS** in most cases. A potential reason for this is that MFCCs, as mentioned before (Section 6.1), are normally distributed, i. e., it is straightforward to model them with a GMM (EM clustering), whereas this property does not apply to all LLDs in eGeMAPS. This is further supported by the observation that **sampling from a Gaussian PDF** seems to be a **reasonable**

choice for the MFCC-based LLD set. This is a meaningful finding as this codebook generation method is the only one discussed here that is fully **data-independent** and thus, suitable also for cross-corpus recognition tasks.

It must be noted that all findings are relatively consistent across databases. Only for **MPSSC**, the results differ evidently: Especially for configuration ⟨1⟩ and MFCCs, there is a **mismatch** for evaluation between development and test conditions. While a *clustering*-based codebook generation seems to perform better on the development set, the *sampling* methods perform better on the test set. This is surprising as the codebooks are kept constant between these conditions and are not re-trained. Anyway, it must be noted that the *error bars* are generally larger for the small MPSSC than for the other corpora.

There is no conclusion whether a clustering on more (50 000) or less (10 000) data leads to better results; however, it can be stated that the **sub-sampling** of the training data has **less influence** on the final result than the codebook generation method itself.

It may be noted that for the test partitions of iHEARu-EAT and MASC, with eGeMAPS, **statistical significance** w. r. t. *Welch's t-test* ($p < 0.05$) is achieved for the **random sampling** (`random`) method, compared to all other methods. A distinct 'winning' approach, however, cannot be identified for the other two databases and the MFCC-based LLDs.

7.4.2 SVM kernel

In the **2nd experiment**, the performance of SVMs with GHIK are compared to those with the linear kernel employed so far. Again, the same two sets of acoustic LLDs from the 1st experiment and configurations ⟨1⟩ and ⟨2⟩ are taken into account (see Section 7.4.1). Codebooks are generated with the **random sampling** (`random`) method, as this method provided the best overall results.

A **z-score normalisation** of all LLDs is performed. The term frequency histograms are subject to a **logarithmic weighting** (log-TF), as this procedure has shown a consistent performance gain in the experience of the author. As described above, **min-max normalisation** is done as the final processing step before SVM training, however, the findings in the following experiments were qualitatively the same without normalisation.

Besides the complexity, also the hyperparameter β of the GHIK is optimised in the same way. The following range is taken into account for β: [0.5, 1.0, 1.5, 2.0, 2.5, 3.0]. The results in terms of the UAR, averaged over 10 random seeds, are reported in Figure 7.2.

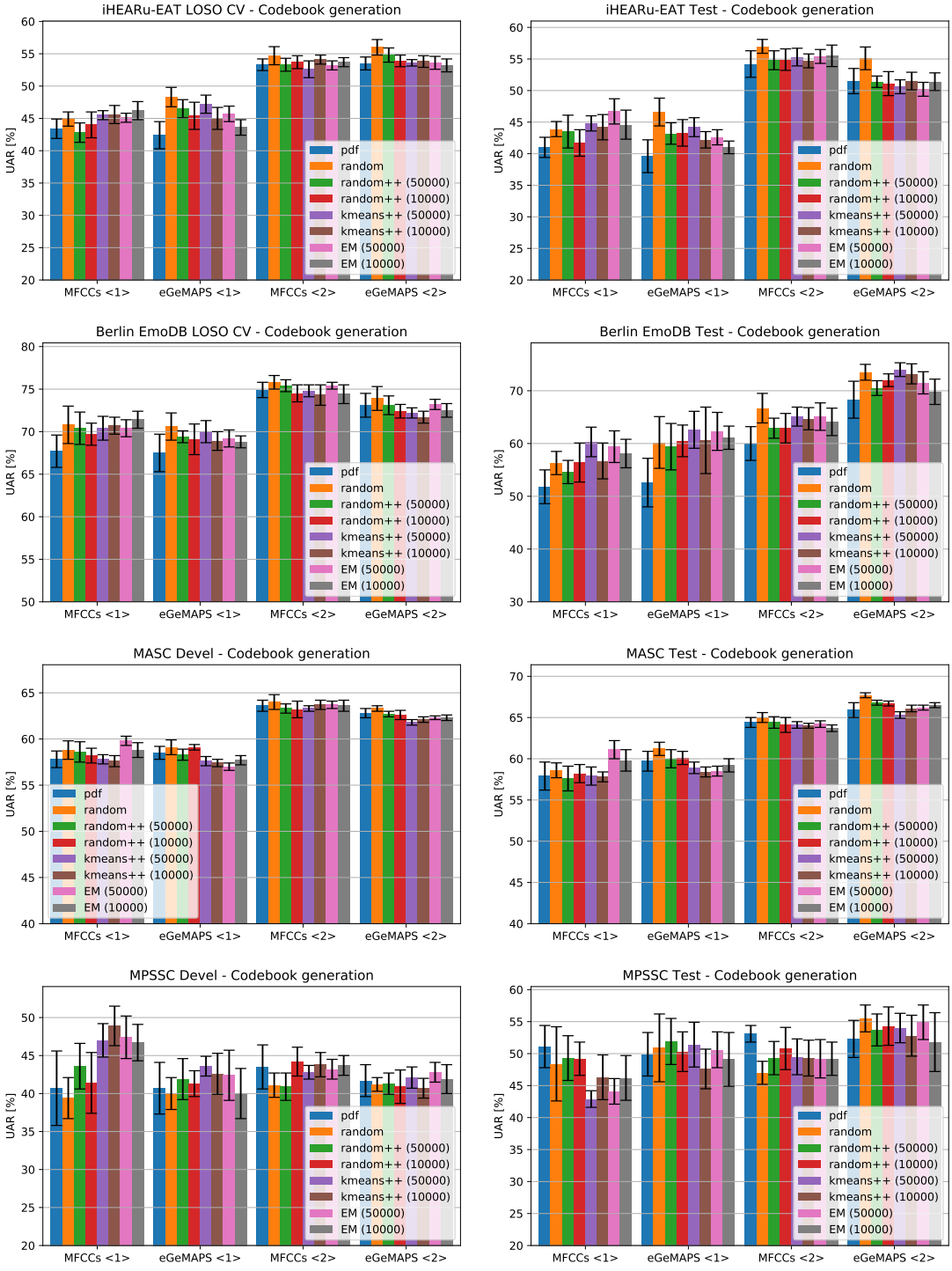

Figure 7.1: Results of the 1st experiment, comparing the influence of the codebook generation method. The UARs (mean and standard deviation over 10 random seeds in openXBOW) achieved for the Development set (Devel) or LOSO CV and Test set, respectively, for each of the four considered datasets, are reported. Eight codebook generation methods (in different colours) are compared to each other. Experiments are run for two LLD sets (MFCCs: MFCCs 1–14 + RMS energy and eGeMAPS) and two configurations of C_S and N_A: $\langle 1 \rangle$ & $\langle 2 \rangle$ as specified in the main text.

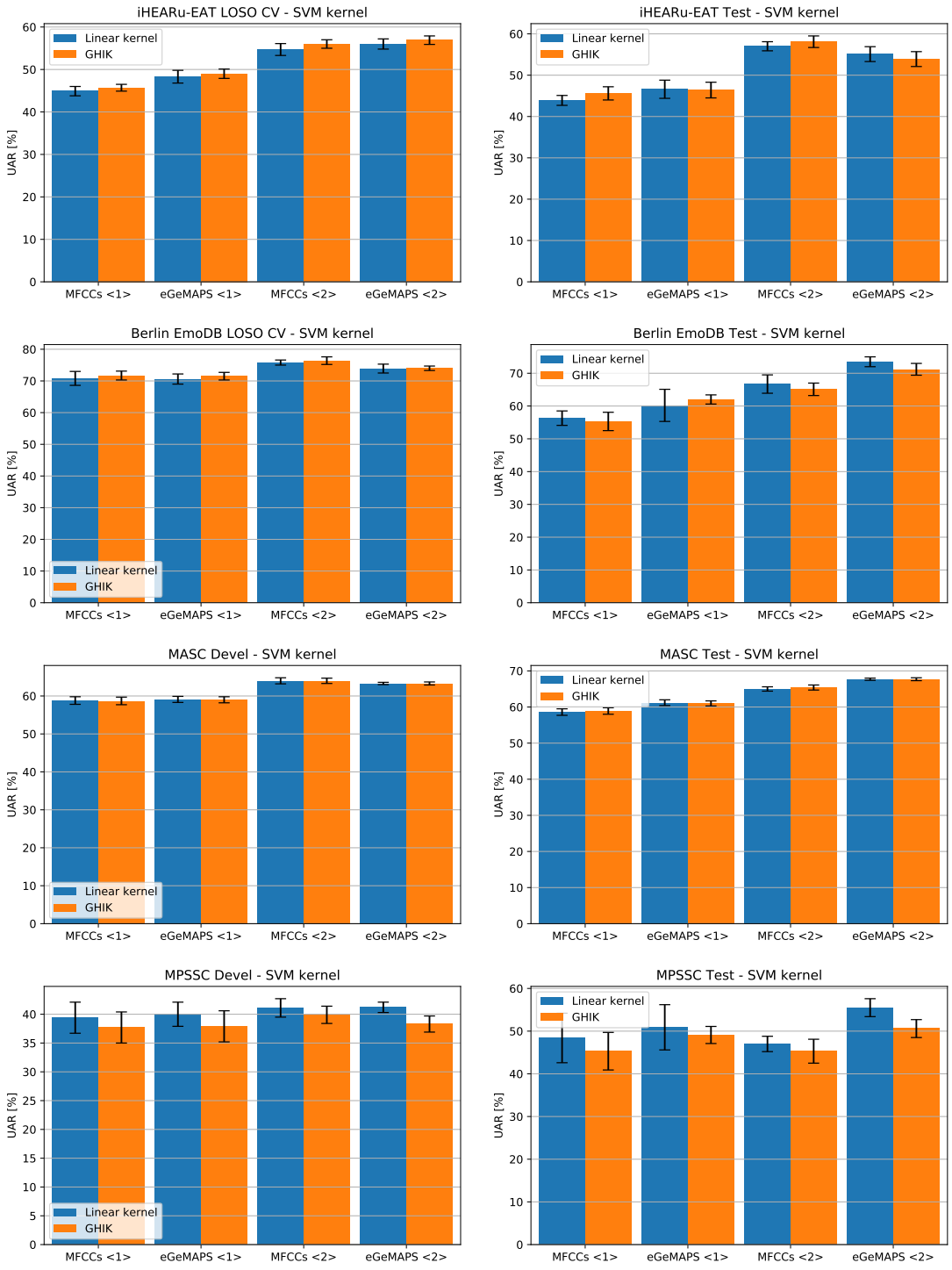

Figure 7.2: Results of the 2nd experiment, comparing linear kernel and GHIK for SVM. The UARs (mean and standard deviation over 10 random seeds in openXBOW) achieved for the Development set (Devel) or LOSO CV and Test set, respectively, for each of the four considered datasets, are reported. Experiments are run for two LLD sets (MFCCs: MFCCs 1–14 + RMS energy and eGeMAPS) and two configurations of C_S and N_A: $\langle 1 \rangle$ & $\langle 2 \rangle$ as specified in the main text.

GHIK outperforms a linear kernel **significantly** (*Welch's t-test*, $p < 0.05$, test set) only for iHEARu-EAT with MFCCs and configuration ⟨2⟩. Nevertheless, GHIK performs **worse** than a linear kernel[1] for MPSSC. For Berlin EmoDB and MASC, the differences are **not statistically significant**. Furthermore, for configurations with a higher degree of *sparseness*, i. e., increasing the ratio between *codebook size* and *number of assignments*, GHIK performs rather worse (not shown here). In summary, previous findings [26, 190] are confirmed, concluding that a **linear kernel** is usually preferable, as the training and optimisation process is faster.

7.4.3 Deltas

In the **3rd experiment**, the relevance of the *deltas* (see Section 2.1.7) is evaluated. Only **MFCCs 1–14 + RMS energy** (15 LLDs) are taken into account in the following experiments, and/or, their *deltas*, as defined in the ComParE feature set (see Section 2.3.1). Consequently, the following four LLD sets are compared to each other:

1. (Raw) LLDs-only (15-dimensional)
2. Deltas-only (15-dimensional)
3. An **LLD-level fusion** of LLDs + deltas (30-dimensional)
4. LLDs + deltas (2×15-dimensional) with a **bag-level fusion**

In order to respect the larger dimensionality of the input space with LLDs + deltas, *four* configurations of *codebook size* and *number of assignments* are evaluated, taking larger codebooks into consideration:

⟨1a⟩ $C_S = 100$, $N_A = 1$

⟨1b⟩ $C_S = 400$, $N_A = 1$

⟨2a⟩ $C_S = 1\,000$, $N_A = 10$

⟨2b⟩ $C_S = 4\,000$, $N_A = 10$

A single codebook is trained for the *LLD-level fusion* and two codebooks are trained for the *bag-level fusion*, using the **random sampling** (`random`) method, as this method provided the best overall results in the 1st experiment. For the *bag-level* fusion, the shown codebook sizes (C_S) are halved for each of the two bags, resulting in the same BoAW feature dimensionality, in order to improve the comparability. As before, a **z-score normalisation** of all LLDs/deltas is performed and the term frequency histograms are subject to a **logarithmic weighting** (log-TF), prior to a **min-max normalisation**. The results in terms of the UAR, averaged over 10 random seeds, are reported in Figure 7.3.

[1] Statistically significant for eGeMAPS ⟨2⟩.

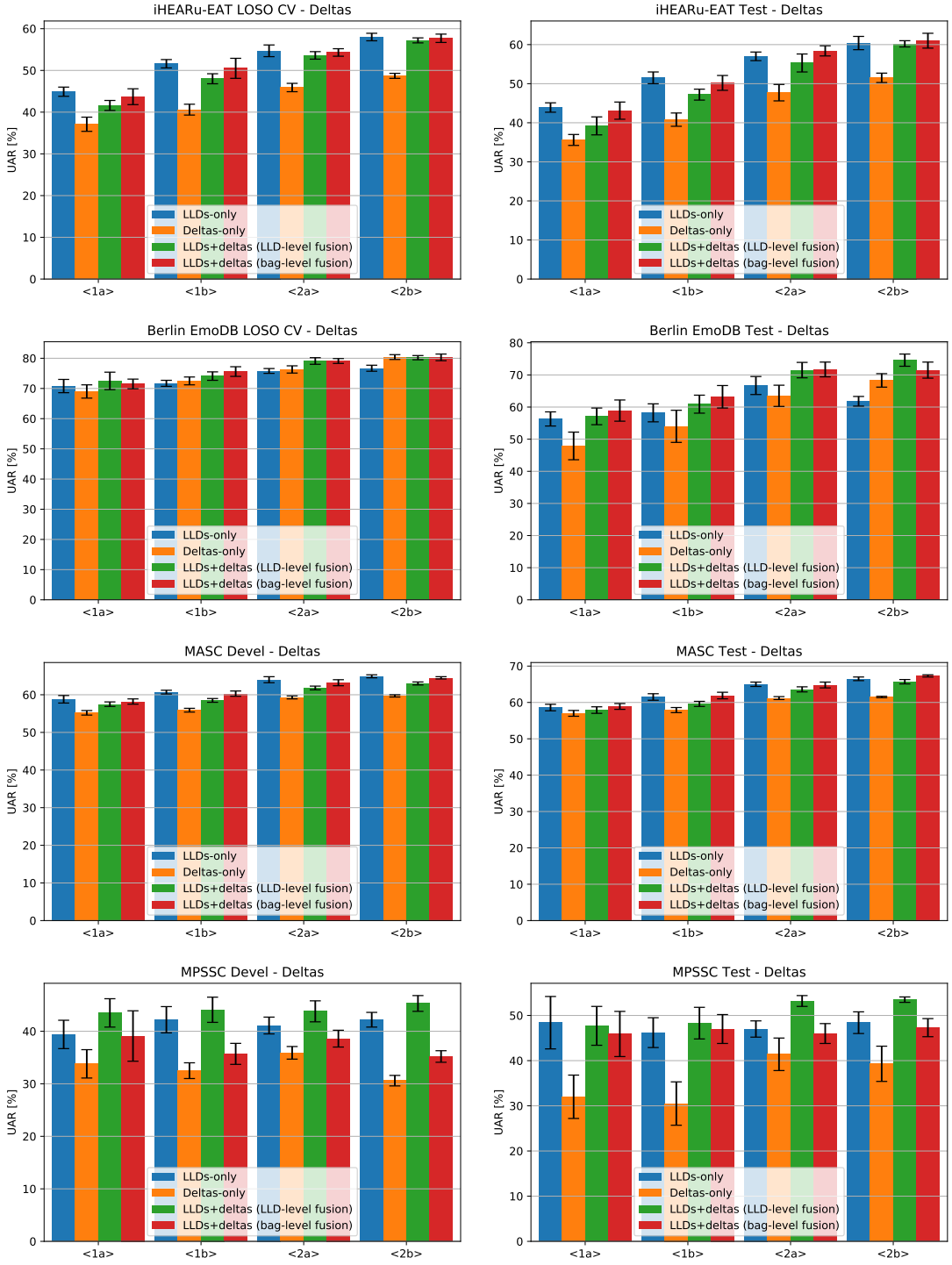

Figure 7.3: Results of the 3rd experiment, comparing the relevance of MFCC-deltas. The UARs (mean and standard deviation over 10 random seeds in openXBOW) achieved for the Development set (Devel) or LOSO CV and Test set, respectively, for each of the four considered datasets, are reported. Experiments are run for four configurations of C_S and N_A: ⟨1a⟩, ⟨1b⟩, ⟨2a⟩, & ⟨2b⟩ as specified in the main text.

It is evident that the **relevance of MFCC-deltas highly depends on the given task**. While for iHEARu-EAT and MASC, BoAW generated from deltas performs worse than the BoAW generated from the raw LLDs and even an *LLD-level fusion* of both leads to worse results, for Berlin EmoDB and MPSSC, the *LLD-level fusion* is at least superior to raw LLDs only.

In contrast, with a *bag-level fusion*, the detrimental effect of the deltas is less distinctive for iHEARu-EAT and MASC, but only for Berlin EmoDB, the best results are achieved with *bag-level fusion* on average. However, for MPSSC, it results in a worse performance than using raw LLDs.

Compared to the findings in Section 6.3, **MFCC-deltas are generally less meaningful** than the raw LLDs for the four CA tasks investigated in this chapter. The configuration of *codebook size* and *number of assignments* does not result in qualitatively different findings w. r. t. the given research question. Nevertheless, a better performance using a larger codebook size can be observed for all corpora except for MPSSC.

7.4.4 Audio word encoding

In the **4th experiment**, **hard** assignment and **soft assigment** are compared. For *hard assignment*, also the influence of **multi-assignment**, i. e., $N_A > 1$ is evaluated. In addition to that, the importance of **log-TF** weighting is examined by computing all results with and without it. Based on the previous experiments, only **MFCCs 1–14 + RMS energy** (15 LLDs) are taken into account, as defined in the ComParE feature set (see Section 2.3.1), without the deltas. As the performance of different assignment methods might depend on the codebook size, two different codebook sizes are taken into account, in analogy with the 1st and 2nd experiment:

1. $C_S = 100$ and
2. $C_S = 1\,000$.

Throughout the experiments, an **EM clustering** is performed, using the outcome of a **k-means++ clustering** for initialisation (`em-kmeans++`). To speed up the training process and as no meaningful differences in performance have been found, the clustering is performed on a sub-sampling of 10 000 instances. The 8 following configurations are considered for the audio word encoding (cf. Section 5.2.2):

1. $N_A = 1$ (single-assignment),
2. $N_A = 2$ (multi-assignment),
3. $N_A = 5$ (multi-assignment),
4. $N_A = 10$ (multi-assignment),
5. $N_A = 20$ (multi-assignment),

6. $N_A = 50$ (multi-assignment),
7. GMM-like soft encoding *without* priors (`-gmm 1`), and
8. GMM-like soft encoding *with* priors (`-gmm 2`).

All these configurations have been applied with and without a log-TF weighting (`-log`). An *IDF weighting* (`-idf`) is not considered further as it is, according to Equation 3.5, a feature-wise linear transform and its effect is nullified by feature normalisation anyway. As before, a **z-score normalisation** of LLDs is performed and the output is subject to a **min-max normalisation** prior to SVM training. The results in terms of the UAR, averaged over 10 random seeds, are reported in Figure 7.4.

It is obvious that a **soft assignment is better than single-assignment** in most cases. As already found before, **multi-assignment performs better than single-assignment**, but the exact **number of assignments** (N_A) needs to be optimised as it is **depending not only on the codebook size but also on the given data**. There is no clear conclusion whether *soft assignment* is superior to *hard assignment*. Also this seems to depend highly on the task or data at hand. Based on the experiments, the soft GMM-like encoding tends to work better for smaller codebook sizes, especially with MASC. This contradicts the findings by Rawat et al. that a soft quantisation is particularly beneficial for large codebooks [201].

Generally, it must be noted that this comparison is not completely fair as the *random sampling* is the optimum codebook generation used with a hard assignment (as shown in Section 7.4.1) and *EM clustering* is used here. Furthermore, the large error bars of the soft encoding for the test set of MPSSC show that it might not be very robust when applied to small datasets. Concerning taking into account GMM-*priors*, there is no preference, given the results, compared to the findings by Barrington et al. [187], claiming that uniform priors work better. In any case, a **log-TF weighting is always to be preferred**.

7.4.5 Optimisation of LLD splits and codebook size

In the **5$^{\text{th}}$ experiment**, a comparison of the performances achieved with different sets of LLDs, including those from the large ComParE feature set, is made. Moreover, experiments are run with two different ways of splitting this relatively large number of LLDs into subsets ('bag-level fusion'). Results are reported for varying codebook sizes. The following **four LLD sets and splits** are taken into account:

- MFCCs 1–14 + RMS energy (15 LLDs)
- eGeMAPS [31] (23 LLDs)
- ComParE (see Section 2.3.1) with two different splits into subsets:

- a split into 2 subsets (2-split), exactly as done for ComParE 2017–2020 (see Section 6.4):
  1. *raw acoustic LLDs* (65 LLDs)
  2. *deltas* (65 'LLDs')
- a split into 12 subsets (12-split), according to the following 6 feature groups
  1. prosody-related: *F0 + Voicing probability + RMS* (3 LLDs)
  2. microprosody: *Jitter (local + DDP) + Shimmer* (3 LLDs)
  3. noise-related: *logHNR + ZCR* (2 LLDs)
  4. *RASTA auditory band energies 1–26 + Loudness + Modulation loudness* (28 LLDs)
  5. *Spectral descriptors* (15 LLDs)
  6. *MFCCs 1–14* (14 LLDs)

  and exactly the same grouping for all corresponding *deltas*.

Generally, a fair comparison of results obtained with different splits of the LLD feature space is difficult, as different configurations of codebook size and number of assignments will be optimal. In the first three experiments, fixed codebook sizes have been employed, independent from the number of LLDs. In this series of experiments, the **codebook size for each subset of LLDs is a multiple of the number of LLDs**. Also the **number of assignments is depending linearly on the codebook size**, with a fixed ratio of $\frac{1}{50}$, motivated by the findings in Section 6.1. The following three multiples of the number of LLDs, N_{LLDs} are evaluated:

1. $C_S = 50 \times N_{\text{LLDs}}$, $N_A = 1 \times N_{\text{LLDs}}$
2. $C_S = 100 \times N_{\text{LLDs}}$, $N_A = 2 \times N_{\text{LLDs}}$
3. $C_S = 200 \times N_{\text{LLDs}}$, $N_A = 4 \times N_{\text{LLDs}}$

Also here, a **z-score normalisation** of all input LLDs is performed, **random sampling** (`random`) is employed for codebook generation, and a **log-TF weighting** is applied. The results in terms of the UAR, averaged over 10 random seeds, are reported in Figure 7.5.

Generally, it is evident that **all tasks benefit the large-scale** ComParE set. Nevertheless, for two tasks, MFCCs and eGeMAPS are outperformed only when using the split into 12 subsets. A larger codebook size leads to higher UARs, by trend. In summary, taking into account a large number of LLDs, split into reasonable subsets, seems to be more meaningful than optimising the codebook size and number of assignments.

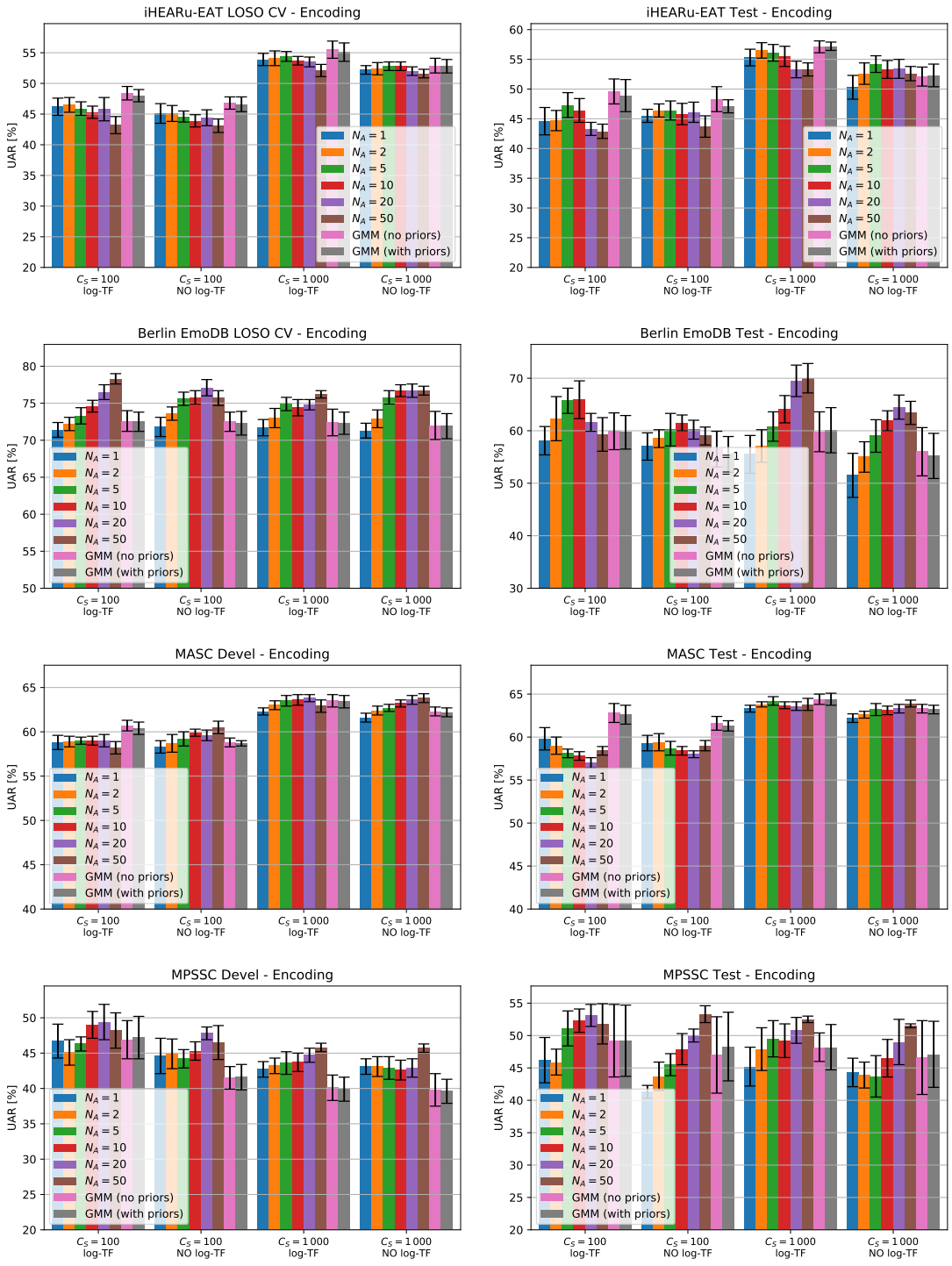

Figure 7.4: Results of the 4$^{\text{th}}$ experiment, comparing audio word encoding techniques and term frequency weighting, as specified in the main text. The UARs (mean and standard deviation over 10 random seeds in openXBOW) achieved for the Development set (Devel) or LOSO CV and Test set, respectively, for each of the four considered datasets, are reported.

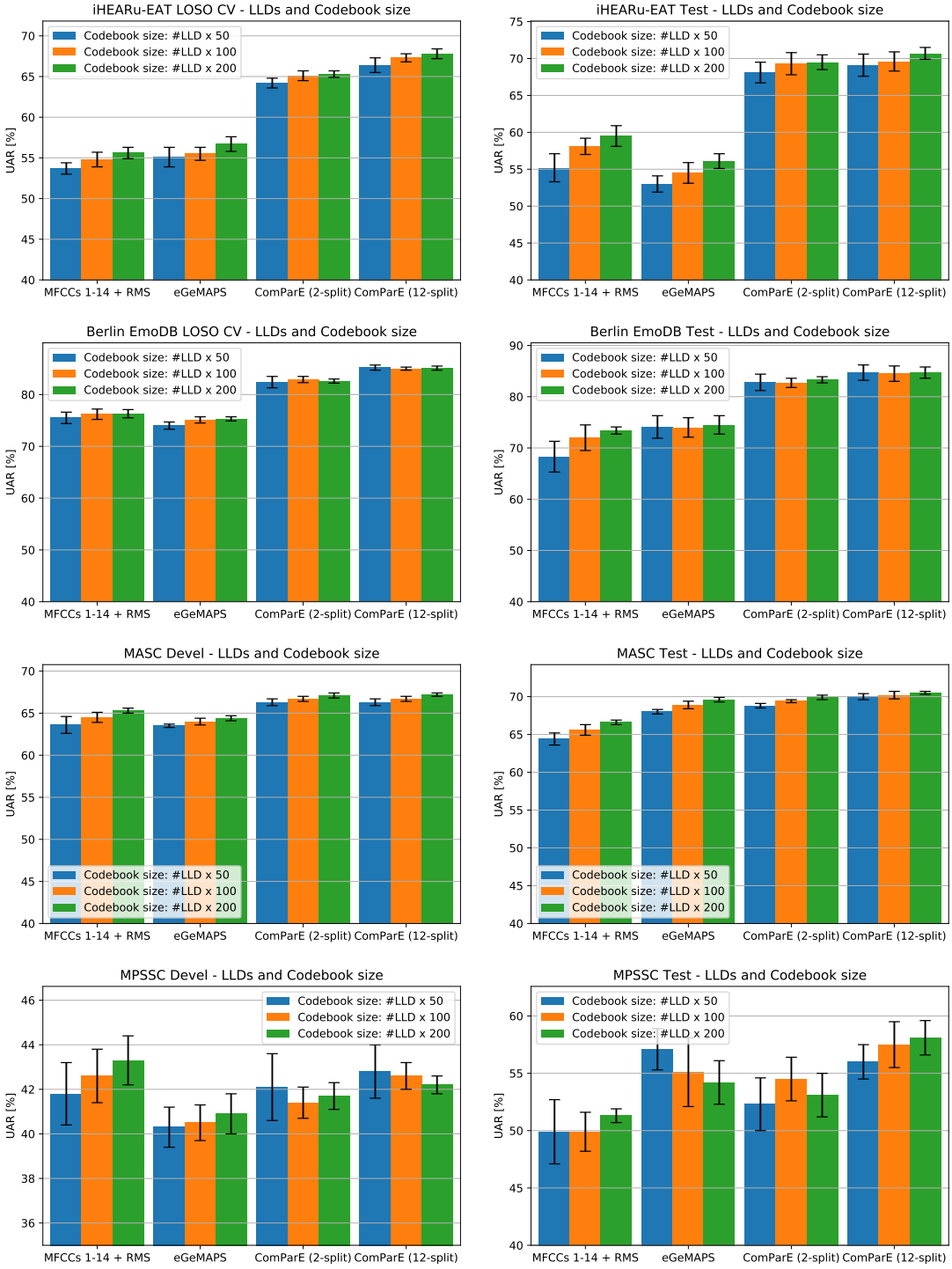

Figure 7.5: Results of the 5$^{\text{th}}$ experiment, comparing different LLD sets and splits. The UARs (mean and standard deviation over 10 random seeds in openXBOW) achieved for the Development set (Devel) or LOSO CV and Test set, respectively, for each of the four considered datasets, are reported. Experiments are run for three configurations of C_S and N_A as a multiple of the number of LLDs, with a fixed ratio of 50 between C_S and N_A.

7.4.6 Comparison and fusion with functionals

Finally, in the **6th experiment**, it is evaluated how good the BoAW features perform in comparison with functionals. In addition to that, results for an *early fusion*, i. e., concatenating BoAW and funcionals-based feature vectors prior to SVM training, are shown.

For BoAW, both eGeMAPS and ComParE LLD sets are evaluated, where the '12-split' is taken into account for ComParE, as explained in Section 7.4.5. Exactly the same configuration as in this section is reused, with a codebook size corresponding to 200 times the number of LLDs (and the number of assignments 4 times the number of LLDs). The corresponding call to openXBOW reflecting all chosen options is shown at the end of Appendix A. It must be noted that optimisation on the development set of MPSSC would return a codebook size lower than optimal, but in order to make a consistent evaluation, a fixed setup is chosen.

As a *baseline*, the original eGeMAPS and ComParE sets with functionals (see Section 2.3) are considered. The results are shown in Table 7.5, with BoAW-based experiments averaged across 10 random seeds. In addition to the results from the experiments, also the *baseline results* from the corresponding challenges are reported, except for Berlin EmoDB, where no official test split exists. Nevertheless, in the literature an *accuracy* of up to 85.2 % is reported for speaker-independent experiments on Berlin EmoDB with a 2-fold CV [377], which closely matches with the experimental results found for a fusion of functionals and BoAW. For MPSSC, the outcomes of further experiments from the literature are given.

First of all, it must be noted that **BoAW features outperform functionals in most of the experiments**, except for eGeMAPS-LLDs on Berlin EmoDB. For iHEARu-EAT and MASC, the **improvement** over the functionals-baseline is **between 3.9 % and 7.3 % throughout partitions and LLD sets**. For MPSSC, there is no meaningful improvement over functionals. Statistical significance testing on the test partitions, performing a *one-sample t-test* ($p < 0.05$), shows that in all experiments where the UAR is higher for BoAW than for functionals[2], this gap is **significant**. Normality of the results has been checked according to Section 4.5.4.

Concerning the **early fusion of BoAW and functionals**, it is evident that **even better measures** are achieved than with functionals or BoAW alone for more or less all experiments with iHEARu-EAT and MASC. For ComParE, these improvements are **statistically significant** ($p < 0.05$) w. r. t. *Welch's t-test* (see Section 4.5.4). Also for Berlin EmoDB with ComParE-LLDs, fusing the functionals with BoAW now provides a small improvement over the baseline for both LOSO CV and the test set. This shows that, similar to the findings in Section 6.1 on time-continuous emotion recognition, functionals and BoAW representations are

[2] I. e., all experiments except for Berlin EmoDB with eGeMAPS and MPSSC with ComParE.

complementary with each other, each conveying meaningful and exclusive information on the task.

The baselines of the 'EAT' and 'ComParE 2017 Snoring' challenges are outperformed, while achieving almost the same result for 'ComParE 2020 Masks'. However, it must be noted that the baseline of the 'Masks' challenge is a late fusion of the predictions of *four* different approaches [116]. With a UAR of 72.3 % (average over 10 seeds), the official winning approach of the EAT challenge [376] is surpassed by 3.2 %, while achieving almost the same performance as the best results reported with 73.3 % [374][3].

A large number of works have been published on MPSSC. As mentioned before, Qian et al. [315] extended the BoAW approach using *wavelet* descriptors (see Section 2.1.3) as LLDs ('bag-of-wavelet-features'). This model, also based on openXBOW, outperforms the baseline by more than 10 %, achieving almost the highest known measure. Demir et al. obtain the best results reported on MPSSC so far (a UAR of 72.6 % on the test set), with a fusion of *histograms of oriented gradients* and *local binary patterns* extracted from the spectrogram and decoded with an SVM [379]. A similar performance has been achieved in a work by the author of this thesis [229], with neural network-based *end-to-end* learning. It is shown that a *fully convolutional* (one-dimensional) network with a global pooling outperforms a DNN consisting of both convolutional and LSTM layers. A very good performance is already achieved on the development set, using only the official training set, while also the baseline on the test set (from ComParE 2017) is outperformed by a large margin. The same UAR is achieved as with the DeepSpectrum (see Section 6.4) + SVM approach by Amiriparian et al. [272]. Overall, it must be stated that 'non-deep' approaches have obtained the best two results so far, with the 'bag-of-wavelet-features' approach [315] and the approach by Demir et al. [379], which employs histogram-like features as well and consequently shares some similarities with BoAW.

Generally, it must be emphasised that the proposed BoAW, or BoAW + functionals, approach for audio classification has not been optimised specifically for a certain task or dataset, but for a collection of **four heterogeneous tasks**. Exactly the same configuration of openXBOW has been applied throughout all experiments. Only the *complexity* hyperparameter of the SVM is tuned on each dataset. Also the *codebooks* are sampled from the LLDs of each respective corpus, but this is a relatively inexpensive procedure and not a limitation, as this is exceeded by the requirements of the classifier training, which requires data anyway and—in addition—the *labels*. Thus, the proposed method can be considered as **task-independent**, in contrast to most of the further approaches reported in Table 7.5, where the model architectures and processing pipelines have presumably been tuned on a specific task.

[3] The authors were not officially participating at the challenge due to an overlap with organisers.

Approach	**Devel / LOSO CV** UAR [%]	**Test** UAR [%]
iHEARu-EAT		
eGeMAPS – Functionals	52.4	51.0
eGeMAPS – BoAW	56.7 ± 0.9	56.1 ± 1.0
eGeMAPS – Functionals + BoAW	56.8 ± 0.6	56.6 ± 1.1
ComParE – Functionals	61.5	63.5
ComParE – BoAW (12-split)	67.8 ± 0.6	70.7 ± 0.8
ComParE – Functionals + BoAW	68.7 ± 0.4	72.3 ± 0.7
Baseline EAT Challenge [84]	64.3	67.2
Official winner EAT Challenge [376]	68.2	69.1
Best result EAT Challenge [374]	75.0	73.3
Berlin EmoDB		
eGeMAPS – Functionals	80.2	75.2
eGeMAPS – BoAW	75.3 ± 0.4	74.5 ± 1.8
eGeMAPS – Functionals + BoAW	75.7 ± 0.4	74.4 ± 1.8
ComParE – Functionals	85.0	80.8
ComParE – BoAW (12-split)	85.1 ± 0.4	84.7 ± 1.1
ComParE – Functionals + BoAW	86.4 ± 0.3	85.9 ± 1.3
MASC		
eGeMAPS – Functionals	60.5	62.3
eGeMAPS – BoAW	64.4 ± 0.3	69.6 ± 0.3
eGeMAPS – Functionals + BoAW	64.6 ± 0.3	69.8 ± 0.3
ComParE – Functionals	62.6	66.5
ComParE – BoAW (12-split)	67.2 ± 0.2	70.5 ± 0.2
ComParE – Functionals + BoAW	67.1 ± 0.3	71.0 ± 0.3
Baseline ComParE 2020 Masks [116]	64.4	71.8
Official winner ComParE 2020 Masks [378]	70.5	80.1
MPSSC		
eGeMAPS – Functionals	41.4	51.8
eGeMAPS – BoAW	40.9 ± 0.9	54.2 ± 1.9
eGeMAPS – Functionals + BoAW	41.6 ± 0.8	55.3 ± 1.9
ComParE – Functionals	37.5	58.9
ComParE – BoAW (12-split)	42.2 ± 0.4	58.1 ± 1.5
ComParE – Functionals + BoAW	39.3 ± 0.5	59.0 ± 1.9
Baseline ComParE 2017 Snoring [11]	40.6	58.5
Official winner ComParE 2017 Snoring [333]	50.1	64.2
DeepSpectrum [272]	44.8	67.0
Spectrogram-based image features [379]	37.8	72.6
Bag-of-wavelet-features [315]	35.0	69.4
End-to-end CNN [229]	59.1	67.0

Table 7.5: Results of the 6$^{\text{th}}$ experiment, showing UARs achieved for functionals, BoAW, and an early fusion of both. The UARs (mean and standard deviation over 10 random seeds in openXBOW) achieved for the Development set (Devel) or LOSO CV and Test set, respectively, for each of the four considered datasets, are reported. In addition, further results published for the respective datasets are included in the table.

Part IV

DISCUSSION

8

Discussion

After having presented a large number of experiments in the previous two chapters, in this chapter, the *main findings* are first summarised. Then, the *benefits* and the *limitations* that come with BoAW as an acoustic feature representation, supplemented by the limitations resulting from the experiments carried out, are discussed. Finally, *ethical implications* are addressed.

8.1 Findings

Most importantly, it was shown that BoAW is an effective method to represent acoustic LLDs on a supra-segmental level, be it a chunk of audio to be classified (e. g., Chapter 7) or a time interval in a continuous stream of audio (e. g., Section 6.1). For the relatively small dataset MPSSC, **recent deep learning approaches are outperformed** by an SVM model based on BoAW features (Section 7.4.6). In the majority of the studies presented in Chapters 6 and 7, a classifier or regressor trained on **BoAW representations** as input **outperforms** a model trained on **statistical functionals**. This was shown especially for the ComParE and eGeMAPS feature sets (see Section 2.3) in Section 7.4.6. The remarkable performance was achieved using a fixed configuration of openXBOW for each set of LLDs, without any further optimisation on the task or the data. Additionally, it was proven that an early fusion of BoAW and functionals by a simple concatenation is beneficial (Sections 6.1 and 7.4.6). This is some evidence for the **complementarity** of both representations. This hypothesis is also confirmed by the benefit of a *late fusion* of predictions from models based on different acoustic (hand-crafted and learnt) feature representations (Section 6.4). Further proofs for the complementarity, also with representations extracted by a DNN, were given by Guo et al. [329] and Gosztolya [330].

In fact, the only scenario where BoAW performs worse (on average) is when used as input for an LSTM-RNN (Section 6.6). Nevertheless, as shown for AVEC

(Section 6.5), for the task of *time-continuous dimensional* emotion recognition with BoAW features, a better performance is achieved with an LSTM-RNN than with a static SVM regression model. Furthermore, at least when tuning the block length, for *valence*, BoAW representations of eGeMAPS-LLDs outperform the functionals representation. It was shown that, for this particular task, a larger block length is required for BoAW than for functionals.

Moreover, the experiments in Section 6.3 showed that BoAW features are **robust against differences between training and testing conditions**. Generally, only an *on-line normalisation* of the LLDs is required, while a normalisation of the BoAW features is not essential.

8.1.1 Notes on acoustic LLDs

The results presented in Section 6.3 have exemplified that the choice of suitable frame-level descriptors is highly task-dependent. In comparison with the experiments from Section 7.4.3, it is evident that the *deltas* of LLDs are relevant only for very specific tasks, such as the *classification of audio effects*, where they are more meaningful than the actual LLDs. However, for the paralinguistic tasks investigated in Chapter 7, the deltas are less meaningful or even detrimental when fused in some of the cases. This applies to both *LLD-level fusion* and *bag-level fusion*; nevertheless, the latter should be preferred if no detailed analysis can be done as the negative effect is softer.

As shown in Section 7.4.5, a **large LLD set**, such as the LLDs from ComParE, is generally **beneficial** for all tasks—even for *emotion recognition*, where eGeMAPS typically gives good results. Furthermore, it was proven that when splitting the 65 LLDs and deltas each into 6 subsets (and corresponding codebooks), the performance of the classifier is often better and thus, should be preferred.

While in Section 6.2, an *LLD-level fusion* of *MFCCs*, *formants*, and *wavelet descriptors* was found to provide better results throughout for the task of snore sound classification, contradictory findings were made in Sections 6.7, 7.4.3 and 7.4.5 for the ComParE features. Thus, it must be concluded that the optimal fusion type depends—at least—on the type and size of the LLD set.

8.1.2 Notes on codebook generation

Extensive comparisons of BoAW representations based on codebooks generated with eight different methods are given in Section 7.4.1. Based on these results, it can be declared that a pure **random sampling** of LLDs from the training set is the **best choice** overall.

The performance of **clustering** methods, especially of *EM clustering*, depends on the type of LLDs. A potential dependency is the *distribution* of the LLDs in the feature space. Based on the experiments, there is some evidence that *EM clustering*

is suitable in cases where also a codebook *sampled from a Gaussian PDF* leads to good results. As this is the case mainly for MFCCs, there is the strong suspicion that the similarity of the LLDs with a *normal distribution* is an appropriate indicator.

Generally, it could be seen that **sampling from a Gaussian PDF** is a good compromise for codebook generation, as the performance is usually close to that achieved by *random sampling* and, most importantly, it has the **big advantage of data-independence**. This means that a codebook is defined only by its *size*, the *number of LLDs*, and the *seed*. The training data is not required to be loaded by the employed software at once, compared with clustering, which comes with technical benefits; however, the parameters for *normalisation* of the LLDs still need to be estimated, which is essential in case of a data-independent codebook.

It must be concluded that a **clustering is generally not required**, which has already been found in some of the related works (Section 3.2.2), and that it has even a **negative effect** on the performance when compared to random sampling, based on the experiments (Section 7.4.1). Moreover, a *codebook reduction* by fusing similar codewords as investigated in Section 6.2 has not proven to be beneficial, as well as a *supervised codebook generation*.

An example of the outcome of codebook generation is shown in Figure 8.1 for the iHEARu-EAT dataset. In analogy with Section 7.4.1, MFCCs 1–14 and RMS energy are employed as LLDs and z-score normalised, but—for the purpose of visualisation—only 20 codewords are defined using each of the four methods: sampling from a normal distribution (`pdf`), random sampling (`random`), random sampling with k-means++ initialisation (`random++`), and k-means++ clustering (`kmeans++`). For visualisation, the LLD space from the training set is reduced to its two main components via a *principal component analysis* [224], a very common technique for dimensionality reduction. The two components are linear combinations of the 15 input dimensions, maximising the variance.

Figure 8.1 shows that the codewords from the simpler sampling techniques, especially from the data-independent sampling from a Gaussian PDF (`pdf`), are situated in the 'denser' area of the plane, i. e., where the majority of the LLDs are concentrated in. The more complex techniques `random++` and `kmeans++` cover the whole LLD space in a better way, giving a larger weight to 'outliers'. However, considering the results, the importance of a wide coverage is questionable. This is, in particular, objecting the claims by Rawat et al. [201], who assume that clusters converge to denser regions of the input space and that random samples cover also the more informative (i. e., sparsely populated) regions. As shown by the relatively competitive performance with sampling from a Gaussian PDF, where 'outlier regions' are not covered at all, the relevance of these might have been disproved.

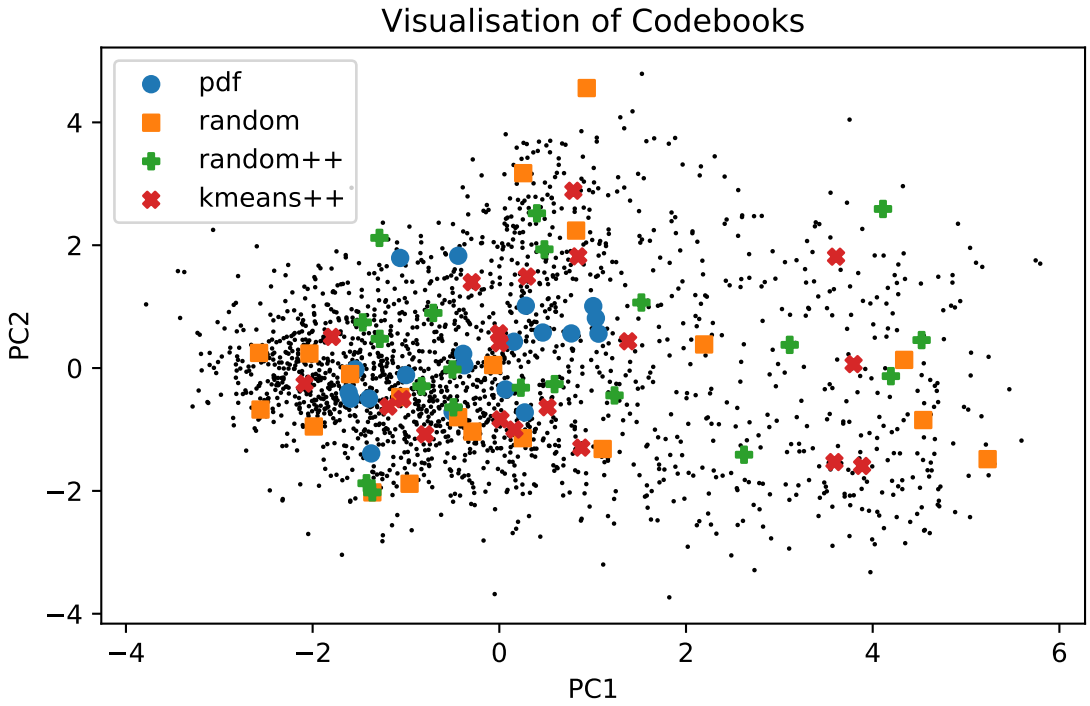

Figure 8.1: Visualisation of codebooks generated with four different techniques for the iHEARu-EAT database. The codebook size is 20 for each of the four techniques (`pdf`, `random`, `random++`, `kmeans++`), in analogy with Section 7.4.1. As LLDs, z-score normalised MFCCs 1–14 and the RMS are used. 2 000 LLDs from the training set are randomly selected to be visualised as dots. The 15-dimensional LLD space is reduced to two dimensions (PC1 & PC2) via a *principal component analysis*.

8.1.3 Notes on codebook size and audio word encoding

Generally, a tendency can be observed that larger *codebook size* and *number of assignments* work better, but not for all LLD sets and tasks (Section 7.4.5). It is noteworthy that a large codebook proves to be suitable especially for MASC, which is the largest dataset investigated (more than 36 000 samples and 10 hours of speech), and a small codebook is on average the best option for MPSSC, the smallest corpus in terms of the total duration. This consolidates the claims made in Section 3.3.3 that smaller codebooks are known to be more robust and *better generalise*, a property which is beneficial in scenarios with a lack of training data.

In Section 6.4 on the *Computational Paralinguistics ChallengE* (ComParE), it was nevertheless shown that optimising the codebook size might result in a better performance and that even for a relatively small size of 125, top results can be achieved, depending on the task and data. For other corpora, such as iHEARu-

EAT, the size of the codebook is not a very critical hyperparameter over a large range (Section 6.7). Moreover, suitable codebook sizes seem to be independent from the *target*, the classifier is trained on, given a fixed dataset.

The experiments carried out, mainly in Sections 6.1 and 6.2, showcase that **multi-assignment**, i. e., a *number of assignments* for each frame larger than 1, is **beneficial and important**. In general, a certain **ratio** must be fulfilled between the codebook size and the number of assignments. Based on the findings in the corresponding sections, a ratio between 1 % and 5 % seems to be suitable over a larger range of codebook sizes. Though these two are critical hyperparameters of the approach, it is evident that they require to be increased when dealing with a larger number of LLDs (Section 6.2).

It must be noted that a **soft encoding** in terms of assigning an audio word to each Gaussian of a GMM-codebook is superior also to a hard multi-assignment in some cases (see Section 7.4.4). Nevertheless, the performance differs depending on the given data and also on the codebook size, where a tendency for the soft encoding was found to work better with smaller codebook sizes. Finally, it must be mentioned that a *soft assignment* in terms of a 'Gaussian encoding' (Equation 3.18) was not found to provide a meaningful improvement of the performance (Section 6.2), contradicting the findings made in related works 3.2.2.

Concerning a *logarithmic weighting* of the term frequency BoAW features (log-TF), this post-processing step was found to be beneficial throughout all configurations when using an SVM model, while an IDF weighting is not relevant in this case.

8.1.4 Notes on the classifier

For the reasoning given in Section 7.3, the ML experiments presented in this thesis have been based mainly on SVM for classification or regression, except for the time-continuous emotion recognition as a *sequence labelling* task (Sections 6.5 and 6.6). For this given task and setup of the *Audio-Visual Emotion Challenge* (AVEC), an LSTM-RNN prove to be more suitable than an SVM regressor.

Concerning the SVM, previous comprehension that a *linear kernel* is adequate to classification in a large feature space—such as BoAW—has been confirmed [30]. Moreover, it has been reassured by the experiments in Section 7.4.2 that a *(generalised) histogram intersection kernel* (GHIK) is usually not superior to a linear one [26, 190], also with optimised *complexity* hyperparameter, except for particular setups and tasks. Generally, optimisation of the *complexity* is an essential procedure when employing SVM.

8.2 Benefits

The BoAW approach comes with some benefits. First of all, the representation is **scalable**, most importantly, w. r. t. to the *codebook size*, which directly translates to the *dimensionality* of the BoAW feature space. As seen in the previous section, an adaptation to the employed dimensionality of frame-level features is required and the optimum depends also on other factors, such as the amount of training data. One advantage, compared with *functionals*, is that the dimensionality can be modified and optimised in an automatic way, whereas each statistical functional needs to be defined and implemented beforehand.

In fact, as shown in Section 7.4.6, exactly the **same fixed configuration** is employed for all investigated tasks, outperforming the baseline throughout. Thus, it can be claimed that BoAW is **task-independent**. The deployment of the method requires solely that normalisation parameters and—depending on the generation method—codebooks are adapted to the training data, in addition to potential hyperparameters of the classifier, such as the *complexity*. Nevertheless, this procedure is typically much less time-consuming than optimisation and training of *deep learning* models. In fact, is was proven that for **small datasets**, such as MPSSC, approaches based on learnt features and SVM classifier outperform deep learning methods.

BoAW is an **unsupervised representation learning** methodology. Unsupervised learning has generally the benefit of generating representations not biased towards the task [380], as opposed to *supervised representation learning* or *supervised end-to-end learning* [229]. This can be useful when multiple labels are to be predicted (cf. Section 6.7).

The BoAW method is further **flexible**—demonstrated by the openXBOW toolkit, as it can be used to fuse different modalities, which can be represented by either *symbolic* or *numeric* descriptors. As in the example of the EAT challenge in Section 6.7, audio and video domains can be fused into a 'comparable', histogram-like representation. The fact that typically, acoustic LLDs are extracted at a higher frequency (e. g., 100 Hz) than video frames (e. g., 25 Hz) is not an issue here, as low-level information is summarised over longer blocks and the final BoAW and BoVW representations can match with an arbitrary target frequency (usually the frequency of the labels).

Finally, it must be pointed out that BoAW (or BoVW) representations are relatively **data efficient**. From a technical point of view, the histogram is represented by just a set of *integers*[1] denoting the assignment frequency of each audio word. This means that, in principle, the number of integers to be stored for one instance is equal to the codebook size. However, given that *low term frequencies* are typically

[1] Alternatively, *floating point numbers*, depending on potential soft assignment, scaling, or normalisation, where the latter two are, however, independent from the particular instance.

represented more often, and, depending on the actual codebook size, BoAW histograms usually show a certain degree of *sparseness*, *lossless compression techniques* could be applied, when transferring BoAW representations via low-bandwidth channels. When a continuous stream of audio data is transferred, only the (integer) *indexes* of the audio word sequence need to be known. The feasible 'compression rate' depends on the *number of LLDs*, the *number of assignments*, the *codebook size* and the *entropy* of the assignment process[2]

8.3 Limitations

Besides the discussed *benefits*, the BoAW approach comes also with some *limitations*. As concluded from the experiments, mainly those in Sections 6.3 and 7.4.3, the selection of LLDs is still task-dependent. This is especially true for the *deltas* and the type of *fusion* (*LLD-level* or *bag-level*). Nevertheless, overall, the performances tend to be better for a larger set of LLDs and bag-level fusion.

As shown in Section 7.4.1, *data-independent codebooks* in terms of sampling from a *normal distribution* does not work well for all types of LLDs and datasets. It has not been proven if the distribution or the sampling process is the reason for this.

Concerning the top results from Section 7.4.6, it must be noted that the *feature space* is relatively *large*. More precisely, for 130 LLDs (including deltas), the BoAW representation has a dimensionality of 26 000. It is evident that an SVM with linear kernel can handle such large inputs, however, other models might have difficulties in handling these. In particular, it could be seen in Section 6.6 that LSTM-RNN performs worse with BoAW than with simple functionals. More advanced neural network architectures or processing steps might be required to tackle this issue. As an example, the feature space might be reduced by *embedding techniques* as known from the large-dimensional BoW representations in NLP [143]. However, this somehow contradicts the vector quantisation principle of the audio words and can be performed on the actual LLDs as well.

Finally, it must be noted that the BoAW method does not increase the *interpretability* of ML approaches, which is a relevant aspect for many use cases [381]. This lack of transparency is due to the fact that *vectors of LLDs* are quantised, obliterating the information of single, semantically meaningful descriptors.

Concerning the presented *systematic* experiments, it must be noted that the optimisation of some BoAW-related modifications has not been taken into account. This is mainly due to the great effort required to optimise in the large hyperparameter space. As an example, the *numeric n-grams* option of openXBOW (Section 5.2.2) using only *2-grams* showed to provide competitive results that were, however, not superior to the default *1-grams*, in the considered range of hyperparameters. Optimisation of the employed LLD sets can provide some room for improvement for

[2]Given that less bits are required for more frequent audio words.

n-grams because their performance, as a method modelling temporal dependencies, might depend on the smoothness of the descriptors over time.

8.4 Ethical Considerations

In 2021, AI already has a meaningful impact on the whole society, on most industries, and on medicine. The number of applications that are including AI technologies and especially, ML-based components, is continuously growing. These applications can have *positive*, *negative*, or also *controversial* implications or even a combination thereof [382]. As an example, *voice forgery* ('deep fakes') can be misused to spread *fake news* [381], but the same technology can be used for the potentially beneficial application of *voice dubbing* in movies [383], i. e., translating actors' speech to another language, but keeping the characteristics of their voice. *Medical diagnosis* or *health monitoring* can be used both in a positive and negative context, when thinking of clinical research to optimise a treatment on the one side and selection criteria in job interviews on the other side [381].

Generally, also from applications that are considered to be mostly positive, many ethical questions arise and are subject to ongoing evaluations and discussions [382], e. g.:

- **Data privacy and anonymity** is a big topic when recording and publishing data for ML, also for research purposes. This is highly problematic for speech and visual data [302], as anonymity cannot be achieved in the raw data. Moreover, when working on a personalised medical treatment, complete metadata are typically required [382].
- The minimum **performance** that must be achieved until an AI system can come to practice is subject to discussion [382]. This is closely connected to the question, if technologies that diagnose the likelihood of future diseases, such as, e. g., *Parkinson's Disease* from speech, should be used [381], as *false positive* decisions might have a harmful psychic impact. Furthermore, it is an open question if a human may override decisions of an AI system (even if the system has proven to provide more reliable results) [382].
- **Explainability** of AI-based decisions is often required legally, especially in health-related applications [382] or whenever an impact on the user is expected [381].
- **Fairness** includes the question whether or not systems have a bias when dealing with certain (ethnic, gender, or age) groups [382]. A typical example in this context is *pedestrian detection* and *protection* in autonomous vehicles [384], which calls for legislatory regulations.
- Finally, **acceptance** of AI in everyday life is an important aspect and of particular relevance for applications in the field of *care* and *social robots* [382].

All these questions are omnipresent when working on ML-based applications in the speech domain and not specially related to BoAW. However, researchers in the field should always be aware of all potential implications.

Nevertheless, the BoAW approach does also present a *benefit* that could be useful in terms of data protection. As mentioned in Section 8.2, the vector quantisation step leads to an LLD representation that is not only very efficient but has also very little semantic meaning, as long as the *codebook* is protected, i. e., kept secret. Thus, the **privacy** of an audio signal to be transmitted through an insecure channel is **preserved** in a better way compared to using the raw LLDs, e. g., MFCCs, from which a speech waveform can be reconstructed to an extent that the spoken words are intelligible [385, 386].

9

Conclusion

To conclude this thesis, a brief *summary* of its content and the achievements are given, followed by the *answers* to the research questions defined in the introduction and an *outlook* on potential future research directions that tie in with the presented work.

9.1 Summary

After the introduction in the first part of this thesis, the theoretical backgrounds of the employed methods were presented in the second part. This started with the description of a large variety of acoustic low-level features, how they are computed and what they represent, and an overview of the application of statistical functionals to summarise them over a whole audio instance. Then, the foundations and properties of the *bag-of-audio-words* (BoAW) method, which are the central topic of this thesis, were introduced, giving also an overview of related approaches and a survey on related works. The second part is concluded with the presentation of the machine learning (ML) methods that were employed in the experiments.

The third part started with the specification of the toolkit openXBOW, which has been developed within the framework of this thesis. The software is able to generate 'bag-of-words' instance-level representations of arbitrary modalities, such as audio, video, text, or physiological signals, either represented by numeric or symbolic descriptors. The architecture and functionalities of openXBOW were explained and some notes on usage examples and its impact were given. Overall, the toolkit has been—so far—part of eight scientific challenges in the field of *computer audition* (CA) and *affective computing* and the accompanying article has received more than 100 citations.

The two succeeding chapters present a large number of experiments on the BoAW method. In Chapter 6, experimental setups and results of previously published works focussing on certain tasks were described, while in Chapter 7, a systematic evaluation

of the methodology was performed across four different datasets, answering some of the open research questions.

The experiments are followed by the fourth part of this thesis, with a discussion of the results, mentioning also the limitations of the approach and ethical implications.

In summary, BoAW has proven to be an alternative instance-level representation of acoustic LLDs that is superior to the standard method of applying statistical functionals. Moreover, it was shown that both representations are complementary and an either early or late fusion of both further improves the performance of the audio classification. Convincing results have been achieved with a fixed set of LLDs and configuration of the processing chain, without an expensive optimisation on the task, which is typical for neural network-based approaches. Especially for applications where only small datasets are available, recent deep learning methods can be outperformed by the proposed method. Given this, the BoAW approach is a relatively simple yet promising way of unsupervised representation learning.

The latter has been exemplified by the very competitive performances of the BoAW-based models throughout the baselines designed for the scientific challenges, as described in Chapter 6. In ComParE, the BoAW approach has been *three times* the *best single approach* provided to the participants, outperforming both the ComParE set of functionals and an *end-to-end* model. For two sub-challenges between 2017 and 2020, the official baseline, given by a fusion of the predictions of the BoAW model and one or two other models, was not surpassed by any of the participants. In AVEC 2018 and 2019, three different acoustic feature representations (ComParE, BoAW, DeepSpectrum) were employed as an input for an RNN model, where BoAW provided the best performance across all sub-tasks and partitions in 2018 and the best performance in 7 out of 12 experiments across sub-tasks and partitions in 2019. In the ICMI EAT challenge in 2018, openXBOW outperformed the *end-to-end* model baseline for all three sub-tasks.

9.2 Answers to the Research Questions

The research questions defined in the introduction (Section 1.3) could be answered:

- **RQ 1:** It can be concluded that a simple **random sampling** is the best choice for codebook generation. It is computationally much cheaper than a clustering and usually superior in terms of the performance achieved by the classifier. Data-independent codebooks in terms of sampling from a Gaussian PDF can be employed as well but usually, they provide a lower performance.
- **RQ 2:** It was shown in Section 7.4.6 that a **fixed configuration** for the BoAW (in terms of *codebook generation*, *codebook size*, *number of assignments*, *split of the LLDs*, and *term-frequency weighting*) **outperforms the baseline** using *functionals*.

- **RQ 3:** Despite the superior performance of BoAW compared to functionals, it could be seen that both representations are still **complementary** and should be fused to retrieve the best performance.

Although the experiments described in this thesis have brought 'light into the darkness' and improved the understanding of BoAW representations in audio classification, there are still some interesting aspects that justify more research on the topic.

9.3 Outlook

One interesting question that has not yet been fully answered is if completely **data-independent codebooks** can be generated, achieving the same optimum performance as codebooks generated by the proposed random sampling. As shown in Section 7.4.1, the performance of a data-independent codebook depends on the type of LLDs employed. Thus, it might be feasible to define a fixed and reusable set of codewords for some sets of acoustic descriptors, such as, e. g., MFCCs. This would continue the work on RQ 1 with the goal of integrating BoAW features into acoustic feature extraction tools like openSMILE [23], which process data *on-the-fly*, i. e., file-by-file, without loading all data at once—a requirement for learning data-dependent codebooks.

In addition to that, room for improvement could be found in the optimisation of the **fusion-level** in large LLD spaces, i. e., a more optimal selection of subsets of LLDs that are combined for the vector quantisation. Moreover, 'overlapping' audio words could be investigated, meaning that an LLD is part of more than one codebook and corresponding encodings, an option that is already integrated in openXBOW.

This will also motivate further analysis of the **noise robustness**, an aspect where BoAW have been shown to outperform other methods [217], and of the capability of **domain adaption**, i. e., differing training and test conditions. As novel tasks and new datasets in CA are introduced almost every day, with an increasing level of complexity and the demand of also understanding the decisions of a classifier [387], a **systematic investigation** of the reasons why BoAW or other representations work better or worse for a particular data type, task, or domain, could be a step towards a more **interpretable ML**.

From a technical point of view, the implementation of the whole method can be massively **parallelised**, as distance computations between LLDs and codewords (and quantisation operations for an arbitrary number of frames) can be executed concurrently. Thus, porting the software to Python, for which up-to-date libraries providing fast computation on both CPU and GPU exist and interoperability with both feature extraction and machine learning tools is ensured, will be a promising concept. This would also encourage and, most importantly, facilitate further research on the open questions.

Appendices

A

openXBOW manual

The following listing shows the manual of openXBOW, which can be printed by calling the toolkit with option `-h`.

```
openXBOW - version 1.3.1 (published under GPL v3)

OpenXBOW generates an ARFF, CSV, or LibSVM file containing a bag-of-words
representation from an ARFF or CSV file of numeric low-level descriptors
and/or symbolic features (e.g., text).

Input format:
The first attribute/column must always contain an identifier for the corres-
ponding file / instance, i.e., a string containing the filename or an index,
e.g. 'instance_001.wav'.
In a CSV file, the last column may be a nominal or numeric class label.
In this case, there must be a header line, otherwise it is optional.
If the class labels are not given in the input data file, an additional CSV
file with class labels can be given: the first line can be a header line,
the first column contains the identifier string for each instance,
the second column the corresponding class label.

Example of a CSV input file:
'instance_001.wav';1.04E+01;2.3E+00;2.7E-01;classA
'instance_001.wav';9.02E+00;7.0E+01;1.1E-01;classA
'instance_001.wav';5.19E+01;4.4E+00;2.7E-01;classA
'instance_002.wav';1.24E+00;1.3E+01;2.8E-01;classB
'instance_002.wav';2.51E+01;6.7E+00;3.1E-01;classB
'instance_002.wav';4.24E+01;2.2E+01;8.0E-02;classB
'instance_003.wav';1.23E+01;4.3E+00;1.6E-01;classA
...

openXBOW options

-h              Print this help text

-i p            Name/Path of an input (ARFF or CSV) file p containing low-level
                feature vectors (over time). The first feature must be a string
                or number which specifies all feature vectors which belong to
                one instance.
-attributes p   An optional string, specifying all input attributes/columns
                (mandatory in case of multiple labels or if multiple codebooks
                are requested):
                n=name, t=time stamp, 0=symbolic feature, 1-9=numeric feature,
```

```
                  c=class label/numeric label, r=remove attribute
                  Using different numbers for numeric features will create a
                  separate codebook and bag for all features belonging to the same
                  index. The codebook index can be followed by brackets [] speci-
                  fying the number of consecutive input features belonging to this
                  index.
                  Example: -attributes nt1[14]2[14]c
                   Input file with the structure: name,timestamp,28 numeric
                                                  features split into two codebooks
                                                  (14 features each) and one label.
-attributesAlt p  An alternative option to specify all input attributes:
                  This option supports splitting the input feature space into
                  more than 9 numeric codebooks; also overlapping codebooks are
                  supported.
                  Specify the input attributes first: n=name, t=time stamp,
                   0=text feature, m=numeric feature, c=class label/numeric label,
                   r=remove attribute
                  Consecutive attributes of the same type may be noted in brackets,
                  specifying the number of consecutive attributes.
                  The attributes are followed by a '_' and the definition of the
                  input features for each codebook.The features are listed in
                  brackets [] specifying the indexes of input features that are
                  considered for the codebook.
                  Indexes start with 1 (for the first numeric feature) and are
                  separated by a +; ranges are defined with a -.
                  Example: -attributes_alt ntm[28]c_[1+2][3+4][5-7][8][9-11][12+13] ...
                             [14-20][1+21][22-24][25][2+26]
                   Input file with the structure: name,timestamp,28 numeric features
                   and one label. The numeric features are split into 11 codebooks,
                   each one covering a different selection of input features.
-o p              Name / Path of an output ARFF, CSV or LibSVM file p containing
                  the bag-of-words representation. The output file format is chosen
                  depending on the given file ending (*.arff, *.csv or *.libsvm).
-csvHeader        Print a header line if a CSV output file is requested (by default,
                  the output CSV file is without a header).
-csvSep p         Use separator p for the CSV output file (default: ; ).
-oi p             Name / Path of an output CSV file p containing the word indexes.
-svmModel p       Name / Path of a Liblinear model (must be L2R_LR_DUAL) to
                  decode the BoW. p specifies the model file. openXBOW outputs a
                  JSON file with the same name (unless given by oJson option).
-oJson p          Name / Path of the JSON output file including the predictions
                  of the Liblinear model (must be given by the option svmModel).
-writeName        Output the id string/number in the output file (only ARFF & CSV).
-writeTimeStamp   Output the time stamp in the output file (only ARFF & CSV; the
                  option -t must be provided).
-noLabels         Do not output the labels in the output file.
                  This option is useful in two cases:
                  1) The input file (-i) contains labels, but they are not desired
                   in the output (-o).
                  2) A labels file (-l) was given only to restrict the output (-o)
                   to a certain interval in time (see -l).
-arffLabels p     String containing all potential class labels (separator comma
                  without whitespaces) for ARFF output file. Only required if not all
                  labels are found in the input data or (?=unknown).
                  Example: -arffLabels class1,class2,class3
-append           Append output to file (if output file already exists).
-l p              CSV file p with the class labels for each analysis window/instance.
                  In case a label file is given, the output is restricted to the
                  instances, where labels are given. Both nominal and numeric classes
                  are supported.
                  Format:
                  1st line (optional): name (according to the input file); label1;
                                       label2; ...
```

```
                2nd line:           'file_1.wav'; class1; ...
                [and so on]

-t p1 p2        Segment the input files with a windows size (segment width) of
                p1 seconds and a hop size (shift) of p2 seconds
                If this option is used, the second column of the input file must
                be a time index (in seconds) of the current frame and the
                (optional) labels file must have the corresponding time stamp as
                the 2nd column (name; time; label).

-e p1 p2        Remove all feature vectors from the input, where the activity (or
                energy) is below p2. Index p1 specifies the index of the activity
                attribute (first index: 1).
-standardizeInput  Standardize (z-score) all numeric input features.
                The parameters are stored in the codebook file (-B) and then used
                for standardization of test data (-b) in an online approach.
-normalizeInput Normalize all numeric input features (min->max is normalized
                 to 0->1).
                The parameters are stored in the codebook file (-B) and then used
                for normalization of test data (-b) in an online approach.

-size p         Set the (initial) size p of the codebook. (default: size=500)
                In case of several codebooks (see -attributes) different sizes can
                be specified using separator comma, e.g., -size 200,500,100
-c p            Method of creating the codebook:
                p=random: Generate the codebook by a random sampling of the input
                  feature vectors.
                p=random++ (default): Generate the codebook by a random sampling
                  of the input feature vectors with a weighting, identical to the
                  initialization of kmeans++.
                p=kmeans: Employ kmeans clustering (Lloyd's algorithm).
                p=kmeans++: Employ kmeans++ clustering (Lloyd's algorithm).
                p=em: Employ EM (expectation maximization) clustering with a random
                  sampling for cluster initialization.
                p=em++: Employ EM (expectation maximization) clustering with a
                  random++ sampling for cluster initialization.
                p=em-kmeans: Employ EM (expectation maximization) clustering with
                  kmeans for cluster initialization.
                p=em-kmeans++: Employ EM (expectation maximization) clustering with
                  kmeans++ for cluster initialization.
                p=pdf: Generate a data-independent codebook with entries sampled
                  from a one-dimensional Gaussian pdf (zero mean, unit variance).
                  The option '-standardizeInput' is highly recommended when using
                  this option.
                p=generic: Generate a generic codebook (independent from data,
                  see option '-gen'). The parameter '-size' is not relevant when
                  selecting this method.
-gen p          Offset p for the values in the generic codebook ('-c generic').
                Example: A codebook with two input features will look like this:
                  -p,-p  -p,+p  +p,-p  +p,+p
-reduce p       Reduce the size of the codebook by merging words which are
                correlated with each other. PCC with threshold p is considered.
-supervised     Generate a codebook for each class separately, first, then merge
                all codebooks. (Not available for numeric labels.)
-seed p         Select the random seed p used for codebook creation.
                (Has no effect on training selection configured by -numTrain.)
-numTrain p     Randomly choose p feature vectors from the input data for the
                creation of the codebook (should not be used for random sampling).
                This option is useful to speed-up the clustering process.

-unigram p      Apply the n-gram approach to numeric features using unigrams.
                Only the p most frequent codewords are taken into account.
-bigram p       Apply the n-gram approach to numeric features using bigrams.
```

```
                The p most frequent codewords are taken into account.
-trigram p      Apply the n-gram approach to numeric features using trigrams.
                The p most frequent codewords are taken into account.
                The uni-/bi-/trigram codebooks are stored in the codebook
                file (-B) and used when loading a codebook (-b).
                In case of several codebooks (see -attributes) different sizes
                can be specified using separator comma, e.g., -bigram 200,600
                In case of using only bi-/trigrams (no unigrams), the standard
                BoW are no longer generated for the respective codebook.
                p=0 results in the standard BoW approach.

-b p            Load codebook p (do not create one).
-B p            Save the created codebook as a file p.

-minTermFreq p  Gives a minimum threshold for the number of occurrences of each
                word/n-gram to be considered for symbolic codebook generation
                (default: minTermFreq=1)
-maxTermFreq p  Gives a maximum threshold for the number of occurrences of each
                word/n-gram to be considered for symbolic codebook generation
                (default: maxTermFreq=0(inf))
-stopChar p     Specifies characters which are removed from all input
                instances (default: .,;:()?!* )
-nGram p        N-gram (symbolic) (default: nGram=1)
-nCharGram p    N-character-gram (symbolic) (default: nCharGram=0)

-a p            When creating the bag-of-words, assign each input feature
                vector to p closest words from the codebook. (default: a=1,
                only closest word)
                In case of several codebooks (see -attributes), a different
                number can be specified for each codebook using separator comma,
                e.g., -a 5,2
                This parameter is stored in the codebook file (-B) and used when
                the respective codebook is loaded (-b).
-gaussian p     Soft assignment using Gaussian encoding with standard deviation
                (stddev) p.
                In case of several codebooks (see -attributes), a different
                stddev can be specified for each codebook using separator comma,
                e.g., -a 25.0,30.0
                This parameter is stored in the codebook file (-B) and used when
                the respective codebook is loaded (-b).
-gmm p          Soft assignment using a GMM-like method.
                p=0 (default): Normal hard assignment (see option -a) is used.
                p=1: GMM-assignment WITHOUT priors.
                p=2: GMM-assignment WITH priors.
                In case of several codebooks (see -attributes), a different option
                can be specified for each codebook using separator comma,
                e.g., -gmm 0,2,1
                This option requires that all corresponding codebooks have been
                generated by an EM clustering method (see option -c)!
                This parameter is stored in the codebook file (-B) and used when
                the respective codebook is loaded (-b).
-off p          Off codebook words: Features with an Euclidean distance above
                threshold p to codewords are not be considered in the assignment
                step.
                In case of several codebooks (see -attributes), a different
                stddev can be specified for each codebook using separator comma,
                e.g., -off 25.0,30.0
                This parameter is stored in the codebook file (-B) and used when
                the respective codebook is loaded (-b).

-log            Logarithmic term weighting 'lg(TF+1)' of the term frequency.
                This parameter is stored in the codebook file (-B) and used when
                the respective codebook is loaded (-b).
```

```
-idf              Inverse document frequency transform: Multiply the term
                  frequency (TF) with the logarithm of the ratio of the
                  total number of instances and the number of instances where the
                  respective word is present.
                  This parameter is stored in the codebook file (-B) and used
                  when the respective codebook is loaded (-b).
-norm p           Normalize the bag-of-words (3 options of normalization).
                  p=1: Divides the term frequencies (TF) by the number of input
                    frames.
                  p=2: Divides the TF by the sum of all TFs.
                  p=3: Divides the TF by a factor so that the resulting Euclidean
                    length is 1.

-standardizeOutput  Standardize (z-score) all output bag-of-words features.
                  The parameters are stored in the codebook file (-B) and then
                  used for standardization of test data (-b) in an online approach.
-normalizeOutput  Normalize all output features (term frequencies, min->max is
                  normalized to 0->1).
                  The parameters are stored in the codebook file (-B) and then
                  used for normalization of test data (-b) in an online approach.

Example:
java -jar openXBOW.jar -i features.arff -o boaw.arff -l labels.csv -size 100
```

In Section 7.4.5, the BoAW features are generated using the following command:

```
java -jar openXBOW.jar -i llds.csv -o boaw.csv
-attributesAlt ntm[130]_[1-2+9][3-5][6+10][7-8+11-36][37-51][52-65]
[66-67+74][68-70][71+75][72-73+76-101][102-116][117-130]
-seed 10 -c random -standardizeInput
-size 600,600,400,5600,3000,2800,600,600,400,5600,3000,2800
-a 12,12,8,112,60,56,12,12,8,112,60,56
-log -B codebook.txt
```

This command presumes that `llds.csv` contains the ComParE LLDs (for the training set) as generated by openSMILE [23] with the included `ComParE_2016.conf` configuration file. The BoAW output is written into the file `boaw.csv` and the codebooks (including the parameters for z-score normalisation) are written into the file `codebook.txt`. For development and test sets, the codebook can be loaded with `-b codebook.txt`.

Acronyms

ACF ..... Autocorrelation function
AED ..... Acoustic event detection
AI ..... Artificial intelligence
ANN ..... Artificial neural network
ASC ..... Acoustic scene classification
ASR ..... Automatic speech recognition
AUC ..... Area under the curve
AVEC ..... Audio/Visual Emotion Challenge
BCE ..... Binary cross-entropy
BLSTM ..... Bidirectional long short-term memory
BoAW ..... Bag-of-audio-words
BoVW ..... Bag-of-visual-words
BoW ..... Bag-of-words
CA ..... Computer audition
CCC ..... Concordance correlation coefficient
CCE ..... Categorical cross-entropy
CNN ..... Convolutional neural network
ComParE ..... Computational Paralinguistics ChallengE
CV ..... Cross-validation
DCT ..... Discrete cosine transform
DDP ..... Difference of differences of periods (jitter)

DFT ..... Discrete Fourier transform
DL ..... Deep learning
DNN ..... Deep neural network
DWT ..... Discrete wavelet transform
E2E ..... End-to-end
EAT ..... Eating Analysis and Tracking (challenge)
EM ..... Expectation-maximisation
EWE ..... Evaluator weighted estimator
F0 ..... Fundamental frequency
FT ..... Fourier transform
GHIK ..... Generalised histogram intersection kernel
GMM ..... Gaussian mixture model
GRU ..... Gated recurrent unit
HIK ..... Histogram intersection kernel
HNR ..... Harmonics-to-noise ratio
IDF ..... Inverse document frequency
LLD ..... Low-level descriptor
log-TF ..... Logarithmic term frequency
LOSO ..... Leave-one-speaker-out
LPC ..... Linear predictive coding
LSTM ..... Long short-term memory
MAE ..... Mean absolute error
MFCCs ..... Mel-frequency cepstral coefficients
MIR ..... Music information retrieval
ML ..... Machine learning
MLP ..... Multi-layer perceptron
MPSSC ..... Munich-Passau snore sound corpus
MSE ..... Mean squared error
NLP ..... Natural language processing
PCC ..... Pearson's correlation coefficient

PDF ..... Probability density function
PLP ..... Perceptual linear prediction
RASTA-PLP ... Relative spectral transform perceptual linear prediction
ReLU ..... Rectified linear unit
RMS ..... Root mean square
RNN ..... Recurrent neural network
ROC ..... Receiver operating characteristic
RQ ..... Research question
SHS ..... Subharmonic summation
SNR ..... Signal-to-noise ratio
STFT ..... Short-time Fourier transform
SVM ..... Support vector machine
TF-IDF ..... Term frequency-inverse document frequency
UAR ..... Unweighted average recall
VAD ..... Voice activity detection
VQ ..... Vector quantisation
VSM ..... Vector space model
WPT ..... Wavelet packet transform
WT ..... Wavelet transform
XBOW ..... Crossmodal bag-of-words
ZCR ..... Zero-crossing rate

List of Symbols

Acc ..... Accuracy

ACF(h, d) ..... Short-time autocorrelation function

a_l ..... Coefficient for training instance l (SVM)

a_p ..... Linear predictive coding (LPC) coefficient

$a(b_i, D_j)$ ..... Audio word assignment function

b ..... Bias term (SVM)

β ..... Hyperparameter of GHIK

B ..... Number of spectral bands (Mel scale)

B_S ..... Block size / amount of context (BoAW, supra-segmental features)

$\mathcal{B}$ ..... Codebook used for BoAW

b_i ..... Codeword (audio word) in codebook $\mathcal{B}$

C ..... Complexity (SVM)

C_S ..... Codebook size (BoAW)

$\mathcal{C}$ ..... Corpus (dataset)

CEP(h) ..... Short-time cepstrum

d ..... Delay (autocorrelation function)

D ..... Delay compensation

d_i ..... Word in dictionary D_W

$D(h), D_j$ ..... Low-level descriptor (LLD) (of frame h, with index j)

$D_{\text{sma}}(h)$ ..... Smoothed LLD of frame h

$D_{\text{z},j}$ ..... z-score normalised LLD

$\Delta D(h)$ ..... Delta regression coefficient of LLD $D(h)$

$\Delta\Delta D(h)$ ..... Double delta regression / acceleration coefficient of LLD $D(h)$

$\Delta_{\text{diff}} D(h)$ ..... Temporal difference of LLD $D(h)$

$\bar{D}, \tilde{D}$ ..... Sample mean/standard deviation of LLDs

D_W ..... Dictionary for BoW-based VSM

$e_{i,j}$ ..... Entry in the confusion matrix

$e(n), E(z)$ ..... Excitation signal in the source-filter model

$E(h)$ ..... Short-time energy

$E_{\text{norm}}(h)$ ..... Normalised short-time energy

$E_{\text{rms}}(h)$ ..... RMS (root mean square) short-time energy

f ..... Frequency

$f()$ ..... Activation function

$F0(h)$ ..... Fundamental frequency

$F_i(h)$ ..... Frequency of the i-th formant

F_s ..... Sample rate

$F_{\text{B},i}(h)$ ..... Bandwidth of the i-th formant

F_{BoW} ..... BoW feature vector / VSM

F_{BoAW} ..... BoAW feature vector

$F_{\text{BoW,log}}$ ..... BoW feature vector / VSM with logarithmic term frequency weighting

$F_{\text{BoW,TFIDF}}$ ..... BoW feature vector / VSM with TFIDF weighting

F_{BoW,L^2} ..... BoW feature vector / VSM with L^2-normalisation

$F_{\text{Func}}(D)$ ..... Functional of the LLD contour $D(h)$

η ..... Learning rate

h ..... Index of the frame (hop)

H_s ..... Hop size, frame step

$H_{\text{voc}}(z)$ ..... Vocal tract filter function

I ..... Number of features

$\mathcal{I}$ ..... One data instance (sample, entity to be classified)

IDF_i ..... Inverse document frequency of the term with index i

j . . . . . Imaginary unit; index of an LLD vector

$J(\theta)$ . . . . . Loss function

k . . . . . Index of the frequency bin; number of cluster centroids (k-means)

k_c . . . . . Mel-frequency cepstral coefficient index (see below $\mathrm{MFCC}_{k_c}(h)$)

k_{emph} . . . . . Pre-emphasis coefficient

K . . . . . Number of classes

$K^{\Phi}(x, x')$ . . . . . Kernel

L . . . . . Number of training instances (samples)

$\mathcal{L}$ . . . . . Training set

L_c . . . . . Liftering coefficient (MFCC)

L_B . . . . . Length of the bag-of-audio-words codebook

L_D . . . . . Length of the bag-of-words dictionary/the feature vector of the VSM

L_S . . . . . Length of word sequence S_W

m . . . . . Index of the samples within one frame

M . . . . . Frame/window length (in samples)

$\mathrm{MFCC}_{k_c}(h)$ . . . . . Mel-frequency cepstral coefficient k_c

n . . . . . Index of the discrete time

N . . . . . Number of samples of a signal

N_h . . . . . Total number of frames

N_A . . . . . Number of assignments (BoAW)

$\xi^{(l)}$ . . . . . Slack variable for instance l

p . . . . . p-value (hypothesis testing)

π_i . . . . . Mixture weight (GMM)

p_i . . . . . Pole of a polynomial or filter in z-transform representation

P . . . . . Order of the linear predictive coding (LPC) model

Prec . . . . . Precision

q . . . . . Quefrency (cepstrum)

$\rho_{\mathrm{PCC}}, \rho_{\mathrm{CCC}}$ . . . . . Pearson/concordance correlation coefficient

Rec ..... Recall
$s(n)$ ..... Signal (discrete-time, one-dimensional)
$\hat{s}(n)$ ..... Approximated signal $s(n)$
$s_t(t)$ ..... Time-domain signal
$s_f(h, m)$ ..... Framed signal
$s_w(h, m)$ ..... Windowed signal
s_y^2 ..... Variance
$s_{x,y}$ ..... Covariance
$S(z)$ ..... Z-transform of a signal $s(n)$
$S_{\mathrm{FT}}(f)$ ..... Fourier transform
$S_{\mathrm{DFT}}(k)$ ..... Discrete Fourier transform (DFT)
$S_{\mathrm{STFT}}(h, k)$ ..... Discrete short-time Fourier transform (STFT)
$S_{\mathrm{STFT,norm}}(h, k)$ ... Normalised STFT
$S_{\mathrm{STFT,P,norm}}(h, k)$ .. STFT-based power spectral density
S_W ..... Word sequence
σ ..... Standard deviation/hyperparameter for Gaussian encoding
θ ..... Model parameters (ANN)
$t_f(h)$ ..... Frame time of a frame with index h
$\mathcal{T}$ ..... Development/test set
T_s ..... Sampling period ($T_s = F_s^{-1}$)
w ..... Normal vector / weight vector (SVM / neural network)
w_i ..... Weight (neural network)
w_j ..... Word in sequence S_W
$w(m)$ ..... Windowing function
$w_{\mathrm{hamming}}(m)$ ..... Hamming window function
$w_{\mathrm{hann}}(m)$ ..... Hann window function
$w_{\mathrm{Gaussian}}(m)$ ..... Gaussian window function
W_Δ ..... Length of the context window of the delta regression
W_{sma} ..... Length of the context window for smoothing with a moving average filter

x ..... Feature vector

$x^{(l)}, y^{(l)}$ ..... Feature vector/label of a training instance

$x^{(t)}, y^{(t)}$ ..... Feature vector/label of a test instance

y ..... Label

$\hat{y}^{(l)}, \hat{y}^{(t)}$ ..... Predicted label of a training or test instance, respectively

z_{mel} ..... Critical band rate (Mel)

ZCR(h) ..... Zero-crossing rate

Φ ..... Feature space transformation

Bibliography

[1] B. Schuller, *Intelligent Audio Analysis*, ser. Signals and Communication Technology. Springer, 2013.

[2] D. Schiller, K. Weitz, K. Janowski, and E. André, "Human-inspired socially-aware interfaces," in *Proceedings of the 8th International Conference on Theory and Practice of Natural Computing (TPNC)*. Kingston, ON, Canada: Springer, 2019, pp. 41–53.

[3] D. Barchiesi, D. Giannoulis, D. Stowell, and M. D. Plumbley, "Acoustic scene classification: Classifying environments from the sounds they produce," *IEEE Signal Processing Magazine*, vol. 32, no. 3, pp. 16–34, 2015.

[4] N. Cummins, S. Scherer, J. Krajewski, S. Schnieder, J. Epps, and T. F. Quatieri, "A review of depression and suicide risk assessment using speech analysis," *Speech Communication*, vol. 71, pp. 10–49, 2015.

[5] N. Cummins, M. Schmitt, S. Amiriparian, J. Krajewski, and B. Schuller, ""you sound ill, take the day off": Automatic recognition of speech affected by upper respiratory tract infection," in *Proceedings of the 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. Jeju Island, Korea: IEEE, 2017, pp. 3806–3809.

[6] C. Janott, C. Rohrmeier, M. Schmitt, W. Hemmert, and B. Schuller, "Snoring – an acoustic definition," in *Proceedings of the 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. Berlin, Germany: IEEE, 2019, pp. 3653–3657.

[7] R. Sonnleitner and G. Widmer, "Robust quad-based audio fingerprinting," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 3, pp. 409–421, 2015.

[8] N. Scaringella, G. Zoia, and D. Mlynek, "Automatic genre classification of music content: a survey," *IEEE Signal Processing Magazine*, vol. 23, no. 2, pp. 133–141, 2006.

[9] S. Sigtia, E. Benetos, and S. Dixon, "An end-to-end neural network for polyphonic piano music transcription," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 5, pp. 927–939, 2016.

[10] U. Zölzer, *Digitale Audiosignalverarbeitung.* Springer, 2013.

[11] B. Schuller, S. Steidl, A. Batliner, E. Bergelson, J. Krajewski, C. Janott, A. Amatuni, M. Casillas, A. Seidl, M. Soderstrom, A. S. Warlaumont, G. Hidalgo, S. Schnieder, C. Heiser, W. Hohenhorst, M. Herzog, M. Schmitt, K. Qian, Y. Zhang, G. Trigeorgis, P. Tzirakis, and S. Zafeiriou, "The interspeech 2017 computational paralinguistics challenge: Addressee, cold & snoring," in *Proceedings of INTERSPEECH.* Stockholm, Sweden: ISCA, 2017, pp. 3442–3446.

[12] B. W. Schuller, S. Steidl, A. Batliner, P. B. Marschik, H. Baumeister, F. Dong, S. Hantke, F. B. Pokorny, E.-M. Rathner, K. D. Bartl-Pokorny, C. Einspieler, D. Zhang, A. Baird, S. Amiriparian, K. Qian, Z. Ren, M. Schmitt, P. Tzirakis, and S. Zafeiriou, "The interspeech 2018 computational paralinguistics challenge: Atypical & self-assessed affect, crying & heart beats." in *Proceedings of INTERSPEECH.* Hyderabad, India: ISCA, 2018, pp. 122–126.

[13] B. W. Schuller, A. Batliner, C. Bergler, F. B. Pokorny, J. Krajewski, M. Cychosz, R. Vollmann, S.-D. Roelen, S. Schnieder, E. Bergelson, A. Cristia, A. Seidl, A. S. Warlaumont, L. Yankowitz, E. Nöth, S. Amiriparian, S. Hantke, and M. Schmitt, "The interspeech 2019 computational paralinguistics challenge: Styrian dialects, continuous sleepiness, baby sounds & orca activity," in *Proceedings of INTERSPEECH.* Graz, Austria: ISCA, 2019, pp. 2378–2382.

[14] "Kaggle: Your machine learning and data science community," https://www.kaggle.com/, 2021, link retrieved on 14 January 2021.

[15] B. Schuller, S. Steidl, and A. Batliner, "The interspeech 2009 emotion challenge," in *Proceedings of INTERSPEECH.* Brighton, UK: ISCA, 2009, pp. 312–315.

[16] B. Schuller, M. Valstar, F. Eyben, G. McKeown, R. Cowie, and M. Pantic, "AVEC 2011 – the first international audio/visual emotion challenge," in *Proceedings of the 4^{th} International Conference on Affective Computing and Intelligent Interaction (ACII)*, AAAC. Memphis, TN, USA: Springer, 2011, pp. 415–424.

[17] F. Ringeval, B. Schuller, M. Valstar, N. Cummins, R. Cowie, L. Tavabi, M. Schmitt, S. Alisamir, S. Amiriparian, E.-M. Messner, S. Song, S. Liu, Z. Zhao, A. Mallol-Ragolta, Z. Ren, M. Soleymani, and M. Pantic, "AVEC 2019 workshop and challenge: state-of-mind, detecting depression with AI, and cross-cultural affect recognition," in *Proceedings of the 9^{th} International Audio/Visual Emotion Challenge and Workshop (AVEC 19).* Nice, France: ACM, 2019, pp. 3–12.

[18] F. Ringeval, B. Schuller, M. Valstar, J. Gratch, R. Cowie, S. Scherer, S. Mozgai, N. Cummins, M. Schmitt, and M. Pantic, "AVEC 2017 – real-life depression, and

affect recognition workshop and challenge," in *Proceedings of the 7th Annual Workshop on Audio/Visual Emotion Challenge (AVEC 17)*. Mountain View, CA, USA: ACM, 2017, pp. 3–9.

[19] F. Ringeval, B. Schuller, M. Valstar, R. Cowie, H. Kaya, M. Schmitt, S. Amiriparian, N. Cummins, D. Lalanne, A. Michaud, E. Çiftçi, H. Güleç, A. A. Salah, and M. Pantic, "AVEC 2018 workshop and challenge: bipolar disorder and cross-cultural affect recognition," in *Proceedings of the 2018 Audio/Visual Emotion Challenge and Workshop (AVEC 18)*. Seoul, Korea: ACM, 2018, pp. 3–13.

[20] B. Schuller, S. Steidl, A. Batliner, A. Vinciarelli, K. Scherer, F. Ringeval, M. Chetouani, F. Weninger, F. Eyben, E. Marchi, M. Mortillaro, H. Salamin, A. Polychroniou, F. Valente, and S. Kim, "The interspeech 2013 computational paralinguistics challenge: Social signals, conflict, emotion, autism," in *Proceedings of INTERSPEECH*. Lyon, France: ISCA, 2013, pp. 148–152.

[21] B. Schuller, S. Steidl, A. Batliner, S. Hantke, F. Hönig, J. R. Orozco-Arroyave, E. Nöth, Y. Zhang, and F. Weninger, "The interspeech 2015 computational paralinguistics challenge: Nativeness, parkinson's & eating condition," in *Proceedings of INTERSPEECH*. Dresden, Germany: ISCA, 2015, pp. 478–482.

[22] B. Schuller, S. Steidl, A. Batliner, J. Hirschberg, J. K. Burgoon, A. Baird, A. Elkins, Y. Zhang, E. Coutinho, and K. Evanini, "The interspeech 2016 computational paralinguistics challenge: Deception, sincerity & native language," in *Proceedings of INTERSPEECH*. San Francisco, CA, USA: ISCA, 2016, pp. 2001–2005.

[23] F. Eyben, F. Weninger, F. Groß, and B. Schuller, "Recent developments in openSMILE, the munich open-source multimedia feature extractor," in *Proceedings of the 21st ACM International Conference on Multimedia (ACM MM)*. Barcelona, Spain: ACM, 2013, pp. 835–838.

[24] C.-C. M. Yeh and Y.-H. Yang, "Supervised dictionary learning for music genre classification," in *Proceedings of the 2nd ACM International Conference on Multimedia Retrieval (ICMR)*, Hong Kong, China, 2012, pp. 1–8.

[25] M. Riley, E. Heinen, and J. Ghosh, "A text retrieval approach to content-based audio hashing," in *Proceedings of the 9th International Conference on Music Information Retrieval*. Philadelphia, PA, USA: ISMIR, 2008, pp. 295–300.

[26] S. Pancoast and M. Akbacak, "Bag-of-audio-words approach for multimedia event classification," in *Proceedings of INTERSPEECH*. Portland, OR, USA: ISCA, 2012, pp. 2105–2108.

[27] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *Proceedings of the European Conference on Machine Learning*. Springer, 1998, pp. 137–142.

[28] G. Forman, "Bns feature scaling: an improved representation over tf-idf for svm text classification," in *Proceedings of the 17th ACM Conference on Information and Knowledge Management*. Napa Valley, CA, USA: ACM, 2008, pp. 263–270.

[29] M. Schmitt and B. Schuller, "openXBOW-introducing the Passau open-source crossmodal bag-of-words toolkit," *The Journal of Machine Learning Research*, vol. 18, pp. 1–5, 2017.

[30] F. Eyben, *Real-time Speech and Music Classification by Large Audio Feature Space Extraction*, ser. Springer Theses, Recognizing Outstanding Ph.D. Research. Springer, 2016.

[31] F. Eyben, K. R. Scherer, B. W. Schuller, J. Sundberg, E. André, C. Busso, L. Y. Devillers, J. Epps, P. Laukka, S. S. Narayanan, and K. P. Truong, "The Geneva minimalistic acoustic parameter set (GeMAPS) for voice research and affective computing," *IEEE Transactions on Affective Computing*, vol. 7, no. 2, pp. 190–202, 2015.

[32] B. Schuller and A. Batliner, *Computational Paralinguistics: Emotion, Affect and Personality in Speech and Language Processing*. John Wiley & Sons, 2014.

[33] E. Zwicker and H. Fastl, *Psychoacoustics: Facts and Models*. Berlin, Heidelberg: Springer, 2013, vol. 22.

[34] J. S. Lim and A. V. Oppenheim, "Enhancement and bandwidth compression of noisy speech," *Proceedings of the IEEE*, vol. 67, no. 12, pp. 1586–1604, 1979.

[35] F. Weninger, H. Erdogan, S. Watanabe, E. Vincent, J. Le Roux, J. R. Hershey, and B. Schuller, "Speech enhancement with lstm recurrent neural networks and its application to noise-robust asr," in *Proceedings of the International Conference on Latent Variable Analysis and Signal Separation*. Springer, 2015, pp. 91–99.

[36] L. Hertel, H. Phan, and A. Mertins, "Comparing time and frequency domain for audio event recognition using deep learning," in *International Joint Conference on Neural Networks (IJCNN)*. Vancouver, BC, Canada: IEEE, 2016, pp. 3407–3411.

[37] J. R. Deller Jr, J. G. Proakis, and J. H. Hansen, *Discrete Time Processing of Speech Signals*. Prentice Hall PTR, 1993.

[38] M. Müller, *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*. Springer, 2015.

[39] D. C. von Grünigen, *Digitale Signalverarbeitung*. Fachbuchverl. Leipzig im Carl Hanser Verlag, 2001.

[40] C.-h. Chen, *Signal Processing Handbook*. New York, NY, USA: Marcel Dekker Inc., 1988.

[41] R. G. Bachu, S. Kopparthi, B. Adapa, and B. D. Barkana, "Voiced/unvoiced decision for speech signals based on zero-crossing rate and energy," in *Advanced Techniques in Computing Sciences and Software Engineering*, K. Elleithy, Ed. Springer, 2010, pp. 279–282.

[42] F. Gouyon, F. Pachet, and O. Delerue, "On the use of zero-crossing rate for an application of classification of percussive sounds," in *Proceedings of the COST G-6 Conference on Digital Audio Effects*. Verona, Italy: DAFx, 2000, 6 pages.

[43] J.-R. Ohm and H. D. Lüke, *Signalübertragung: Grundlagen der digitalen und analogen Nachrichtenübertragungssysteme*, 11st ed. Berlin, Heidelberg: Springer, 2015.

[44] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of computation*, vol. 19, no. 90, pp. 297–301, 1965.

[45] D. Gabor, "Theory of communication. part 1: The analysis of information," *Journal of the Institution of Electrical Engineers-Part III: Radio and Communication Engineering*, vol. 93, no. 26, pp. 429–457, 1946.

[46] W. D. Ward, "Musical perception," in *Foundations of modern auditory theory*, J. V. Tobias, Ed., 1970, vol. 1, pp. 407–447.

[47] F. Burkhardt, A. Paeschke, M. Rolfes, W. F. Sendlmeier, and B. Weiss, "A database of German emotional speech," in *Proceedings of INTERSPEECH*, vol. 5, Lisbon, Portugal, 2005, pp. 1517–1520.

[48] D. P. W. Ellis and D. F. Rosenthal, "Mid-level representations for computational auditory scene analysis," in *Proceedings of the International Joint Conference on Artificial Intelligence – Workshop on Computational Auditory Scene Analysis*. Montreal, QC, Canada: AAAI, CSCSI, 1995, 7 pages.

[49] F. Weninger, J. R. Hershey, J. Le Roux, and B. Schuller, "Discriminatively trained recurrent neural networks for single-channel speech separation," in *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. Atlanta, GA, USA: IEEE, 2014, pp. 577–581.

[50] H. Hermansky, "Perceptual linear predictive (plp) analysis of speech," *The Journal of the Acoustical Society of America*, vol. 87, no. 4, pp. 1738–1752, 1990.

[51] H. Hermansky, N. Morgan, A. Bayya, and P. Kohn, "Rasta-plp speech analysis," in *Proceedings of the 17th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. San Francisco, CA, USA: IEEE, 1992, pp. 121–124.

[52] J. Ohm, *Multimedia communication technology: Representation, transmission and identification of multimedia signals*. Berlin, Heidelberg: Springer, 2012.

[53] S. G. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 674–693, 1989.

[54] A. N. Akansu and Y. Liu, "On signal decomposition techniques," *Optical Engineering*, vol. 30, no. 7, pp. 912–921, 1991.

[55] R. N. Khushaba, "Application of biosignal-driven intelligent systems for multifunction prosthesis control," Ph.D. dissertation, University of Technology, Sydney, 2010.

[56] R. N. Khushaba, S. Kodagoda, S. Lal, and G. Dissanayake, "Driver drowsiness classification using fuzzy wavelet-packet-based feature-extraction algorithm," *IEEE Transactions on Biomedical Engineering*, vol. 58, no. 1, pp. 121–131, 2010.

[57] K. Qian, "Automatic general audio signal classification," Ph.D. dissertation, Technische Universität München, Germany, 2018.

[58] Y. Sagisaka, N. Campbell, and N. Higuchi, *Computing Prosody: Computational Models for Processing Spontaneous Speech*. Springer Science & Business Media, 2012.

[59] N. Campbell and P. Mokhtari, "Voice quality: the 4th prosodic dimension," in *Proceedings of the 15th International Congress of Phonetic Sciences (ICPhS)*. Barcelona, Spain: International Phonetic Association, 2003, pp. 2417–2420.

[60] A. N. Chasaide and C. Gobl, "Voice quality and f0 in prosody: towards a holistic account," in *Proceedings of the International Conference on Speech Prosody*. Nara, Japan: ISCA, 2004, pp. 189–196.

[61] A. M. Sluijter and V. J. v. Heuven, "Perceptual cues of linguistic stress: intensity revisited," in *ESCA Workshop on Prosody*, Lund, Sweden, 1993, pp. 246–249.

[62] D. Hirst and A. Di Cristo, *A survey of intonation systems*. Cambridge University Press Cambridge, 1998, ch. 1, pp. 1–44.

[63] A. De Cheveigné and H. Kawahara, "YIN, a fundamental frequency estimator for speech and music," *The Journal of the Acoustical Society of America*, vol. 111, no. 4, pp. 1917–1930, 2002.

[64] C. T. Ishi, H. Ishiguro, and N. Hagita, "Automatic extraction of paralinguistic information using prosodic features related to f0, duration and voice quality," *Speech Communication*, vol. 50, no. 6, pp. 531–543, 2008.

[65] K. Sönmez, E. Shriberg, L. Heck, and M. Weintraub, "Modeling dynamic prosodic variation for speaker verification," in *Proceedings of the 5th International Conference on Spoken Language Processing (ICSLP)*. Sydney, Australia: Australian Speech Science and Technology Association (ASSTA), 1998, 4 pages.

[66] W. Hess, *Pitch Determination of Speech Signals: Algorithms and Devices*. Berlin, Heidelberg: Springer, 2012, vol. 3.

[67] A. d. Cheveigné and H. Kawahara, "Comparative evaluation of f0 estimation algorithms," in *Proceedings of the 7th European Conference on Speech Communication and Technology (EUROSPEECH)*. Aalborg, Denmark: ISCA, 2001, pp. 2451–2454.

[68] C. Wendt and A. P. Petropulu, "Pitch determination and speech segmentation using the discrete wavelet transform," in *Proceedings of the IEEE International Symposium on Circuits and Systems. Circuits and Systems Connecting the World (ISCAS)*, vol. 2. Atlanta, GA, USA: IEEE, 1996, pp. 45–48.

[69] F. Kurth, A. Cornaggia-Urrigshardt, and S. Urrigshardt, "Robust f0 estimation in noisy speech signals using shift autocorrelation," in *Proceedings of the 39th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Florence, Italy: IEEE, 2014, pp. 1468–1472.

[70] S. Xu and H. Shimodaira, "Direct f0 estimation with neural-network-based regression," in *Proceedings of INTERSPEECH*. Graz, Austria: ISCA, 2019, pp. 1995–1999.

[71] D. E. Hall, *Musikalische Akustik: Ein Handbuch*, J. Goebel, Ed. Mainz: Schott, 2008, veröffentlichung des Zentrums für Kunst und Medientechnologie Karlsruhe, Institut für Musik und Akustik.

[72] D. J. Hermes, "Measurement of pitch by subharmonic summation," *The Journal of the Acoustical Society of America*, vol. 83, no. 1, pp. 257–264, 1988.

[73] A. Batliner, J. Buckow, R. Huber, V. Warnke, E. Nöth, and H. Niemann, "Prosodic feature evaluation: Brute force or well designed," in *Proceedings of the 14th International Congress of Phonetic Sciences (ICPhS)*, vol. 3. San Francisco, CA, USA: International Phonetic Association, 1999, pp. 2315–2318.

[74] H. Sierro, "Automatically detected acoustic landmarks for assessing natural emotion from speech," Master's thesis, University of Fribourg, Switzerland, 2013.

[75] S. Tilsen and K. Johnson, "Low-frequency fourier analysis of speech rhythm," *The Journal of the Acoustical Society of America*, vol. 124, no. 2, pp. EL34–EL39, 2008.

[76] S. E. Linville, "Intraspeaker variability in fundamental frequency stability: An age-related phenomenon?" *The Journal of the Acoustical Society of America*, vol. 83, no. 2, pp. 741–745, 1988.

[77] R. F. Orlikoff and R. Baken, "The effect of the heartbeat on vocal fundamental frequency perturbation," *Journal of Speech, Language, and Hearing Research*, vol. 32, no. 3, pp. 576–582, 1989.

[78] K. Koike, H. Suzuki, and H. Saito, "Prosodic parameters in emotional speech," in *Proceedings of the 5th International Conference on Spoken Language Processing (ICSLP)*. Sydney, Australia: Australian Speech Science and Technology Association (ASSTA), 1998, 4 pages.

[79] E. Zetterholm, "Prosody and voice quality in the expression of emotions," in *Proceedings of the 5th International Conference on Spoken Language Processing (ICSLP)*. Sydney, Australia: Australian Speech Science and Technology Association (ASSTA), 1998, 5 pages.

[80] R. Cowie, E. Douglas-Cowie, N. Tsapatsoulis, G. Votsis, S. Kollias, W. Fellenz, and J. G. Taylor, "Emotion recognition in human-computer interaction," *IEEE Signal Processing Magazine*, vol. 18, no. 1, pp. 32–80, 2001.

[81] B. Schuller, S. Steidl, A. Batliner, F. Burkhardt, L. Devillers, C. Müller, and S. S. Narayanan, "The interspeech 2010 paralinguistic challenge," in *Proceedings of INTERSPEECH*. Makuhari, Chiba, Japan: ISCA, 2010, pp. 2794–2797.

[82] B. Schuller, S. Steidl, A. Batliner, E. Nöth, A. Vinciarelli, F. Burkhardt, R. v. Son, F. Weninger, F. Eyben, T. Bocklet, G. Mohammadi, and B. Weiss, "The interspeech 2012 speaker trait challenge," in *Proceedings of INTERSPEECH*. Portland, OR, USA: ISCA, 2012, pp. 148–152.

[83] B. Schuller, S. Steidl, A. Batliner, F. Schiel, and J. Krajewski, "The interspeech 2011 speaker state challenge," in *Proceedings of INTERSPEECH*. Florence, Italy: ISCA, 2011, pp. 3201–3204.

[84] S. Hantke, M. Schmitt, P. Tzirakis, and B. Schuller, "Eat– the icmi 2018 eating analysis and tracking challenge," in *Proceedings of the 20th ACM International Conference on Multimodal Interaction (ICMI)*. Boulder, CO, USA: ACM, 2018, pp. 559–563.

[85] B. Schuller, S. Steidl, A. Batliner, J. Epps, F. Eyben, F. Ringeval, E. Marchi, and Y. Zhang, "The interspeech 2014 computational paralinguistics challenge: Cognitive & physical load," in *Proceedings of INTERSPEECH*. Singapore, Singapore: ISCA, 2014, pp. 427–431.

[86] T. Bocklet, E. Nöth, G. Stemmer, H. Ruzickova, and J. Rusz, "Detection of persons with parkinson's disease by acoustic, vocal, and prosodic analysis," in *Proceedings of the IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU)*. Big Island, HI, USA: IEEE, 2011, pp. 478–483.

[87] J. McCann and S. Peppé, "Prosody in autism spectrum disorders: a critical review," *International Journal of Language & Communication Disorders*, vol. 38, no. 4, pp. 325–350, 2003.

[88] R. B. Grossman, R. H. Bemis, D. P. Skwerer, and H. Tager-Flusberg, "Lexical and affective prosody in children with high-functioning autism," *Journal of Speech, Language, and Hearing Research*, vol. 53, no. 3, pp. 778–793, 2010.

[89] J. A. Russell, "A circumplex model of affect," *Journal of Personality and Social Psychology*, vol. 39, no. 6, p. 1161, 1980.

[90] H. S. Cheang and M. D. Pell, "The sound of sarcasm," *Speech communication*, vol. 50, no. 5, pp. 366–381, 2008.

[91] A. Chen, "Perception of paralinguistic intonational meaning in a second language," *Language Learning*, vol. 59, no. 2, pp. 367–409, 2009.

[92] M. P. Gelfer and V. A. Mikos, "The relative contributions of speaking fundamental frequency and formant frequencies to gender identification based on isolated vowels," *Journal of Voice*, vol. 19, no. 4, pp. 544–554, 2005.

[93] P. Vary and R. Martin, *Digital Speech Transmission: Enhancement, Coding and Error Concealment*. John Wiley & Sons, 2006.

[94] W. Han, C.-F. Chan, C.-S. Choy, and K.-P. Pun, "An efficient mfcc extraction method in speech recognition," in *IEEE International Symposium on Circuits and Systems*. Island of Kos, Greece: IEEE, 2006, pp. 145–148.

[95] E. Loweimi, S. M. Ahadi, T. Drugman, and S. Loveymi, "On the importance of pre-emphasis and window shape in phase-based speech recognition," in *Proceedings of the International Conference on Nonlinear Speech Processing (NOLISP)*. Mons, Belgium: Springer, 2013, pp. 160–167.

[96] P. Boersma, "Praat, a system for doing phonetics by computer," *Glot International*, vol. 5, no. 9, pp. 341–345, 2001.

[97] R. C. Snell and F. Milinazzo, "Formant location from lpc analysis data," *IEEE Transactions on Speech and Audio Processing*, vol. 1, no. 2, pp. 129–134, 1993.

[98] J. W. Picone, "Signal modeling techniques in speech recognition," *Proceedings of the IEEE*, vol. 81, no. 9, pp. 1215–1247, 1993.

[99] F. Zheng, G. Zhang, and Z. Song, "Comparison of different implementations of mfcc," *Journal of Computer Science and Technology*, vol. 16, no. 6, pp. 582–589, 2001.

[100] B. P. Bogert, M. J. R. Healy, and J. W. Tukey, "The quefrency alanysis of time series for echoes; cepstrum, pseudo-autocovariance, cross-cepstrum and saphe cracking," in *Proceedings of the Symposium on Time Series Analysis*. John Wiley & Sons, 1963, pp. 209–243.

[101] A. M. Noll, "Short-time spectrum and "cepstrum" techniques for vocal-pitch detection," *The Journal of the Acoustical Society of America*, vol. 36, no. 2, pp. 296–302, 1964.

[102] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland, "The HTK Book," *Cambridge University Engineering Department*, vol. 3.4, 2009.

[103] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Transactions on Computers*, vol. 100, no. 1, pp. 90–93, 1974.

[104] V. Britanak, P. C. Yip, and K. R. Rao, *Discrete Cosine and Sine Transforms: General Properties, Fast Algorithms and Integer Approximations*. Elsevier, 2010.

[105] M. Dorfer, B. Lehner, H. Eghbal-zadeh, H. Christop, P. Fabian, and W. Gerhard, "Acoustic scene classification with fully convolutional neural networks and i-vectors," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop*, Surrey, U. K., 2018.

[106] F. De Leon and K. Martinez, "Enhancing timbre model using mfcc and its time derivatives for music similarity estimation," in *Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, EURASIP. Bucharest, Romania: IEEE, 2012, pp. 2005–2009.

[107] B. Schuller, "Automatische Emotionserkennung aus sprachlicher und manueller Interaktion," Ph.D. dissertation, Technische Universität München, 2006.

[108] F. Weninger, F. Eyben, and B. Schuller, "On-line continuous-time music mood regression with deep recurrent neural networks," in *Proceedings of the 39th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Florence, Italy: IEEE, 2014, pp. 5412–5416.

[109] R. Kehrein, "The prosody of authentic emotions," in *Proceedings of the International Conference on Speech Prosody*. Aix-en-Provence, France: ISCA, 2002, 4 pages.

[110] M. Schmitt and B. W. Schuller, *Machine-Based Decoding of Paralinguistic Vocal Features*. Oxford University Press, 2018, ch. 33, pp. 719–742.

[111] M. Schmitt, E. Marchi, F. Ringeval, and B. Schuller, "Towards cross-lingual automatic diagnosis of autism spectrum condition in children's voices," in *Proceedings of the 11th ITG Symposium on Speech Communication (ITG SC)*, VDE. Paderborn, Germany: IEEE, 2016, pp. 264–268.

[112] M. Schmitt, F. Ringeval, and B. Schuller, "At the border of acoustics and linguistics: Bag-of-audio-words for the recognition of emotions in speech," in *Proceedings of INTERSPEECH*. San Francisco, CA, USA: ISCA, 2016, pp. 495–499.

[113] T. L. Nwe, S. W. Foo, and L. C. D. Silva, "Speech emotion recognition using hidden markov models," *Speech Communication*, vol. 41, no. 4, pp. 603–623, 2003.

[114] B. Schuller, G. Rigoll, and M. Lang, "Hidden markov model-based speech emotion recognition," in *Proceedings of the 28th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. II. Hong Kong, China: IEEE, 2003, pp. 1–4.

[115] J. Han, Z. Zhang, M. Schmitt, M. Pantic, and B. Schuller, "From hard to soft: Towards more human-like emotion recognition by modelling the perception uncertainty," in *Proceedings of the 25th ACM International Conference on Multimedia (ACM MM)*. Mountain View, CA; USA: ACM, 2017, pp. 890–897.

[116] B. W. Schuller, A. Batliner, C. Bergler, E.-M. Messner, A. Hamilton, S. Amiriparian, A. Baird, G. Rizos, M. Schmitt, L. Stappen, H. Baumeister, A. D. MacIntyre, and

S. Hantke, "The interspeech 2020 computational paralinguistics challenge: Elderly emotion, breathing & masks," in *Proceedings of INTERSPEECH*. Shanghai, China: ISCA, 2020, 5 pages, to be published.

[117] C. Janott, M. Schmitt, Y. Zhang, K. Qian, V. Pandit, Z. Zhang, C. Heiser, W. Hohenhorst, M. Herzog, W. Hemmert, and B. Schuller, "Snoring classified: The Munich-Passau snore sound corpus," *Computers in Biology and Medicine*, vol. 94, pp. 106–118, 2018.

[118] R. Banse and K. R. Scherer, "Acoustic profiles in vocal emotion expression," *Journal of Personality and Social Psychology*, vol. 70, no. 3, pp. 614–636, 1996.

[119] P. N. Juslin and P. Laukka, "Communication of emotions in vocal expression and music performance: Different channels, same code?" *Psychological Bulletin*, vol. 129, no. 5, pp. 770–814, 2003.

[120] J. Sundberg, S. Patel, E. Bjorkner, and K. R. Scherer, "Interdependencies among voice source parameters in emotional speech," *IEEE Transactions on Affective Computing*, vol. 2, no. 3, pp. 162–174, 2011.

[121] F. Eyben, M. Wöllmer, and B. Schuller, "OpenSMILE: the Munich versatile and fast open-source audio feature extractor," in *Proceedings of the 18th ACM International Conference on Multimedia (ACM MM)*. Firenze, Italy: ACM, 2010, pp. 1459–1462.

[122] "Opensmile 2.3," https://www.audeering.com/opensmile/, 2016, link retrieved on 28 February 2020.

[123] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd ed., Elsevier, Ed. Morgen Kaufmann, 2011.

[124] G. Salton, A. Wong, and C.-S. Yang, "A vector space model for automatic indexing," *Communications of the ACM*, vol. 18, no. 11, pp. 613–620, 1975.

[125] C. D. Manning, P. Raghavan, and H. Schütze, *An Introduction to Information Retrieval*. Cambridge, U. K.: Cambridge University Press, 2009, online Edition.

[126] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, and J. Gao, "Deep learning based text classification: A comprehensive review," *arXiv preprint arXiv:2004.03705*, pp. 1–42, 2020.

[127] A. Moreno-Ortiz, "Lingmotif: Sentiment analysis for the digital humanities," in *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. Valencia, Spain: ACL, 2017, pp. 73–76.

[128] M. A. Boukhaled and J.-G. Ganascia, "Using function words for authorship attribution: Bag-of-words vs. sequential rules," in *Proceedings of the 11th International Workshop on Natural Language Processing and Cognitive Science*. Venice, Italy: De Gruyter, 2015, pp. 115–122.

[129] T. S. Guzella and W. M. Caminhas, "A review of machine learning approaches to spam filtering," *Expert Systems with Applications*, vol. 36, no. 7, pp. 10 206–10 222, 2009.

[130] P. Kolari, A. Java, T. Finin, T. Oates, and A. Joshi, "Detecting spam blogs: A machine learning approach," in *Proceedings of the National Conference on Artificial Intelligence*, vol. 21, no. 2. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006, p. 1351.

[131] G. Salton, "Developments in automatic text retrieval," *Science*, vol. 253, no. 5023, pp. 974–980, 1991.

[132] T. Danisman and A. Alpkocak, "Feeler: Emotion classification of text using vector space model," in *Proceedings of the AISB 2008 Convention Communication, Interaction and Social Intelligence*, Aberdeen, Scotland, U. K., 2008, pp. 897–906.

[133] N. Colnerič and J. Demšar, "Emotion recognition on twitter: Comparative study and training a unison model," *IEEE Transactions on Affective Computing*, vol. 11, no. 3, pp. 433–446, 2018.

[134] Z. S. Harris, "Distributional structure," *Word*, vol. 10, no. 2-3, pp. 146–162, 1954.

[135] K. Erk, "Vector space models of word meaning and phrase meaning: A survey," *Language and Linguistics Compass*, vol. 6, no. 10, pp. 635–653, 2012.

[136] T. Joachims, "A probabilistic analysis of the rocchio algorithm with tfidf for text categorization." School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA, Tech. Rep. CMU-CS-96-118, 1996.

[137] C. Apté, F. Damerau, and S. M. Weiss, "Automated learning of decision rules for text categorization," *ACM Transactions on Information Systems (TOIS)*, vol. 12, no. 3, pp. 233–251, 1994.

[138] J. Rocchio and G. Salton, "Information search optimization and interactive retrieval techniques," in *Proceedings of the 1965 Fall Joint Computer Conference, Part I.* Las Vegas, NV, USA: American Federation of Information Processing Societies (AFIPS), 1965, pp. 293–305.

[139] G. Salton, "A document retrieval system for man-machine interaction," in *Proceedings of the 19th ACM National Conference.* ACM, 1964, pp. 122–301.

[140] ——, "The smart document retrieval project," in *Proceedings of the 14th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Chicago, IL, USA, 1991, pp. 356–358.

[141] G. Salton and C.-S. Yang, "On the specification of term values in automatic indexing," Department of Computer Science, Cornell University, Ithaca, NY, USA, Tech. Rep. 73-173, 1973.

[142] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing & Management*, vol. 24, no. 5, pp. 513–523, 1988.

[143] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, pp. 1–12, 2013.

[144] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the International Conference on Machine Learning (ICML)*, Beijing, China, 2014, pp. 1188–1196.

[145] C. J. Hutto and E. Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," in *Proceedings of the 8th International AAAI Conference on Weblogs and Social Media*. Ann Arbor, MI, USA: AAAI, 2014, pp. 216–225.

[146] W. J. Wilbur and K. Sirotkin, "The automatic identification of stop words," *Journal of Information Science*, vol. 18, pp. 45–55, 1992.

[147] Y. Yang and J. Wilbur, "Using corpus statistics to remove redundant words in text categorization," *Journal of the American Society for Information Science*, vol. 47, no. 5, pp. 357–369, 1996.

[148] J. B. Lovins, "Development of a stemming algorithm," *Mechanical Translatation & Computational Linguistics*, vol. 11, no. 1-2, pp. 22–31, 1968.

[149] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[150] A. G. Jivani, "A comparative study of stemming algorithms," *International Journal of Computer Technology and Applications*, vol. 2, no. 6, pp. 1930–1938, 2011.

[151] E. Gabrilovich and S. Markovitch, "Feature generation for text categorization using world knowledge," in *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, vol. 5, Edinburgh, Scotland, UK, 2005, pp. 1048–1053.

[152] A. Alahmadi, A. Joorabchi, and A. E. Mahdi, "A new text representation scheme combining bag-of-words and bag-of-concepts approaches for automatic text classification," in *Proceedings of the 7th IEEE GCC Conference and Exhibition*. Doha, Qatar: IEEE, 2013, pp. 108–113.

[153] G. A. Miller, *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[154] E. Gabrilovich and S. Markovitch, "Overcoming the brittleness bottleneck using wikipedia: Enhancing text categorization with encyclopedic knowledge," in *Proceedings of the 21st National Conference on Artificial Intelligence*, vol. 6. Boston, MA, USA: Association for the Advancement of Artificial Intelligence (AAAI), 2006, pp. 1301–1306.

[155] K. Spärck Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of Documentation*, vol. 28, no. 1, pp. 11–21, 1972.

[156] J. Yang, Y.-G. Jiang, A. G. Hauptmann, and C.-W. Ngo, "Evaluating bag-of-visual-words representations in scene classification," in *Proceedings of the* 9^{th} *ACM SIGMM International Workshop on Multimedia Information Retrieval*, Augsburg, Germany, 2007, pp. 197–206.

[157] C. Wengert, M. Douze, and H. Jégou, "Bag-of-colors for improved image search," in *Proceedings of the* 19^{th} *ACM International Conference on Multimedia (ACM MM)*. Scottsdale, AZ, USA: ACM, 2011, pp. 1437–1440.

[158] W. W. Cohen and Y. Singer, "Context-sensitive learning methods for text categorization," *ACM Transactions on Information Systems (TOIS)*, vol. 17, no. 2, pp. 141–173, 1999.

[159] K. Erk and S. Padó, "A structured vector space model for word meaning in context," in *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*. Honolulu, HI, USA: Association for Computational Linguistics (ACL), 2008, pp. 897–906.

[160] T. Wilson and S. Raaijmakers, "Comparing word, character, and phoneme n-grams for subjective utterance recognition," in *Proceedings of INTERSPEECH*. Brisbane, Australia: ISCA, 2008, pp. 1614–1617.

[161] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD Eplorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[162] M. Damashek, "Gauging similarity with n-grams: Language-independent categorization of text," *Science*, vol. 267, no. 5199, pp. 843–848, 1995.

[163] F. M. Idris and S. Panchanathan, "Image indexing using vector quantization," in *Storage and Retrieval for Image and Video Databases III*, vol. 2420. San Josë, CA, USA: International Society for Optics and Photonics, 1995, pp. 373–380.

[164] F. Idris and S. Panchanathan, "Storage and retrieval of compressed images," *IEEE Transactions on Consumer Electronics*, vol. 41, no. 3, pp. 937–941, 1995.

[165] M. J. Swain and D. H. Ballard, "Color indexing," *International Journal of Computer Vision*, vol. 7, no. 1, pp. 11–32, 1991.

[166] G. Lu and S. Teng, "A novel image retrieval technique based on vector quantization," in *Proceedings of the International Conference on Computational Intelligence for Modeling, Control and Automation (CIMCA)*, Vienna, Austria, 1999, pp. 36–41.

[167] T. Leung and J. Malik, "Representing and recognizing the visual appearance of materials using three-dimensional textons," *International Journal of Computer Vision*, vol. 43, no. 1, pp. 29–44, 2001.

[168] B. Julesz, "Textons, the elements of texture perception, and their interactions," *Nature*, vol. 290, pp. 91–97, 1981.

[169] L. Zhu, A. B. Rao, and A. Zhang, "Theory of keyblock-based image retrieval," *ACM Transactions on Information Systems (TOIS)*, vol. 20, no. 2, pp. 224–257, 2002.

[170] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *Proceedings of the 9^{th} IEEE International Conference on Computer Vision (ICCV)*. Nice, France: IEEE, 2003, pp. 1–8.

[171] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *Proceedings of the Workshop on Statistical Learning in Computer Vision, Satellite Event of ECCV*. Prague, Czech Republic, 2004, pp. 1–16.

[172] F. Monay, P. Quelhas, D. Gatica-Perez, and J.-M. Odobez, "Constructing visual models with a latent space approach," in *Proceedings of the International Statistical and Optimization Perspectives Workshop: Subspace, Latent Structure and Feature Selection*. Bohinj, Slovenia: Springer, 2005, pp. 115–126.

[173] E. Nowak, F. Jurie, and B. Triggs, "Sampling strategies for bag-of-features image classification," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Graz, Austria: Springer, 2006, pp. 490–503.

[174] M.-E. Nilsback and A. Zisserman, "A visual vocabulary for flower classification," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2. New York, NY, USA: IEEE, 2006, pp. 1447–1454.

[175] A. Bosch, A. Zisserman, and X. Muñoz, "Scene classification via plsa," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Graz, Austria: Springer, 2006, pp. 517–530.

[176] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2. New York, NY, USA: IEEE, 2006, pp. 2169–2178.

[177] X. Peng, L. Wang, X. Wang, and Y. Qiao, "Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice," *Computer Vision and Image Understanding*, vol. 150, pp. 109–125, 2016.

[178] G. Cheng, Z. Li, X. Yao, L. Guo, and Z. Wei, "Remote sensing image scene classification using bag of convolutional features," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 10, pp. 1735–1739, 2017.

[179] S. Hafdhellaoui, Y. Boualleg, and M. Farah, "Collaborative clustering approach based on dempster-shafer theory for bag-of-visual-words codebook generation," in

Proceedings of the 32nd Canadian Conference on Artificial Intelligence. Kingston, ON, Canada: Springer, 2019, pp. 263–273.

[180] J. T. Foote, "Content-based retrieval of music and audio," in *Multimedia Storage and Archiving Systems II*, vol. 3229. International Society for Optics and Photonics, 1997, pp. 138–147.

[181] D. Pye, "Content-based methods for the management of digital music," in *Proceedings of the 25th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 4. Istanbul, Turkey: IEEE, 2000, pp. 2437–2440.

[182] E. Leopold, J. Kindermann, G. Paaß, S. Volmer, R. Cavet, M. Larson, S. Eickeler, and T. Kastner, "Integrated classification of audio, video and speech using partitions of low-level features," in *Proceedings of the Workshop on Multimedia Discovery and Mining*, 2003.

[183] F. Vignoli and S. Pauws, "A music retrieval system based on user driven similarity and its evaluation," in *Proceedings of the 6th International Conference on Music Information Retrieval*. London, U. K.: ISMIR, 2005, pp. 272–279.

[184] T. Kohonen, *Self-Organizing Maps*, 3rd ed. Berlin, Heidelberg: Springer, 2001.

[185] S. Kullback and R. A. Leibler, "On information and sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[186] A. Coates and A. Y. Ng, "The importance of encoding versus training with sparse coding and vector quantization," in *Proceedings of the 28th International Conference on Machine Learning (ICML)*, Bellevue, WA, USA, 2011, 8 pages.

[187] L. Barrington, A. Chan, D. Turnbull, and G. Lanckriet, "Audio information retrieval using semantic similarity," in *Proceedings of the 32nd International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 2. Honolulu, HI, USA: IEEE, 2007, pp. 725–728.

[188] J.-C. Wang, H.-S. Lee, H.-M. Wang, and S.-K. Jeng, "Learning the similarity of audio music in bag-of-frames representation from tagged music data," in *Proceedings of the 12th International Conference on Music Information Retrieval*. Miami, FL, USA: ISMIR, 2011, pp. 85–90.

[189] L. Su, C.-C. M. Yeh, J.-Y. Liu, J.-C. Wang, and Y.-H. Yang, "A systematic evaluation of the bag-of-frames representation for music information retrieval," *IEEE Transactions on Multimedia*, vol. 16, no. 5, pp. 1188–1200, 2014.

[190] C.-C. M. Yeh, L. Su, and Y.-H. Yang, "Dual-layer bag-of-frames model for music genre classification," in *Proceedings of the 38th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Vancouver, BC, Canada: IEEE, 2013, pp. 246–250.

[191] B. McFee, L. Barrington, and G. Lanckriet, "Learning content similarity for music recommendation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 8, pp. 2207–2218, 2012.

[192] S. Sundaram and S. Narayanan, "Audio retrieval by latent perceptual indexing," in *Proceedings of the 33rd International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Las Vegas, NV, USA: IEEE, 2008, pp. 49–52.

[193] G. Chechik, E. Ie, M. Rehn, S. Bengio, and D. Lyon, "Large-scale content-based audio retrieval from text queries," in *Proceedings of the 1st ACM International Conference on Multimedia Information Retrieval (MIR)*, Vancouver, BC, Canada, 2008, pp. 105–112.

[194] K. Seyerlehner, G. Widmer, and P. Knees, "Frame level audio similarity-a codebook approach," in *Proceedings of the 11th International Conference on Digital Audio Effects (DAFx-08)*. Espoo, Finland: DAFX, 2008, pp. 1–8.

[195] G. Marques, M. Lopes, M. Sordo, T. Langlois, and F. Gouyon, "Additional evidence that common low-level features of individual audio frames are not representative of music genres," in *Proceedings of the 7th Sound and Music Computing Conference (SMC)*, Barcelona, Spain, 2010, 6 pages.

[196] Z. Zeng, W. Liang, H. Li, and S. Zhang, "A novel video classification method based on hybrid generative/discriminative models," in *Proceedings of the Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*. Orlando, FL, USA: Springer, 2008, pp. 705–713.

[197] Y. Liu, W.-L. Zhao, C.-W. Ngo, C.-S. Xu, and H.-Q. Lu, "Coherent bag-of audio words model for efficient large-scale video copy detection," in *Proceedings of the International Conference on Image and Video Retrieval (CIVR)*. Xi'an, China: ACM, 2010, pp. 89–96.

[198] Z. Fu, G. Lu, K. M. Ting, and D. Zhang, "Music classification via the bag-of-features approach," *Pattern Recognition Letters*, vol. 32, no. 14, pp. 1768–1777, 2011.

[199] S. Pancoast and M. Akbacak, "Softening quantization in bag-of-audio-words," in *Proceedings of the 39th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Florence, Italy: IEEE, 2014, pp. 1370–1374.

[200] ——, "N-gram extension for bag-of-audio-words," in *Proceedings of the 38th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Vancouver, BC, Canada: IEEE, 2013, pp. 778–782.

[201] S. Rawat, P. F. Schulam, S. Burger, D. Ding, Y. Wang, and F. Metze, "Robust audio-codebooks for large-scale event detection in consumer videos," in *Proceedings of INTERSPEECH*. Lyon, France: ISCA, 2013, pp. 2929–2933.

[202] A. Plinge, R. Grzeszick, and G. A. Fink, "A bag-of-features approach to acoustic event detection," in *Proceedings of the 39th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Florence, Italy: IEEE, 2014, pp. 3704–3708.

[203] R. Grzeszick, A. Plinge, and G. A. Fink, "Temporal acoustic words for online acoustic event detection," in *Proceedings of the 37th German Conference on Pattern Recognition (GCPR)*. Aachen, Germany: Springer, 2015, pp. 142–153.

[204] ——, "Bag-of-features methods for acoustic event detection and classification," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 6, pp. 1242–1252, 2017.

[205] H. Lim, M. J. Kim, and H. Kim, "Robust sound event classification using LBP-HOG based bag-of-audio-words feature representation," in *Proceedings of INTERSPEECH*. Dresden, Germany: ISCA, 2015, pp. 3325–3329.

[206] L. Nanni, Y. M. Costa, A. Lumini, M. Y. Kim, and S. R. Baek, "Combining visual and acoustic features for music genre classification," *Expert Systems with Applications*, vol. 45, pp. 108–117, 2016.

[207] O. Mangin, P.-Y. Oudeyer, and D. Filliat, "A bag-of-features framework for incremental learning of speech invariants in unsegmented audio streams," in *Proceedings of the 10th International Conference on Epigenetic Robotics (EpiRob)*. Örenäs Slott, Sweden: Lund University Cognitive Studies, 2010, 8 pages.

[208] F. Pokorny, F. Graf, F. Pernkopf, and B. Schuller, "Detection of negative emotions in speech signals using bags-of-audio-words," in *Proceedings of the 1st International Workshop on Automatic Sentiment Analysis in the Wild (WASA), held in conjunction with ACII*, AAAC. Xi'an, China: IEEE, 2015, pp. 879–884.

[209] Y.-G. Jiang, X. Zeng, G. Ye, D. Ellis, S.-F. Chang, S. Bhattacharya, and M. Shah, "Columbia-ucf trecvid2010 multimedia event detection: Combining multiple modalities, contextual concepts, and temporal matching." in *TRECVID*, vol. 2, 2010, pp. 3–2.

[210] J. Joshi, R. Goecke, S. Alghowinem, A. Dhall, M. Wagner, J. Epps, G. Parker, and M. Breakspear, "Multimodal assistive technologies for depression diagnosis and monitoring," *Journal on MultiModal User Interfaces*, vol. 7, no. 3, pp. 217–228, 2013.

[211] S. Bhatia, M. Hayat, and R. Goecke, "A multimodal system to characterise melancholia: cascaded bag of words approach," in *Proceedings of the 19th ACM International Conference on Multimodal Interaction (ICMI)*. Glasgow, Scotland: ACM, 2017, pp. 274–280.

[212] M. G. Baydogan, G. Runger, and E. Tuv, "A bag-of-features framework to classify time series," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2796–2802, 2013.

[213] N. Passalis and A. Tefas, "Spectral clustering using optimized bag-of-features," in *Proceedings of the 9th Hellenic Conference on Artificial Intelligence*. Thessaloniki, Greece: ACM, 2016, pp. 1–9.

[214] H. Steinhaus, "Sur la division des corps matériels en parties," *Bulletin de L'Académie Polonaise des Sciences*, vol. III-4, no. 12, pp. 801–804, 1956.

[215] S. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[216] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proceedings of the 18th annual ACM-SIAM symposium on Discrete Algorithms*. New Orleans, LA, USA: SIAM, 2007, pp. 1027–1035.

[217] P. Foggia, N. Petkov, A. Saggese, N. Strisciuglio, and M. Vento, "Reliable detection of audio events in highly noisy environments," *Pattern Recognition Letters*, vol. 65, pp. 22–28, 2015.

[218] F. Perronnin and C. Dance, "Fisher kernels on visual vocabularies for image categorization," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*. Minneapolis, MN, USA: IEEE, 2007, pp. 1–8.

[219] H. Kaya, A. A. Karpov, and A. A. Salah, "Fisher vectors with cascaded normalization for paralinguistic analysis," in *Proceedings of INTERSPEECH*. Dresden, Germany: ISCA, 2015, pp. 909–913.

[220] K. West and S. Cox, "Features and classifiers for the automatic classification of musical audio signals," in *Proceedings of the 5th International Conference on Music Information Retrieval*. Barcelona, Spain: ISMIR, 2004, 6 pages.

[221] B. Schuller and G. Rigoll, "Recognising interest in conversational speech-comparing bag of frames and supra-segmental features," in *Proceedings of INTERSPEECH*. Brighton, U. K.: ISCA, 2009, pp. 1999–2002.

[222] J.-J. Aucouturier, B. Defreville, and F. Pachet, "The bag-of-frames approach to audio pattern recognition: A sufficient model for urban soundscapes but not for polyphonic music," *The Journal of the Acoustical Society of America*, vol. 122, no. 2, pp. 881–891, 2007.

[223] M. Lagrange, G. Lafay, B. Defreville, and J.-J. Aucouturier, "The bag-of-frames approach: a not so sufficient model for urban soundscapes," *The Journal of the Acoustical Society of America*, vol. 138, no. 5, pp. 487–492, 2015.

[224] C. M. Bishop, *Pattern Recognition and Machine Learning*, ser. Information Science and Statistics. Springer, 2006.

[225] J. R. Searle, "Minds, brains, and programs," *Behavioral and Brain Sciences*, vol. 3, no. 3, pp. 417–424, 1980.

[226] T. Jebara, *Machine Learning: Discriminative and Generative*. New York, NY, USA: Springer Science & Business Media, 2012, vol. 755.

[227] A. Y. Ng and M. I. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes," in *Advances in Neural Information Processing Systems*. Vancouver, BC, Canada: NeurIPS, 2002, pp. 841–848.

[228] G. Trigeorgis, F. Ringeval, R. Brueckner, E. Marchi, M. A. Nicolaou, B. Schuller, and S. Zafeiriou, "Adieu features? end-to-end speech emotion recognition using a deep convolutional recurrent network," in *Proceedings of the 41st International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Shanghai, China: IEEE, 2016, pp. 5200–5204.

[229] M. Schmitt and B. Schuller, "End-to-end audio classification with small datasets – making it work," in *Proceedings of the 27th European Signal Processing Conference (EUSIPCO)*, EURASIP. A Coruña, Spain: IEEE, 2019, 5 pages.

[230] S. Ali and K. A. Smith, "On learning algorithm selection for classification," *Applied Soft Computing*, vol. 6, no. 2, pp. 119–138, 2006.

[231] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[232] A. Sze-To and A. K. Wong, "A weight-selection strategy on training deep neural networks for imbalanced classification," in *Proceedings of the International Conference Image Analysis and Recognition (ICIAR)*. Montreal, QC, Canada: Springer, 2017, pp. 3–10.

[233] A. B. Graf, A. J. Smola, and S. Borer, "Classification in a normalized feature space using support vector machines," *IEEE Transactions on Neural Networks*, vol. 14, no. 3, pp. 597–605, 2003.

[234] J. T. Geiger, B. Schuller, and G. Rigoll, "Large-scale audio feature extraction and svm for acoustic scene classification," in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. New Paltz, NY, USA: IEEE, 2013, pp. 1–4.

[235] Y. Yang and J. O. Pedersen, "A comparative study on feature selection in text categorization," in *Proceedings of the 14th International Conference on Machine Learning (ICML)*. Nashville, TN, USA: International Machine Learning Society, 1997, pp. 412–420.

[236] T. Vogt and E. André, "Comparing feature sets for acted and spontaneous speech in view of automatic emotion recognition," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*. Amsterdam, The Netherlands: IEEE, 2005, pp. 474–477.

[237] V. Vapnik and A. Chervonenkis, "Theory of pattern recognition," 1974.

[238] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[239] M. P. Deisenroth, A. A. Faisal, and C. S. Ong, *Mathematics for Machine Learning*. Cambridge University Press, 2020.

[240] C.-W. Hsu, C.-C. Chang, C.-J. Lin *et al.*, "A practical guide to support vector classification," 2003.

[241] J. Nocedal and S. Wright, *Numerical optimization*, 2nd ed. Springer Science & Business Media, 2006.

[242] B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett, "New support vector algorithms," *Neural Computation*, vol. 12, no. 5, pp. 1207–1245, 2000.

[243] J. Milgram, M. Cheriet, and R. Sabourin, ""One against one" or "one against all": Which one is better for handwriting recognition with svms?" in *Proceedings of the 10th International Workshop on Frontiers in Handwriting Recognition (IWFHR)*, La Baule, France, 2006, pp. 1–6.

[244] S. S. Keerthi and C.-J. Lin, "Asymptotic behaviors of support vector machines with gaussian kernel," *Neural Computation*, vol. 15, no. 7, pp. 1667–1689, 2003.

[245] A. Barla, F. Odone, and A. Verri, "Histogram intersection kernel for image classification," in *Proceedings of the International Conference on Image Processing (ICIP)*, vol. 3. Barcelona, Spain: IEEE, 2003, pp. 513–516.

[246] S. Boughorbel, J.-P. Tarel, and N. Boujemaa, "Generalized histogram intersection kernel for image recognition," in *Proceedings of the International Conference on Image Processing (ICIP)*, vol. 3. Genova, Italy: IEEE, 2005, pp. III–161.

[247] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[248] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, p. 386, 1958.

[249] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[250] T. J. Sejnowski, *The Deep Learning Revolution*. Cambridge, MA, USA: MIT Press, 2018.

[251] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[252] M. Abadi *et al.*, "Tensorflow: A system for large-scale machine learning," in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2016, pp. 265–283.

[253] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, Vancouver, BC, Canada, 2019, pp. 8026–8037.

[254] A. M. Zador, "A critique of pure learning and what artificial neural networks can learn from animal brains," *Nature Communications*, vol. 10, no. 1, pp. 1–7, 2019.

[255] A. Tjandra, S. Sakti, and S. Nakamura, "Sequence-to-sequence asr optimization via reinforcement learning," in *Proceedings of the 43rd International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Calgary, AB, Canada: IEEE, 2018, pp. 5829–5833.

[256] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 2, no. 4, pp. 303–314, 1989.

[257] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*. Fort Lauderdale, FL, USA: Knowledge 4 All Foundation, 2011, pp. 315–323.

[258] A. Ghosh, H. Kumar, and P. S. Sastry, "Robust loss functions under label noise for deep neural networks," in *Proceedings of the 31st AAAI Conference on Artificial Intelligence*. San Francisco, CA, USA: Association for the Advancement of Artificial Intelligence (AAAI), 2017, pp. 1919–1925.

[259] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 9–48.

[260] L. Bottou, "Stochastic gradient descent tricks," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 421–436.

[261] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, 2015, pp. 1–13.

[262] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, pp. 1–18, 2012.

[263] K. Hara, D. Saitoh, and H. Shouno, "Analysis of dropout learning regarded as ensemble learning," in *Proceedings of the 25th International Conference on Artificial Neural Networks (ICANN)*, ENNS. Barcelona, Spain: Springer, 2016, pp. 72–79.

[264] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization?" in *Advances in Neural Information Processing Systems*, Montreal, QC, Canada, 2018, pp. 2483–2493.

[265] H. Noh, T. You, J. Mun, and B. Han, "Regularizing deep neural networks by noise: Its interpretation and optimization," in *Advances in Neural Information Processing Systems*, Long Beach, CA, USA, 2017, pp. 5109–5118.

[266] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition," in *Proceedings of INTERSPEECH*. Dresden, Germany: ISCA, 2015, pp. 3586–3589.

[267] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, 2017.

[268] J. Schlüter and T. Grill, "Exploring data augmentation for improved singing voice detection with neural networks," in *Proceedings of the 16th International Conference on Music Information Retrieval*. Malaga, Spain: ISMIR, 2015, pp. 121–126.

[269] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*. Providence, RI, USA: IEEE, 2012, pp. 3642–3649.

[270] A. M. Badshah, J. Ahmad, N. Rahim, and S. W. Baik, "Speech emotion recognition from spectrograms with deep convolutional neural network," in *Proceedings of the International Conference on Platform Technology and Service (PlatCon)*. IEEE, 2017, pp. 1–5.

[271] M. Ravanelli and Y. Bengio, "Speaker recognition from raw waveform with sincnet," in *Proceedings of the Spoken Language Technology Workshop (SLT)*. IEEE, 2018, pp. 1021–1028.

[272] S. Amiriparian, M. Gerczuk, S. Ottl, N. Cummins, M. Freitag, S. Pugachevskiy, A. Baird, and B. W. Schuller, "Snore sound classification using image-based deep spectrum features," in *Proceedings of INTERSPEECH*. Stockholm, Sweden: ISCA, 2017, pp. 3512–3516.

[273] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *European Journal of Operational Research*, vol. 270, no. 2, pp. 654–669, 2018.

[274] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems*, Montreal, QC, Canada, 2014, pp. 3104–3112.

[275] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*. Vancouver, BC, Canada: IEEE, 2013, pp. 6645–6649.

[276] K. Gurney, *An Introduction to Neural Networks*. CRC Press, 2004.

[277] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[278] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[279] F. Eyben, S. Böck, B. Schuller, and A. Graves, "Universal onset detection with bidirectional long-short term memory neural networks," in *Proceedings of the 11th International Conference on Music Information Retrieval*. Utrecht, The Netherlands: ISMIR, 2010, pp. 589–594.

[280] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, pp. 1–15, 2014.

[281] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, Montreal, QC, Canada, 2014, pp. 2672–2680.

[282] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, pp. 1–14, 2014.

[283] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA: IEEE, 2016, pp. 770–778.

[284] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Advances in Neural Information Processing Systems*, Long Beach, CA, USA, 2017, pp. 3856–3866.

[285] Y. Aytar, C. Vondrick, and A. Torralba, "Soundnet: Learning sound representations from unlabeled video," in *Advances in Neural Information Processing Systems*, Barcelona, Spain, 2016, pp. 892–900.

[286] E. Fonseca, J. Pons Puig, X. Favory, F. Font Corbera, D. Bogdanov, A. Ferraro, S. Oramas, A. Porter, and X. Serra, "Freesound datasets: a platform for the creation of open audio datasets," in *Proceedings of the 18th International Conference on Music Information Retrieval*. Suzhou, China: ISMIR, 2017, pp. 486–493.

[287] D. Krstajic, L. J. Buturovic, D. E. Leahy, and S. Thomas, "Cross-validation pitfalls when selecting and assessing regression and classification models," *Journal of Cheminformatics*, vol. 6, no. 1, pp. 1–15, 2014.

[288] I. Lawrence and K. Lin, "A concordance correlation coefficient to evaluate reproducibility," *Biometrics*, pp. 255–268, 1989.

[289] C. A. E. Nickerson, "A note on "a concordance correlation coefficient to evaluate reproducibility"," *Biometrics*, pp. 1503–1507, 1997.

[290] J. H. Zar, "Significance testing of the spearman rank correlation coefficient," *Journal of the American Statistical Association*, vol. 67, no. 339, pp. 578–580, 1972.

[291] R. A. Fisher, *Statistical Methods for Research Workers*, 5th ed. Oliver and Boyd, Edinburgh and London, 1934.

[292] R. S. Nickerson, "Null hypothesis significance testing: a review of an old and continuing controversy," *Psychological Methods*, vol. 5, no. 2, p. 241, 2000.

[293] M. Krzywinski and N. Altman, "Significance, p values and t-tests," *Nature Methods*, vol. 10, no. 11, pp. 1041–1042, 2013.

[294] J. Bortz, *Statistik: für Human-und Sozialwissenschaftler*, 6th ed. Heidelberg: Springer, 2006.

[295] B. L. Welch, "The generalization of student's problem when several different population variances are involved," *Biometrika*, vol. 34, no. 1/2, pp. 28–35, 1947.

[296] B. Rüger, *Test-und Schätztheorie: Band II: Statistische Tests*. München: Walter de Gruyter, 2010.

[297] R. D'Agostino and E. S. Pearson, "Tests for departure from normality. empirical results for the distributions of b_2 and $\sqrt{b}_1$," *Biometrika*, vol. 60, no. 3, pp. 613–622, 1973.

[298] C. Ullenboom, *Java ist auch eine Insel*. Galileo Press, 2020, vol. 15.

[299] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[300] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *The Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

[301] J. Link, *Softwaretests mit JUnit: Techniken der testgetriebenen Entwicklung*, 2nd ed. Heidelberg, Germany: dpunkt.verlag, 2005.

[302] J. Kossaifi, R. Walecki, Y. Panagakis, J. Shen, M. Schmitt, F. Ringeval, J. Han, V. Pandit, A. Toisoul, B. W. Schuller, K. Star, E. Hajiyev, and M. Pantic, "Sewa db: A rich database for audio-visual emotion and sentiment research in the wild," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 3, pp. 1022–1040, 2021.

[303] A. Asthana, S. Zafeiriou, S. Cheng, and M. Pantic, "Incremental face alignment in the wild," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*. Columbus, OH, USA: IEEE, 2014, pp. 1859–1866.

[304] B. Schuller, A. E.-D. Mousa, and V. Vasileios, "Sentiment analysis and opinion mining: On optimal parameters and performances," *WIREs Data Mining and Knowledge Discovery*, vol. 5, pp. 255–263, September/October 2015.

[305] R. K. Bakshi, N. Kaur, R. Kaur, and G. Kaur, "Opinion mining and sentiment analysis," in *Proceedings of the 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*. New Delhi, India: IEEE, 2016, pp. 452–455.

[306] I. Naji, "Twitter sentiment analysis training corpus," Online resource: http://thinknook.com/Twitter-sentiment-analysis-training-corpus-dataset-2012-09-22/, 2012, visited on November 13, 2020.

[307] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, "Speaker verification using adapted gaussian mixture models," *Digital Signal Processing*, vol. 10, no. 1-3, pp. 19–41, 2000.

[308] V. Pandit, M. Schmitt, N. Cummins, F. Graf, L. Paletta, and B. Schuller, "How good is your model 'really'? on 'wildness' of the in-the-wild speech-based affect recognisers," in *Proceedings of the International Conference on Speech and Computer (SPECOM)*. Leipzig, Germany: Springer, 2018, pp. 490–500.

[309] J. Han, Z. Zhang, M. Schmitt, Z. Ren, F. Ringeval, and B. Schuller, "Bags in bag: Generating context-aware bags for tracking emotions from speech," in *Proceedings of INTERSPEECH*. Hyderabad, India: ISCA, 2018, pp. 3082–3086.

[310] J. Han, "Automated deep audiovisual emotional behaviour analysis in the wild," Ph.D. dissertation, University of Augsburg, Germany, 2020.

[311] S. Amiriparian, M. Gerczuk, S. Ottl, N. Cummins, S. Pugachevskiy, and B. Schuller, "Bag-of-deep-features: Noise-robust deep feature representations for audio analysis," in *Proceeedings of the International Joint Conference on Neural Networks (IJCNN)*. Rio de Janeiro, Brazil: IEEE, 2018, pp. 1–7.

[312] S. Amiriparian, "Deep representation learning techniques for audio signal processing," Ph.D. dissertation, Technische Universität München, Germany, 2019.

[313] G. Gosztolya, T. Grósz, and L. Tóth, "General utterance-level feature extraction for classifying crying sounds, atypical & self-assessed affect and heart beats," in *Proceedings of INTERSPEECH*. Hyderabad, India: ISCA, 2018, pp. 531–535.

[314] G. Gosztolya, "Using the bag-of-audio-word feature representation of asr dnn posteriors for paralinguistic classification," in *Proceedings of INTERSPEECH*. Graz, Austria: ISCA, 2019, pp. 3940–3944.

[315] K. Qian, M. Schmitt, C. Janott, Z. Zhang, C. Heiser, W. Hohenhorst, M. Herzog, W. Hemmert, and B. Schuller, "A bag of wavelet features for snore sound classification," *Annals of Biomedical Engineering*, vol. 47, no. 4, pp. 1000–1011, 2019.

[316] K. Qian, H. Kuromiya, Z. Ren, M. Schmitt, Z. Zhang, T. Nakamura, K. Yoshiuchi, B. W. Schuller, and Y. Yamamoto, "Automatic detection of major depressive disorder via a bag-of-behaviour-words approach," in *Proceedings of the 3rd International Symposium on Image Computing and Digital Medicine (ISICDM)*. Xi'an, China: ACM, 2019, pp. 71–75.

[317] R. Elbarougy, B. T. Atmaja, and M. Akagi, "Continuous audiovisual emotion recognition using feature selection and lstm," *Journal of Signal Processing*, vol. 24, no. 6, pp. 229–235, 2020.

[318] M. S. S. Syed, Z. S. Syed, M. Lech, and E. Pirogova, "Automated screening for alzheimer's dementia through spontaneous speech," in *Proceedings of INTERSPEECH*. Shanghai, China: ISCA, 2020, pp. 2222–2226.

[319] S. Amiriparian, M. Schmitt, N. Cummins, K. Qian, F. Dong, and B. Schuller, "Deep unsupervised representation learning for abnormal heart sound classification," in *Proceedings of the 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. Honolulu, HI, USA: IEEE, 2018, pp. 4776–4779.

[320] S. Amiriparian, S. Pugachevskiy, N. Cummins, S. Hantke, J. Pohjalainen, G. Keren, and B. Schuller, "Cast a database: Rapid targeted large-scale big data acquisition via small-world modelling of social media platforms," in *Proceedings of the 7th International Conference on Affective Computing and Intelligent Interaction (ACII)*, AAAC. San Antonio, TX, USA: IEEE, 2017, pp. 340–345.

[321] Z. Zhang, A. Cristia, A. S. Warlaumont, and B. Schuller, "Automated classification of children's linguistic versus non-linguistic vocalisations," in *Proceedings of INTERSPEECH*. Hyderabad, India: ISCA, 2018, pp. 2588–2592.

[322] Z. Ren, N. Cummins, J. Han, S. Schnieder, J. Krajewski, and B. Schuller, "Evaluation of the pain level from speech: Introducing a novel pain database and benchmarks," in *Proceedings of the 13th ITG Symposium on Speech Communication (ITG SC)*, VDE. Oldenburg, Germany: IEEE, 2018, pp. 1–5.

[323] N. Cummins, S. Amiriparian, G. Hagerer, A. Batliner, S. Steidl, and B. W. Schuller, "An image-based deep spectrum feature representation for the recognition of emotional speech," in *Proceedings of the 25th ACM International Conference on Multimedia (ACM MM)*. Mountain View, CA, USA: ACM, 2017, pp. 478–484.

[324] S. Hantke, N. Cummins, and B. Schuller, "What is my dog trying to tell me? the automatic recognition of the context and perceived emotion of dog barks," in *Proceedings of the 43rd International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Calgary, AB, Canada: IEEE, 2018, pp. 5134–5138.

[325] P. Tzirakis, A. Shiarella, R. Ewers, and B. W. Schuller, "Computer audition for continuous rainforest occupancy monitoring: The case of bornean gibbons' call detection," in *Proceedings of INTERSPEECH*. Shanghai, China: ISCA, 2020, pp. 1211–1215.

[326] K. Qian, M. Schmitt, H. Zheng, T. Koike, J. Han, J. Liu, W. Ji, J. Wei, M. Song, Z. Yang, Z. Ren, S. Liu, Z. Zhang, Y. Yamamoto, and B. Schuller, "Computer audition for fighting the sars-cov-2 corona crisis – introducing the multi-task speech corpus for covid-19," *IEEE Internet of Things Journal*, pp. 1–12, 2021.

[327] P. Buitelaar, I. D. Wood, S. Negi, M. Arcan, J. P. McCrae, A. Abele, C. Robin, V. Andryushechkin, H. Ziad, H. Sagha, M. Schmitt, B. W. Schuller, J. F. Sánchez-Rada, C. A. Iglesias, C. Navarro, A. Giefer, N. Heise, V. Masucci, F. A. Danza, C. Caterino, P. Smrž, M. Hradiš, F. Povolný, M. Klimeš, P. Matějka, and G. Tummarello, "Mixedemotions: An open-source toolbox for multimodal emotion analysis," *IEEE Transactions on Multimedia*, vol. 20, no. 9, pp. 2454–2465, 2018.

[328] S. Hantke, C. Cohrs, M. Schmitt, B. Tannert, F. Lütkebohmert, M. Detmers, H. Schelhowe, and B. Schuller, "Introducing an emotion-driven assistance system for cognitively impaired individuals," in *Proceedings of the International Conference on Computers Helping People with Special Needs (ICCHP)*. Linz, Austria: Springer, 2018, pp. 486–494.

[329] Y. Guo, J. Han, Z. Zhang, B. Schuller, and Y. Ma, "Exploring a new method for food likability rating based on dt-cwt theory," in *Proceedings of the 20th ACM International Conference on Multimodal Interaction (ICMI)*. Boulder, CO, USA: ACM, 2018, pp. 569–573.

[330] G. Gosztolya, "Using Fisher vector and bag-of-audio-words representations to identify styrian dialects, sleepiness, baby & orca sounds," in *Proceedings of INTERSPEECH*. Graz, Austria: ISCA, 2019, pp. 2413–2417.

[331] B. Vlasenko, J. Sebastian, P. K. DS, and M. Magimai-Doss, "Implementing fusion techniques for the classification of paralinguistic information," in *Proceedings of INTERSPEECH*. Hyderabad, India: ISCA, 2018, pp. 526–530.

[332] M. Vetráb and G. Gosztolya, "Investigating the corpus independence of the bag-of-audio-words approach," in *Proceedings of the International Conference on Text, Speech, and Dialogue*. Brno, Czech Republic: Springer, 2020, pp. 285–293.

[333] H. Kaya and A. A. Karpov, "Introducing weighted kernel classifiers for handling imbalanced paralinguistic corpora: Snoring, addressee and cold," in *Proceedings of INTERSPEECH*. Stockholm, Sweden: ISCA, 2017, pp. 3527–3531.

[334] J. Huang, Y. Li, J. Tao, Z. Lian, Z. Wen, M. Yang, and J. Yi, "Continuous multimodal emotion prediction based on long short term memory recurrent neural network," in *Proceedings of the 7th Annual Workshop on Audio/Visual Emotion Challenge (AVEC 17)*. Mountain View, CA, USA: ACM, 2017, pp. 11–18.

[335] D. Cai, Z. Ni, W. Liu, W. Cai, G. Li, M. Li, D. Cai, Z. Ni, W. Liu, and W. Cai, "End-to-end deep learning framework for speech paralinguistics detection based on perception aware spectrum," in *Proceedings of INTERSPEECH*. Stockholm, Sweden: ISCA, 2017, pp. 3452–3456.

[336] F. Ringeval, A. Sonderegger, J. Sauer, and D. Lalanne, "Introducing the RECOLA multimodal corpus of remote collaborative and affective interactions," in *Proceedings of the International Workshop on Emotion Representation, Analysis and Synthesis in Continuous Time and Space (EmoSPACE)*, Shanghai, China, 2013, 8 pages.

[337] F. Ringeval, B. Schuller, M. Valstar, S. Jaiswal, E. Marchi, D. Lalanne, R. Cowie, and M. Pantic, "AV+EC 2015 – the first affect recognition challenge bridging across audio, video, and physiological data," in *Proceedings of the 5th International Workshop on Audio/Visual Emotion Challenge (AVEC 15)*. Brisbane, Australia: ACM, 2015, pp. 3–8.

[338] M. Valstar, J. Gratch, B. Schuller, F. Ringeval, D. Lalanne, M. T. Torres, S. Scherer, G. Stratou, R. Cowie, and M. Pantic, "AVEC 2016 – depression, mood, and emotion recognition workshop and challenge," in *Proceedings of the 6th International Workshop on Audio/Visual Emotion Challenge (AVEC 16)*. Amsterdam, The Netherlands: ACM, 2016, pp. 3–10.

[339] A. Batliner and R. Huber, *Speaker Classification I*. Berlin, Heidelberg: Springer, 2007, ch. Speaker Characteristics and Emotion Classification, pp. 138–151.

[340] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002.

[341] T. L. Li and A. B. Chan, "Genre classification and the invariance of mfcc features to key and tempo," in *Proceedings of the International Conference on MultiMedia Modeling*. Taipei, Taiwan: Springer, 2011, pp. 317–327.

[342] F. Ringeval, F. Eyben, E. Kroupi, A. Yuce, J.-P. Thiran, T. Ebrahimi, D. Lalanne, and B. Schuller, "Prediction of asynchronous dimensional emotion ratings from audiovisual and physiological data," *Pattern Recognition Letters*, vol. 66, pp. 22–30, 2015.

[343] R. A. Fisher, "On the "probable error" of a coefficient of correlation deduced from a small sample," *Metron*, vol. 1, pp. 3–32, 1921.

[344] L. He, D. Jiang, L. Yang, E. Pei, P. Wu, and H. Sahli, "Multimodal affective dimension prediction using deep bidirectional long short-term memory recurrent neural networks," in *Proceedings of the 5th International Workshop on Audio/Visual Emotion Challenge (AVEC 15)*. Brisbane, Australia: ACM, 2015, pp. 73–80.

[345] M. Schmitt, C. Janott, V. Pandit, K. Qian, C. Heiser, W. Hemmert, and B. Schuller, "A bag-of-audio-words approach for snore sounds' excitation localisation," in *Proceedings of the 11th ITG Symposium on Speech Communication (ITG SC)*, VDE. Paderborn, Germany: IEEE, 2016, pp. 230–234.

[346] T. Young, M. Palta, J. Dempsey, J. Skatrud, S. Weber, and S. Badr, "The occurrence of sleep-disordered breathing among middle-aged adults," *New England Journal of Medicine*, vol. 328, no. 17, pp. 1230–1235, 1993.

[347] P. J. Strollo Jr and R. M. Rogers, "Obstructive sleep apnea," *New England Journal of Medicine*, vol. 334, no. 2, pp. 99–104, 1996.

[348] B. Mokhlesi, S. A. Ham, and D. Gozal, "The effect of sex and age on the comorbidity burden of osa: an observational analysis from a large nationwide us health claims database," *European Respiratory Journal*, vol. 47, no. 4, pp. 1162–1169, 2016.

[349] M. B. Blumen, M. A. Q. Salva, I. Vaugier, K. Leroux, M.-P. d'Ortho, F. Barbot, F. Chabolle, and F. Lofaso, "Is snoring intensity responsible for the sleep partner's poor quality of sleep?" *Sleep and Breathing*, vol. 16, no. 3, pp. 903–907, 2012.

[350] M. R. El Badawey, G. McKee, H. Marshall, N. Heggie, and J. A. Wilson, "Predictive value of sleep nasendoscopy in the management of habitual snorers," *Annals of Otology, Rhinology & Laryngology*, vol. 112, no. 1, pp. 40–44, 2003.

[351] K. Qian, Z. Xu, H. Xu, Y. Wu, and Z. Zhao, "Automatic detection, segmentation and classification of snore related signals from overnight audio recording," *IET Signal Processing*, vol. 9, no. 1, pp. 21–29, 2015.

[352] R. Khushaba, "Multiscale wavelet transform features," https://github.com/RamiKhushaba/getmswtfeat/blob/master/getmswtfeat.m, 2012, link retrieved on 26 November 2020, original publication in a different repository.

[353] K. Qian, C. Janott, Z. Zhang, C. Heiser, and B. Schuller, "Wavelet features for classification of vote snore sounds," in *Proceedings of the 41st International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Shanghai, China: IEEE, 2016, pp. 221–225.

[354] M. Schmitt and B. Schuller, "Recognising guitar effects – which acoustic features really matter?" in *Proceedings of INFORMATIK 2017, Lecture Notes in Informatics (LNI)*, M. G. Maximilian Eibl, Ed. Chemnitz, Germany: Gesellschaft für Informatik, 2017, pp. 177–190.

[355] U. Zölzer, *DAFX: Digital Audio Effects*. John Wiley & Sons, 2011.

[356] C. Kehling, J. Abeßer, C. Dittmar, and G. Schuller, "Automatic tablature transcription of electric guitar recordings by estimation of score-and instrument-related parameters." in *Proceedings of the International Conference on Digital Audio Effects*. Erlangen, Germany: DAFx, 2014, pp. 219–226.

[357] C. Xu, N. C. Maddage, and X. Shao, "Automatic music classification and summarization," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 3, pp. 441–450, 2005.

[358] M. Stein, J. Abeßer, C. Dittmar, and G. Schuller, "Automatic detection of audio effects in guitar and bass recordings," in *Proceedings of the 128th AES Convention*. London, U. K.: Audio Engineering Society, 2010, 12 pages.

[359] A. Shahid, K. Wilkinson, S. Marcu, and C. M. Shapiro, "Karolinska sleepiness scale (kss)," in *STOP, THAT and One Hundred Other Sleep Scales*. Springer, 2012, pp. 209–210.

[360] S. Amiriparian, M. Freitag, N. Cummins, and B. Schuller, "Sequence to sequence autoencoders for unsupervised representation learning from audio," in *Proceedings of the Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE 2017)*, Munich, Germany, 2017, 5 pages.

[361] M. Freitag, S. Amiriparian, S. Pugachevskiy, N. Cummins, and B. Schuller, "auDeep: Unsupervised learning of representations from audio with deep recurrent neural networks," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 1–5, 2017.

[362] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, pp. 1–16, 2018.

[363] P. Tzirakis, S. Zafeiriou, and B. W. Schuller, "End2you–the imperial toolkit for multimodal profiling by end-to-end learning," *arXiv preprint arXiv:1802.01115*, pp. 1–5, 2018.

[364] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-learn: Machine learning in python," *The Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[365] E. Çiftçi, H. Kaya, H. Güleç, and A. A. Salah, "The turkish audio-visual bipolar disorder corpus," in *Proceedings of the 1st Asian Conference on Affective Computing and Intelligent Interaction (ACII Asia)*. Beijing, China: IEEE, 2018, pp. 1–6.

[366] E.-M. Rathner, Y. Terhorst, N. Cummins, B. W. Schuller, and H. Baumeister, "State of mind: Classification through self-reported affect and word use in speech," in *Proceedings of INTERSPEECH*. Hyderabad, India: ISCA, 2018, pp. 267–271.

[367] D. DeVault, R. Artstein, G. Benn, T. Dey, E. Fast, A. Gainer, K. Georgila, J. Gratch, A. Hartholt, M. Lhommet, G. Lucas, S. Marsella, F. Morbini, A. Nazarian, S. Scherer, G. Stratou, A. Suri, D. Traum, R. Wood, Y. Xu, and A. Rizzo, "Simsensei kiosk: A virtual human interviewer for healthcare decision support," in *Proceedings of the* 13th *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. Paris, France: ACM, 2014, pp. 1061–1068.

[368] M. Schmitt and B. Schuller, "Deep recurrent neural networks for emotion recognition in speech," in *Proceedings of the 44th Jahrestagung für Akustik (DAGA)*. Munich, Germany: Deutsche Gesellschaft für Akustik, 2018, pp. 1537–1540.

[369] S. Chen, Q. Jin, J. Zhao, and S. Wang, "Multimodal multi-task learning for dimensional and continuous emotion recognition," in *Proceedings of the 7th Annual*

Workshop on Audio/Visual Emotion Challenge (AVEC 17). Mountain View, CA, USA: ACM, 2017, pp. 19–26.

[370] F. Eyben, M. Wöllmer, and B. Schuller, "A multitask approach to continuous five-dimensional affect sensing in natural speech," *Transactions on Interactive Intelligent Systems*, vol. 2, no. 1, pp. 1–29, 2012.

[371] J. Huang, Y. Li, J. Tao, Z. Lian, M. Niu, and M. Yang, "Multimodal continuous emotion recognition with data augmentation using recurrent neural networks," in *Proceedings of the 2018 Audio/Visual Emotion Challenge and Workshop (AVEC 18)*. Seoul, Korea: ACM, 2018, pp. 57–64.

[372] M. Schmitt, N. Cummins, and B. Schuller, "Continuous emotion recognition in speech – do we need recurrence?" in *Proceedings of INTERSPEECH*. Graz, Austria: ISCA, 2019, pp. 2808–2812.

[373] S. Hantke, F. Weninger, R. Kurle, F. Ringeval, A. Batliner, A. E.-D. Mousa, and B. Schuller, "I hear you eat and speak: Automatic recognition of eating condition and food type, use-cases, and impact on asr performance," *PloS One*, vol. 11, no. 5, pp. 1–24, 2016.

[374] B. Sertolli, N. Cummins, A. Sengur, and B. W. Schuller, "Deep end-to-end representation learning for food type recognition from speech," in *Proceedings of the 20th ACM International Conference on Multimodal Interaction (ICMI)*. Boulder, CO, USA: ACM, 2018, pp. 574–578.

[375] Y. Gao, O. Beijbom, N. Zhang, and T. Darrell, "Compact bilinear pooling," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA: IEEE, 2016, pp. 317–326.

[376] D. Pir, "Functional-based acoustic group feature selection for automatic recognition of eating condition," in *Proceedings of the 20th ACM International Conference on Multimodal Interaction (ICMI)*. Boulder, CO, USA: ACM, 2018, pp. 579–583.

[377] Q. Mao, M. Dong, Z. Huang, and Y. Zhan, "Learning salient features for speech emotion recognition using convolutional neural networks," *IEEE Transactions on Multimedia*, vol. 16, no. 8, pp. 2203–2213, 2014.

[378] J. Szep and S. Hariri, "Paralinguistic classification of mask wearing by image classifiers and fusion," in *Proceedings of INTERSPEECH*. Shanghai, China: ISCA, 2020, pp. 2087–2091.

[379] F. Demir, A. Sengur, N. Cummins, S. Amiriparian, and B. Schuller, "Low level texture features for snore sound discrimination," in *Proceedings of the 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, EMBS. Honolulu, HI, USA: IEEE, 2018, pp. 413–416.

[380] S. Pascual de la Puente, "Efficient, end-to-end and self-supervised methods for speech processing and generation," Ph.D. dissertation, Universitat Politècnica de Catalunya, 2019.

[381] A. Batliner, S. Hantke, and B. W. Schuller, "Ethics and good practice in computational paralinguistics," *IEEE Transactions on Affective Computing*, 2020, 19 pages.

[382] E. Gómez-González, E. Gomez, J. Márquez-Rivas, M. Guerrero-Claro, I. Fernández-Lizaranzu, M. I. Relimpio-López, M. E. Dorado, M. J. Mayorga-Buiza, G. Izquierdo-Ayuso, and L. Capitán-Morales, "Artificial intelligence in medicine and healthcare: a review and classification of current and near-future applications and their ethical and social impact," *arXiv preprint arXiv:2001.09778*, pp. 1–20, 2020.

[383] M. Federico, Y. Virkar, R. Enyedi, and R. Barra-Chicote, "Evaluating and optimizing prosodic alignment for automatic dubbing," in *Proceedings of INTERSPEECH*. Shanghai, China: ISCA, 2020, pp. 1481–1485.

[384] V. K. Singh, E. André, S. Boll, M. Hildebrandt, and D. A. Shamma, "Legal and ethical challenges in multimedia research," *IEEE MultiMedia*, vol. 27, no. 2, pp. 46–54, 2020.

[385] B. Milner and X. Shao, "Speech reconstruction from mel-frequency cepstral coefficients using a source-filter model," in *Proceedings of the 7th International Conference on Spoken Language Processing (ICSLP)*. Denver, CO, USA: ISCA, 2002, pp. 2421–2424.

[386] L. Juvela, B. Bollepalli, X. Wang, H. Kameoka, M. Airaksinen, J. Yamagishi, and P. Alku, "Speech waveform synthesis from mfcc sequences with generative adversarial networks," in *Proceedings of the 43rd International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Calgary, AB, Canada: IEEE, 2018, pp. 5679–5683.

[387] B. Schuller, S. Amiriparian, G. Keren, A. Baird, M. Schmitt, and N. Cummins, "The next generation of audio intelligence: A survey-based perspective on improving audio analysis," in *Proceedings of the 7th International Symposium on Auditory and Audiological Research (ISAAR)*. Nyborg, Denmark: The Danavox Jubilee Foundation, 2019, pp. 101–112.