Deep Learning Techniques for Computer Audition

Inaugural-Dissertation
zur Erlangung des Doktorgrades an der
Fakultät für Angewandte Informatik
der Universität Augsburg

vorgelegt von

Zhao Ren

2022

Erstgutachter:	Prof. Dr.-Ing. habil. Björn Schuller
Zweitgutachter:	Prof. Dr. Elisabeth André
	Prof. Dr.-Ing. habil. Andreas Maier
Tag der müdlichen Prüfung:	22. Februar 2022

Acknowledgement

First of all, I would like to sincerely thank my supervisor Prof. Björn W. Schuller for providing me with the opportunity to study and work in University of Augsburg, for inspiring me to carry out research into the fascinating field of deep learning on computer audition, for his kind encouragement, guidance and help during the past four years of my studies, and for his supervision of this thesis.

Moreover, I wish to express my gratitude to all colleagues and collaborators for their fruitful discussions and joint efforts to my previous publications and parts of this thesis. Especially, I would like to sincerely thank Prof. Mark D. Plumbley, Dr. Nicholas Cummins, Dr. Jing Han, Alice Baird, Dr. Kun Qian, Dr. Qiuqiang Kong, Dr. Zixing Zhang, Yi Chang, Adria Mallol-Ragolta, and many others. I would like to also extend my thanks to all my colleagues and friends in the chair for making the time of studying and working here pleasant and memorable.

Most of all, I would like to thank my parents for their love, and their constant support in various aspects of life.

My research was supported by the DFG's Reinhart Koselleck project No. 442218748 (AUDI0NOMOUS), the European Union's Horizon H2020 research and innovation programme under Marie Skłodowska-Curie grant agreement No. 766287 (TAPAS), the German national BMBF IKT2020-Grant under grant agreement No. 16SV7213 (EmotAsS), and the European Union's Seventh Framework under grant agreement No. 338164 (ERC StG iHEARu).

Abstract

Automatically recognising audio signals plays a crucial role in the development of intelligent computer audition systems. Particularly, audio signal classification, which aims to predict a label for an audio wave, has promoted many real-life applications. Amounts of efforts have been made to develop effective audio signal classification systems in the real world. However, several challenges in deep learning techniques for audio signal classification remain to be addressed. For instance, training a deep neural network (DNN) from scratch is time-consuming to extracting high-level deep representations. Furthermore, DNNs have not been well explained to construct the trust between humans and machines, and facilitate developing realistic intelligent systems. Moreover, most DNNs are vulnerable to adversarial attacks, resulting in many misclassifications.

To deal with these challenges, this thesis proposes and presents a set of deep-learning-based approaches for audio signal classification. In particular, to tackle the challenge of extracting high-level deep representations, the transfer learning frameworks, benefiting from pre-trained models on large-scale image datasets, are introduced to produce effective deep spectrum representations. Furthermore, the attention mechanisms at both the frame level and the time-frequency level are proposed to explain the DNNs by respectively estimating the contributions of each frame and each time-frequency bin to the predictions. Likewise, the convolutional neural networks (CNNs) with an attention mechanism at the time-frequency level is extended to atrous CNNs with attention, aiming to explain the CNNs by visualising high-resolution attention tensors. Additionally, to interpret the CNNs evaluated on multi-device datasets, the atrous CNNs with attention are trained in the conditional training frameworks. Moreover, to improve the robustness of the DNNs against adversarial attacks, models are trained in the adversarial training frameworks. Besides, the transferability of adversarial attacks is enhanced by a lifelong learning framework. Finally, the experiments conducted with various datasets demonstrate that these presented approaches are effective to address the challenges.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

Sound is a key component of human perception of the world in our daily life. A variety of information is transmitted to human through audio signals. For instance, a human being can perceive scenes through hearing the environmental sounds, a physician diagnoses patients by auscultation, and a person can feel the other's emotions from their conversation. In the past decades, the era of Artificial Intelligence (AI) sprang up to enable machines to automatically perceive audio information. Thanks to the rapid development of machine learning [4], particularly deep learning techniques [5], machine perception of audio signals, which is called *computer audition* [6], is advanced to facilitate numerous applications in intelligent systems, e. g., scene recognition in mobile robots [7], automatic medical diagnosis systems [8], affective social robots [9], etc.

As one of the well-known computer audition techniques, the audio classification problem aims to produce a label from an audio signal, while the audio samples are grouped into two or multiple pre-determined classes [10]. For example, each environmental audio signal is labelled with one of the classes, including park, street, shopping mall, tram, etc. Different types of classes promote a series of audio classification tasks, such as Acoustic Scene Classification (ASC) [11], Heart Sound Classification (HSC) [12], Speech Emotion Recognition (SER) [13], and many more.

In this thesis, audio classification tasks are formulated as the major task in a supervised learning framework [14], where a model is optimised to learn a mapping function from each input to a label based on a pre-annotated training set. To achieve audio classification, machine learning approaches, like Support Vector Machines (SVMs) and decision trees, have been effectively used in most studies [15, 16]. With the increasing amounts of data due to the advances of data storage technologies [17], the traditional machine learning methods are struggling to process massive data [18]. The recent advent of deep learning techniques has shown a great contribution to the classification of large-scale audio data sets compared to conventional

machine learning methods. For example, Convolutional Neural Networks (CNNs) can yield good performance by extracting highly abstract representations from audio signals [19], and Recurrent Neural Networks (RNNs) can perform well via learning sequential features [20]. The connected structure – Convolutional Recurrent Neural Networks (CRNNs) was also employed in recent studies for taking advantage of CNNs and RNNs [21].

With these successful applications of deep learning on audio signal classification, amounts of significant challenges posted by audio classification tasks remain to be addressed by developing novel deep learning topologies and frameworks. The challenges to be focused in this thesis are listed as below:

(1) *Extraction of deep representations.* With a strong capability of extracting highly abstract representations, many deep learning architectures have been successful in audio classification tasks [22, 23]. However, there are two limitations of training deep learning models (i. e., Deep Neural Networks (DNNs)) needed to be overcome. Firstly, a small amount of data might result in over-fitting during training DNNs [24]. Therefore, training a deep learning model requires a tremendous amount of data in order to learn a huge number of parameters during the training procedure. In some practical cases, data collection is expensive and time-consuming. For example, medical data sets are mostly small because of the legal restraint and lack of resources in hospitals [25]. Secondly, developing a deep learning model is time-consuming from designing the model architecture to the training procedure [26], making it difficult to spread deep learning-based applications to the real world, e. g., mobile devices. Therefore, processing unavoidable small-scale data sets and saving the training time are two key points in extracting deep representations.

(2) *Explanation of deep learning models.* Even though deep learning has achieved high performance, it is still mostly presented as an unexplainable "black box" in practical scenes [27]. Lack of explainability in a deep learning model poses many challenges during solving real-world problems, particularly in safety-sensitive domains, e. g., autonomous driving, healthcare, etc. For instance, in the medical area, an uninterpretable auxiliary diagnosis system cannot gain the physicians' trust, as a wrong predicted decision may lead to severe health problems and even death of a patient [28]. To this end, constructing transparent and explainable DNNs is needed to promote the development of deep learning in safety crucial fields. More recently, explanation of deep learning models was mostly investigated in image processing tasks via visualisation of the models [29, 30]. However, only a few studies have worked on explaining deep learning models for audio classification [31]. Enhancing the explainability of DNNs for audio signal classification tasks is still urgently required to break the "black box".

(3) *Robustness of deep learning models.* Because of the development of more and more mature deep learning techniques, the robustness of deep learning models is becoming an important factor in the real world. In a recent study [32], deep learning was found to be vulnerable to adversarial attacks generated by adding very small and human indistinguishable perturbations to the original input data. DNNs may produce a wrong prediction with high confidence due to such perturbations. Adversarial attacks are a serious threat to security-sensitive fields, e. g., finance, autonomous driving, and healthcare. For example, a wrong prediction given by a deceived deep learning model may result in misdiagnosis and wrong treatment in the medical domain. With this in mind, improving the robustness of deep learning models against adversarial attacks is a serious requirement in applications of deep learning. Adversarial attacks have been studied on image processing tasks in many research works [33], yet, only a few works have focused on adversarial attacks for audio classification [34]. The topic of adversarial attacks is still an active field of research in the audio signal classification.

Overall, this thesis mainly focuses on analysing, discussing, and overcoming the above three challenges on audio signal classification tasks via developing novel deep learning architectures and frameworks. The audio signal classification tasks contain ASC, HSC, Pain Level Evaluation (PLE), and SER. Due to the specific requirements of different tasks, the study of each challenge is investigated on partial tasks with emphasis.

1.2 Aims of the Thesis

To address the above three challenges introduced in Section 1.1, the three corresponding major contributions, in which the last two contributions are respectively followed by two extended contributions, are listed as follows:

(1) *Extracting high-level representations through transfer learning.* To address the first challenge, transfer learning is a potential way to enable a DNN model working on a small database, and save the time of the model design and the training procedure. This thesis contributes to extracting deep representations from audio signals with the use of pre-trained CNN models from large-scale image datasets [35], while the spectrum representations of the original audio signals are extracted as the input of CNNs [36].

(2) *Explaining deep learning models via an attention mechanism.* To face the second challenge, the focus of this thesis is to visualise DNNs using an attention mechanism. With the aim of estimating the contribution of each unit in the high-level representations to the final predictions, the attention mechanism

has the potential to improve the performance of DNNs. More importantly, it is interesting to visualise the representations learnt by the attention, and further help on understanding the decisions made by the "black-box" neural networks.

(2-1) *Explaining deep learning models in a multi-device condition.* Because of the development of audio recording devices, processing multi-device data sets in a training procedure is needed to reduce the time and space costs of training models on each single-device data set independently. The most straightforward solution to save the costs is jointly training a model on multi-device data, yet, the difference of the data distributions is a limitation to achieve a good performance. This thesis, for the first time, proposes the conditional training to improve the performance of both independently training models and jointly training a model. Furthermore, an attention mechanism is applied to visualise the DNNs and help analyse the difference of DNNs in multi-device data.

(3) *Generating and protecting deep learning models against adversarial attacks.* To deal with the third challenge, adversarial training is performed in this thesis to protect DNNs against adversarial attacks. This approach has two advantages: i) the training set is augmented by the generated adversarial data, and ii) the DNN model is trained to converge on adversarial data. Furthermore, a similarity-based adversarial training approach is proposed to improve the robustness of DNNs by reducing the difference between the deep representations of the original real data and the generated adversarial data.

(3-1) *Improving the transferability of adversarial attacks.* Besides protecting the DNNs against adversarial attacks, a key measurement of the attacks is their transferability of generating adversarial data to deceive multiple target DNNs. The focus of this thesis is set on enhancing the transferability of adversarial attacks via lifelong learning for the first time. In particular, instead of training an attack model in a multi-task learning or a transfer learning framework, a widely used lifelong learning approach of Elastic Weight Consolidation (EWC) is explored.

1.3 Outline

With the motivation and goals of this thesis introduced in *Chapter 1*, to provide a good overview for the readers, the rest of the thesis is structured into the four chapters as follows.

Chapter 2 reviews the theoretical background of audio signal classification systems. A general framework of audio signal classification systems is firstly overviewed.

Afterwards, feature-based machine learning is introduced with a brief survey of existing feature sets and models that are commonly used in audio signal classification. Furthermore, the end-to-end deep learning is presented with pre-processing audio signals into spectrum representations and typical deep learning architectures.

Chapter 3 concentrates on a set of the employed and contributed methodologies corresponding to the challenges mentioned in Section 1.1. Firstly, deep representation learning using transfer learning paradigms is given. Next, an attention mechanism is investigated to visualise DNNs, and conditional training is proposed to train device-robust DNNs. Finally, adversarial training is studied to train robust deep learning models, and enhancing the transferability of adversarial attacks is investigated.

Chapter 4 presents the practical evaluations of methods described in Chapter 3. Following an introduction of the databases and evaluation metrics, the performances on each database are presented and discussed.

Chapter 5 summarises the achievements and limitations of the work in this thesis, discusses the ethics in computer audition, and suggests the outlook of potential future perspectives.

2

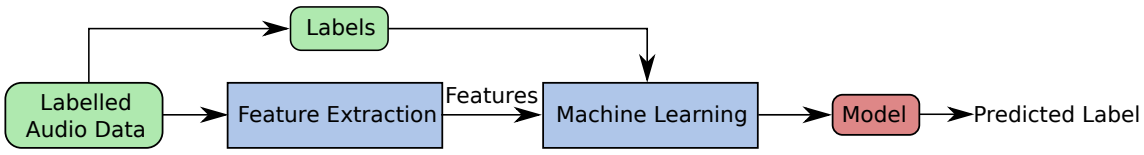
Theoretical Background

This chapter mainly introduces the theoretical background of intelligent audio signal classification to the readers. To be more specific, two general frameworks of audio signal classification systems, including traditional machine learning and end-to-end deep learning, are introduced and compared in Section 2.1. For better understanding and comparing the two frameworks, the traditional machine learning is briefly described in Section 2.2, and the end-to-end deep learning is then presented in Section 2.3.

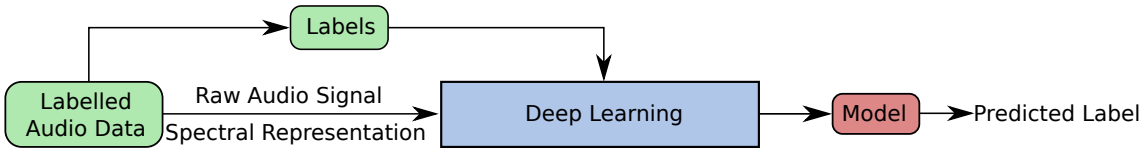
2.1 General Framework of Audio Signal Classification Systems

A machine learning algorithm aims to optimise the parameters of a model through an iterative progress with a goal of improving the performance on a specific task. A standard machine learning framework for audio signal classification consists of two stages: a training stage and a test stage. The goal of the training stage is to optimise the model's parameters on a labelled training set. At the test stage, the trained model is used to predict the labels of an unseen test set.

In particular, from the perspective of the model architectures and the corresponding inputs, machine learning contains two subsets: traditional machine learning (i. e., feature-based machine learning), and end-to-end deep learning (cf. Figure 2.1). In feature-based machine learning (cf. Figure 2.1(a)), the first step is to extract features, which are generally represented as vectors, from the labelled training set. Furthermore, the extracted feature vectors and the annotated labels are provided as the input of a feature-based machine learning algorithm. Typically, the feature vectors are hand-crafted, such as pitch, formants, spectral features, etc. Lastly, the model in the feature-based machine learning algorithm is optimised to accurately classify the audio signals. Compared to the feature-based machine learning, the goal of end-to-end deep learning is to learn a more complex model directly from



(a) Feature-based machine learning.



(b) End-to-end deep learning.

Figure 2.1: General pipelines of the training procedure in an audio signal classification system using feature-based machine learning and end-to-end deep learning, respectively.

the audio signals or the simple spectrum representations by skipping the process of feature extraction (cf. Figure 2.1(b)) [37]. The idea of end-to-end deep learning is to save the costs of designing the feature sets, and directly learn highly abstract representations, which are more helpful for a classification task than hand-crafted features. The approaches in this thesis are achieved by end-to-end deep learning, which is therefore called *deep learning* for simplification in the following sections.

2.2 Feature-based Machine Learning

In the following, the feature-based machine learning procedure for audio signal classification will be given, including the feature extraction in Section 2.2.1 and the classification models in Section 2.2.2.

2.2.1 Feature Extraction

Essentially, the original audio signals are not directly fed into a traditional machine learning model, as the useless information inside the audio signals may pull the classification performance down. For instance, noises and human speech voices are not helpful for an HSC task. To extract more useful information for improving the classification performance, feature vectors are computed from the raw audio signals regarding a crafted feature set. Feature extraction has been successfully employed in a set of audio classification tasks, e. g., HSC [38], SER [39], etc. The idea of feature extraction is to represent raw audio signals as more abstract representations, which are more suitable for machine learning [40]. As the input of a traditional machine

learning model, the acoustic feature vectors are typically extracted using a set of signal processing methods, such as Fourier transform and filters.

Since 2013, the feature set employed by the baseline of INTERSPEECH COMputaional PARalinguistic ChallengE (ComParE), has shown good performances in audio classification tasks [38]. The ComParE feature set consists of a variety of segmental Low-Level Descriptors (LLDs) and supra-segmental functionals. For an audio signal, the LLDs are obtained from the short-term analysis on every segment, while the functionals aim to project the sequential LLDs onto an independent vector.

In the ComParE feature set, the LLDs mainly contains the raw LLDs and the derived LLDs. The raw LLDs are extracted from each short-time segment covered by a sliding window, which is mostly set as $25 - 32$ ms [41]. The raw LLDs in the ComParE feature set include prosodic features (e. g., loudness, Root Mean Square (RMS) energy, F_0 via subharmonic summation, etc), spectral features (e. g., spectral roll-off point 0.25, 0.50, 0.75, 0.90, spectral flux, spectral entropy, etc), cepstral features (e. g., Mel-Frequency Cepstral Coefficients (MFCCs) $1 - 14$), and voice quality features (e. g., probability of voicing, jitter, and shimmer). Additionally, the derived LLDs are added into the feature set as well, resulting in 130-dimensional LLDs. To further compute the potential information among a time series of the extracted LLDs, functionals are applied inside each LLD and over multiple LLDs. Especially, functionals are computed through summarising the statistical features from the LLDs, such as root-quadratic mean, flatness, standard deviation, temporal centroid, linear regression slope, etc. Finally, a $6,373$-dimensional feature vector is obtained from an audio sample.

Besides the ComParE feature set, the extended Geneva Minimalistic Acoustic Parameter Set (eGeMAPS) was designed especially for the affective speech analysis [42]. The eGeMAPS feature set has shown good performances in many speech-related tasks, such as SER [43], detection of Alzheimer's dementia [44], etc. Similar to the ComParE feature set, the eGeMAPS feature set also consists of LLDs and functionals. In the eGeMAPS feature set, 25 LLDs are employed, including frequency related parameters (e. g., pitch, jitter, formant 1, 2, and 3 frequency, formant 1, and formant $2 - 3$ bandwidth), energy/amplitude related parameters (e. g., shimmer, loudness, and harmonics-to-noise ratio), and spectral (balance/shape/-dynamics) parameters (e. g., alpha ratio, hammarberg index, harmonic difference H1–H2, MFCCs $1 - 4$, etc). Moreover, the functionals (e. g., arithmetic mean, coefficient of variation, etc) are applied to the LLDs, generating the 88-dimensional eGeMAPS feature set. To facilitate the acoustic feature extraction process, the openSMILE toolkit [45] provides several predefined and well-established feature sets, such as the ComParE and the eGeMAPs feature sets.

Additionally, the technique of Bag-of-Audio-Words (BoAW) [46] was proposed to estimate the understandable representations based on the segment-level LLDs for SER. The procedure of BoAW consists of three steps: codebook generation, vector

quantisation, and BoAW aggregation. The purpose of codebook generation is to extract a series of audio words from the LLDs using either random sampling or k-means clustering. Random sampling is selected to achieve codebook generation in this thesis, as the computation of k-means has been shown much slower than random sampling [46]. Notably, a codebook is only generated from the LLDs of the training set. Given a generated codebook, each LLD vector is then assigned to the closest audio word based on the Euclidean distance. This procedure is called vector quantisation. Once the codebook is generated, a histogram is built to count the frequencies of the audio words in the BoAW aggregation step. Through the BoAW procedure, the segmental LLDs are transformed to high-level representations. The openXBOW [47], an open-source toolkit for BoAW features extraction, has been successful in a number of audio classification tasks [38, 48]. Therefore, the openXBOW is used in the partial experiments of this thesis.

2.2.2 Classification

With the extracted features as the input, a classifier is trained to predict the labels for the audio samples. In general, machine learning techniques contain two major types of models: generative models and discriminative models. Given the feature vectors $\boldsymbol{x}$ and the labels y, generative models aim to calculate the posterior probability $P(y|\boldsymbol{x})$ using the Bayes rule on the modelled joint probability distribution $P(\boldsymbol{x}, y)$, whereas discriminative models attempt to compute $P(y|\boldsymbol{x})$ directly from the given features and labels.

In a generative model, the likelihood probability is firstly obtained using $P(\boldsymbol{x}|y) = \frac{P(\boldsymbol{x},y)}{P(y)}$, where $P(y)$ is the prior distribution of y. $P(y|\boldsymbol{x})$ is then calculated using the Bayes rule $P(y|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|y)P(y)}{P(\boldsymbol{x})}$, where $P(\boldsymbol{x})$ is the prior distribution of $\boldsymbol{x}$. In this context, a set of generative models have been proposed, such as naive Bayes, Gaussian mixture models, and hidden Markov models. Although the generative models are more suitable for semi-supervised and unsupervised learning due to the calculation of the joint probability distribution, they have been successfully applied to a set of audio signal classification tasks [49, 50, 51].

A discriminative model focuses on producing $P(y|\boldsymbol{x})$ through computing the boundary between classes, instead of estimating the joint probability distribution $P(\boldsymbol{x}, y)$. At this point, discriminative models can often achieve high performances on audio classification tasks, such as ASC [52], because of their direct computing of $P(y|\boldsymbol{x})$. Typically, discriminative models include SVMs, decision trees, k-nearest neighbours, and neural networks [53]. An SVM model attempts to learn a soft decision boundary between classes via constructing a hyperplane based on the features; a decision tree constructs an interpretable model with a variety of nodes; k-nearest neighbours evaluate each feature vector's distances to its neighbours. Herein, the neural networks are the artificial neural network with a single layer, which is the

foundation of Feedforward Neural Networks (FNNs) with a deeper structure in Section 2.3.2.1.

As SVMs have shown good performances in audio classification tasks [52], they are chosen to represent the feature-based machine learning method in this thesis. The basic idea of a linear SVM is to build a hyperplane for a binary classification task, through optimising the function

$$\arg\min \frac{||\boldsymbol{w}||^2}{2} + C\sum_{i=1}^{N}\alpha_i, \text{ subject to } y_i(\boldsymbol{w}^T x_i + b) \geq 1 - \alpha_i, \alpha_i \geq 0, \tag{2.1}$$

where $y_i \in \{-1, 1\}, i = 1, ..., N$, N is the number of samples, $\boldsymbol{w}$ is the weights of each value in a feature vector, b is the bias, and C, the complexity parameter, controls the slack variable α_i to penalize the misclassified samples.

Regarding the SVM model for binary classification, two major strategies are available for a multi-class classification: one-versus-all strategy classifies each class with the rest classes, and one-versus-one strategy classifies each pair classes [54]. Therefore, for a multi-class classification task, SVMs optimise the parameters to construct a set of maximum-margin hyperplanes in the feature space. However, it might be challenging to classify non-linear feature spaces with a linear SVM model. In this regard, a linear SVM model can also be improved to nonlinear models by integrating kernel functions, such as the radial basis function [55].

2.3 Deep Learning

Despite that the feature-based machine learning has been widely applied to many audio classification tasks [52, 56], its application is limited by the time-consuming feature extraction procedure [24]. To overcome this limitation, deep learning takes the raw data as the input, and trains multi-layer neural networks for classification. As mentioned in Section 2.1, the input of a deep learning model can be either the original audio signals or the simple spectrum representations. In this section, the data pre-processing procedure of generating spectrum representations will be introduced in Section 2.3.1, and the typical DNNs for classification will be described in Section 2.3.2.

2.3.1 Data Pre-processing

Spectrum representations, i. e., time-frequency representations of audio signals, have been used as input of DNNs in a number of research studies for audio signal processing. For instance, in [36], spectrograms were employed as the input of a CNN models for music onset detection. Spectrograms were also investigated to extract deep spectrum features for snore sound classification in [22]. Log-mel spectrograms

were employed to train a CNN model for large-scale audio classification in [19]. MFCC features was used in [57] for SER. Wavelet representations were fed into RNNs for HSC in [58]. In the following, a series of typical spectrum representations will be introduced, including spectrogram in Section 2.3.1.1, log mel spectrogram in Section 2.3.1.2, MFCCs in Section 2.3.1.3, and scalogram in Section 2.3.1.4.

2.3.1.1 Spectrogram

Through Short-Time Fourier Transform (STFT), a specific Fourier transform, spectrograms are extracted from audio signals. Fourier transform [59] decomposes the global frequency components of a whole audio signal, yet, loses the local frequency information in shorter periods. The purpose of STFT is to provide the frequency components in every short-time interval of an audio wave. Given an audio signal s, the spectrogram is obtained via applying Fourier transform to each audio segment split by a sliding window,

$$f(t_s, \omega) = \sum_{t=-\infty}^{\infty} s[t]w[t - t_s]e^{-j\omega t}, \tag{2.2}$$

where t_s is a constant value indicating the translation distance of the window function w at each step, and ω is the angular frequency. At this point, the spectrograms are represented with two axes of time frames and frequencies, where each time frame holds the time length of the window function. A set of window functions are available to conduct STFT, such as Triangular window, Hann window, Hamming window, Gaussian window, etc [60].

2.3.1.2 Log Mel Spectrogram

An early study [61] proposed the mel scale to define the relations between pitch and frequency, similar to the non-linear human auditory system. To accurately mimic the perception of human ears, the mel scale was then investigated in a set of studies [62, 63]. The mel scale was typically defined by

$$f_{mel}(f) = 2595 \log_{10}(1 + \frac{f}{700}), \tag{2.3}$$

where f is the spectrogram obtained by STFT. In practice, to obtain a mel spectrogram, a set of Triangular mel filter banks are constructed according to the mel scale, and then multiplied with the original spectrogram. The passing frequency range of each mel filter bank is increasing with the frequency. The number of the mel filter banks determines the dimensions of the mel spectrograms at the mel frequency level.

Recently, mel spectrograms have shown the potential on audio signal classification tasks [64, 65]. Moreover, in some classification tasks, as some useful information

is hidden in the low frequencies, the logarithms of mel spectrograms (i. e., log mel spectrograms) were employed to enhance the information of low-frequency bands in a lot of research studies [66, 67].

2.3.1.3 Mel-Frequency Cepstral Coefficients

When obtaining log mel spectrograms, filter bank coefficients are most highly correlated. Therefore, a Discrete Cosine Transform (DCT) is employed to decorrelate the log mel spectrograms [68], resulting in a compressed representation of the filter banks (called MFCCs). The operation of DCT is defined by

$$f_{mfcc}(n) = \sum_{t=1}^{N_{mel}} \log(f_{mel}[t]) \cos(\frac{\pi n(t+0.5)}{N_{mel}}), n = 0, ..., N_{mfcc}, \tag{2.4}$$

where N_{mel} is the dimensional number of mel frequency, and N_{mfcc} denotes the dimensional number of the obtained MFCCs.

2.3.1.4 Scalogram

The above three kinds of spectrum representations only provide a fixed resolution at the time-frequency level, as STFT cannot reach a high-resolution representation at both time and frequency domains according to the Heisenberg uncertainty principle [69]. For example, increasing the resolution at the time domain (i. e., increasing the number of the windows) leads to the reduction of the frequency resolution (i. e., reducing the width of the sliding window), and vice versa. In this regard, wavelet transform [70, 71] was proposed to overcome this shortcoming of STFT by a wavelet function, producing multi-scale time-frequency representations, which are called scalograms. In this thesis, the Continuous Wavelet Transform (CWT) will be introduced to generate scalograms. The CWT of a discrete audio signal is the convolution of the signal with a scaled and translated version of a wavelet basis, which replaces the window function in STFT [72]. In relative studies, several wavelet basis functions have been proposed, such as *bump* [73] and *morse* [74], which will be introduced as below.

Using the *bump* wavelet transform, the wavelet basis Ψ_{bump} is defined by

$$\Psi_{bump}(\epsilon\omega) = e^{\left(1-\frac{1}{1-(\epsilon\omega-\mu)^2/\sigma^2}\right)} 1_{[(\mu-\sigma)/\epsilon,(\mu+\sigma)/\epsilon]}, \tag{2.5}$$

where ϵ is the wavelet scale, μ and σ are two constant parameters, σ affects the frequency and time localisation, and $1_{[(\mu-\sigma)/\epsilon,(\mu+\sigma)/\epsilon]}$ is the indicator function for the interval $\omega \in [(\mu-\sigma)/\epsilon, (\mu+\sigma)/\epsilon]$.

Besides, the wavelet basis of the *morse* transform is defined as

$$\Psi_{morse}(a, \gamma, \omega) = u(\omega)\beta_{a,\gamma}\omega^{\frac{a^2}{\gamma}} e^{-\omega^\gamma}, \tag{2.6}$$

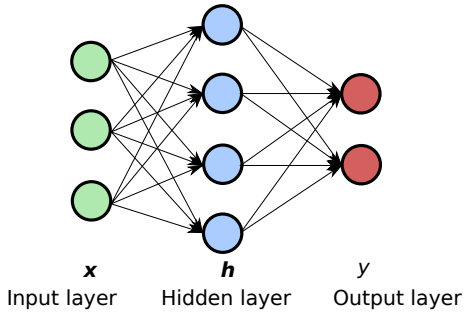


Figure 2.2: An example of FNNs with an input layer, a hidden layer, and an output layer. The input is denoted by $\boldsymbol{x}$, the units in the hidden layer is $\boldsymbol{h}$, and the output is y.

where $u(\omega)$ is the unit step, $\beta_{a,\gamma}$ stands for a normalising constant, a^2 is the time-bandwidth product, and γ is the symmetry parameter.

2.3.2 Classification

With the obtained spectrum representations as the input, deep learning models are then trained to classify the data. In this section, three classical deep learning structures will be introduced, including FNNs in Section 2.3.2.1, CNNs in Section 2.3.2.2, and RNNs in Section 2.3.2.3.

2.3.2.1 Feedforward Neural Networks

Herin, an FNN model is defined by the simplest artificial neural network with multiple layers, which are connected by feeding forward the output of each layer to the next layer (cf. Figure 2.2). Besides the input and output layers, an FNN model generally holds at least one hidden layer. Each unit of a layer in an FNN model is connected to all units of the next layer. The computational procedure in each layer is a linear transformation from the input to the output. For example, at the n-th layer, $n = 1, ..., N_l$, where N_l is the number of layers, the output activation $\boldsymbol{h}^n$ is calculated from the input $\boldsymbol{h}^{n-1}$ by

$$\boldsymbol{h}^n = \boldsymbol{w}^n \boldsymbol{h}^{n-1} + b^n, \tag{2.7}$$

where $\boldsymbol{w}$ is the weight values of each unit, and b^n stands for the biases. In the input layer, the input $\boldsymbol{h}^0$ is equal to the input data $\boldsymbol{x}$. Before feeding the activation $\boldsymbol{h}^n$ into the next layer, $\boldsymbol{h}^n$ is mostly passed to a non-linear transformation conducted by an activation function, e. g., tanh function, Rectified Linear Unit (ReLU) function, sigmoid function, softmax function, etc. Especially, a softmax function or a log-softmax function, is applied to the output of the final layer to compute the prob-

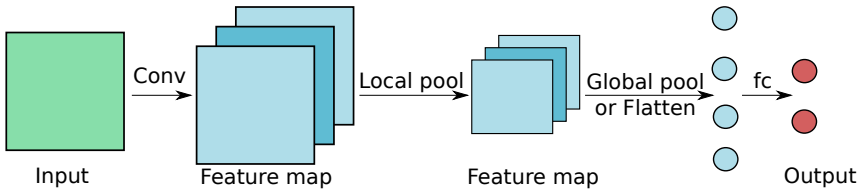


Figure 2.3: An example of two-dimensional CNNs. This CNN model includes a convolutional layer, a local pooling layer, a global pooling or a flattening layer, and a fully connected layer. The "fc" means the fully connected layer.

abilities of every class. For each audio sample, the class with the largest probability is finally selected to be the predicted label.

During the training procedure, backpropagation is often employed to optimise the neural network for the best weights and biases in Equation (2.7) [75, 76]. The whole procedure of backpropagation consists of two stages: forward pass and backward pass. In the forward pass, the activations of all layers in an FNN model are computed using Equation (2.7), while in the backward pass, an objective function (i. e., loss function), such as cross-entropy loss, is minimised through calculating the gradients of the loss function regarding the parameters. However, it is challenging to find the global minimum, as the relation between the loss function and parameters is highly non-linear. Therefore, gradient descent is mostly used to adjust the parameters towards the negative error gradient in a number of small steps. At this point, the objective function is approximately minimised to the global minimum. Based on gradient descent, a set of optimisation algorithms have been proposed and widely used, such as Stochastic Gradient Descent (SGD) [77], RMSProp [78], Adam [79], etc.

2.3.2.2 Convolutional Neural Networks

With a strong capability of learning high-level representations, CNNs have been applied to process both image data and audio signals [23, 80]. Due to various dimensions of the data, CNNs have been mainly developed as one-dimensional, two-dimensional, and three-dimensional structures [23, 80], each of which is corresponding to the dimensions of the input data. In this thesis, since the spectrum representations of audio signals are two-dimensional (time-frequency), they are fed into two-dimensional CNNs. Therefore, two-dimensional CNNs are introduced in this section.

A structure of two-dimensional CNNs is depicted in Figure 2.3. We can see that, a CNN model generally contains a set of convolutional layers, pooling layers, and fully connected layers. As the key component of CNNs, the convolutional layers consist of many trainable filters (i. e., kernels), which are convoluted on the input data or the feature maps outputted by the internal layers. During the convolutional

operation, the dot product of each filter and each local area is computed as a unit of the output, so that the convolutional layers can extract shift-invariant features [81]. Given the $C^{n-1} \times P^{n-1} \times Q^{n-1}$ feature map $\boldsymbol{h}^{n-1}$ from the $(n-1)$-th layer, where C^{n-1} is the channel number and $P^{n-1} \times Q^{n-1}$ is the size of the feature map at the time and frequency axes, the output of the n-th convolutional layer is computed by

$$\boldsymbol{h}_j^n = \sum_{i=1}^{C^{n-1}} \boldsymbol{w}_{ij}^n * \boldsymbol{h}_i^{n-1} + b_j^n, \tag{2.8}$$

where $\boldsymbol{h}_j^n$ is the j-th channel of $\boldsymbol{h}^n$, $\boldsymbol{w}_{ij}$ is the (i,j)-th convolutional kernel, $*$ denotes the convolutional operation, and b_j^n is the bias at the j-th channel. In general, convolutional layers are followed by batch normalisation and a ReLU activation function for convergence acceleration, and batch normalisation can also improve the stability of CNNs [82, 83].

The pooling layers reduce the size or the dimensions of the feature maps through combining multiple units inside a feature map into a single unit using a set of kernels. Regarding the methods of combining the neurons, pooling layers typically contain max pooling and average pooling. In an area covered by a kernel, a max pooling layer summarises the units through selecting the maximum unit, whereas an average pooling layer computes the average value of the units. Furthermore, the pooling layers consist of local and global pooling layers regarding the kernel size. As shown in Figure 2.3, local pooling aims to reduce the size of the feature map with keeping it as a three-dimensional tensor, while global pooling aims to convert the three-dimensional feature map into a one-dimensional tensor. A local pooling layer mostly follows a convolutional layer with batch normalisation and a ReLU activation function, aiming to reduce the computational cost and improve the robustness of CNNs against the input variation [84]. Global pooling is usually set between the final convolutional layer and fully connected layer(s) to generate the fed feature map as a one/two-dimensional tensor. Alternatively, the flattening operation, which simply flattens the feature map into a vector with retaining all units, is also available to reduce the three dimensions of a feature map into one dimension.

Finally, same as the layers in an FNN model for a classification task, the fully connected layers predict the probabilities of every class by processing the feature vectors obtained from either global pooling or flattening.

2.3.2.3 Recurrent Neural Networks

RNNs, constructed by a set of recurrent layers, can extract effective representations from sequential data, e. g., audio signals. As illustrated in Figure 2.4, the hidden layer is recurrent from each time step to the next one, resulting in many sequences in each hidden layer. Each sequence in a recurrent layer is trained to process the input data at the corresponding time step, producing a hidden state, which is represented

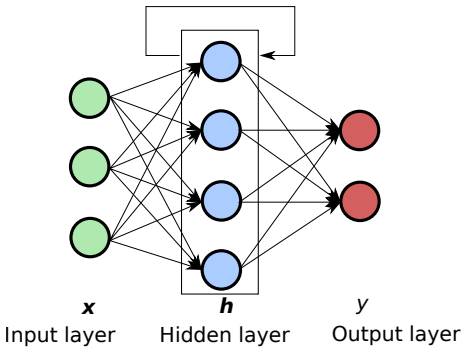


Figure 2.4: An RNN structure. The input is $\boldsymbol{x}$, the units in the hidden layer are represented as $\boldsymbol{h}$, and the output is y.

as a vector. The hidden states at all time steps are then fed into the next recurrent layer, or used to predict the probabilities of each class when these hidden states are produced by the final recurrent layer. While the total number of time steps is T, the hidden state h_t at the time step t, $t = 1, ..., T$, is defined by

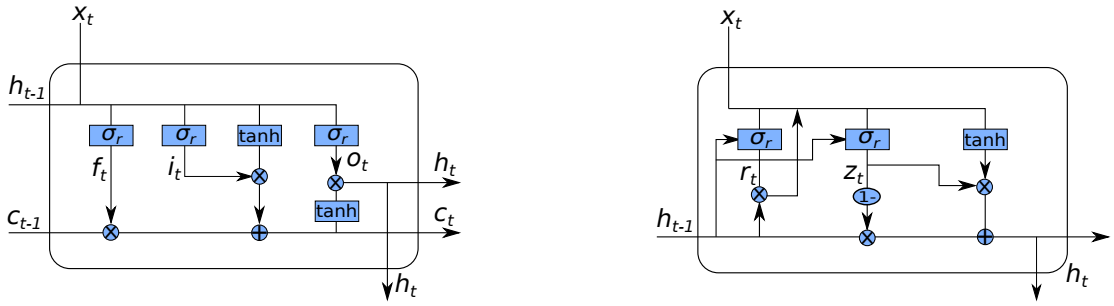
$$h_t = \sigma_h(\boldsymbol{w}_h x_t + \boldsymbol{u}_h h_{t-1} + b_h), \tag{2.9}$$

where $\boldsymbol{w}_h$ and $\boldsymbol{u}_h$ are the weights, b_h is the bias, x_t denotes the input vector at the time step t, h_{t-1} stands for the previous hidden state at the time step $t-1$, and σ_h is an activation function. In classification tasks, the hidden states obtained from the final recurrent layer are generally integrated to a single vector, which is then forwarded to a fully connected layer in Equation (2.7). Typically, both extracting the hidden state at the last time step and computing the average of the hidden states are available to generate a vector as the input of the fully connected layer.

The above simple RNN structure can better capture the sequential context information [85] than an FNN model and a CNN model. However, the simple RNNs cannot process long-term context information due to the exploding and vanishing gradient problem, which describes that the backpropagated error either blows up or decays over time [86]. In this regard, the Long Short-Term Memory (LSTM) RNN structure [87] and the Gated Recurrent Unit (GRU) RNN structure [88] were proposed to address the exploding and vanishing gradient problem. In LSTM-RNN and GRU-RNN models, the neurons inside a simple RNN model are replaced by memory blocks (cf. Figure 2.5), which are connected recurrently.

At the time step t, an LSTM memory block(cf. Figure 2.5(a)) consists of a cell state c_t and three gate units, including an input gate i_t, an output gate o_t, and a forget gate f_t. These three gate units are in charge of writing, reading, and resetting the cell state. The procedure inside an LSTM memory block at the time step t is defined by

$$i_t = \sigma_r(\boldsymbol{w}_i x_t + \boldsymbol{u}_i h_{t-1} + b_i), \tag{2.10}$$



(a) An LSTM memory block.

(b) A GRU memory block.

Figure 2.5: The structures of the LSTM and the GRU memory blocks. (a) An LSTM memory block contains an input gate i_t, an output gate o_t, and a forget gate f_t, and (b) a GRU memory block consists of a reset gate r_t and an update gate z_t.

$$f_t = \sigma_r(\boldsymbol{w}_f x_t + \boldsymbol{u}_f h_{t-1} + b_f), \tag{2.11}$$

$$o_t = \sigma_r(\boldsymbol{w}_o x_t + \boldsymbol{u}_o h_{t-1} + b_o), \tag{2.12}$$

$$\tilde{c_t} = \tanh(\boldsymbol{w}_c x_t + \boldsymbol{u}_c h_{t-1} + b_c), \tag{2.13}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c_t}, \tag{2.14}$$

$$h_t = o_t \odot \tanh(c_t), \tag{2.15}$$

where $\tilde{c_t}$ stands for a candidate cell state, $\boldsymbol{w}_i$, $\boldsymbol{w}_f$, $\boldsymbol{w}_o$, $\boldsymbol{w}_c$, $\boldsymbol{u}_i$, $\boldsymbol{u}_f$, $\boldsymbol{u}_o$, $\boldsymbol{u}_c$ are the weight matrices, b_i, b_f, b_o, b_c are the biases, σ_r is a logistic sigmoid function, and $\odot$ denotes the element-wise multiplication. We can see that, the input gate controls how much a new value flows into the cell state, the output gate controls how much the cell state is used to compute the output hidden state h_t, and the forget gate controls how much a value remains in the cell state. For example, while the input gate is closed and the forget gate is open, the cell state cannot be updated by the new inputs. Therefore, only the information from the previous time steps is accessed by the output gate.

Compared to the LSTM-RNN model, a GRU-RNN model has simpler memory blocks, each of which has a reset gate r_t and an update gate z_t (cf. Figure 2.5(b)). A GRU memory block processes the input and the previous hidden state using

$$r_t = \sigma_r(\boldsymbol{w}_r x_t + \boldsymbol{u}_r h_{t-1} + b_r), \tag{2.16}$$

$$z_t = \sigma_r(\boldsymbol{w}_z x_t + \boldsymbol{u}_z h_{t-1} + b_z), \tag{2.17}$$

$$\tilde{h_t} = \tanh(\boldsymbol{w}_g x_t + \boldsymbol{u}_g(r_t \odot h_{t-1}) + b_g), \tag{2.18}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h_t}, \tag{2.19}$$

where $\tilde{h_t}$ is a candidate hidden state, $\boldsymbol{w}_r$, $\boldsymbol{w}_z$, $\boldsymbol{w}_g$ are the weights, and b_r, b_z, b_g are the biases. The update gate aims to control both how much to forget the existed

contextual information h_{t-1} and how much to update the hidden states with the current information $\tilde{h}_t$. For instance, when the update gate is open (gate activation values are close to one), the hidden state h_{t-1} is forgotten and the information $\tilde{h}_t$ is remembered. Due to less parameters than those of LSTM-RNNs, GRU-RNNs converged faster than LSTM-RNNs in [89].

Another RNN structure worthwhile introducing is the bi-directional RNN (BRNN) model [90], which accesses both past and future information. A BRNN layer contains two independent recurrent layers scanning the input sequences in two opposite directions. With the bi-directional recurrent layers in each BRNN layer, BRNNs are efficient to learn context inter-dependent features. Furthermore, bi-directional LSTM-RNNs (BLSTM-RNNs) and bidirectional GRU-RNNs (BGRU-RNNs) can also be extended from LSTM-RNNs and GRU-RNNs with the same idea of BRNNs.

3 Methodology

This chapter aims to analyse and address the three challenges of deep learning mentioned in Section 1.1, with the targeted contributions in mind (cf. Section 1.2). To tackle the first challenge, i. e., extracting effective deep representations, the transfer-learning-based deep representation learning approaches are firstly discussed in Section 3.1. Regarding the second challenge about explaining DNNs, the deep learning models are interpreted using an attention mechanism in Section 3.2, aiming to open the "black-box" in DNNs. The attention mechanism is applied to explain the DNNs in both single-device and multi-device conditions. Finally, to cope with the third challenge (i. e., robustness of DNNs) caused by adversarial attacks, both the robustness of deep learning models and the transferability of adversarial attacks are analysed and further improved in Section 3.3.

3.1 Deep Representation Learning

When training a DNN model, the inputs are either the original audio signals or the spectrum representations. In recent years, spectrum representations, containing the time and frequency information of audio signals, have shown good potential in audio signal classification tasks, such as snore sound classification [22] and ASC [91]. Inspired by the CNNs' successful applications in image processing, e. g., image classification [92], image captioning [30], etc, they have been employed to analyse the spectrum representations of audio signals effectively [22]. CNNs have a strong capability of mapping the inputs to the predicted labels using highly nonlinear convolutional layers. Consequently, CNNs can learn highly abstract representations from the inputs. However, training CNNs is limited by the over-fitting problem caused by small-scale data sets [93]. Furthermore, the time-consuming training procedure with high computational costs could be a difficulty to run deep learning models on mobile devices [94].

In this regard, to overcome these limitations and difficulties, transfer learning using pre-trained CNNs is promising to extract useful deep spectrum representations

from spectrum features instead of training a CNN model from scratch. The pre-trained CNNs on large-scale image datasets are considered in this thesis, since they are more robust to extract high-level representations than those CNNs learnt from small databases. More importantly, transfer learning can effectively save the time of designing CNNs' structure and training CNNs.

To develop transfer learning for extracting deep spectrum representations, novel transfer learning approaches will be introduced to avoid over-fitting on small-scale data sets and save the training costs, as proposed by the author and her collaborators in [95, 1, 3, 2]. In this section, the typical pre-trained CNN topologies will be firstly introduced in Section 3.1.1. Next, the CNN-based transfer learning approaches will be presented in detail in Section 3.1.2. Finally, an approach of fusing the classifiers, each of which processes a type of deep spectrum representation, will be elaborated in Section 3.1.3.

3.1.1 Pre-trained Convolutional Neural Networks

Over the past decades, a set of CNN models have been trained for image classification tasks [96, 97]. Particularly, a large-scale database, ImageNet, was collected for the ImageNet Large Scale Visual Recognition Competition (ILSVRC) challenges [98], and promoted the development of effective CNN structures. This section provides the structures of three CNN models: AlexNet [99], VGG-16, and VGG-19. [100]. All of these three models have achieved good performances on the ImageNet database.

The topologies of the three CNNs are given in Table 3.1. AlexNet, VGG-16, and VGG-19 consist of 8, 16, and 19 layers, respectively. Except for the final three fully connected layers *fc6*, *fc7*, and *fc*, AlexNet contains three local max pooling layers and five convolutional layers with output channel numbers of 96, 256, 384, 384, and 256, and kernel sizes of 11×11, 5×5, 3×3, 3×3, and 3×3. The two VGG models (VGG-16/VGG-19) are constructed by five local max pooling layers and 13 (VGG-16: 2, 2, 3, 3, and 3)/16 (VGG-19: 2, 2, 4, 4, and 4) convolutional layers with output channel numbers of 64, 128, 256, 512, and 512, and a kernel size of 3×3. For the each CNN model, a softmax layer is employed finally to predict the probabilities of $1,000$ image classes for the $1,000$-class classification task in ILSVRC.

Notably, as the three CNNs were proposed to work on the ImageNet database, the inputs of the models are three-channel RGB (red, green, and blue) images with a fixed size of $3 \times 224 \times 224$ defined by the hyperparameters in these models. To process one-channel spectrum representations with these introduced pre-trained CNNs, the spectrum representations can be resized and saved with a colourful colour map, such as *viridis*, which varies from blue (low range) to green (mid range) to yellow (upper range). Then, the scaled image-like spectrum representations are available to be fed into the pre-trained CNNs. Finally, the high-level features can be extracted from the activations of the fully connected layers (either *fc6* or *fc7*).

Table 3.1: Structures of the three CNNs: AlexNet, VGG-16, and VGG-19. The "conv" stands for the convolutional layer, "k" is the kernel size, and "ch" means the number of output channels.

<table>
<tr><th>AlexNet</th><th>VGG-16</th><th>VGG-19</th></tr>
<tr><td colspan="3">Input: images</td></tr>
<tr><td>1×conv, k: 11, ch: 96</td><td>2×conv, k: 3, ch: 64</td><td>2×conv, k: 3, c: 64</td></tr>
<tr><td colspan="3">Local max pooling</td></tr>
<tr><td>1×conv, k: 5, ch: 256</td><td>2×conv, k: 3, ch: 128</td><td>2×conv, k: 3, ch: 128</td></tr>
<tr><td colspan="3">Local max pooling</td></tr>
<tr><td rowspan="2">1×conv, k: 3, ch: 384</td><td>3×conv, k: 3, ch: 256</td><td>4×conv, k: 3, ch: 256</td></tr>
<tr><td colspan="2">Local max pooling</td></tr>
<tr><td rowspan="2">1×conv, k: 3, ch: 384</td><td>3×conv, k: 3, ch: 512</td><td>4×conv, k: 3, ch: 512</td></tr>
<tr><td colspan="2">Local max pooling</td></tr>
<tr><td>1×conv, k: 3, ch: 256</td><td>3×conv, k: 3, ch: 512</td><td>4×conv, k: 3, ch: 512</td></tr>
<tr><td colspan="3">Local max pooling</td></tr>
<tr><td colspan="3">Fully connected layer fc6, neurons: 4096</td></tr>
<tr><td colspan="3">Fully connected layer fc7, neurons: 4096</td></tr>
<tr><td colspan="3">Fully connected layer fc, neurons: 1000</td></tr>
<tr><td colspan="3">Output: softmax</td></tr>
</table>

3.1.2 CNN-based Transfer Learning

In the framework of transfer learning (cf. Figure 3.1), pre-trained CNNs with the prior knowledge learnt from a large-scale image database are employed for audio signal classification tasks. The transfer learning in our work contains two major branches: one branch is feeding the extracted high-level representations from a pre-trained CNN model into a classifier, which is either a traditional machine learning model or a deep learning model; the other branch is fine-tuning a pre-trained CNN model by adapting partial or the whole CNN parameters. In the following, the first branch, which aims to extract high-level representations will be introduced in Section 3.1.2.1, and the second branch, which focuses on the fine-tuning procedures, will then be given in Section 3.1.2.2.

3.1.2.1 High-level Representation Extraction

With the extracted representations by pre-trained CNNs, a set of classifiers can be used to map the high-level representations into classes. Next, both procedures for constructing traditional machine learning and deep learning classifiers will be introduced, respectively.

Classification using traditional machine learning. In Section 2.2.2, SVMs were introduced as the major feature-based machine learning classifier in this thesis.

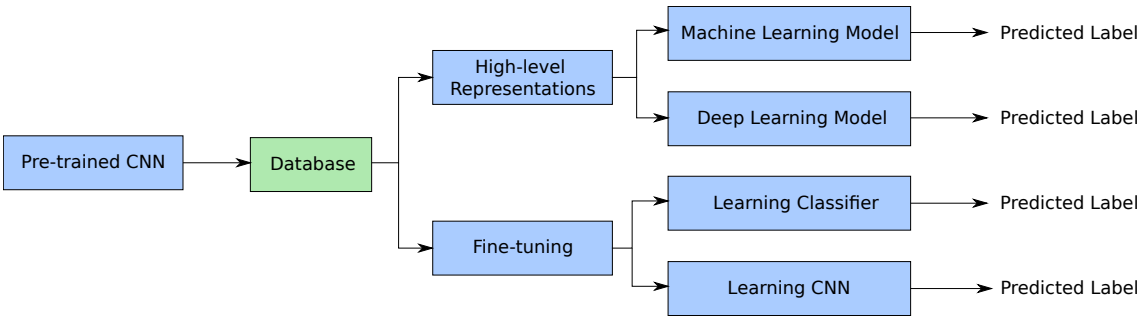


Figure 3.1: An overview of the CNN-based transfer learning framework.

Therefore, SVMs are employed herein to classify the deep spectrum representations. Depending on the segmentation procedure of the audio signals, the high-level representations can be either fed into an SVM model directly when they are extracted from every complete audio sample, or processed by BoAW first and then fed into an SVM model when they are from segmented audio samples. Specifically, if each audio signal is cut into segments, the BoAW features are firstly extracted through summarising the LLD-like deep representations into a feature vector for each audio sample. Afterwards, an SVM model is optimised to predict the probabilities of every class with the BoAW feature vectors as the input.

Classification using Deep Neural Networks. Despite the good performances achieved by traditional machine learning models [52], DNNs can further model non-linear transformations between the high-level representations and the predicted labels. Particularly, in the work of the author and her colleagues [1], the (B)RNNs were proposed to process the deep representations extracted from the fully connected layers of pre-trained CNNs. The purpose of using (B)RNNs is to learn the sequential representations from the spatial features produced by CNNs. As shown in the pipeline depicted in Figure 3.2, the audio samples are firstly cut into shorter audio segments, and the spectrum representations are then extracted from every audio segment. Next, the deep representations are extracted from each spectrum representation by a pre-trained CNN model. Furthermore, these deep representations are arranged according to the time steps, and fed into the (B)RNNs to learn deep sequential representations. Finally, the (B)RNNs are followed by a softmax layer for classification.

3.1.2.2 Fine-tuning Procedure

As mentioned before, training CNNs from scratch on small datasets suffers from over-fitting, since training a CNN model requires a large amount of labelled data to ensure the convergence. Therefore, training CNNs on small-scale datasets is a challenging task. Using pre-trained CNN models from large-scale datasets is promising to solve the over-fitting problem. However, a pre-trained CNN model on a large-scale

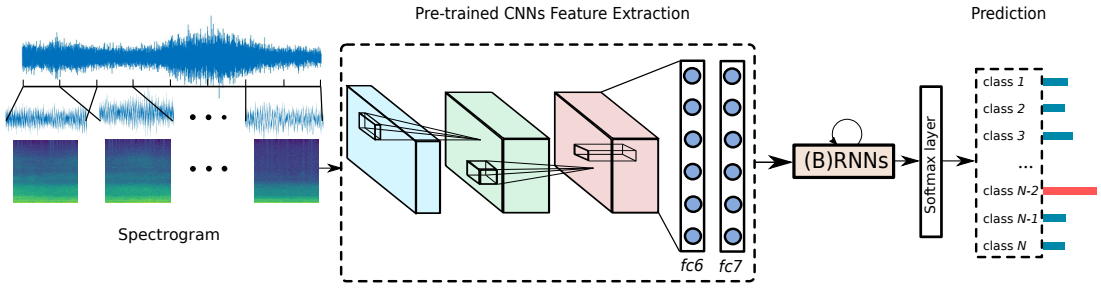


Figure 3.2: The framework of the approach which employs (B)RNNs to classify the extracted high-level representations from the pre-trained CNNs.

dataset mostly cannot work well on a small dataset because of the substantial differences between the two datasets. Therefore, fine-tuning was investigated to solve the problem caused by the data differences [101]. The author and her collaborators also attempted to fine-tune the pre-trained CNNs in [3], in order to improve the performance of training an additional classifier on the extracted deep representations from pre-trained CNNs. Two kinds of fine-tuning procedures in [3] will be introduced as below.

Learning Classifier of CNNs. To construct a CNN model for a classification task, the number of the outputted neurons from the final *fc* layer is set as the number of classes. During the training of the CNNs, the classifier (*fc7* and *fc*) is updated, whereas the convolutional layers and *fc6* are frozen by setting the corresponding parameters of the pre-trained CNN model untrainable. Learning the classifier of the CNNs can solve the data difference problem, yet, it is still challenging for only training the classifier to perform well when there is a big difference between two data resources, such as natural images and spectrum representations.

Learning CNNs. The aim of learning CNNs is to train a more suitable CNN model on a database than that trained by learning the classifier only. The last fully connected layer *fc*, herein, is also replaced by a new one to generate neurons whose number is equal to the number of classes. To better suit the database, all parameters of the pre-trained CNN model are updated in the training procedure, instead of setting the parameters of the classifier trainable only. The obtained CNN model is called *learnt CNNs*.

Additionally, to compare the pre-trained CNNs and the *learnt CNNs*, the high-level representations extracted from the *learnt CNNs* are fed into a traditional machine learning classifier. This approach is based on the assumption that the extracted representations from the learnt CNNs are potentially more helpful than those from a pre-trained CNN model, in targeted classification tasks.

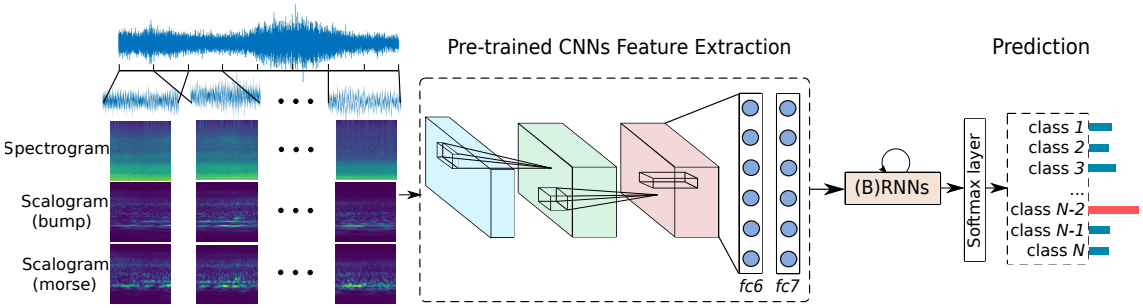


Figure 3.3: An overview of the transfer learning for multiple deep features extraction. In this framework, spectrograms and scalograms (*bump* and *morse*) are firstly generated from the segmented audio waveforms. All spectrum representations are then fed into the pre-trained CNNs, from which high-level features are further extracted at a subsequent fully connected layer. Finally, when the deep features are the input of the (B)RNNs followed by a softmax layer, the final predictions are obtained by fusing the predictions of multiple (B)RNN models via the MSV strategy.

3.1.3 Transfer Learning for Multiple Deep Features Extraction

Apart from extracting deep spectrum representations in Section 3.1.2.1, transfer learning was proposed to extract multiple types of deep representations in the research work of the author and her colleagues [1]. The aim of extracting multiple deep representations is to improve the performance of a single one. As shown in Figure 3.3, three kinds of time-frequency representations are fed into the pre-trained CNNs for deep features extraction. The extracted three sets of the deep features are then used as the input of the (B)RNNs for a classification task. Finally, for each audio sample, three predictions are produced by the three learnt (B)RNN models. At this point, to obtain the final prediction for each audio sample, the decision-level (i. e., late) fusion is applied. Since Margin Sampling Value (MSV) [102] has been effectively used to fuse multiple machine learning models [103], we apply the MSV to fuse the predictions from the three (B)RNNs.

The MSV strategy measures the confidence of each model, and the final decision comes from the model with the highest confidence. For an audio sample, each single-model prediction is defined by $\{\tilde{y}_j, p_j\}$, $j = 1, ..., N_c$, where $\tilde{y}_j$ denotes the predicted label, p_j is the probabilities of every class, and N_c is the number of the classifiers. The final prediction $\tilde{y}$ is selected using MSV:

$$\tilde{y} = \{\tilde{y}_j \mid \max_{j=1}^{N_c} (p_j^1 - p_j^2)\}, \tag{3.1}$$

where p_j^1 and p_j^2 are the first and second highest probabilities produced by the j-th classifier. Herein, the difference of the probabilities $(p_j^1 - p_j^2)$ is considered as the

confidence of the j-th model for the predicted label $\tilde{y}_j$. At this point, the label corresponding to the maximum confidence value is selected as the final prediction.

3.2 Explainable Deep Learning Models

In applications of deep learning, to construct the trust between humans and machines, it is essential to train interpretable deep learning models referred as explainable AI [104]. Several approaches have been explored to achieve explainable AI, such as identifying the elements which could provide key information, constructing data structures to describe the logic inside DNNs, and visualising DNNs [104]. For example, Local Interpretable Model-agnostic Explanations (LIME) was proposed to explain which input feature is important towards an outcome in [105]. In [29], a decision tree was trained to semantically explain the reason of the predicted class for an image made by a pre-trained CNN model, through interpreting which parts of an image were used for the prediction and how much they contributed to the prediction. An explanatory graph was learnt by disentangling multiple part patterns from each filter in a pre-trained CNN model [106]. Regarding explaining DNNs through visualisation techniques in [107], the descriptive image regions were visualised through the guided backpropagation. A CNN model was trained to reconstruct an image from a layer of CNNs in [108]. Specifically, a more realistic image can be reconstructed from lower layers, and the colours/rough contours can be reconstructed from higher layers. However, these above methods require either an additional model or an extra approach.

To learn an explainable model in a single training procedure, the study in [109] trained an interpretable CNN model, where a high convolutional layer had multiple filters representing specific object parts. A spatial and channel-wise attention mechanism was applied to a CNN model, aiming to achieve a visual image captioning task in [30]. An attention mechanism was used to identify the important temporal parts for an audio tagging task in [31]. Inspired by these successful research works on the attention mechanism, the author and her collaborators focused on investigating visualising neural networks by training attention-based models in [110, 111, 112, 113], which will be discussed in this section.

In the following, the basic attention mechanism will be firstly introduced in Section 3.2.1. The approaches of visualising DNNs using an attention mechanism, including localising both the important temporal and the time-frequency parts of high-level representations, will be then presented in Section 3.2.2. Next, an atrous CNN model with attention, aiming to visualise high-resolution feature maps of attention-based CNNs, will be explained in Section 3.2.3. Finally, the atrous CNN model with an attention mechanism will be extended to visualise CNNs in multi-device conditions through a novel conditional training approach (cf. Section 3.2.4).

3.2.1 Attention Mechanism

In early studies, the attention mechanism was mostly applied in speech-to-text tasks [114, 115, 116]. In a speech-to-text task, an encoder-decoder neural network is designed and trained. The encoder, an RNN model, reads the speech signals into a context vector c, whereas the decoder is trained to predict the word given the vector c. Typically, the vector c is equal to the representation at the final time step of the hidden state, which is outputted by the final layer. However, selecting the representation at the last time step may lose the context information at other potential time steps. In this regard, the attention mechanism was proposed to identify which context information the model should pay attention to [114]. While the hidden state at the time step t is defined as h_t and the total number of time steps is T, the i-th word is determined by the context vector c_i, the hidden state s_{i-1} of the decoder, and the $(i-1)$-th word. The context vector c_i is calculated by

$$c_i = \sum_{t=1}^{T} \rho_{it} h_t, \tag{3.2}$$

$$\rho_{it} = \frac{\exp\left(e_{it}\right)}{\sum_{k=1}^{T} \exp\left(e_{ik}\right)}, \tag{3.3}$$

$$e_{it} = f_a(h_t, s_{i-1}), \tag{3.4}$$

where ρ_{it} denotes the normalised evaluation score, and e_{it} is the evaluation score computed by an alignment model $f_a(h_t, s_{i-1})$, which aims to evaluate a matching score between the hidden state h_t and the decoder's hidden state s_{i-1}. During the training procedure of the speech-to-text model, the alignment model is also involved to be updated through backpropagation.

3.2.2 Visualising Deep Neural Networks via Attention

With the inspiration of the attention mechanism in speech-to-text tasks, it is promising to apply an attention mechanism to classification tasks due to its capability of evaluating matching scores between the hidden states and the outputs. Therefore, the attention-based approaches, proposed by the author and her colleagues in [110, 113], will be presented to compute how much a neural network pays attention to each unit for audio classification tasks with an attention mechanism at the decision-level. In a similar way to the attention mechanism in a speech-to-text task, the attention tensor herein is computed to evaluate the matching score between the neurons and the predicted label. Attention at the frame-level working on RNNs will be introduced in Section 3.2.2.1, and attention at the time-frequency level working on CNNs will be discussed in Section 3.2.2.2.

In addition to the goal of visualising neural networks, the attention mechanism will be analysed from the perspective of the performance in the next two subsections (Sections 3.2.2.1 and 3.2.2.2). In an RNN structure, an attention mechanism summarises each two-dimensional hidden state (time frame and feature vector) into a vector. While in a CNN structure, the attention mechanism reduces the three-dimensional feature maps into a vector. The above procedure reduces the dimensions of hidden states and is referred as *global pooling* (cf. Section 2.3.2.2). In classification tasks, global pooling has been often used to follow recurrent layers or convolutional layers [117]. However, typical global pooling layers, e. g., global max pooling and global average pooling, can not achieve the optimal performance, since they either overestimate or underestimate the contribution of each unit in the high-level representations. Hence, in the next two sections (Section 3.2.2.1 and Section 3.2.2.2), the attention mechanism for classification tasks will be also compared to the traditional global pooling methods.

3.2.2.1 Attention at Frame Level

In an RNN classification model, to reduce dimensions of the hidden states from the final recurrent layer, the simplest way is selecting the representation h_T at the final time step (cf. Section 3.2.1). However, the feature vector at the final time step may lose helpful context information, resulting in sub-optimal classification performance. In this regard, three global pooling methods will be presented: the global max pooling, the global average pooling, and the proposed global attention pooling [113].

Global max pooling selects the maximum value from the hidden state at each fixed frequency position among all time steps. Then, the selected maximum value is considered as the unit at the corresponding position of the output vector. Global max pooling is defined by

$$r_j = \max_{1 \leq t \leq T} h_{jt}, \tag{3.5}$$

where r_j is the j-th unit of the vector r. Global max pooling can select the most useful units for classification, but underestimates the contribution of the other units in the hidden state to the prediction.

Global average pooling calculates the average value of the hidden state at each frequency position among all of the time steps to better utilise all units in the hidden state. The j-th unit of the vector r is obtained by

$$r_j = \frac{1}{T} \sum_{t=1}^{T} h_{jt}. \tag{3.6}$$

Although global average pooling considers the contribution of the hidden state at all time steps to classification, it always underestimates helpful units and overestimates other useless ones.

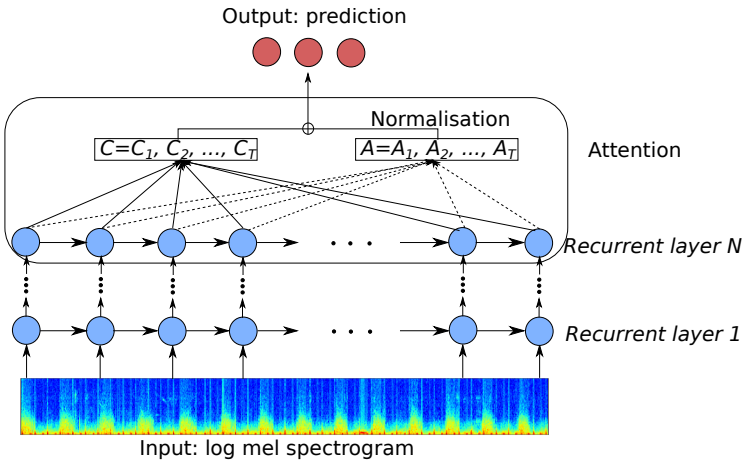


Figure 3.4: An RNN structure with an attention mechanism. The input, which is the log mel spectrogram of an audio signal, is fed into the RNN model with N recurrent layers and an attention mechanism. $C = C_1, C_2, ..., C_T$ is the classification tensor, and $A = A_1, A_2, ..., A_T$ stands for the attention tensor.

Global attention pooling reasonably evaluates the contribution of a hidden state at each time frame to a classification task, therefore, it can overcome the above shortcomings of global max pooling and global average pooling. An RNN model with a global attention pooling at the frame level is depicted in Figure 3.4. The attention mechanism has two branches: the attention branch computes the contribution of a hidden state at each time step to a classification task; the classification branch aims to achieve the classification procedure. When F is the length of hidden state at each time frame and K is the number of classes, both branches convert an $F \times T$ hidden state into a tensor A (in the attention branch) or C (in the classification branch) with a size of $K \times T$, by a one-dimensional convolutional layer with a kernel size of one and an output channel number of K. In the attention branch, the one-dimensional convolutional layer is followed by an activation function (softmax or sigmoid) to rectify the values in the obtained tensor A into $[0, 1]$, representing the weighted values of each time frame. Then, the tensor A^*, obtained by applying the activation function to A, is normalised by

$$P_{kt} = \frac{A^*_{kt}}{\sum_{t=1}^{T} A^*_{kt}}, \tag{3.7}$$

where P is the probability tensor. Finally, the predicted probability p_k of the k-th class is obtained by

$$p_k = \sum_{t=1}^{T} C_{kt} \odot P_{kt}, \tag{3.8}$$

where $\odot$ means the element-wise multiplication. Notably, to achieve classification tasks, a softmax or log-softmax function is employed after obtaining the classification tensor. Such a function can be applied to either the classification tensor C or the probability p.

To this end, the attention mechanism has the potential to improve the performance of RNNs with global max pooling or global average pooling, as it evaluates the contribution of all time steps to the predictions. Furthermore, the attention mechanism at the frame level can explain the contribution of each time step through visualising the attention tensor, producing an understandable and interpretable RNN model.

3.2.2.2 Attention at Time-frequency Level

In CNN models, it is essential to reduce the three-dimensional feature map (channel, time frame, feature vector) into a vector for classification. A direct way to achieve the reduction of dimensions is to flatten the feature map into a vector. However, the flattening may lead to a sub-optimisation, as it might retain redundant information. Therefore, to produce a smaller vector with less redundant information than that produced by flattening, the global max pooling, global average pooling, global Region-of-Interest (ROI) pooling in CNNs will be introduced in the following, and the global attention pooling will be then presented and discussed.

Global max pooling selects the maximum value across all of the time-frequency bins of a feature map at each channel. Mathematically, given an $H \times P_w \times Q_w$ feature map $\boldsymbol{h}$, where H is the channel number and $P_w \times Q_w$ is the size of each sub-feature-map, global max pooling is defined by

$$r_j = \max_{1 \le p \le P_w} \max_{1 \le q \le Q_w} \boldsymbol{h}_{jpq}, \tag{3.9}$$

where r_j is the j-th unit of the output vector. Through global max pooling, the three-dimensional feature map $\boldsymbol{h}$ is converted into a vector r with a length of H. However, since global max pooling only considers the maximum value in each sub-feature-map, it may underestimate the contribution of other time-frequency bins with smaller values than the maximum one to a classification task.

Global average pooling calculates the average value of each sub-feature-map, with considering that each time-frequency bin has the same contribution to classification. The summarised vector using global average pooling is calculated by

$$r_j = \frac{1}{P_w} \frac{1}{Q_w} \sum_{p=1}^{P_w} \sum_{q=1}^{Q_w} \boldsymbol{h}_{jpq}. \tag{3.10}$$

However, global average pooling is also not suitable to estimate the contribution of each unit, as it overestimates or underestimates the time-frequency units in a feature map.

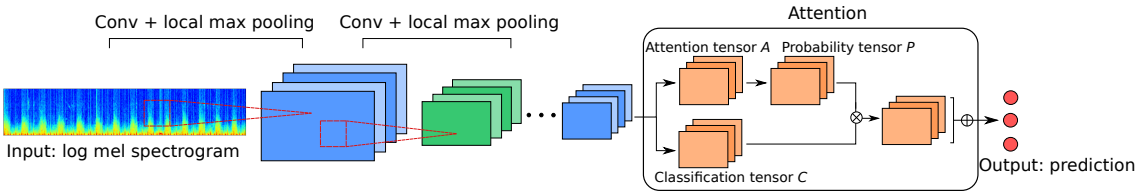


Figure 3.5: A CNN structure with an attention mechanism. The log mel spectrogram of an audio signal is fed into the CNN model, which consists of several convolutional layers, local max pooling layers, and an attention mechanism.

Global ROI pooling, which retains more useful time-frequency units during the dimension reduction than those in global max/average pooling, can be helpful to overcome the above shortcomings of them [118]. Global ROI pooling splits each sub-feature-map into smaller areas with a size of $P_s \times Q_s$, and then employs global max pooling to these small areas. Finally, the produced tensor from these areas by global max pooling is flattened into a vector for classification. Additionally, global ROI pooling can be combined with an attention mechanism. During the combination of ROI and attention, the output of global ROI pooling is fed into a global attention pooling layer, in order to evaluate the contribution of each unit in the tensor from global ROI pooling. However, both global ROI pooling and ROI with attention cannot estimate the contribution of all units in a feature map.

Global attention pooling in CNNs reduces the dimensions of a feature map through estimating the contribution of each time-frequency unit. Therefore, global attention pooling is promising in improving the performance of CNNs with the above global pooling methods. In Figure 3.5, a CNN structure with a global attention pooling layer for classification is depicted. Following a set of convolutional layers and local max pooling layers, global attention pooling consists of two branches: an attention branch and a classification branch. The attention branch has a two-dimensional convolutional layer with a kernel size of 1×1 and an output channel number equal to the classes number K, producing an attention tensor A. Next, the convolutional layer is followed by an activation function (softmax or sigmoid), generating a tensor A^*, in which the values are in $[0, 1]$. The tensor A^* is then normalised by

$$P_{kpq} = \frac{A^*_{kpq}}{\sum_{p=1}^{P_w} \sum_{q=1}^{Q_w} A^*_{kpq}}, \tag{3.11}$$

where P is the probability tensor. In the classification branch, an additional two-dimensional convolutional layer with a kernel size of 1×1 is used to convert the feature map into a new one C with the channel number of K. The produced classification tensor C is then multiplied with P to predict the probability of each class

using

$$p_k = \sum_{p=1}^{P_w} \sum_{q=1}^{Q_w} C_{kpq} \odot P_{kpq}. \tag{3.12}$$

Herein, similar to the attention mechanism at the frame level, a softmax or a log-softmax function is used to work on C or p to achieve a classification task.

The global attention pooling at the time-frequency level can evaluate the contribution of each time-frequency bin to classification, thereby achieving more optimal predictions. Furthermore, visualising the attention tensor is promising in presenting the effectiveness of all time-frequency bins to the final prediction, and help humans understand a CNN model.

3.2.3 Visualising Atrous CNNs via Attention

In the presented CNNs in Section 3.2.2.2, the convolutional layers are mostly followed by local pooling layers or controlled by strides, which are the moving steps of the convolutional kernels. Both local pooling layers and strides can reduce the computational cost via downsampling the feature maps. However, these two downsampling procedures result in low-resolution attention tensors. These procedures may lose the time-frequency details, which is a limitation of analysing the potential contribution of each time-frequency bin in the spectrum representations to the predictions. In this regard, high-resolution attention tensors are needed to better visualise and understand CNNs. In the following, let us now move to an attention-based atrous CNN model which is used to produce high-resolution feature maps and firstly proposed by the author and her colleagues [111].

As both the local pooling layers and strides can reduce the size of feature maps, local pooling layers are considered only. To generate high-resolution attention tensors, the simplest way is to remove the local pooling layers. However, CNNs without local pooling have more convolutional operations than CNNs with local pooling, so it may result in sub-optimisation and low performances. Recently, a set of approaches have been proposed to produce high-resolution feature maps from the final convolutional layer in image processing tasks. For instance, encoder-decoder CNNs, where an encoder downsamples the feature maps and a decoder upsamples the feature maps, were trained to learn internal representations [119]. Fully Convolutional Networks (FCNs) were employed to upsample the low-resolution feature maps using a set of deconvolutional layers [120]. Deconvolutional layers were also used in Generative Adversarial Networks (GANs) to generate high-resolution synthetic images, in order to improve the classification performance by augmenting the training data [121]. However, these above image-based approaches mostly require strongly labelled data, where each pixel of an input image is annotated. The datasets for classification tasks are usually weakly labelled, as each audio signal is annotated with one label, rather than each frame is annotated with a separate label. Hence,

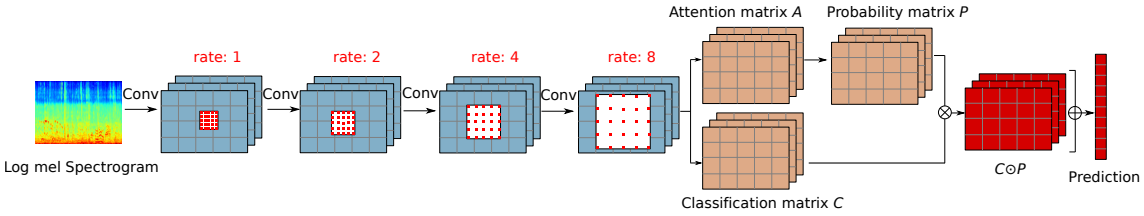


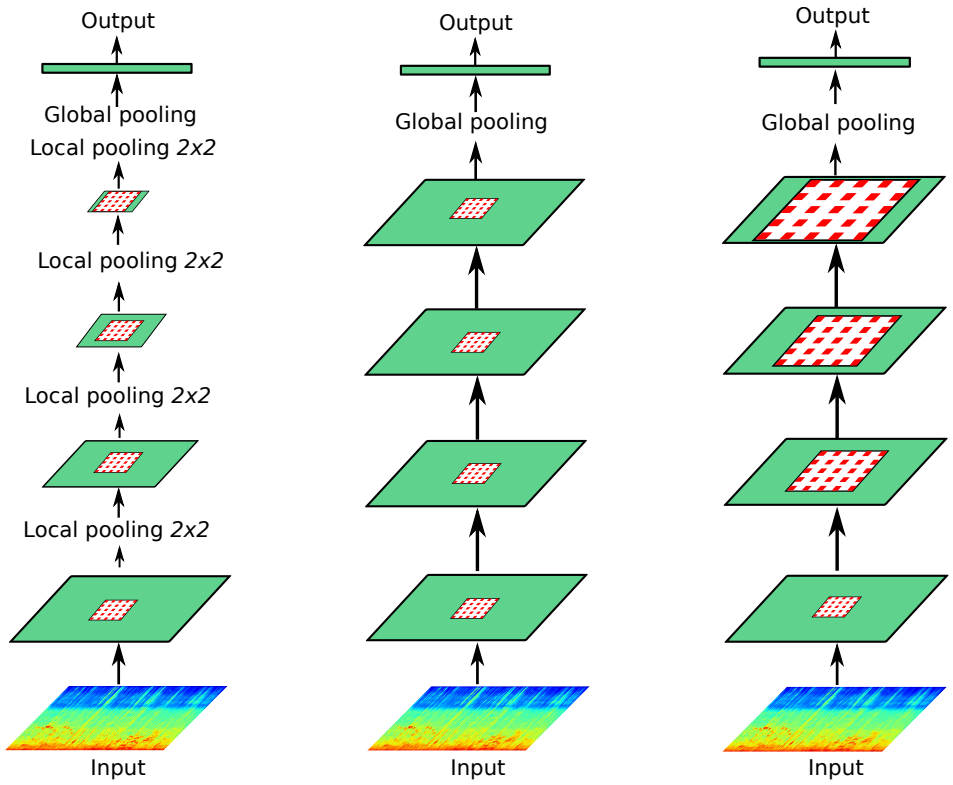
Figure 3.6: An overview of an atrous CNN model with an attention mechanism. The dilation rates in the first four convolutional layers of the atrous CNNs are set to 1, 2, 4, and 8. Then, the four convolutional layers are followed by an attention mechanism.

it is difficult to generate high-resolution attention tensors using these image-based approaches. In a recent study [122], atrous CNNs were proposed to retain the size of feature maps at each convolutional layer by dilated convolutions, instead of using downsampling and upsampling. Inspired by this work [122], atrous CNNs with an attention mechanism will be discussed to visualise high-resolution attention tensors in this section.

An overview of the atrous CNNs with attention is depicted in Figure 3.6. An extracted log mel spectrogram is fed into the CNN model with four convolutional layers and a global attention pooling layer. In each convolutional layer, the kernel size is controlled by a dilation rate (1, 2, 4, and 8 in the four convolutional layers), so that the size of the feature maps is the same as that of the original log mel spectrogram. Finally, a global attention pooling layer can help visualise the feature maps from the internal CNN layers with a high resolution. Next, to further analyse the effectiveness of the dilated convolution, atrous CNNs will be compared to the other two CNN structures: CNNs with local pooling and CNNs without local pooling.

CNNs with local pooling employ local pooling layers with a kernel size of 2×2 following each convolutional layer, as shown in figure 3.7(a). Local pooling layers can help CNNs extract time-frequency shift-invariant features, and accelerate the convergence speed by reducing the computational cost [123]. However, as aforementioned, local pooling layers lead to low-resolution feature maps, which are a limitation of visualising high-resolution attention tensors in the CNNs.

CNNs without local pooling is the simplest way to retain the size of the feature maps at each convolutional layer through removing the local pooling layers, as shown in Figure 3.7(b). However, training CNNs without local pooling is time-consuming due to more convolutional operations than those in CNNs with local pooling, and may lead to worse classification performance. The reason for the underperformance is assumed that, the sizes of the receptive fields in CNNs without local pooling are mostly smaller than those in CNNs with local pooling. Herein, the receptive field denotes the size of the input area which affects a pixel in the output of a convolutional layer. As shown in Figure 3.8(a), the size of the receptive field is

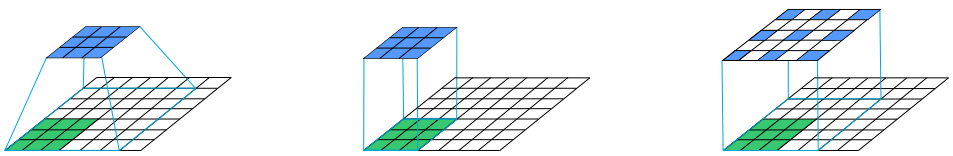


(a) CNNs with local pooling.

(b) CNNs without local pooling.

(c) Atrous CNNs.

Figure 3.7: The three CNN structures. The red grids in each convolutional layer denotes the convolutional kernels.



(a) CNNs with local pooling.

(b) CNNs without local pooling.

(c) Atrous CNNs.

Figure 3.8: The receptive fields of the three CNN structures. The white grids are the input, the green grids are the receptive field of a convolutional kernel in the first convolutional layer, and the blue grids denote the receptive field of the kernel in the second convolutional layer.

$s_r \times s_r$ at the first convolutional layer, and is $2s_r \times 2s_r$ at the second convolutional layer due to local pooling. Consequently, at the n-th convolutional layer, the size of receptive field is $2^{n-1}s_r \times 2^{n-1}s_r$. Compared to CNNs with local pooling, CNNs without local pooling have the size of the receptive field $s_r \times s_r$ at both the first and the second convolutional layers (cf. Figure 3.8(b)). At the n-th convolutional layer, the size of the receptive field is also $s_r \times s_r$ in CNNs without local pooling. Hence, a big difference exists between the receptive fields of these two CNN architectures: the size of the receptive field in CNNs with local pooling increases exponentially with the number of layers, but it stays the same at all of the layers in CNNs without local pooling.

Atrous CNNs aim to preserve the size of the feature maps using the dilated convolutional kernels (cf. Figure 3.7(c)). Different from the convolutional kernels which are used in CNNs with/without pooling, a dilated kernel is sparse due to the holes between each two kernel bins. The size of the holes inside a dilated kernel is determined by the dilation rate, and the hole size affects the size of the receptive field. The size of the receptive field becomes larger when the dilation rate is increased. For example, the size of the receptive field is $s_r \times s_r$ at the first convolutional layer when the dilation rate is set to 1, and is $(2s_r - 1) \times (2s_r - 1)$ at the second convolutional layer when the dilation rate is set to 2. Furthermore, at the n-th convolutional layer, the size of the receptive field is $(2^{n-1}(s_r - 1) + 1) \times (2^{n-1}(s_r - 1) + 1)$ when the dilation rate is equal to 2^{n-1}. Therefore, in atrous CNNs, the increasing trend of the size of the receptive field is approximately exponential as the dilation rate increases exponentially.

3.2.4 Visualising Conditional Atrous CNNs via Attention

Because of the development of audio recording devices, processing a multi-device dataset has been essential in the applications of audio signal classification in the daily life [124]. Audio signals are often recorded with different devices, such as professional sound recording devices [125] and mobile devices [126]. The qualities of audio data, such as sampling rate, amplitude, frequency responses, and data distributions, vary due to various characteristics of audio recording devices [127]. Therefore, deep learning models trained on each single-device data set could be distinct from each other due to the mismatch in audio qualities. As proposed by the author and her colleagues in [112], conditional training will be introduced as a novel framework to train a CNN model on a multi-device dataset.

To process a multi-device dataset, training a separate model on each single-device data is time-consuming. To address the above constraints, supervised domain adaption transfers the model knowledge from a source database to a target one [128]. Yet, it is also time-consuming to train models before and after transferring the learnt knowledge. Although training a robust deep learning model on a multi-device audio dataset is challenging due to various data distributions among single-device

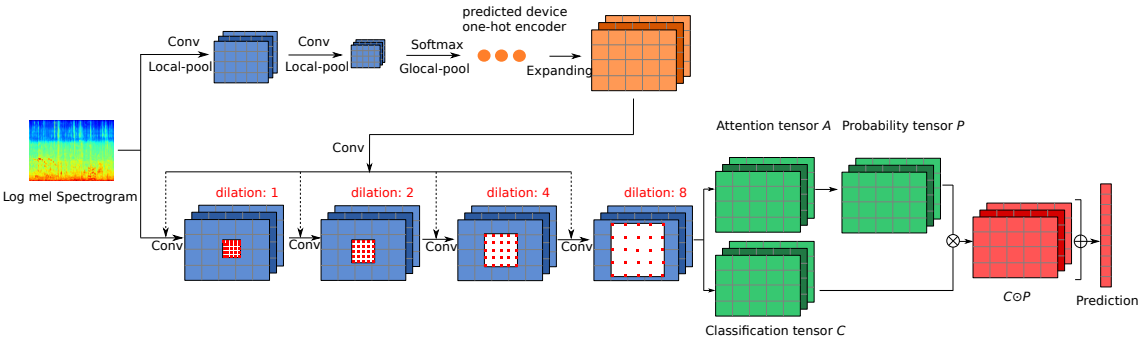


Figure 3.9: The proposed CAA-Net in the multi-task conditional training framework. The top branch is a CNN model which predicts the device information, and the bottom branch is an atrous CNN model with attention for an audio signal classification task.

datasets [129], a set of studies have investigated to train a model on a multi-device audio dataset. For example, in [124], a joint model was trained on a multi-device dataset, and learnt common parameters from each single-device dataset. A joint training approach was proposed to process multi-source data by sharing the context information in [130]. Therefore, it is helpful to reduce the computational burden through jointly training a single model on a multi-device dataset. However, jointly training a model is difficult to distinguish each single-device dataset, so that improving the performance is hard. To address this difficulty, in [131], CNNs were trained with sharing the parameters of the low-level convolutional layers only, and the parameters of the high-level convolutional layers were learnt separately using a multi-task learning framework. However, more separate high-level convolutional layers might lead to a sub-optimisation [132]. In a study of multi-pose face recognition [133], conditional training was proposed to train CNNs, in which the convolutional filters of each convolutional layer were sparsely activated using a conditional activation function. Similarly, the speaker information was fed into a deep learning model to achieve multi-speaker speech generation in [134, 135]. In [136], the predicted difficulty levels of the samples in a multi-task learning framework were fed back into DNNs for a multi-modal emotion recognition task. Inspired by these approaches [134, 135, 136], a novel approach of conditionally training CNNs with the device information, namely Conditional Atrous CNNs with Attention (CAA-Net), will be discussed. Through the CAA-Net, the learnt attention tensors on each single-device dataset can be visualised with a high resolution.

The CAA-Net is developed under two conditions: i) the device information is the prior knowledge, and ii) the device information is unknown for the trained audio classification model. As the device information is sometimes unknown in practical, the CAA-Net without knowing device information in the multi-task conditional training framework, namely multi-task CAA-Net, is depicted in Figure 3.9. The multi-task

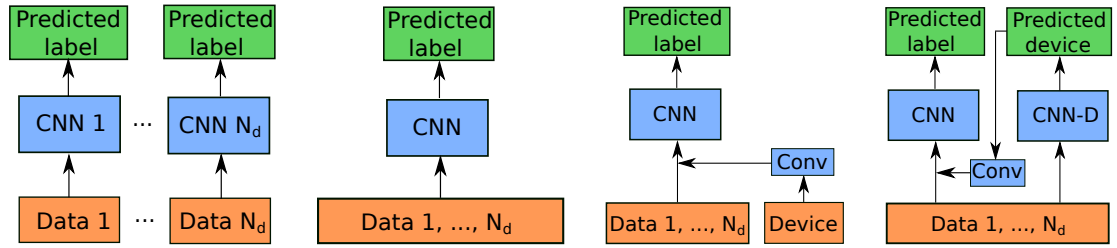
CAA-Net contains two branches: one branch aims to achieve the original audio signal classification task, and the other branch classifies the device information. The first branch is constructed by an atrous CNN model with attention referred to as Section 3.2.3. The second branch is a CNN model, including a set of convolutional layers, local pooling layers, a global max pooling layer, and a fully connected layer. Specifically, in the second branch, each convolutional layer is followed by a local pooling layer. The global max pooling layer is used to summarise the feature maps into a vector for classifying the device information. Then, the predicted device information from the fully connected layer is represented as a one-hot encoder, which is a binary vector converted from the integer value of the device number. Next, the one-hot encoder is expanded into a three-dimensional tensor along the time and frequency axes. Finally, the three-dimensional tensor is fed into a convolutional layer of the CNN model in the first branch to condition this CNN model with the predicted device information.

In the following, to further discuss the details and the effectiveness of the conditional training approaches, the four training strategies will be compared: single-device training, joint training, teacher forcing conditional training (device information is known), and multi-task conditional training (device information is unknown).

Single-device training aims to train an independent CNN model for a separate single-device dataset as part of a multi-device dataset, resulting in N_d CNN models for an N_d-device dataset, as shown in Figure 3.10(a). However, single-device training requires many resources to train multiple models for data from various recording devices, and is not suitable to train a robust CNN model if a single-device dataset is small [137].

Joint training optimises a CNN model with the input of a multi-device dataset (cf. Figure 3.10(b)). All of the parameters are shared among different single-device data. Joint training is based on the assumption that common features could be learnt from all single-device datasets [130]. However, as the CNN parameters are shared among datasets from various devices, it is difficult to learn device-related representations by joint training.

Teacher forcing conditional training, a specific conditional training approach, has been employed in speech recognition [135] and speaker verification [138]. For example, in [139], the ground truth of a previous time step was fed into the current time step in a conditional RNN model. In the teacher forcing conditional training, which is depicted in Figure 3.10(c), the device information is firstly processed by a convolutional layer, and is then fed into the "CNN" model for a classification task. Since the output of the convolutional layer can give weights to each time-frequency bin of the feature map in the "CNN" model, the feature map produced by this convolutional layer is similar to a "mask", filtering out the time-frequency bins which are useless for classifying a single-device data. Therefore, the produced feature map from the convolutional layer which processes the device information herein is called a "mask".



(a) Single-device training. (b) Joint training. (c) Teacher forcing conditional training. (d) Multi-task conditional training.

Figure 3.10: A comparison of the four training strategies. The N_d denotes the number of devices. The "CNN" model aims to achieve the original audio signal classification task, and the "CNN-D" model in (d) multi-task conditional training is used to predict the device information.

To calculate the "mask", the device information represented as an N_d-length one-hot encoder is firstly expanded into an $N_d \times P_w \times Q_w$ tensor, where $P_w \times Q_w$ is the size of the sub-feature-map at a convolutional layer of the "CNN" model. Referring to the linear transformation procedure of the speaker one-hot encoders in a multi-speaker text-to-speech task [134], the expanded one-hot encoder, which represents the device information, is fed into a convolutional layer with a kernel size of 1×1. To further combine the feature map learnt from this convolutional layer and the feature map in the "CNN" model, the output channel number of the convolutional layer is the same as the feature map's channel numbers in the "CNN" model. While the feature map $\boldsymbol{h}^{n-1}$ in the "CNN" model is obtained from the $(n-1)$-th convolutional layer, the combination procedure is defined by

$$\boldsymbol{h}_j^n = \sigma_u\Big(\sum_{i=1}^{C^{n-1}} \boldsymbol{w}_{ij}^n * \boldsymbol{h}_i^{n-1} + \sum_{i=1}^{N_d} \boldsymbol{v}_{ij} * \boldsymbol{e}_i + b_j^n\Big), \tag{3.13}$$

where $\boldsymbol{w}_{ij}^n$ and $\boldsymbol{v}_{ij}$ are the (i,j)-th convolutional kernels, $\boldsymbol{e}_i^{n-1}$ is the i-th channel of the expanded device one-hot encoder, and σ_u denotes a ReLU activation function. Teacher forcing conditional training requires the device information to be the prior knowledge, yet, the device information is sometimes unknown for an audio classification model.

Multi-task conditional training predicts the audio classes and the device information in a single model using a multi-task learning framework to overcome the limitation that the device information has to be prior knowledge in the teacher forcing conditional training. As shown in Figure 3.10(d), an additional CNN model "CNN-D" is trained to predict the device information, which will be expanded to a one-hot encoder. The one-hot encoder is then processed by Equation (3.13). During the training procedure, the loss function $\mathcal{L}_{caa}$ is defined by the weighted sum of the

two CNN models ("CNN" and "CNN-D"),

$$\mathcal{L}_{caa} = \mathcal{L}_s + \lambda \times \mathcal{L}_d, \tag{3.14}$$

where $\mathcal{L}_s$ and $\mathcal{L}_d$ are the loss functions of the audio classification model "CNN" and the "CNN-D" model for device prediction, respectively. The weight factor λ is employed to balance the gradient difference between the two CNN models [140].

3.3 Robust Deep Learning Models Against Adversarial Attacks

In recent years, due to the rapid development of deep learning, developing robust DNNs has been an essential part in their applications. Several studies have shown that deep learning models are vulnerable to adversarial attacks, which are well-designed and imperceptible for humans [33, 141]. Especially, a set of studies have investigated adversarial attacks in the research field of image processing. In [141, 142], fake data was generated to deceive deep learning models for image classification and captioning. CNN models for semantic segmentation and object detection were also deceived by adversarial attacks in [143]. However, only a few works focused on investigating adversarial attacks for audio signal classification models [34], especially in the field of healthcare.

In this regard, the novel approaches proposed by the author and her colleagues in [144, 145] will be presented in this section. Firstly, the adversarial attacks will be introduced in Section 3.3.1. Next, adversarial training will be employed to protect deep learning models against adversarial attacks in Section 3.3.2. Finally, the transferability of adversarial attacks will be improved in Section 3.3.3.

3.3.1 Adversarial Attacks

The procedure of generating adversarial attacks and deceiving a CNN defence model is depicted in Figure 3.11. Given a real data x, a small perturbation (i. e., an additional noise) θ is generated by an attack model, and the adversarial data (i. e., fake data) x' is then calculated by adding the perturbation to the real data. While both x and x' are fed into a defence model (e. g., CNN), the defence model classifies the real data correctly, but wrongly classifies the fake data. The misclassification of the adversarial data is caused by the structure of the defence model. Each layer of the defence model can be simplified into a linear transformation $y = \boldsymbol{wx}$, where y is the labels, and $\boldsymbol{w}$ is the weight parameters. Given the adversarial data $\boldsymbol{x'}$ obtained by $\boldsymbol{x'} = \boldsymbol{x} + \boldsymbol{\theta}$, the linear transformation is then updated to $\boldsymbol{wx'} = \boldsymbol{wx} + \boldsymbol{w\theta}$. Although $\boldsymbol{\theta}$ is very small, the difference between $\boldsymbol{wx'}$ and $\boldsymbol{wx}$ becomes bigger when the model goes deeper, leading to wrong predictions on $\boldsymbol{x'}$. Similarly, the generated

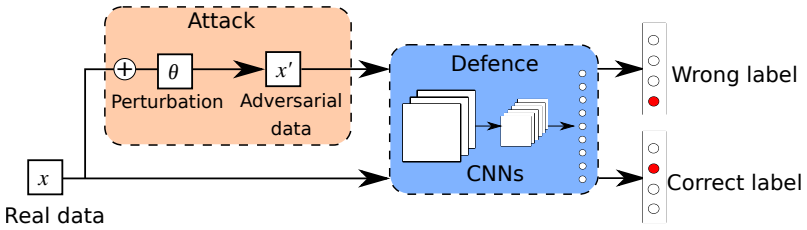


Figure 3.11: An overview of the attack procedure, including generating adversarial data and deceiving a CNN defence model. Through the attack procedure, a defence model wrongly classifies the adversarial data, whereas the real data is classified correctly.

adversarial data can also fool non-linear DNNs (e. g., CNNs), which are mostly used in practice.

Recently, both targeted and non-targeted adversarial attacks were proposed to deceive a deep learning model. Targeted adversarial attacks are generated by giving the attack model a fixed label, so that the defence model classifies all adversarial data into this label [146]. Non-targeted adversarial attacks make a defence model predicting wrong labels without a targeted label [147]. In the following sections, non-targeted adversarial attacks will be generated and defended.

Regarding the prior knowledge for an attack model, adversarial attacks contain white-box attacks and black-box attacks. White-box attacks are obtained when an attack model knows the data and the parameters of the defence model, while black-box attacks are generated when either the data or the defence model's parameters are unknown for the attacker. Both white-box and black-box adversarial attacks will be investigated to generate adversarial spectrum representations. In the following, white-box adversarial attacks will be firstly introduced in Section 3.3.1.1, and a CNN model for generating black-box adversarial attacks will be then described in Section 3.3.1.2.

3.3.1.1 White-box Adversarial Attacks

To generate white-box adversarial attacks, the Fast Gradient Sign Method (FGSM) has shown effectiveness in [148]. The FGSM calculates the gradient of the loss function in a defence model as the perturbations. While the loss function is defined by $\mathcal{L}(\boldsymbol{w}, \boldsymbol{x}, y)$, the gradient $\nabla_{\boldsymbol{x}}\mathcal{L}(\boldsymbol{w}, \boldsymbol{x}, y)$ is obtained during the back propagation procedure. The adversarial data $\boldsymbol{x'}$ is calculated by

$$\boldsymbol{x'} = \boldsymbol{x} + \epsilon_w \times sign(\nabla_{\boldsymbol{x}}\mathcal{L}(\boldsymbol{w}, \boldsymbol{x}, y)), \tag{3.15}$$

$$\boldsymbol{x'} = clip(\boldsymbol{x'}, \boldsymbol{x} - \eta, \boldsymbol{x} + \eta), \tag{3.16}$$

where ϵ_w is a perturbation factor, i. e., a weight value of the adversarial data relative to the real data, and η is a constant hyperparameter for clipping the adversarial data

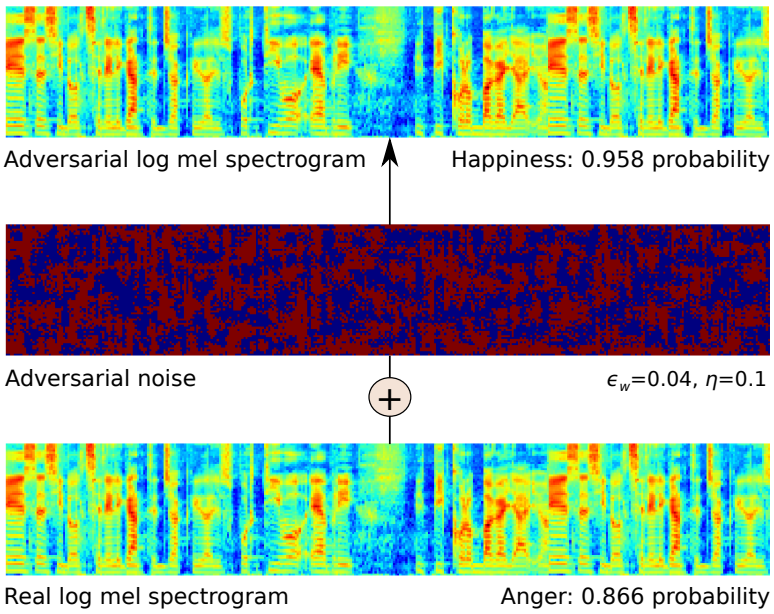


Figure 3.12: An adversarial log mel spectrogram generated from the log mel spectrogram of the speech sample *NP_m_27_ang07b.wav* in the DEMoS database (cf. Section 4.1.3.2).

into the interval of $[\boldsymbol{x}-\eta, \boldsymbol{x}+\eta]$. Finally, the fake data $\boldsymbol{x'}$ with a very small difference (smaller than η) from $\boldsymbol{x}$ is obtained using the FGSM.

Figure 3.12 shows an example of the adversarial log mel spectrograms which are generated by the FGSM. In an SER task, the real log mel spectrogram is correctly classified into *anger* with a probability of 0.866. However, the generated adversarial log mel spectrogram is wrongly classified into *happiness* with a probability of 0.958, although it is difficult to distinguish the adversarial log mel spectrogram from the real log mel spectrogram by human eyes.

3.3.1.2 Black-box Adversarial Attacks

Different from white-box adversarial attacks computed by the parameters of the defence model, black-box adversarial attacks are generated without the parameters of the defence model. With the spectrum representations $\boldsymbol{x}$ as the input, the attack model can produce two-dimensional perturbations $\boldsymbol{\theta}$, where the value of each time-frequency bin is very small. The fake data is then obtained by $\boldsymbol{x'} = \boldsymbol{x} + \boldsymbol{\theta}$.

In the following, the approach of training an atrous CNN model for perturbation generation will be introduced, as proposed by the author and her colleagues in [145]. Herein, the purpose of using an atrous CNN model is to generate adversarial spectrum representations with the same size as the real spectrum representations through setting suitable dilation rates in the convolutional layers (cf. Section 3.2.3).

At this point, the outputs of the atrous CNN model can be added to the real spectrum representations for generating the fake data. During the training procedure of the atrous CNN model, the loss function is minimised to achieve two goals: one goal is to fool the defence model f^D, and the other goal is to generate the fake data which is similar to the real data. The loss function $\mathcal{L}_A$ of the attack model is defined by

$$\mathcal{L}_A = \alpha \mathcal{L}_{cla}(f^D(\boldsymbol{x'})) + (1-\alpha)\mathcal{L}_{MSE}(\boldsymbol{x'}, \boldsymbol{x}), \tag{3.17}$$

$$\mathcal{L}_{cla}(f(x')) = \max(f^D(x')_{gt} - \max(f^D(x')_{other}), 0), \tag{3.18}$$

where $\mathcal{L}_{cla}$ is the loss function leading to misclassification on the fake data, $\mathcal{L}_{MSE}$ is a Mean Squared Error (MSE) loss function to minimize the difference between the fake and real data, and α is a constant factor to balance the gradients of the two loss functions $\mathcal{L}_{cla}$ and $\mathcal{L}_{MSE}$. The loss function $\mathcal{L}_{cla}$ is calculated by the difference between the predicted probability $f^D(x')_{gt}$ on the ground truth gt and the maximum predicted probability $\max(f^D(x')_{other})$ on the other classes. The computation of $\mathcal{L}_{cla}$ is referred to as the Carlini-Wagner loss function [149], which has shown effectiveness in generating image-based adversarial attacks [33, 150]. According to the definition of the loss function in Equations (3.17) and (3.18), black-box adversarial data is obtained by training an attack model.

3.3.2 Adversarial Training

In this section, to protect the defence models against the adversarial attacks, a novel approach of adversarial training will be introduced, as proposed by the author and her colleagues in [144].

In a recent study [34], fake emotional speech signals were generated using an end-to-end model, yet, the approaches of defending against the attacks were not further investigated. To train a robust deep learning model against the adversarial attacks, adversarial training has proven to be effective in previous studies of image processing [142, 143]. Adversarial training is achieved by optimising the defence model with both real and fake data. There are two advantages of adversarial training: i) adversarial training can train a defence model to perform well on classifying the fake data, and ii) through the data augmentation by the fake data, the performance of the defence model on the real data is potentially better. Therefore, adversarial training a robust audio signal classification model with real and fake spectrum representations will be discussed. In the following, the vanilla adversarial training will be firstly introduced, and then a novel similarity-based adversarial training will be proposed.

Vanilla adversarial training aims to optimise the loss functions on the real data and fake data in a single training procedure, so that the defence model can perform well on not only the original spectrum representations, but also the generated fake

spectrum representations. The loss function of the vanilla adversarial training is defined by

$$\mathcal{L}_{vat} = \beta_v \mathcal{L}_s(\boldsymbol{w}, \boldsymbol{x}, y) + (1 - \beta_v)\mathcal{L}_s(\boldsymbol{w}, \boldsymbol{x'}, y), \tag{3.19}$$

where $\mathcal{L}_s(\boldsymbol{w}, \boldsymbol{x}, y)$ and $\mathcal{L}_s(\boldsymbol{w}, \boldsymbol{x'}, y)$ are the loss functions for classifying $\boldsymbol{x}$ and $\boldsymbol{x'}$, respectively, and β_v is a constant factor to balance the gradients of the two loss functions $\mathcal{L}_s(\boldsymbol{w}, \boldsymbol{x}, y)$ and $\mathcal{L}_s(\boldsymbol{w}, \boldsymbol{x'}, y)$.

Similarity-based adversarial training, as an extension of the vanilla adversarial training, is proposed to further minimise the difference of the learnt representations from the real and fake data. Since the difference between the representations of the real and fake data is bigger when the defence model goes deeper, minimising the difference of the high-level representations is promising to help improve the performance on the fake data. The loss function of the similarity-based adversarial training is defined by

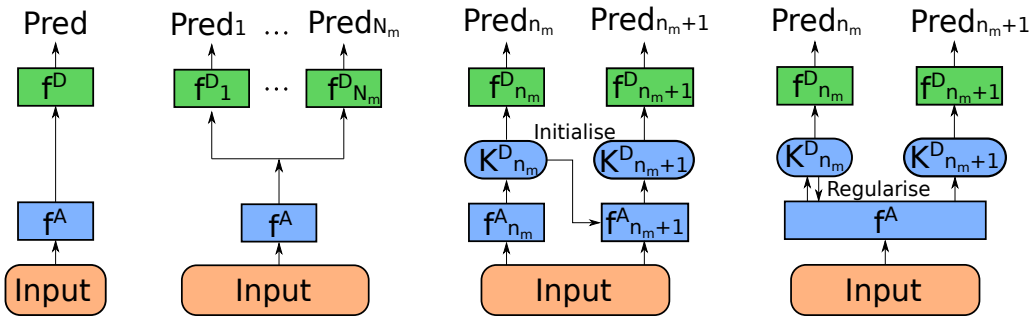
$$\mathcal{L}_{sat} = \zeta_1 \mathcal{L}_s(\boldsymbol{w}, \boldsymbol{x}, y) + \zeta_2 \mathcal{L}_s(\boldsymbol{w}, \boldsymbol{x'}, y) + (1 - \zeta_1 - \zeta_2)\|\boldsymbol{h} - \boldsymbol{h'}\|_2, \tag{3.20}$$

where ζ_1 and ζ_2 are the constant factors to balance the gradients of the three parts ($\mathcal{L}_s(\boldsymbol{w}, \boldsymbol{x}, y)$, $\mathcal{L}_s(\boldsymbol{w}, \boldsymbol{x'}, y)$, and $\|\boldsymbol{h} - \boldsymbol{h'}\|_2$) in the loss function, and $(\boldsymbol{h} - \boldsymbol{h'})$ is the difference between the representation $\boldsymbol{h}$ from the real data and the representation $\boldsymbol{h'}$ from the fake data. The difference of the representations is minimised via an L2 loss function.

3.3.3 Improving Transferability of Adversarial Attacks

The transferability of adversarial attacks, as one of the measurements for adversarial attacks, measures an attack model's capability of transferring among different targeted defence models, i. e., how many defence models can be deceived by an attack model [147, 151]. A highly transferable attacker can not only deceive the already disposed defence models, but also fool a new defence. To save the cost of training an attack model for deceiving each defence and promote the development of more robust defence models, it is essential to enhance the transferability of adversarial attacks. With this in mind, in this section, a novel lifelong learning framework for improving the transferability of adversarial attacks will be presented as investigated by the author and her colleagues in [145].

While single-task learning trains an attack model to deceive a target model, multi-task learning trains an attacker to deceive multiple target models simultaneously for better transferability. In [152], multi-task learning was proposed to train a universal attack model for fooling both semantic segmentation and depth estimation DNNs. Additionally, an ensemble of models and an ensemble of inputs were proposed to train a transferable attack model in [148, 153]. The study in [147] trained an attack model with an ensemble of target models. Random transformations of the original inputs were fed into an attack model in [154]. However, both multi-task



(a) Single-task learning.

(b) Multi-task learning.

(c) Transfer learning.

(d) Lifelong learning.

Figure 3.13: A comparison of the four learning strategies for transferable adversarial attacks. The f^A is an attack model, f^D is a target model, K^D is the learnt knowledge of an attack model to deceive a target model, and N_m denotes the number of the target models.

learning and the ensemble-based methods require a substantial memory space to save multiple target models in a single training procedure. In this regard, transfer learning was employed in [155], and a smooth regularisation was demonstrated in [153] to train a transferable attacker with a smaller memory space. However, the learnt prior knowledge is mostly forgotten in a transfer learning framework, so it is challenging to train a highly transferable attack model. Lifelong learning [156], which enables a model to transfer generalised knowledge across multiple tasks and domains [157], is promising to overcome the problems of multi-task learning and transfer learning. Therefore, the lifelong learning framework for training a transferable attack model will be introduced as follows.

To investigate the effectiveness of lifelong learning, four training strategies of adversarial attacks are depicted in Figure 3.13, including single-task learning, multi-task learning, transfer learning, and lifelong learning. Next, the four training strategies will be analysed and compared.

Single-task learning trains an attack model f^A for deceiving a target model f^D in a training procedure (cf. Figure 3.13(a)). Consequently, N_m attackers are trained to deceive N_m target models. Single-task learning is time-consuming to train multiple independent attack models.

Multi-task learning trains a common attack model to deceive N_m defence models simultaneously (cf. Figure 3.13(b)). However, a large memory space is required to save multiple defence models in multi-task learning.

Transfer learning aims to sequentially train an attacker for each defence model (cf. Figure 3.13(c)). During the transfer learning procedure, the attack model is updated when it is trained to deceive a new defence model. The $(n_m + 1)$-th attack

model $f^A_{n_m+1}$ is fine-tuned with an initialisation of the parameters from $f^A_{n_m}$, $n_m \in [1, N_m - 1]$. However, the learnt knowledge $K^D_{n_m}$ is mostly forgotten during training $f^A_{n_m+1}$.

Lifelong learning trains an attack model for fooling each defence model sequentially. Specifically, lifelong learning transfers the learnt knowledge $K^D_{n_m}$ and enables the attacker remember partial of $K^D_{n_m}$ by regularisation (cf. Figure 3.13(d)). Herein, Elastic Weight Consolidation (EWC) [158] is employed to achieve the lifelong learning framework. EWC aims to help the attack model remember the important information of the learnt knowledge from each target model by giving elastic constraints to the parameters of the attack model. The important parameters are subjected to strict constraints, while the unimportant parameters are less constrained. The diagonal of the Fisher information matrix [159] herein is calculated as the importance of each parameter. The calculation of the diagonal in the Fisher information matrix is equivalent to computing the second derivative of the loss close to minimum for every parameters. With the importance of each parameter, the loss function $\mathcal{L}_{ewc}$ of the attack model is defined by

$$\mathcal{L}_{ewc} = \mathcal{L}_A + \lambda \sum_i F^A_i (\delta_i - \delta^*_i)^2, \tag{3.21}$$

where $\mathcal{L}_A$ is the loss function of an attacker to deceive a target model (cf. Equation (3.17)), λ is a constant factor to balance the global importance of the parameters $\boldsymbol{\delta}^*$ learnt from the previous target model, $\boldsymbol{F}^A$ is the Fisher information matrix, and $\boldsymbol{\delta}$ is the current parameters of the attack model. The square of the difference between $\boldsymbol{\delta}$ and $\boldsymbol{\delta}^*$ is minimised to remember $\boldsymbol{\delta}^*$ according to $\boldsymbol{F}^A$. At this point, the attack model is trained with a high transferability through remembering the previous knowledge with the regularisation.

4

Experimental Evaluations

In this chapter, comprehensive experiments of audio signal classification are designed and executed to verify the effectiveness of the presented approaches in Chapter 3. More specifically, the databases containing both audio and speech data sets are firstly described in Section 4.1. The evaluation metrics of the performance are then presented in Section 4.2. Afterwards, the experimental setups and the result discussions for every proposed approaches are elaborated from Section 4.3 to Section 4.8.

4.1 Databases

This section gives a brief description of the audio and speech databases used for the experimental evaluations in this thesis. Specifically, three types of databases corresponding to different applications are included for the experiments: three open databases for ASC tasks are firstly introduced in Section 4.1.1, two heart sound databases for HSC are then given in Section 4.1.2, and Section 4.1.3 finally describes two speech databases for speech-based classification tasks, including PLE and SER.

4.1.1 ASC Databases

ASC [11, 160], aiming to automatically identify the acoustic environment in an audio stream, has become a major research field of computer audition in recent years [161]. Using computational approaches of signal processing and machine learning, ASC has been employed in manifold applications, such as context-aware services [162], wearable devices [163], robot navigation systems [164], and serious games [165]. Moreover, ASC could be performed as a pre-processing step for some other applications, e. g., source separation of speech signals from different types of background noise [11].

In this light, the ASC task of the IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE) [124] provides a unique opportunity to develop machine learning models on the open databases. In the following

subsections, three ASC databases will be described: the DCASE 2017 database (cf. Section 4.1.1.1), the DCASE 2018 database (cf. Section 4.1.1.2), and the DCASE 2019 database (cf. Section 4.1.1.3).

4.1.1.1 DCASE 2017 Database

During the collection of the DCASE 2017 database [166], the audio signals were recorded in various locations using a Soundman OKM II Klassik/studio A3, an electret binaural microphone, and a Roland Edirol R-09 wave recorder with a sampling rate of 44.1 kHz and a resolution of 24 bits. To increase the number of audio samples, each audio recording was split into a set of independent segments with a time length of 10 seconds. Each audio segment was labelled as one of the 15 classes, including *beach*, *bus*, *cafe/restaurant*, *car*, *city center*, *forest path*, *grocery store*, *home*, *library*, *metro station*, *office*, *park*, *residential area*, *train*, and *tram*.

The whole database was split into a development set and an evaluation set considering the locations of the original audio recordings. In other words, the development and evaluation sets contain the same classes' recordings, but the recordings in the two sets are from different geographical locations. In the development set, there are 312 audio segments (52 minutes) for each acoustic scene, whereas each acoustic scene has 108 segments (18 minutes) in the evaluation set. A cross-validation setup was provided for the development set. This setup contains four folds, each of which consists of a training subset and an evaluation subset. The average performance over the four folds is computed as the final performance on the development set.

4.1.1.2 DCASE 2018 Database

The DCASE 2018 acoustic scene database [124] was recorded in six large European cities with four recording devices. The major recording device (called device A) incorporates a Soundman OKM II Klassik/studio A3, an electret binaural in-ear microphone, and a Zoom F8 audio recorder. All of the other three recording devices are customer devices, including a Samsung Galaxy S7 (called device B), an iPhone SE (called device C), and a GoPro Hero5 Session (called device D). While recording the audio signals, device A was worn in the ears, device B was held in a hand, device C was worn in a sleeve of the strap of a backpack, and device D was mounted on the other strap. Devices A and D were used to record the audio signals with a sampling rate of 48 kHz, and both sampling rates of devices B and C were 44.1 kHz. The recorded audio signals were then cut into separate 10-second segments.

All of the audio segments were labelled as one of the ten classes, including *airport*, *shopping mall*, *metro station*, *street pedestrian*, *public square*, *street traffic*, *tram*, *bus*, *metro*, and *park*. Based on the audio recordings from the four devices, the ASC task of DCASE 2018 contains two subtasks (called subtask A and subtask B). The recordings from device A are used in subtask A, while the recordings from all of the

four devices are utilised in subtask B. In each subtask, the corresponding database was split into a development set and an evaluation set. Particularly, in subtask B, the development set involves part of the recordings from device A, B, and C, while the evaluation set comprises the remaining part of recordings from device A, B, and C, and all recordings from device D. Since the labels of the evaluation sets were not publicly released, only experimental results on the development sets will be presented.

In subtask A, the development set contains 8,640 audio segments (24 hours). There are 864 segments (144 minutes) for each acoustic scene. The development set is split into two subsets: a training subset and a test subset. The training subset consists of 6,122 segments, and the test subset includes 2,518 segments.

The development set in subtask B consists of the same development set in subtask A, and the audio recordings from devices B and C. Devices B and C each contribute 720 segments (2 hours), and each scene contains 72 segments. As for the recordings from each mobile device (i. e., devices B and C), 540 segments were split into the training subset, and 180 segments were split into the test subset. Finally, the training subset of subtask B contains 7,202 segments, and the test subset consists of 2,878 segments.

4.1.1.3 DCASE 2019 Database

The database of the ASC task in DCASE 2019 was recorded in the same ten acoustic scenes as those in the DCASE 2018 database. As an extension of the DCASE 2018 database, the recording locations in the DCASE 2019 database were extended from six cities to 12 large European cities. The audio recording devices for the DCASE 2019 database were the same as those in DCASE 2018. The audio recordings were cut into smaller segments with a length of 10 seconds. The ASC task of DCASE 2019 also contains two subtasks, i. e., subtask A and subtask B. The databases in both subtasks were split into a development set and an evaluation set. Similar to the experiments on the DCASE 2018 database, the experimental evaluation will be discussed on the development set only.

In subtask A, the development set consists of 14,400 segments (40 hours) recorded by device A in ten cities. For each acoustic scene, there are 144 segments per city. The development set was split into 9,185 segments in the training subset, 4,185 in the test subset, and 1,030 segments from Milan.

In subtask B, the development set contains the audio recordings in subtask A and the recordings from devices B and C (amounting to 3 hours for each). In the development set, there are 16,560 audio segments, including 10,265 audio segments in the training subset, 5,265 segments in the test subset, and 1,030 segments from Milan. In the training subset, there are 9,185 segments from device A, and 540 segments from each of devices B and C. The test subset consists of 4,185 segments from device A, and 540 segments from each of devices B and C. Notably, the 1,030

audio segments from Milan were split into neither the training subset nor the test subset. Therefore, only the official training and test subsets will be utilised in the experiments.

4.1.2 Heart Sound Databases

As a leading cause of death, Cardiovascular Diseases (CVDs) caused 17.9 million deaths in 2016 (representing 31 % of all global deaths), and have been a worldwide health burden today [167, 168]. To mitigate the high costs and social burdens, early-stage diagnosis is essential to cope with serious CVDs [169, 170]. Auscultation of the heart sounds has been widely utilised in clinics over a century as a cheap, convenient, and non-invasive means [171]. However, auscultation of heart sounds requires experienced physicians [172]. In this regard, developing automatic auscultation techniques of heart sounds can help early screening of heart diseases, and is an auxiliary measure of the diagnosis by physicians. Typical automatic auscultation techniques of heart sounds contain HSC (e. g., normal/abnormal) [173, 174] and heart sound segmentation [175]. Specifically, the HSC tasks will be focused in the experiments.

In the following subsections, two heart sound databases for HSC tasks will be introduced. The PhysioNet/Computing in Cardiology (CinC) database will be firstly described in Section 4.1.2.1, and the Heart Sounds Shenzhen (HSS) database will be then given in Section 4.1.2.2.

4.1.2.1 PhysioNet/CinC Database

The PhysioNet/CinC Challenge 2016 provided an opportunity for developing HSC algorithms by constructing a large-scale heart sound database via collecting multiple heart sound data sets from various research groups [12, 176]. The heart sounds in the PhysioNet/CinC database were annotated into two classes: *normal* and *abnormal*. Only the training set of the PhysioNet/CinC database will be used in the experiments, since the test set was not publicly released. Different from the official data distribution, the training set is split into a new training subset and a new test subset, as shown in Table 4.1. As the official test set is not considered, the training subset and the test subset herein are called training set and test set, respectively. The training set consists of four independent data sources, including the Massachusetts Institute of Technology (MIT) heart sounds database, the Aristotle University of Thessaloniki (AUTH) heart sounds database, the University of Haute Alsace (UHA) heart sounds database, and the Dalian University of Technology (DLUT) heart sounds database. The test set contains two data sources: one is the Aalborg University (AAD) heart sounds database, and the other is the Shiraz University adult (SUA) heart sounds database. The details of every data source are summarised in Table 4.1.

Table 4.1: An overview of the training and test partitions. The training set is structured by four sub-databases from the PhysioNet/CinC database, and the test set is by two. The heart sound signals herein are annotated by the two-class labels (normal/abnormal).

#	Data source	Recordings	Normal	Abnormal	Durations (seconds)			
					$\sum$	Min	Max	Average
Train	MIT	409	117	292	13,328.08	9.27	36.50	32.59
	AUTH	31	7	24	1,532.49	9.65	122.00	49.44
	UHA	55	27	28	833.14	6.61	48.54	15.15
	DLUT	2,141	1,958	183	49,397.15	8.06	101.67	23.07
$\sum$		2,636	2,109	527	65,090.86	–	–	–
Test	AAD	490	386	104	3,910.20	5.31	8.00	7.98
	SUA	114	80	34	3,775.45	29.38	59.62	33.12
$\sum$		604	466	138	7,685.65	–	–	–

Before training a model on the whole training set, the training set is split into three folds for cross-validation. In the three folds (fold 1, fold 2, and fold 3), the data sources MIT, AUTH, or UHA are excluded for validation, respectively. Please note that, DLUT is always used for model training due to its large scale.

4.1.2.2 HSS Database

The HSS database [177], collected by the Shenzhen University General Hospital, China, consists of 845 audio recordings from 170 subjects (female: 55, male: 115, age: 65.4 ± 13.2 years), as shown in Table 4.2. These subjects have various health conditions, including coronary heart disease, heart failure, arrhythmia, hypertension, hyperthyroid, valvular heart disease, and congenital heart disease amongst others. The heart sound recordings were collected from the four locations on the body of each subject, i. e., auscultatory mitral, aortic valve auscultation, pulmonary valve auscultation, and auscultatory areas of the tricuspid valve. An electronic stethoscope (Eko CORE, USA) was used to record the heart sounds at a sampling rate of 4 kHz. Each heart sound signal was recorded with a duration of 30 seconds on average (ranging from 29.808 seconds to 30.152 seconds). Finally, the audio recordings were annotated with three classes, i. e., *normal*, *mild*, *moderate/severe*, by experienced cardiologists using Echocardiography as the gold standard.

The database was split into training, development, and test sets with the consideration of age and gender balances in the INTERSPEECH ComParE 2018 [38]. This data distribution will be used in the experiments for a comparison with the state-of-the-art methods on the HSS database.

Table 4.2: The number [#] of the instances and the subjects in each data set of the HSS database. The HSS database is split into three data sets, including the (train)ing, (dev)elopment and test sets, for a three-class classification task (normal, mild, and (mod)erate/(sev)ere).

#	Normal	Mild	Mod./Sev.	Σ	Subject
Train	84	276	142	502	100
Dev	32	98	50	180	35
Test	28	91	44	163	35
Σ	144	465	236	845	170

4.1.3 Speech Databases

Apart from the above audio databases, two speech-based applications and the corresponding corpora will be given in this subsection. Firstly, the Duessel-dorf Acute Pain (DAP) Corpus will be described in section 4.1.3.1 for PLE. Then, the Database of Elicited Mood in Speech (DEMoS) mainly designed for SER will be presented in Section 4.1.3.2.

4.1.3.1 DAP Corpus

As a neural perception within the human brain, pain is a key reaction from individuals in terms of their physical and psychological health [178, 179]. To diagnose a set of pathologies, such as cancer [180] and Alzheimer [181], pain evaluation is widely used as an auxiliary role in clinical practice. For instance, the diagnosis of cancer and its severity can benefit from pain information. In practice, clinical examinations of pain rely on self-reports from patients, e. g., questionnaires which contain the details of pain information, including location, type, time, length, and level [182]. Moreover, the pain information is evaluated inside the self-reports using various assessment scales, like the Numerical Pain Rating Scale (NPRS) and the visual analogue scale [183]. However, these evaluation methods of pain are highly subjective, especially with age and gender. Such a subjectivity can possibly lead to biases during the procedure of evaluating the pain level.

To mitigate the biases caused by the conventional evaluation methods, automatic evaluation of pain is potential to construct an objective and unified standard. In recent studies, models for automatic detection of pain have been investigated and proposed based on multiple modalities, including facial expression [184, 185, 186], body gestures, and motion descriptors [187, 188]. As an important factor of evaluating the physiological health like the cardiovascular system [189] and the mental health such as depression [190], voice is potential to evaluate the pain level. In [191], a speech-based corpus with 400 short samples from 27 participants was collected for pain detection, yet, the small size of this corpus is a limitation for training the

Table 4.3: An overview of the partition in the DAP corpus. The speech recordings are divided into three data sets, including the (train)ing, (dev)elopment and test sets, for a three-class classification task (mild/(mod)erate/severe).

Dataset	$\sum$	Mild	Mod.	Severe	Durations (s)			
					$\sum$	Min	Max	Avg
Train	526	320	127	79	6,704.4	3.5	66.9	12.7
Dev	163	95	42	26	2,062.7	3.7	39.7	12.7
Test	155	108	25	22	2,018.0	3.8	36.0	13.0
$\sum$	844	523	194	127	10,785.0	3.5	66.9	12.8

state-of-the-art deep learning models. In this context, as presented by the author and her colleagues in [2], a bigger corpus – the DAP corpus, which contains 844 audio samples from 80 subjects (female: 39, male: 41), will be introduced.

The ages of the participants in the DAP corpus vary from 19 to 64 years, with 35.3 years on average and a standard deviation of 14.9 years. A cold pressor test, as the pain stimulus source, was utilised during the data collection procedure. The left hand was immersed up to the wrist in ice-chilled water ($0.5 - 1.5$°C), and the water tub (2.8 l) was shaken manually by the experimenter every 30 seconds to prevent the water from warming up around the skin. During the cold pressor test, the participants were asked to read sentences regarding the voice commands in German as used for driver assistance systems and a German short story "The North Wind and the Sun". Additionally, spontaneous dialogues were elicited by asking the participants to book a doctor's appointment. The speech recordings were collected with a sampling rate of 44.1 kHz, and then down-sampled to 16 kHz with a quantisation of 16 bits. All of the speech samples were annotated by the participants themselves using the clinically reliable and valid 11-point NPRS, where 0 means "no pain" and 10 points at "worst imaginable pain".

To make the machine learning experiments available on this database, the DAP corpus is split into training, development, and test sets considering the balance of gender and age, for a three-class classification task as shown in Table 4.3. The three classes are obtained via splitting the NPRS into three intervals: i) *mild*: $0 - 2$, ii) *moderate*: $3 - 5$, and iii) *severe*: $6 - 10$.

4.1.3.2 DEMoS Database

In recent years, SER, which plays an essential role in Human-Computer Interaction (HCI), has become a popular research topic [192]. SER is a task of recognising the emotional aspects of a speech signal [193]. The emotional aspects can be either continuous dimensional affect ratings in terms of arousal and valence, or discrete emotional categories (e. g., happiness, sadness, etc). Thanks to the rapid develop-

Table 4.4: An overview of the speaker-independent partitions, i. e., (train)ing, (dev)elopment, and test sets, created from the DEMoS, including the distribution of the seven classes as well as the subjects and the genders, i. e., (f)emale and (m)ale.

#	Train	Dev	Test	$\sum$	Gender (F:M)
Subjects	24	22	22	68	23: 45
Anger	492	472	513	1,477	516: 961
Disgust	525	556	597	1,678	596:1,082
Fear	380	383	393	1,156	415: 741
Guilt	351	366	412	1,129	400: 729
Happiness	447	434	514	1,395	524: 871
Sadness	493	486	551	1,530	532: 998
Surprise	336	327	337	1,000	349: 651
$\sum$	3,024	3,024	3,317	9,365	3,332:6,033

ment of machine learning, especially deep learning, SER has been employed in a large number of real-life applications, such as call centre conversations [194], educational settings [195], and diagnosis tools for conditions like depression [196].

In this thesis, predicting discrete emotional categories from the speech will be the task of SER. An Italian emotional speech corpus – the DEMoS [197], is employed. DEMoS, containing $9,365$ emotional speech samples and 332 neutral speech signals, was recorded from 68 speakers (female: 23, male: 45). The emotions in DEMoS were induced by an arousal-valence progression. Since *neutral* is a minority class, only the emotional speech samples will be considered. As shown in Table 4.4, the $9,365$ speech samples were annotated with seven classes of emotional states, including *anger*, *disgust*, *fear*, *guilt*, *happiness*, *sadness*, and *surprise*. Then, the whole database was split into speaker-independent training, development, and test sets with the consideration to gender and emotional class balancing. For more details of the DEMoS, the readers are suggested to refer to [197].

4.2 Evaluation Metrics

To evaluate the performance of the proposed approaches for audio signal classification, a set of measurements will be employed. In the following, a brief introduction of the frequently-used measurements will be presented to evaluate the classification performance.

In a binary-class classification task, the sensitivity (i. e., True Positive Rate (TPR)), as a class-wise evaluation, is defined by

$$\text{sensitivity} = \frac{N_{TP}}{N_{TP} + N_{FN}}, \tag{4.1}$$

where N_{TP} is the number of true positive samples, N_{FN} denotes the number of false negative samples. Correspondingly, the specificity (i. e., True Negative Rate (TNR)) is computed by

$$\text{specificity} = \frac{N_{TN}}{N_{TN} + N_{FP}}, \tag{4.2}$$

where N_{TN} denotes the number of true negative samples, and N_{FP} denotes the number of false positive samples.

To evaluate the performance of a model for a binary-class classification task on both classes, the Mean Accuracy (MAcc) is generally utilised:

$$\text{MAcc} = \frac{\text{sensitivity} + \text{specificity}}{2}. \tag{4.3}$$

In addition to binary-class classification tasks, to evaluate the performance on each class in a multi-task classification problem, the recall (i. e., class-wise accuracy) score and precision score are defined by

$$\text{recall}_k = \frac{\tilde{N}_k}{N_k}, \tag{4.4}$$

$$\text{precision}_k = \frac{\tilde{N}_k}{\hat{N}_k}, \tag{4.5}$$

where k is the k-th class of all K classes, recall_k is the recall score of the k-th class, representing the ratio of the number of correctly predicted samples $\tilde{N}_k$ and the total number of samples N_k at the k-th class, and precision_k, the precision score, is computed as the ratio of $\tilde{N}_k$ and the total number of the predicted samples $\hat{N}_k$ at the k-th class. Specifically, the recall score is an extension of sensitivity in the context of multi-class classification.

Apart from the class-wise evaluation metrics, several measurements over all classes have been proposed. A widely-used measurement is the Weighted Average Recall (WAR) (i. e., accuracy), which aims to evaluate the performance of an approach over all classes. WAR is computed based on the weighted recall scores over all classes:

$$\text{WAR} = \sum_{k=1}^{K} \frac{N_k}{N} \text{recall}_k. \tag{4.6}$$

Although WAR can give an effective evaluation of the performance, a classification system might be overestimated if the correctly classified samples labelled with the majority class.

To overcome the aforementioned problem of WAR, the Unweighted Average Recall (UAR) was proposed as a frequently-used measurement. UAR is computed as the average value of the recall scores over all classes:

$$\text{UAR} = \frac{\sum_{k=1}^{K} \text{recall}_k}{K}. \tag{4.7}$$

As UAR is computed as the average of all recall scores without weights, a classification system will not be overestimated on an imbalanced data set. Notably, MAcc is a specific form of UAR in binary-class classification.

4.3 Transfer Learning based Deep Representation Learning

In this section, to verify the proposed approaches of deep representation learning in Section 3.1, the experiments will be carried out on three databases for three tasks, respectively. Specifically, the DCASE 2017 database will be selected for an ASC task (cf. Section 4.3.1), the DAP corpus will be chosen to evaluate the pain level from speech (cf. Section 4.3.2), and the PhysioNet/CinC database will be used for an HSC task (cf. Section 4.3.3). Finally, the experiments will be summarised in Section 4.3.4.

4.3.1 Performance on DCASE 2017 Database

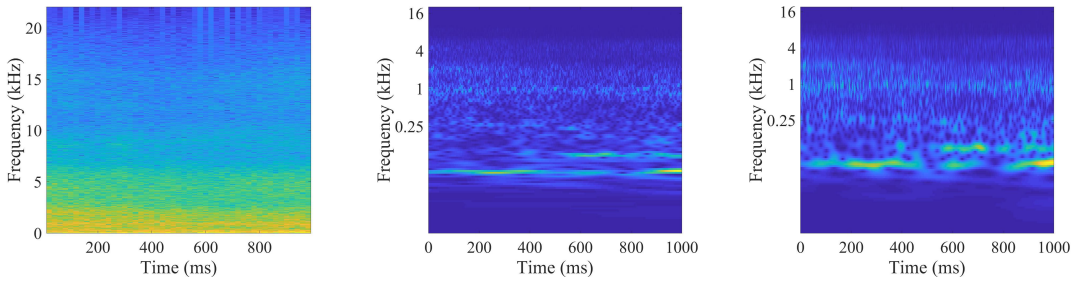
The first experiment of transfer learning is on the DCASE 2017 database. Next, the experimental details will be introduced in Section 4.3.1.1, and then the results will be presented and discussed in Section 4.3.1.2.

4.3.1.1 Experimental Setup

In the experiment, each audio sample is firstly cut into a sequence of 19 audio segments with a length of 1 second and an overlap of 50 %. Then, the hand-crafted features and the deep representations are extracted from all of these audio segments, respectively.

The hand-crafted features consist of two kinds of LLDs: MFCCs $1-14$ and log Mel-Frequency Bands (MFBs) $1-8$. Both of the two kinds of features are obtained according to the feature sets which were provided in the ComParE challenges [198]. Therefore, 100 functionals are applied to each LLD, yielding $14 \times 100 = 1,400$ MFCCs features and $8 \times 100 = 800$ log MFBs features. Specifically, the hand-crafted feature extraction is implemented using openSMILE [199].

To obtain the deep representations, three kinds of time-frequency representations, including spectrograms, *bump* scalograms, and *morse* scalograms, are firstly extracted from the audio segments, as shown in Figure 4.1. Then, the spectrum representations are fed into pre-trained AlexNet, VGG-16, and VGG-19, yielding three sets of deep representations from each kind of time-frequency representations. As all of the three CNNs were designed for a $1,000$-class classification task, the deep representations are extracted from the activations of the second fully connected layer



(a) spectrogram (b) *bump* scalogram (c) *morse* scalogram

Figure 4.1: The spectrogram, the *bump* scalogram, and the *morse* scalogram extracted from the first audio segment of DCASE2017's *a001_10_20.wav* with a label *residential area* [1].

fc7. Notably, the pre-trained AlexNet is obtained from MATLAB R2017a[3], and the VGG-16 and VGG-19 models are from MatConvNet [200].

Next, the extracted features are fed into (B)GRU-RNNs with 120 and 60 neurons respectively with a tanh activation function. The (B)GRU-RNNs are followed by a single highway network layer with a linear activation function and a softmax layer, as highway networks were found to be mostly more efficient than conventional fully connected layers [201]. Empirically, the (B)GRU-RNNs are trained using an RMSProp optimiser with a learning rate of 0.0002 and a batch size of 65. The accuracy of each model are evaluated at the epochs of $20, 30, ..., 120$. Finally, the MSV strategy is used to fuse the predictions from all of the (B)GRU-RNN models for producing the final predicted labels.

4.3.1.2 Results and Discussion

According to the official evaluation metrics of DCASE 2017, the mean accuracy on the 4-fold partitioned development set is calculated in the experimental results. As shown in Figure 4.2, the performances of both GRU-RNNs and BGRU-RNNs on a set of feature sets are compared, when the training procedures are stopped at multiple epochs. We can see that, the accuracies of (B)GRU-RNNs on the hand-crafted features, i. e., MFCCs and log MFBs, are lower than the baseline. The performances of the (B)GRU-RNN models on the deep spectrum representations, particularly the representations extracted by the pre-trained VGG-16 and VGG-19, are comparable to the baseline performance. This indicates that the deep spectrum representations are more effective than hand-crafted features in this ASC task.

The performances of the (B)GRU-RNN models on each kind of deep spectrum representations are presented in Table 4.5. Each accuracy score on the development set is denoted as the best performance across all of the epochs, and the perfor-

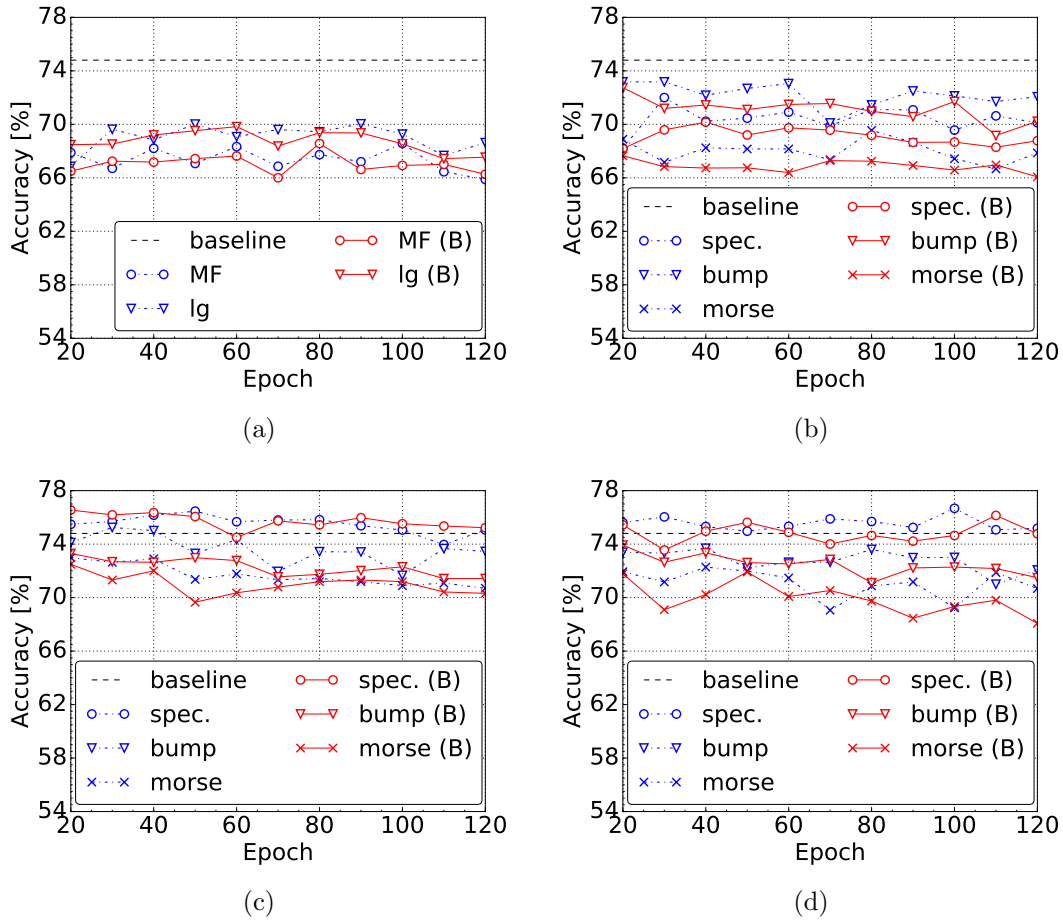


Figure 4.2: The performances of GRU-RNNs and BGRU-RNNs on different features. (a) (MF)CCs and log MFB (lg) features. The features from the spectrogram and scalograms (*bump* and *morse*) are extracted by the three CNNs: (b) AlexNet, (c) VGG-16, and (d) VGG-19.

mance on the evaluation set is obtained using the consistent epoch number on the development set. The results of the fused models on multiple deep representations outperform those of the models on a single deep feature set. For instance, the performances of the fused models on spectrograms and *bump* scalograms are mostly better than those of a single model on each of the three spectrum representations (spectrograms, *bump* scalograms, and *morse* scalograms). Furthermore, the performances of GRU-RNNs and BGRU-RNNs are comparable on the development set, yet, the performances of BGRU-RNNs are slightly better than those of GRU-RNNs on the evaluation set. This improvement of BGRU-RNNs may be due to the ability of BGRU-RNNs for covering the overall information in both forward and backward time directions. Through GRU-RNNs with the input of deep representa-

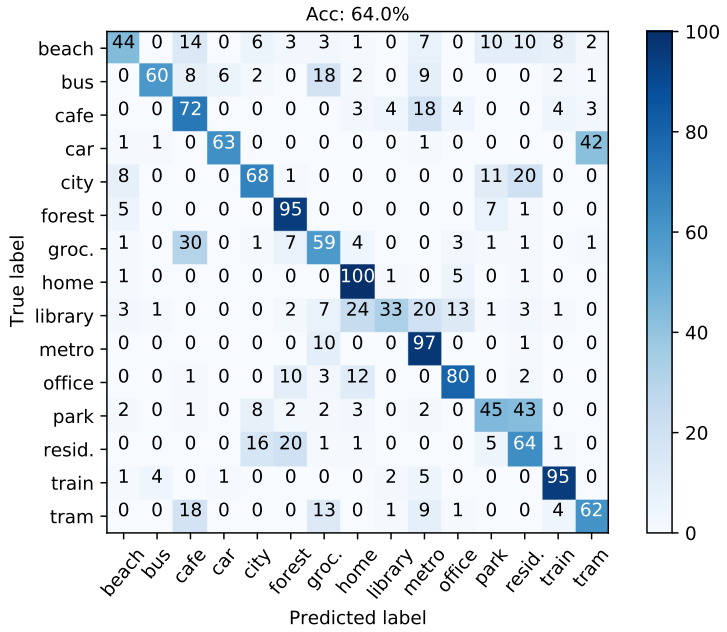


Figure 4.3: Confusion matrix of the best accuracy of 64.0 % on the evaluation set. This best accuracy is obtained by the late fusion of BGRU-RNNs on the deep features extracted by VGG-16 from the spectrograms and the *bump* scalograms.

tions extracted by VGG-16 from the spectrograms and *bump* scalograms, the best performance of 84.4 % on the development set achieves a significant improvement over the official baseline of 74.8 % in the DCASE 2017 challenge ($p < 0.001$ by a one-tailed z-test). Correspondingly, the deep representations extracted by VGG-16 from spectrograms and *bump* scalograms yield the best performance of 64.0 % on the evaluation set, when the BGRU-RNNs are trained for the classification task. This result of 64.0 % on the evaluation set is also an improvement upon the official baseline of 61.0 %.

To discuss the effectiveness of our proposed approach in each class, the confusion matrix of the best result on the evaluation set is depicted in Figure 4.3. The model performs well on the classes like *forest path*, *home*, and *metro station*. However, some classes such as *library* and *residential area* are challenging to be correctly classified. This difficulty might be caused by noises, or that the audio waves labelled with different acoustic scenes have similar environments. For example, some audio waves recorded in *library* are classified into *home*, which is also an indoor environment as *library*.

Furthermore, as for the class-wise performance of the spectrograms and scalograms, the best performance on the evaluation set, obtained by fusing the results of BGRU-RNNs on the deep features from the spectrograms and the *bump* scalograms, are compared to the performance before the decision-level fusion in Table 4.6. The

Table 4.5: Performance (accuracy [%]) comparisons on the development and the evaluation sets with the GRU-RNNs and the BGRU-RNNs on the hand-crafted features ((MF)CCs and log MFBs (lg)) and the deep features extracted by the pre-trained CNNs from the spectrograms (S), *bump* scalograms (B), and *morse* scalograms (M).

	GRU-RNNs						BGRU-RNNs					
	Development set			Evaluation set			Development set			Evaluation set		
	MF	*lg*	*MF+lg*	*MF*	*lg*	*MF+lg*	*MF*	*lg*	*MF+lg*	*MF*	*lg*	*MF+lg*
Acc [%]	68.6	70.0	75.6	49.3	56.0	56.9	68.6	69.8	74.7	48.7	53.7	52.1
Acc [%]	*AlexNet*	*VGG-16*	*VGG-19*	*AlexNet*	*VGG-16*	*VGG-19*	*AlexNet*	*VGG-16*	*VGG-19*	*AlexNet*	*VGG-16*	*VGG-19*
S	72.0	76.5	76.7	56.3	57.7	57.3	70.2	76.5	76.1	54.3	60.3	56.2
B	73.2	75.2	73.7	52.1	48.8	50.4	72.7	73.3	73.9	50.9	53.9	52.0
M	69.5	73.0	72.3	46.1	51.1	49.0	67.6	72.5	71.9	46.1	50.4	49.7
S+B	78.9	**84.4**	82.3	55.9	**61.7**	61.4	78.0	81.9	**83.4**	58.5	**64.0**	59.4
S+M	76.8	82.6	81.5	54.6	61.0	57.8	76.5	82.4	82.1	57.2	60.7	59.5
B+M	76.1	77.4	80.1	47.5	54.1	54.8	73.7	76.8	78.6	48.5	53.4	53.0
S+B+M	79.7	82.6	83.7	56.5	60.7	61.3	78.1	81.3	82.8	57.1	62.2	59.0

Table 4.6: Performance (precision [%]) on the evaluation set from before and after late fusion of the BGRU-RNNs on the deep features extracted from the spectrogram (S) and the *bump* scalogram (B).

Precision [%]	beach	bus	cafe	car	city	forest	groc.	home	library	metro	office	park	resid.	train	tram
S	54.6	30.6	52.8	64.8	51.9	81.5	62.0	69.4	35.2	83.3	88.0	48.1	58.3	71.3	52.8
B	10.2	62.0	61.1	47.2	65.7	88.0	36.1	98.1	25.0	87.0	17.6	24.1	49.1	88.0	49.1
S+B	40.7	55.6	66.7	58.3	63.0	88.0	54.6	92.6	30.6	89.8	74.1	41.7	59.3	88.0	57.4

result from the spectrograms performs better than that from the *bump* scalograms on the classes of *beach*, *grocery store*, *office*, and *park*, while the result from the *bump* scalograms is better on the scenes of *bus*, *city*, *home*, and *train*. Through the decision-level fusion, the precisions of classes, like *cafe/restaurant*, *metro station*, *residential area*, and *tram*, are improved over the results from a single deep feature set. Therefore, it is effective to employ both spectrograms and scalograms in order to obtain more accurate precisions.

Through the training set augmentation using GANs, the best results on the ASC task of DCASE 2017 are 87.1 % on the development set and 83.3 % on the evaluation set [202]. Since the approach of deep representation learning herein focuses on comparing the performances of deep representations and omits data augmentation, there is a significant difference between the result from the champion and the best performance in this work ($p < 0.001$ by one-tailed z-test). The GAN-based data augmentation in combination with the proposed method shown herein is promising to lead to better performance. Therefore, it is worthwhile to improve the proposed method through data augmentation in future work.

4.3.2 Performance on DAP Corpus

In this section, the experiments of deep representation learning will be evaluated on the DAP corpus. The experimental setup will be given in Section 4.3.2.1, and the results will be presented and discussed in Section 4.3.2.2.

4.3.2.1 Experimental Setup

Before training the classifiers, three feature sets are firstly extracted, including ComParE, MFCCs features, and deep spectrum representations. Moreover, the BoAW features based on these three kinds of representations are extracted, respectively. The details of these feature sets are described as follows.

1. The ComParE feature set [203] is extracted herein using openSMILE [199], yielding a $6,373$-dimensional feature vector for each audio sample. It is worth noting that, due to the relations between pain and emotion [179], the eGeMAPS [42] designed for affective computing was also considered in the initial experiment of this study. However, the preliminary results of the ComParE feature set outperformed those of eGeMAPS.

2. The MFCCs features, which produced good performances in tasks related to pain evaluation, like SER [204] and infant cry recognition [205], are extracted with MFCCs $1-14$ and 100 functionals. Finally, for each audio sample, a $1,400$ dimensional feature vector is extracted.

3. The deep spectrum representations are generated by a pre-trained VGG-16 with the input of mel spectrograms, three examples of which are depicted in Figure 4.4. The parameters of VGG-16 are obtained from Pytorch [206]. During this feature extraction procedure, the audio waves are firstly cut into segments, whose length and overlap depend on the classifiers introduced later. The mel spectrograms with 128 mel filter banks are then generated from each segment. Finally, the deep spectrum representations are obtained from the activations of the first fully connected layer in VGG-16.

4. The BoAW features contain three sub-feature sets corresponding to the above three feature sets. For the ComParE features, 65 extracted LLDs are quantised; for MFCC features, 14 LLDs are quantised. As deep spectrum representations cannot be extracted in terms of LLDs, a set of deep spectrum representations are generated from smaller non-overlapping audio segments with a time length of 3.5 seconds, and then quantised by the BoAW method. The codebook generation is achieved by openXBOW, and an iterative search is designed to select the optimal *codebook size* ($C_s \in \{125, 250, 500, 1,000, 2,000\}$ with the number of assignments consistently set to 10.)

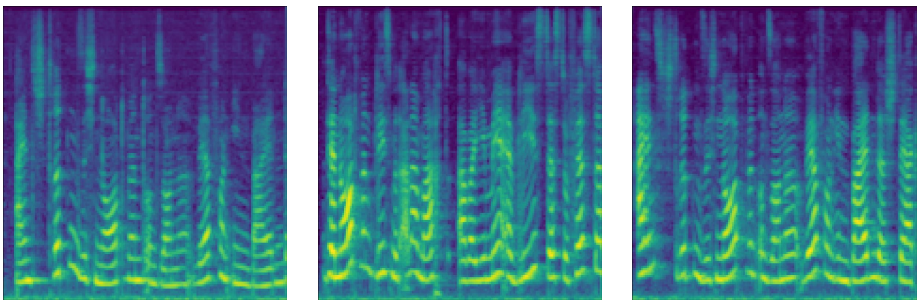
With the extracted features, two classifiers are then trained for the classification task.

1. SVMs, as the baseline in this experiment, are used to process both the functionals and the deep spectrum representations, as well as the BoAW features. All SVM models are set to linear kernels, and tuned with the complexity parameter $C \in [10^{-6}; 10^{-1}]$ on the development set.

   Notably, the deep spectrum representations are generated from the audio segments with a time length of 3.5 seconds (as per BoAW features). Then, the final prediction of each audio sample is calculated from the predictions of audio segments using MSV.

2. LSTM-RNNs herein are designed with three structures by varying the number of recurrent layers (N_l): the first structure has one layer with 60 neurons, the second one has two layers which hold the number of neurons $120 - 60$, and the third one is set as a three-layer model with the number of neurons $480 - 120 - 60$. The final recurrent layer in each LSTM-RNN model is then followed by a highway network layer and a softmax layer for classification.

   Please note that, to feed the sequential feature sets into LSTM-RNNs, all of the audio segments with a length of 3.5 seconds are cut into smaller audio chunks with a length of 0.6 seconds and an overlap of 0.3 seconds, and the features are then extracted from each chunk. Each output of the LSTM-RNNs is obtained from the prediction of the last audio chunk in a sequence. Finally,



(a) mild (b) moderate (c) severe

Figure 4.4: The three mel spectrogram images are extracted from the first 3.5 s segments of the three speech samples (labels: mild/moderate/severe) recorded from the same person: (a) *train_250.wav*, (b) *train_53.wav*, (c) *train_337.wav* [2].

the prediction of each audio sample is selected from the results of the audio segments using MSV.

During the training procedure, the learning rate is set to 0.0001, and the batch size is 128. The results in Table 4.7 are the strongest among the epoch number $epoch \in [30; 90]$.

Finally, since the DAP corpus is highly imbalanced, the evaluation metric of UAR is used herein instead of WAR.

4.3.2.2 Results and Discussion

Table 4.7 presents the results on both the development and the test sets. In the results of "BoAW+SVM", the performance on each C_s is the best one among the C values ($C \in [10^{-6}; 10^{-1}]$) on the development set. The results of "BoAW+SVM" with the input of the deep spectrum representations when $C_s = 2,000$ are not given, because the number of the training samples is less than $2,000$.

From Table 4.7, we can see that both ComParE and MFCCs feature sets perform well when SVMs are employed as the classifiers. The deep spectrum representations perform better than the other two feature sets (i. e., ComParE and MFCCs) when LSTM-RNNs are trained for classification. Finally, the best UAR of 42.7 % evaluated on the test set is obtained by the two-hidden-layer LSTM-RNNs with the deep spectrum representations as the input. This indicates that the deep spectrum representations extracted by transfer learning are effective to slightly improve the performance over the conventional hand-crafted features.

The confusion matrix of the best result on the test set is depicted in Figure 4.5. The class of *mild* is well classified. However, the other two classes (i. e., *moderate* and *severe*) are difficult to be recognised. The reason for this difficulty might be the imbalanced nature of the DAP corpus.

Table 4.7: Performances (UAR [%]) of the speech-based PLE approaches evaluated on the development and test sets of the DAP corpus.

	ComParE		**MFCC**		**Deep Spec.**	
UAR [%]	*Dev*	*Test*	*Dev*	*Test*	*Dev*	*Test*
C	**Functionals + SVM**					
10^{-6}	47.3	**42.0**	39.9	41.0	37.0	33.7
10^{-5}	45.2	38.3	39.9	41.0	37.6	31.8
10^{-4}	40.3	39.1	39.2	39.0	32.9	32.6
10^{-3}	42.2	36.8	39.0	38.5	31.3	32.5
10^{-2}	40.8	34.5	35.3	35.1	31.6	30.9
10^{-1}	40.8	34.5	33.5	36.7	31.1	34.7
C_s	**BoAW + SVM**					
125	46.8	35.2	39.6	30.7	44.0	35.7
250	47.9	38.9	40.6	40.0	41.7	28.6
500	46.0	33.8	45.7	38.4	40.8	30.5
$1,000$	43.9	39.4	43.3	38.8	40.0	35.5
$2,000$	50.3	33.0	40.2	**41.3**	—	—
N_l	**LSTM–RNNs**					
1	36.1	31.7	38.5	31.2	38.6	38.4
2	39.3	36.9	39.1	34.8	40.0	**42.7**
3	34.5	31.0	39.1	33.6	36.9	37.3

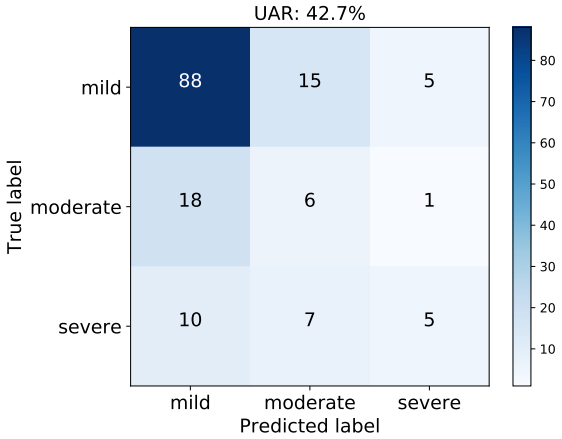


Figure 4.5: Confusion matrix of the best result (UAR: 42.7 %) on the test set. This result is obtained by the two-hidden-layer LSTM-RNNs from the deep spectrum features.

4.3.3 Performance on PhysioNet/CinC Database

In this section, the experiments of the deep representation learning evaluated on the PhysioNet/CinC database will be presented, including the experimental setup in Section 4.3.3.1 and the result discussions in Section 4.3.3.2.

4.3.3.1 Experimental Setup

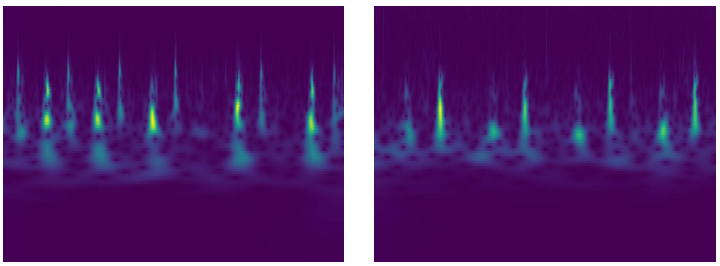
In the experiment, a baseline system using the 6,373-dimensional ComParE audio feature set and SVMs for classification, is compared to the four proposed transfer-learning-based methods (cf. Section 3.1.2): the *classification using traditional machine learning*, the *learnt CNNs with classification using traditional machine learning*, the *learning classifier of CNNs*, and the *learning CNNs*.

In the proposed approaches, each heart sound file is firstly cut into non-overlapping segments with an equal length of 4 seconds. The *morse* scalograms are then generated from each audio segment with a sampling rate of 2 kHz (cf. Figure 4.6), and fed into VGG-16 to achieve the HSC task.

For the first two aforementioned proposed approaches, which are denoted as *pre-trained VGG+SVM* and *learnt VGG+SVM* in this experiment, the deep spectrum representations are extracted from the activations of the first fully connected layer *fc6* in the VGG-16 model. Finally, the extracted deep representations are fed into a linear SVM for the binary-class classification task (normal/abnormal).

Herein, the third and the fourth proposed approaches are called *learning classifier of VGG* and *learning VGG*, respectively. The last fully connected layer of VGG-16 is replaced by a new fully connected layer with two output neurons, in order to achieve the binary-class classification task. While learning the classifier of VGG-16, the parameters of the convolutional layers and *fc6* are frozen, and only the parameters of the final two fully connected layers are trained. All of the parameters in the VGG-16 model are trained in the method of *learning VGG*. During the training procedures in these two methods, the learning rate is set to 0.001, the batch size is 64, the epoch number is 50, and the SGD is employed as the optimiser.

To evaluate the performance, as the official evaluation metrics in the PhysioNet/CinC Challenge 2016, the sensitivity, the specificity and the MAcc are used in this experiment. Notably, the class of *abnormal* is considered as the positive state, and the class of *normal* is considered as the negative state. Therefore, according to Equations (4.1) and (4.2), N_{TP} denotes the number of true positive abnormal samples, N_{FN} denotes the number of false negative abnormal samples, N_{TN} denotes the number of true negative normal samples, and N_{FP} denotes the number of false positive normal samples.



(a) normal (*a0007.wav*) (b) abnormal (*a0001.wav*)

Figure 4.6: The scalogram images with the *viridis* colour map are extracted from the first 4-second segments of the normal/abnormal heart sounds. The audio file names corresponding to the presented scalogram images are described in parentheses [3].

Table 4.8: Performances comparison of the proposed approaches with the baseline. The methods are evaluated on the 3-fold development set and the test set. The experimental results are evaluated by *(se)nsitivity*, *(sp)ecificity*, and *(MA)cc*.

	Development set												Test set		
	fold 1			fold 2			fold 3			mean					
performance [%]	*Se*	*Sp*	*MA*	*Se*	*Sp*	*MA*	*Se*	*Sp*	*MA*	*Se*	*Sp*	*MA*	*Se*	*Sp*	*MA*
ComParE+SVM (baseline)	23.6	93.2	58.4	58.3	100.0	79.2	00.0	100.0	50.0	27.3	97.7	**62.5**	76.8	17.0	46.9
pre-trained VGG+SVM	57.2	70.9	64.1	41.7	85.7	63.7	17.9	81.5	49.7	38.9	79.4	59.1	24.6	87.1	55.9
learnt VGG+SVM	58.6	57.3	57.9	83.3	57.1	70.2	32.1	70.4	51.3	58.0	61.6	59.8	24.6	87.8	**56.2**
learning Classifier of VGG	68.2	51.3	59.7	79.2	14.3	46.7	35.7	40.7	38.2	61.0	35.4	48.2	33.3	63.7	48.5
learning VGG	83.6	40.2	61.9	95.8	28.6	62.2	53.6	44.4	49.0	77.7	37.7	57.7	12.3	95.7	54.0

4.3.3.2 Results and Discussion

As shown in Table 4.8, all of the VGG-based approaches outperform the baseline at the metric of MAcc on the test set, although this outperformance is not obtained on the development set. This indicates that the deep spectrum representations perform better than the conventional hand-crafted features for this HSC task. In the results of the two adapted VGG-16 models, *learning VGG* performs better than *learning classifier of VGG*. The reason could be that all of the VGG-16 parameters are optimised to suit the PhysioNet/CinC dataset in the method of *learning VGG*. However, only the parameters of the final two fully connected layers are optimised in the method of *learning classifier of VGG*. When comparing the classifiers, the SVMs mostly perform stronger than the CNN classifiers. The reason is perhaps that the SVM classifiers are more suitable to relatively smaller amounts of training data than the CNN classifiers.

The best performance (MAcc: 56.2 %) on the test set is obtained by the *learnt VGG+SVM*, which achieves a significant improvement over the baseline ($p < 0.001$ by one-tailed z-test). Therefore, the proposed method of extracting deep spectrum representations using the *learnt VGG* is effective to perform on this HSC task.

4.3.4 Summary

To this end, the deep representation learning approaches were applied to multiple audio classification tasks, including ASC, PLE, and HSC. The time-frequency representations of the audio waves were fed into a CNN model to extract deep spectrum representations, which were further classified by the CNN model itself or an additional classifier. These proposed approaches outperformed either the baselines or the conventional SVMs with the hand-crafted features as the input.

In future efforts, data augmentation methods, such as GANs, will be investigated, as they have shown effectiveness in the champion's work in DCASE challenge 2017 [202]. Data augmentation is also promising to compensate for the imbalanced nature of the dataset, especially the DAP corpus and the PhysioNet/CinC database. In particular, due to the potential relations between pain and emotional states [179], in the work of PLE, it is worthwhile to annotate the data with emotional states, and explore multi-task classification paradigms to improve the performance of PLE.

4.4 Deep Learning Models with Attention

To train an explainable deep learning model, the attention mechanisms will be applied to DNNs in this section. In the following subsections, the attention-based DNNs will be validated on the applications of ASC (cf. Section 4.4.1) and HSC (cf. Section 4.4.2), respectively.

4.4.1 Performance on DCASE 2018 Database

On the DCASE 2018 database, an attention mechanism is employed to explain the potential contributions of every time-frequency bin in the log mel spectrograms to the predicted scenes. Next, the details of the experimental setup will be given in Section 4.4.1.1, and the results will be presented and discussed in Section 4.4.1.2.

4.4.1.1 Experimental Setup

With the input of 320×64 log mel spectrograms extracted with a Hamming window size of 2,048, an overlap of 672, and 64 mel bands, the three CNN models presented in Table 4.9 are trained on the training sets of the two subtasks (i. e., subtask A and subtask B). Since the DCASE 2018 dataset is much smaller than the ImageNet database [35], both AlexNet and VGG-4 are designed with fewer parameters than the models in Table 3.1. Additionally, "Net-4", a CNN structure with a stride with a size of 2 instead of local max pooling among the convolution layers, is designed to weaken the effect of local max pooling layers. This Net-4 has a kernel size of 5×5 in between those of AlexNet and VGG-4 to explore the effect of the kernel size and achieve better performance. To reduce the feature dimensions, a global pooling

Table 4.9: Configurations of the convolutional layers and local pooling layers in the three CNNs. The convolutional layers are denoted as "the number of convolution layers × conv(kernel size – number of output channels)" with the stride "s(stride size)".

AlexNet	**VGG-4**	**Net-4**
Input: log mel spectrogram		
1×conv11-64; s1 Maxpooling	1×conv3-64; s1 Maxpooling	1×conv5-64; s2
1×conv5-192; s1 Maxpooling	1×conv3-128; s1 Maxpooling	1×conv5-128; s2
1×conv3-384; s1 2×conv3-256; s1 Maxpooling	1×conv3-256; s1 1×conv3-512; s1 Maxpooling	1×conv5-256; s2 1×conv5-512; s2
Output		

layer follows the convolutional layers to convert the three-dimensional feature maps into one-dimensional tensors.

During the training procedure, the CNNs are optimised by an *Adam* optimiser with a learning rate of 0.001 during 3,000 maximum iteration steps. In the DCASE challenge 2018, the accuracy computed as an average of the class-wise accuracy was utilised as the official evaluation metric. Mathematically, the accuracy herein is the same as the UAR defined in Section 4.2. Yet, to be consistent with the DCASE challenge 2018, we call the measurement "accuracy" in this experiment.

4.4.1.2 Results and Discussion

The performances of the three CNN models, including AlexNet, VGG-4, and Net-4, are shown in Table 4.10. Nearly all of the CNN structures with a global pooling layer outperform the official baseline system. Specifically, the performances of AlexNet with attention are comparable to those of AlexNet with global max and average pooling, and the Net-4 with attention achieves improvements over the Net-4 with global max and average pooling. However, the VGG-4 with a global attention pooling layer performs slightly worse than VGG-4 with global max pooling. This underperformance might be caused by overfitting due to the larger number of hyper-parameters in VGG-4 with attention. Finally, the best results are achieved by the Net-4. This indicates that CNNs, with a kernel size of five and without local max pooling among convolutional layers, appear more suitable for this ASC task. The proposed Net-4 with a global attention pooling layer achieves an accuracy of 72.6 %

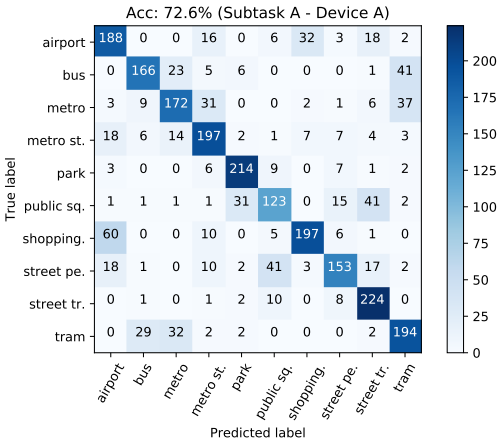
Table 4.10: Performance (accuracy [%]) comparison of the baseline and the CNN topologies of AlexNet, VGG-4, and Net-4, with three global pooling methods – max, average, and attention, evaluated on the official development sets of subtask A (SubA) and subtask B (SubB). The dataset recorded by device A is employed in subtask A, and the datasets from devices A, B, and C are used in subtask B. $\overline{(B, C)}$ stands for the mean evaluation performance on the data of devices B and C.

		SubA	**SubB**			
Model	Pooling	A	A	B	C	$\overline{(B, C)}$
Baseline		59.7	58.9	45.1	46.2	45.6
AlexNet	max	67.2	62.2	51.7	54.4	53.1
AlexNet	average	64.3	60.8	51.1	52.2	51.7
AlexNet	attention	67.2	64.2	53.3	46.7	50.0
VGG-4	max	68.7	66.8	53.9	56.1	55.0
VGG-4	average	63.7	63.2	52.8	48.9	50.8
VGG-4	attention	67.6	64.6	50.6	46.1	48.3
Net-4	max	68.1	68.7	58.3	55.6	56.9
Net-4	average	68.1	67.3	**59.4**	56.1	57.8
Net-4	attention	**72.6**	**71.8**	58.3	**58.3**	**58.3**

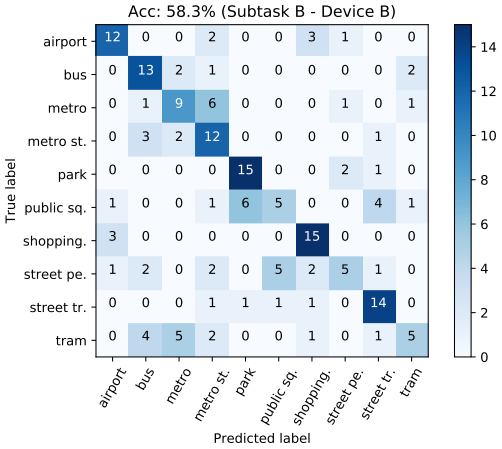
for subtask A, which is a significant improvement over the baseline ($p < 0.001$ in a one-tailed z-test). For subtask B, the Net-4 achieves accuracies of 71.8 %, 58.3 %, and 58.3 % on the data from devices A, B, and C, respectively, which are significant improvements of the baseline as well (in a one-tailed z-test, $p < 0.001$ for device A , $p < 0.01$ for device B, and $p < 0.05$ for device C).

From Table 4.10, we can also see that the CNN models mostly perform better for subtask A than subtask B. The reason might be that multiple data recording devices were used in subtask B. The data mismatch may lead to a difficulty for the models to achieve high performance. In subtask B, the models perform better on the data from device A than the one from devices B and C, perhaps because of the data imbalance among the three devices, i. e., more audio recordings were recorded with device A than devices B and C.

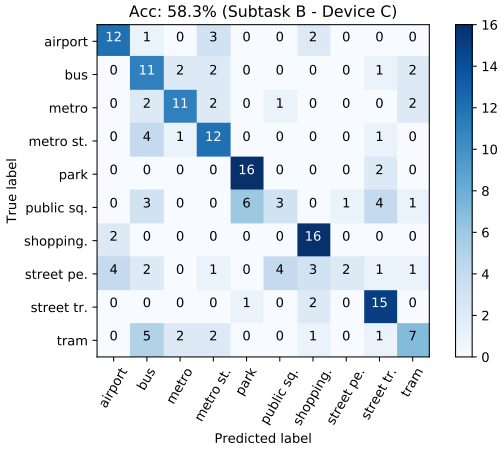
To analyse the class-wise performances achieved by the best model, i. e., Net-4 with attention, the confusion matrices of the results on the data from the three devices are depicted in Figure 4.7. The Net-4 model performs well on several classes, such as *park*, *shopping mall*, and *street traffic*. However, it is challenging to recognise some classes, like *public square* and *street pedestrian*, perhaps due to the background noise in these classes.



(a)



(b)



(c)

Figure 4.7: Confusion matrices of the best results obtained by the Net-4 with attention on the data from device A in subtask A and devices B and C in subtask B.

As the attention mechanism can estimate the contributions of every time-frequency bin to the predictions, the heat maps during the global attention pooling procedure are visualised in Figure 4.8. We can see that, even though it is challenging to analyse the characteristics of each acoustic scene from the visualised heat maps due to their small size, each time-frequency bin in the heat map of each acoustic scene is assigned a different weight. We think the weight difference thereby leads to more optimal predictions.

Figure 4.8: Heat maps of every scene class computed by the probability tensors in the attention-based Net-4 model. Each heat map is the transpose matrix of the probability tensor with a size of 20×4 for a better display. The horizontal axis represents the time steps, and the vertical axis represents the feature vectors.

4.4.2 Performance on HSS Database

To explain the DNNs performed on the HSS database, both attention mechanisms at the frame level and the time-frequency level are applied in this experiment. The experimental setup will be introduced in Section 4.4.2.1, and the results will be presented and discussed in Section 4.4.2.2.

4.4.2.1 Experimental Setup

Firstly, the log mel spectrograms with a size of 936×64 are extracted from the heart sound signals using a Hamming window with a length of 256, an overlap with a length of 128, and 64 mel frequency bins. Then, two types of DNNs, including CNNs and RNNs, are used to process the log mel spectrograms. The details of the employed models are described as follows.

1. The CNN models contain four convolutional layers with output channels 64, 128, 256, and 256. Each convolutional layer is followed by a local max pooling layer with a 2×2 kernel.

2. The RNN models consist of LSTM-RNNs and GRU-RNNs. Both RNN models contain three recurrent layers with output channels 256, $1,024$, and 256.

To achieve the HSC task, the above convolutional/recurrent layers are followed by a flattening layer or a global pooling layer, and the final log softmax layer.

During the training procedure, an *Adam* optimiser is utilised to optimise the models with a batch size of 32. To stabilise the training procedure, the learning rate initialised by 0.0001 is reduced by a factor of 0.9 at every 100-th iteration. The training process is finally stopped at the $3,000$-th iteration step. To be consistent with the ComParE challenge [38], the performances are evaluated by the UAR.

Table 4.11: The performance (UAR [%]) comparison of various deep learning topologies on the HSS database.

	w/o upsampling		w/ upsampling	
UAR [%]	Dev	Test	Dev	Test
CNN				
Flattening	35.6	37.6	35.6	39.9
Max-pooling	41.7	38.4	39.3	38.5
Attention-softmax	31.5	43.1	38.3	47.3
Attention-sigmoid	40.1	**51.2**	39.6	50.5
LSTM-RNN				
Last-time stamp	39.3	36.1	40.7	35.7
Max-pooling	32.9	38.9	34.6	38.1
Attention-softmax	40.0	39.6	39.0	39.4
Attention-sigmoid	39.6	38.9	42.0	**42.6**
GRU-RNN				
Last-time stamp	39.0	36.5	37.4	36.1
Max-pooling	38.7	35.8	40.7	35.2
Attention-softmax	30.8	44.7	35.3	**46.8**
Attention-sigmoid	34.9	44.2	34.5	45.7

Specifically, in order to investigate the effect of data balance status to the classification performance on the HSS database, the results on the original imbalanced database and random-upsampling-based balanced data [207] are compared.

4.4.2.2 Results and Discussion

Table 4.11 presents and compares the experimental results. The attention mechanism can mostly improve the corresponding models in this HSC task. For example, CNNs with a sigmoid-attention (UAR: 51.2 %) perform better than CNNs with flattening (UAR: 37.6 %) and CNNs with global max pooling (UAR: 38.4 %) ($p < 0.01$ in a one-tailed z-test). Moreover, the upsampling strategy slightly improves the performances of the best RNN models, but such improvement is not significant. The reason might be that this simple upsampling strategy is not able to sufficiently generate informative instances to improve performance. Finally, the best result (UAR: 51.2 %) is achieved by the CNN model with a sigmoid-attention mechanism at the time-frequency level. This best result outperforms those of LSTM-RNNs (UAR: 42.6 %) and GRU-RNNs (UAR: 46.8 %).

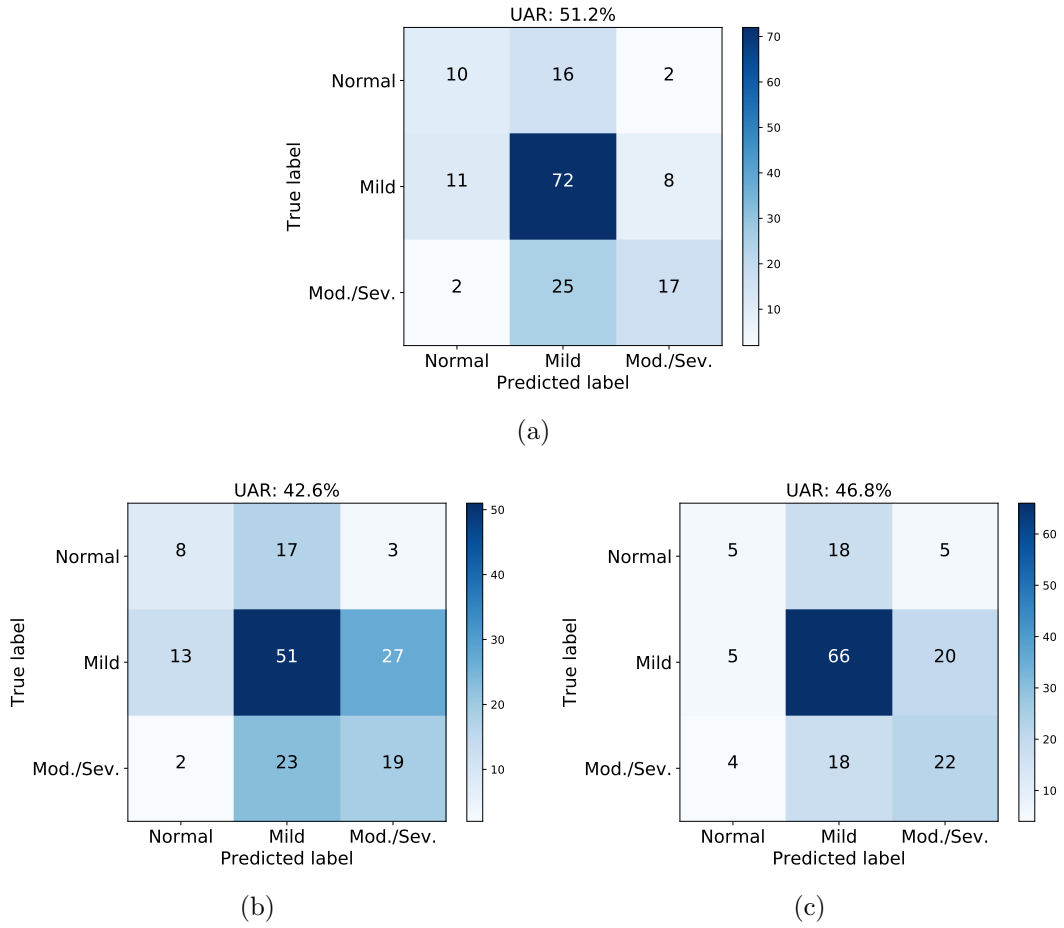


Figure 4.9: Confusion matrices achieved by the best models on the test set. The best three models are (a) CNN with sigmoid-attention, (b) LSTM-RNN with sigmoid-attention, and (c) GRU-RNN with softmax-attention, respectively.

To analyse the class-wise performances of the three best models, the confusion matrices are shown in Figure 4.9. Specifically, the best CNN model can classify the class of *mild* better than the other two models. Although all of the three models perform well on the class of *mild*, it is difficult for them to accurately predict the other two classes, i. e., *normal* and *moderate/severe*.

Furthermore, the best result of our proposed model, i. e., CNN with sigmoid-attention, is compared to those of the state-of-the-art methods on the HSS database in Table 4.12. The proposed approach outperforms most of these methods, but underperforms the ComParE baseline (fusion). The reason of the underperformance might be that multiple models are fused to improve the performance of a single model in the ComParE baseline (fusion).

Table 4.12: The performance (UAR [%]) comparison between the proposed model and the state-of-the-art methods.

UAR [%]	Dev	Test
ComParE baseline (End2You) [208]	41.2	37.7
ComParE baseline (openSMILE) [208]	50.3	46.4
ComParE baseline (openXBOW) [208]	42.6	52.3
ComParE baseline (fusion) [208]	–	56.2
Ensemble of transfer learning [209]	57.9	42.1
Utterance-level feature and SVMs [210]	53.2	49.3
Seq2Seq autoencoders and SVMs [65]	35.2	47.9
Log Mel features and SVMs [177]	46.5	49.7
Our proposed approach	**40.1**	**51.2**

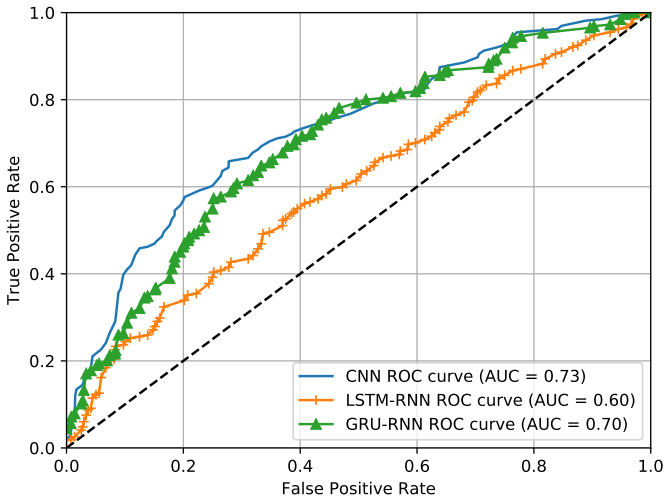


Figure 4.10: A comparison of the macro-average ROC curves of the best three models on the test set. The corresponding AUC is also computed for each model.

Figure 4.10 presents the macro-averaged Receiver Operating Characteristic (ROC) curves of the three best models' performance on the test set. The ROC curves of the CNN model and the GRU-RNN model mostly have higher TPRs at a given False Positive Rate (FPR) than that of the LSTM-RNN model. The TPRs of the CNN model are comparable to or higher than those of the GRU-RNN model. The Area Under the ROC Curve (AUC) of the CNN model is the largest among those of the three models, indicating that the CNN model is more effective than the other two models.

To explain the attention-based models, the heat maps in the three models are visualised in Figure 4.11; the original audio waveforms and the log mel spectrograms

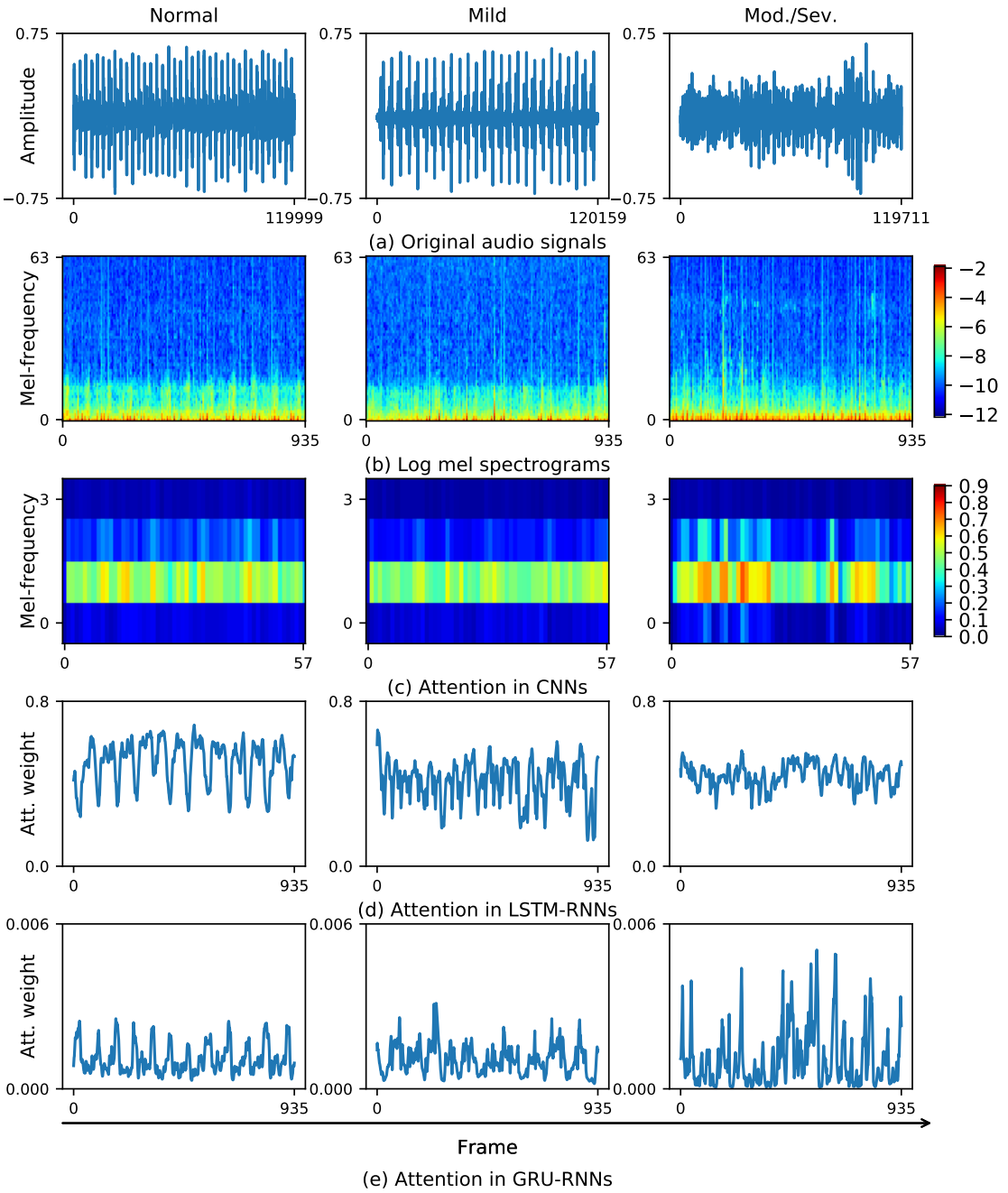


Figure 4.11: Visualisation of the three examples with the classes of *normal*, *mild*, and *moderate/severe*, respectively. Each example consists of the original audio waveform, its corresponding log mel spectrogram, the attention tensor in the CNNs, the attention tensor in the LSTM-CNNs, and the attention tensor in the GRU-RNNs.

are also depicted. We can see that the class of *moderate/severe* shows more irregular than the other two classes (i. e., *normal* and *mild*) from the perspective of the audio waveforms (cf. Figure 4.11 (a)). Correspondingly, the high-level representations also present the irregularity on the attention tensors. For example, the class of *moderate/severe* has higher weights than the other two at the similar frequency bands of the CNNs' attention tensors. In the attention tensors of RNNs, we can see the periodic signal's characteristics in the class of *normal*, and the relative irregular changes along the time axis in the class of *moderate/severe*.

4.4.3 Summary

In this section, the DNNs with attention were applied to the DCASE 2018 database and the HSS database. Specifically, CNNs with attention at the time-frequency level were used on the DCASE 2018 database, and DNNs with attention at both the frame and the time-frequency levels were utilised on the HSS database. On both databases, the attention mechanism achieved improvements over the corresponding models without attention.

In future work, since the proposed attention mechanism is only at the decision level of DNNs, it is worthwhile to investigate the attention-based models at the feature level in order to explain the contributions of the representations in each intermediate layer to the predictions. Furthermore, the size of the feature maps in CNNs are reduced due to the local max pooling layers among the convolutional layers or the strides inside the convolutional operations, therefore it is essential to explain CNNs in detail through visualising high-resolution attention heat maps.

4.5 Atrous CNNs with Attention

To increase the size of the attention tensors in CNNs, the atrous CNNs with attention (cf. Section 3.2.3) will be employed and developed on the DCASE 2018 database. The details of the experiments will be introduced, including the experimental setup in Section 4.5.1 and the result discussions Section 4.5.2. Finally, a summary of this work will be given in Section 4.5.3.

4.5.1 Experimental Setup

The log mel spectrograms with a size of 64×320 are firstly extracted from the acoustic signals using a Hamming window with a size of $2,048$ and 64 mel bins. The overlap is set to satisfy that each log mel spectrogram contains 320 time frames. Then, the log mel spectrograms are fed into CNNs to achieve the ASC task. Three types of CNN models are used for comparison: baseline CNN, CNN without local pooling, and atrous CNN. All of the three CNN models have four convolutional

Table 4.13: Performance (accuracy [%]) comparison of the CNN topologies with flattening and five global pooling models, including max, average (avg), ROI, attention (att), and the combination of ROI and attention (roi+att), evaluated on the two subtasks (SubA on the data of device A, and SubB on the data of devices A, B, and C).

Accuracy [%]		SubA	SubB		
Network	Pooling	A	A	B	C
Baseline CNN	flatten	60.9	61.6	49.4	46.7
Baseline CNN	max	68.6	69.8	57.2	57.8
Baseline CNN	avg	69.1	65.8	57.2	57.8
Baseline CNN	att	72.4	72.6	62.2	56.1
CNN w/o local pool	max	60.4	61.9	46.7	52.2
CNN w/o local pool	avg	62.8	59.1	54.4	50.0
CNN w/o local pool	roi	61.6	61.7	50.6	43.9
CNN w/o local pool	att	62.1	59.6	45.0	43.3
CNN w/o local pool	roi+att	68.1	69.2	56.1	50.6
Atrous CNN	max	68.8	69.7	60.0	59.4
Atrous CNN	avg	69.1	67.2	62.8	60.0
Atrous CNN	roi	65.2	62.6	48.3	43.9
Atrous CNN	att	**72.7**	**73.2**	**64.4**	**62.2**
Atrous CNN	roi+att	72.6	72.2	57.2	56.7

layers with the number of output channels 64, 128, 256, and 512. In the baseline CNN, each convolutional layer is followed by a local max pooling layer with a kernel size of 2×2 to extract shift-invariant features, whereas the CNN without local pooling and atrous CNN have no local max pooling layers. Particularly, the dilation rates are set to 1, 2, 4, and 8 in the four convolutional layers, respectively. Finally, a global pooling layer and a softmax non-linearity are applied to the final feature maps, in order to predict the probabilities of all scene classes. Notably, only a global attention pooling layer is applied for classification when an attention mechanism is used.

During the training procedure, the CNNs are optimised by an *Adam* optimiser with a batch size of 16 and an initial learning rate of 0.001. The learning rate is reduced by a factor of 0.9 at every 200 iteration steps for stabilising the training procedure. The training procedure is finally stopped at the $15,000$ iteration step.

Table 4.14: The class-wise accuracies [%] of the attention-based atrous CNNs, which lead to the best results on the two subtasks (SubA on the data of device A, and SubB on the data of devices A, B, and C).

Accuracy [%]	**SubA**	**SubB**		
Class	A	A	B	C
airport	59.6	74.0	61.1	38.9
bus	77.7	69.4	66.7	94.4
metro	64.0	81.6	94.4	55.6
metro_station	75.7	82.2	66.7	66.7
park	84.3	86.8	77.8	77.8
public_square	59.3	45.4	50.0	33.3
shopping_mall	88.5	68.1	94.4	100.0
street_pedestrian	52.2	68.0	44.4	61.1
street_traffic	89.4	90.2	83.3	88.9
tram	76.2	66.3	5.6	5.6
Average	72.7	73.2	64.4	62.2

4.5.2 Results and Discussion

The performances of the three types of CNNs in Section 4.5.1 are presented in Table 4.13. To avoid the potential overfitting caused by excessive parameters, flattening is applied to the baseline CNN only, which produces smaller feature maps with a size of 4×20 than the other two CNNs. In the baseline CNNs, the global max, average, and attention pooling layers perform better than flattening on both subtasks. The CNNs without local pooling underperform the baseline CNNs when the global pooling layer is max, average, or attention. The proposed atrous CNN model achieves the best results of accuracies (72.7 % on subtask A, and 73.2 %, 64.4 %, and 62.2 % on subtask B). The proposed model significantly outperforms the CNNs without local pooling, which achieve accuracies of 60.4 % on subtask A, and 61.9 %, 46.7 %, and 52.2 % on subtask B (in a one-tailed z-test, $p < 0.001$ for subtask A and subtask B (devices A and B), and $p < 0.05$ for subtask B (device C)). This indicates that the size of the receptive field can affect the performance more than a local max pooling layer.

The class-wise accuracies are presented in Table 4.14. We can see that, except *tram*, most classes are classified with high accuracies on devices B and C. The reason might be there is noise in the recordings labelled as *tram* from devices B and C.

Due to the atrous CNNs, the size of the feature maps is retained as that of the log mel spectrograms, which is 64×320. The high-resolution feature maps are visualised to explain the CNNs. Therefore, the feature maps in the global attention pooling

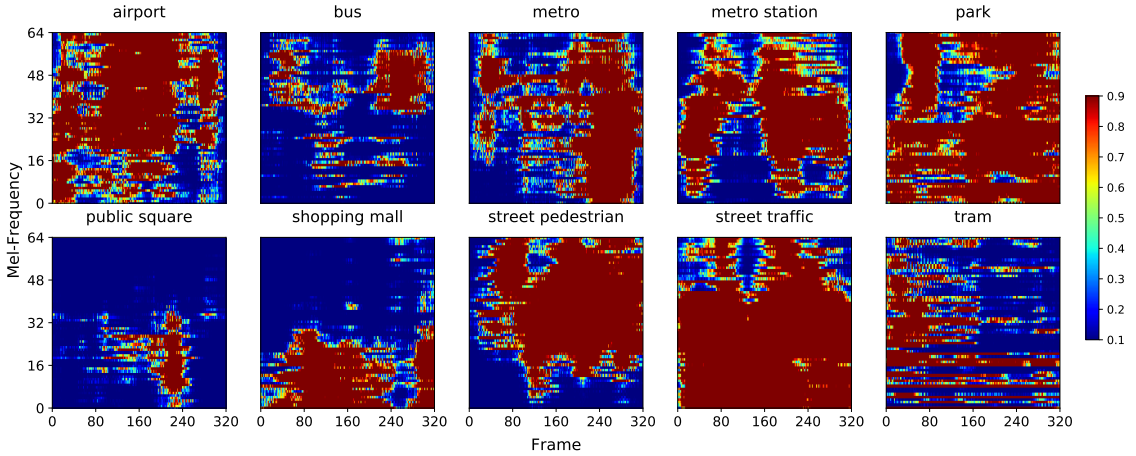


Figure 4.12: Heat maps with a size of 64×320 are the visualisation of the attention tensors in the attention-based atrous CNNs. The horizontal and vertical axes respectively represent the time frames and frequency bins.

layer are presented in Figure 4.12. We can see that each time-frequency bin has different contributions to the ten acoustic scenes. For instance, most time-frequency bins of the heat maps have similar weight values in the classes of *airport*, *park*, and *street traffic*, perhaps due to stationary background noise in these acoustic scenes. In the traffic environments, including *bus*, *metro*, and *tram*, the temporal continuity appears at several mel-frequency bins. The heat maps of *public square*, *shopping mall*, and *street pedestrian* show that there might be audio events such as speech.

4.5.3 Summary

The attention-based atrous CNNs were proposed to visualise and understand the feature maps from the intermediate layers of CNNs with a high resolution. The proposed model performed better than the baseline CNNs and the CNNs without local pooling on the DCASE 2018 database. Moreover, the attention tensors in the CNNs were visualised and analysed.

In future efforts, it is worthwhile to investigate an attention mechanism at the feature level to reach a deeper visualisation of CNNs. To consider the temporal information inside the audio signals, CNNs followed by sequence-to-sequence learning methods (i. e., CRNNs [21]) and three-dimensional CNNs will be focused. Furthermore, as the CNNs share the parameters among the data recorded with three devices in subtask B, it is challenging to learn device-related characteristics for better performance. Therefore, training a device-robust CNN model will be investigated.

4.6 Conditional Atrous CNNs with Attention

In the subtask B of the DCASE 2018 ASC task, multi-device audio data was simply fed into a CNN model together in the experiments of Section 4.5, without considering the device information. To train device-robust CNN models, the proposed approach in Section 3.2.4 will be achieved on both DCASE 2018 and 2019 ASC databases in this experiment. The experimental setup will be introduced in Section 4.6.1, and the results will be presented and discussed in Section 4.6.2. Finally, a summary of this work will be given in Section 4.6.3.

4.6.1 Experimental Setup

Firstly, the 64×320 log mel spectrograms are extracted from the audio waves resampled to 44.1 kHz by a Hamming window with a sample length of $2,048$, an overlap of 672, and 64 mel bins. Next, the log mel spectrograms are fed into a CNN model for the ASC task.

In the experiments, the CNN with local pooling, the CNN without local pooling, and the atrous CNN, are followed by a flattening layer or five global pooling layers (max, average, ROI, attention, and the combination of ROI and attention). In the three kinds of CNNs, four convolutional layers with the number of output channels 64, 128, 256, and 512 are optimised. Specifically, in the CNN with local pooling, each convolutional layer is followed by a local max pooling layer with a kernel size of 2×2. The convolutional layers in the atrous CNN model are designed to have the dilation rates 1, 2, 4, and 8. Furthermore, these CNNs are trained in the four training strategies, including single-device training, joint training, teacher forcing conditional training, and multi-task conditional training. Notably, in the multi-task conditional training, the "CNN-D" model (cf. Figure 3.10(d)) for predicting the device information consists of two convolutional layers with the number of output channels 64 and 128, each of which is followed by a local pooling layer.

During the training procedure, the CNNs are optimised by an *Adam* optimiser with an initial learning rate of 0.001. The learning rate is reduced by a factor of 0.9 at every 200-th iteration. The training procedure is stopped at the $15,000$-th iteration. To obtain a robust "CNN-D" model before training the CNN model for ASC, the value of λ (cf. Equation (3.14)) is experientially set to 1 and 0.0001 before and after the accuracy of "CNN-D" on the test subset achieves 98 %, respectively.

To be consistent with the official evaluation metrics in the ASC tasks of DCASE 2018 and 2019, the accuracy, which is the average of class-wise accuracies, is used in this experiment.

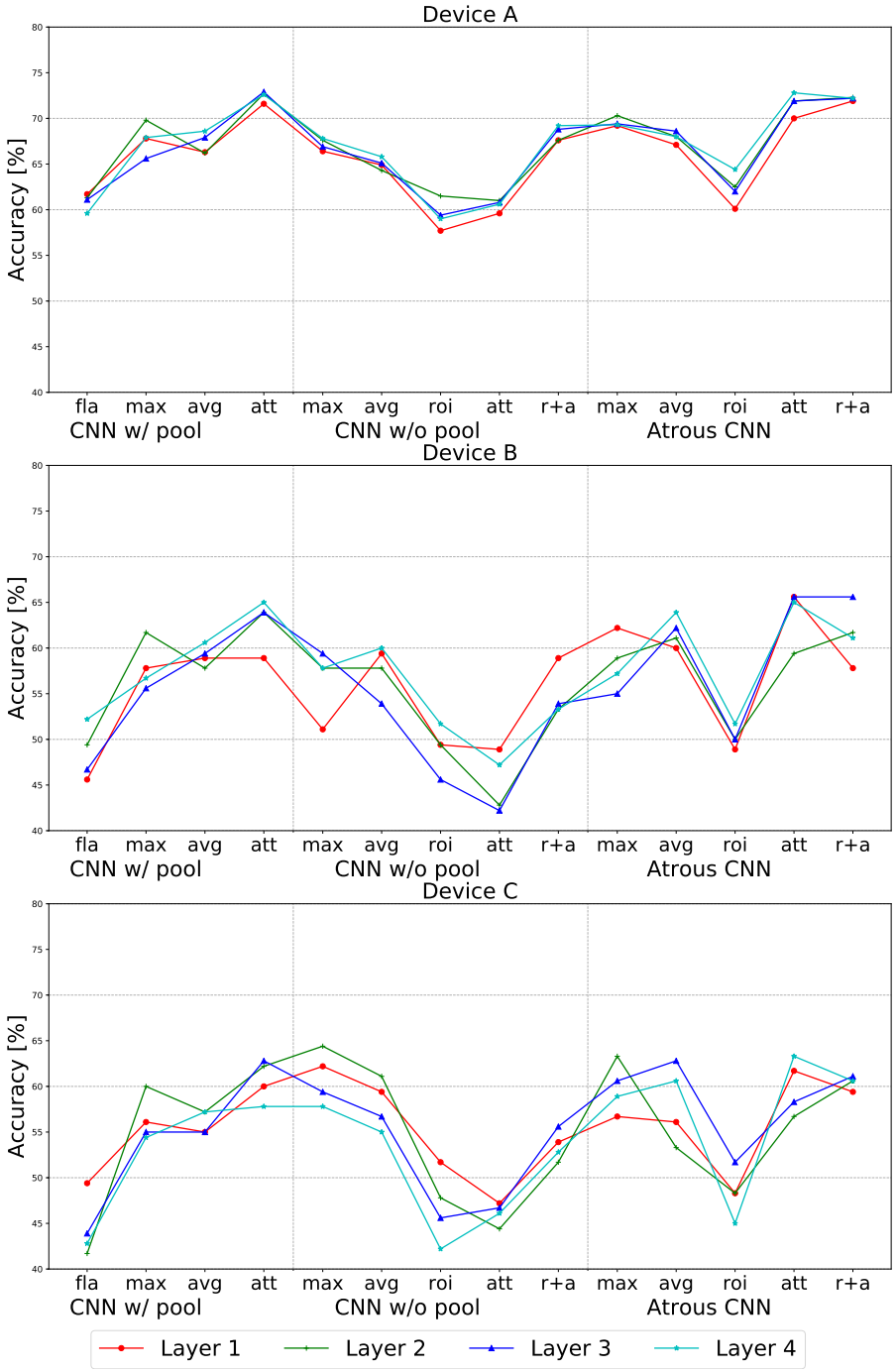


Figure 4.13: Performance (accuracy [%]) comparison of the CNN models evaluated on the DCASE 2018 dataset, when the teaching forcing conditional training works at different convolutional layers. The three CNN topologies contain CNN with local pooling, CNN without local pooling, and atrous CNN. The CNNs are followed by (fla)ttening and five global pooling layers, including max, average (avg), ROI, (att)ention, and the combination of ROI and attention (r+a). The performance is evaluated on the data from the three devices A, B, and C, respectively.

Table 4.15: The average performance (accuracy [%]) comparison of the CNN topologies evaluated on the data from the three devices (A, B, and C) available in the DCASE 2018 dataset, when the teaching forcing conditional training works at different convolutional layers. The three types of CNN topologies contain CNN with local pooling, CNN without local pooling, and atrous CNN. The CNNs are followed by flattening and five global pooling layers, including max, average (avg), ROI, (att)ention, and the combination of ROI and attention (roi+att). The best result chosen from the four layers in each CNN model is highlighted.

Accuracy [%]		**Layer 1**	**Layer 2**	**Layer 3**	**Layer 4**
CNN w/ pool	flatten	**52.2**	50.8	50.6	51.5
CNN w/ pool	max	60.6	**63.8**	58.7	59.7
CNN w/ pool	avg	60.1	60.4	60.8	**62.1**
CNN w/ pool	att	63.5	66.3	**66.5**	65.1
CNN w/o pool	max	59.9	**63.3**	61.9	61.1
CNN w/o pool	avg	**61.2**	61.1	58.6	60.3
CNN w/o pool	roi	**52.9**	52.9	50.2	51.0
CNN w/o pool	att	**51.9**	49.4	49.9	51.3
CNN w/o pool	roi+att	**60.1**	57.5	59.4	58.4
Atrous CNN	max	62.7	**64.2**	61.7	61.8
Atrous CNN	avg	61.1	60.8	**64.5**	64.2
Atrous CNN	roi	52.4	53.6	**54.6**	53.7
Atrous CNN	att	65.8	62.7	65.3	**67.0**
Atrous CNN	roi+att	63.0	64.9	**66.3**	64.6

4.6.2 Results and Discussion

The DCASE 2018 database is used to validate the teacher forcing conditional training, where the device information is conditioned at the four convolutional layers. The performances of the CNNs in the framework of teacher forcing conditional training evaluated on the DCASE 2018 database are depicted and compared in Figure 4.13. We can see that the performances on the data from device A are mostly better than those on the data from devices B and C. The result might be due to the data imbalance among the audio samples from devices A, B, and C, and the potential existence of more noise in the audio waves recorded with mobile devices B and C. When comparing the performances of the CNNs with global pooling layers on the data from each device, the atrous CNNs mostly perform slightly better than the other two (CNN with local pooling and CNN without local pooling). When each of their four convolutional layers is conditioned by the device information, the performances of the CNNs are similar to each other.

In each CNN architecture, to select the best convolutional layer to be conditioned, the average performances on the three-device data are presented in Table 4.15. The global attention pooling method performs the best in the CNNs with local pooling and atrous CNNs, perhaps due to its capability for estimating the contribution of every time-frequency bin in the feature maps to the predictions. Conditioning different layers with the device information can affect the performances. For example, the best performances of CNNs without local pooling are mostly obtained when the first convolutional layer is conditioned. The atrous CNNs, in which the third convolutional layer is conditioned, mostly perform better than those in which the other layers are conditioned. Therefore, the best performances of the teacher forcing conditional training are selected to be compared with the other three training approaches in Table 4.16.

In Table 4.16, the performances of the multi-task conditional training are obtained when conditioning the same convolutional layers as those best layers in the teacher forcing conditional training. Except CNNs without local pooling with global attention pooling and global ROI combined with attention pooling, conditional training outperforms the single-device training and joint training. The reason might be that a huge number of convolutional operations are needed in those two CNN architectures (CNNs without local pooling using global attention pooling and global ROI combined with attention pooing). When comparing the two conditional training approaches, teacher forcing conditional training performs slightly better than multi-task conditional training. This is possibly because the ground truth of the device information is employed in teacher forcing conditional training, while the predicted device information is used in multi-task conditional training. The performances of joint training and single-device training are comparable on the data from devices B and C. The proposed conditional training can learn the device-related complementary features. Therefore, conditional training mostly performs better than the other two training methods on the data from devices B and C. Finally, the best performance of 68.0 % on average, which is obtained by the proposed multi-task CAA-Net, achieves a significant improvement over the performance of 65.0 % obtained by the atrous CNNs with attention in single-device training ($p < 0.01$ in a one-tailed z-test), and the performance of 49.0 % achieved by the CNNs without local pooling using a global ROI pooling layer in single-device training ($p < 0.001$ in a one-tailed z-test).

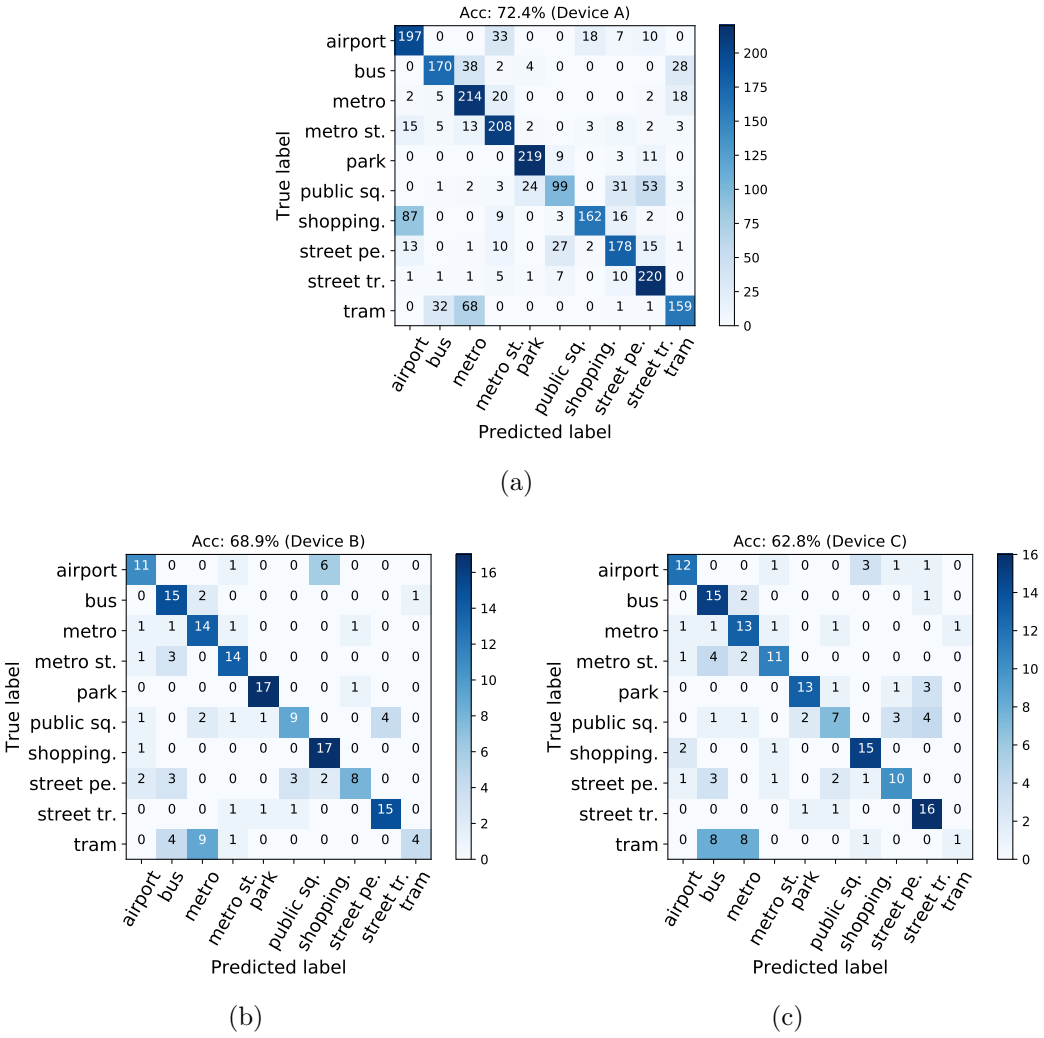
To investigate the class-wise performance of the best model (i. e., multi-task CAA-Net), the confusion matrices are depicted in Figure 4.14. The proposed multi-task CAA-Net produces good performances on some classes, such as *metro*, *park*, and *street traffic*. However, it is difficult to recognise other classes, such as *public square* and *street pedestrian*. The reason for the underperformance might be that the audio recordings in these two classes contain more noise. Moreover, some classes of scenes are classified into similar environments. For example, the class of *tram* is easily recognised as *bus* and *metro*, perhaps due to the similar characteristics in traffic

Table 4.16: Performance (accuracy [%]) comparison of the CNN topologies assessed on the DCASE 2018 dataset using the four training strategies. The performance is evaluated on the data from the three devices *A*, *B*, and *C*. The best result of the four training strategies for each CNN model is highlighted.

Accuracy [%]		**Single-device Training**				**Joint Training [111]**				**Conditional Training**							
										Teacher Forcing				**Multi-task**			
Network	Pooling	*A*	*B*	*C*	*Avg*	*A*	*B*	*C*	*Avg*	*A*	*B*	*C*	*Avg*	*A*	*B*	*C*	*Avg*
CNN w/ pool	flatten	61.1	49.4	40.6	50.4	61.6	49.4	46.7	52.6	61.7	45.6	49.4	52.2	**63.5**	**52.8**	**50.6**	**55.6**
CNN w/ pool	max	70.0	57.8	56.1	61.3	69.8	57.2	57.8	61.6	**69.8**	**61.7**	**60.0**	**63.8**	68.0	58.3	57.8	61.4
CNN w/ pool	avg	63.6	50.0	62.8	58.8	65.8	57.2	57.8	60.3	**68.6**	**60.6**	**57.2**	**62.1**	65.4	60.0	58.3	59.7
CNN w/ pool	att	73.1	51.1	58.3	60.8	72.6	62.2	56.1	63.6	**72.9**	**63.9**	**62.8**	**66.5**	72.6	60.6	59.4	64.2
CNN w/o pool	max	67.8	53.9	56.1	59.3	61.9	46.7	52.2	53.6	**67.6**	**57.8**	**64.4**	**63.3**	65.4	56.7	57.2	59.8
CNN w/o pool	avg	61.9	61.7	42.8	55.5	59.1	54.4	50.0	54.5	**64.9**	**59.4**	**59.4**	**61.2**	66.1	61.7	55.4	61.1
CNN w/o pool	roi	60.4	43.3	43.3	49.0	61.7	50.6	43.9	52.1	57.7	49.4	51.7	52.9	**60.7**	**50.0**	**48.9**	**53.2**
CNN w/o pool	att	**62.7**	**56.7**	**60.0**	**59.8**	59.6	45.0	43.3	49.3	59.6	48.9	47.2	51.9	59.2	46.1	49.4	51.6
CNN w/o pool	roi+att	**67.8**	**61.1**	**60.0**	**63.0**	69.2	56.1	50.6	58.6	67.6	58.9	53.9	60.1	67.5	58.3	53.3	59.7
Atrous CNN	max	66.2	60.6	42.8	56.5	69.7	60.0	59.4	63.0	**70.3**	**58.9**	**63.3**	**64.2**	67.0	61.1	58.9	62.3
Atrous CNN	avg	69.5	61.1	55.6	62.1	67.2	62.8	60.0	63.3	**68.6**	**62.2**	**62.8**	**64.5**	66.9	56.1	55.0	59.3
Atrous CNN	roi	61.2	45.6	40.6	49.1	62.6	48.3	43.9	51.6	**62.0**	**50.0**	**51.7**	**54.6**	63.5	47.2	45.0	51.9
Atrous CNN	att	70.7	61.1	63.3	65.0	73.2	64.4	62.2	66.6	72.8	65.0	63.3	67.0	**72.4**	**68.9**	**62.8**	**68.0**
Atrous CNN	roi+att	72.7	58.9	49.4	60.3	72.2	57.2	56.7	62.0	**72.2**	**65.6**	**61.1**	**66.3**	72.2	56.7	62.2	63.7

Table 4.17: Performance (accuracy [%]) comparison of the CNN topologies assessed on the DCASE 2019 dataset using the four training strategies. The performance is evaluated on the data from the three devices *A*, *B*, and *C*. The best result of the four training strategies for each CNN model is highlighted.

Accuracy [%]		**Single-device Training**				**Joint Training**				**Conditional Training**							
										Teacher Forcing				**Multi-task**			
Network	Pooling	*A*	*B*	*C*	*Avg*	*A*	*B*	*C*	*Avg*	*A*	*B*	*C*	*Avg*	*A*	*B*	*C*	*Avg*
CNN w/ pool	att	67.2	47.0	47.0	53.7	69.9	44.1	49.8	54.6	70.9	46.7	48.1	55.2	**70.9**	**46.1**	**49.1**	**55.4**
CNN w/o pool	att	60.9	48.9	48.3	52.7	63.5	36.9	36.5	45.6	**65.0**	**50.6**	**56.3**	**57.3**	63.3	43.3	46.3	51.0
Atrous CNN	att	72.2	45.4	50.2	55.9	71.4	44.3	50.9	55.5	72.0	43.7	51.1	55.6	**71.3**	**44.4**	**52.6**	**56.1**



(a)

(b)

(c)

Figure 4.14: Confusion matrices of the results on the data from devices A, B, and C obtained by the proposed multi-task CAA-Net, which achieves the best performance on the DCASE 2018 dataset.

sounds. This suggests that despite the performance is improved by considering the device information, the type of acoustic scenes could be an additional factor to be considered in the ASC task.

Apart from the performances on the DCASE 2018 database, the effectiveness of the proposed conditional training approach is also validated on the DCASE 2019 database. Table 4.17 presents the performances of the three CNNs with attention, including CNN with local pooling, CNN without local pooling, and atrous CNN. The device information is used to condition the same convolutional layers as those best layers in Table 4.15. The proposed conditional training achieves the best per-

formances among the four training methods on the DCASE 2019 database. This is consistent with the results on the DCASE 2018 database, indicating that the proposed conditional training is more effective and robust than single-device training and joint training on multi-device data sets.

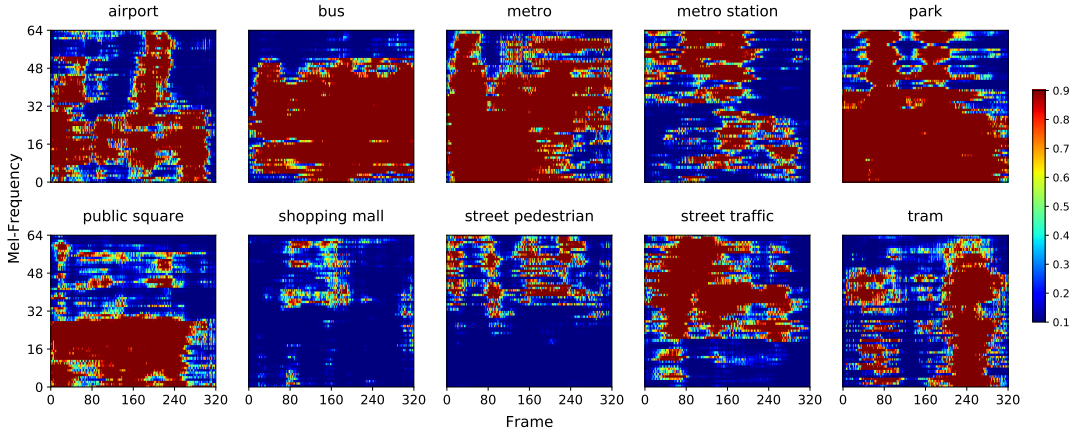
To further visualise the intermediate layers of CNNs, the heat maps of the attention tensors in the proposed multi-task CAA-Net are depicted in Figure 4.15. The audio examples at each class of scene were recorded at the same time and place by the three devices. In Figure 4.15, the attention tensors are mostly similar across the data from the three devices. In some classes, e. g., *airport*, *metro*, *public square*, and *tram*, more time-frequency bins in the heat maps on the data from device A are learnt with high values than those in the heat maps on the data from devices B and C. The reason might be that the data qualities of the three devices differ from each other. Some characteristics of specific acoustic scenes can be seen in the heat maps. For example, the heat maps of traffic scenes, such as *bus*, *metro*, and *tram*, consist of time-continuity bins. Non-traffic sounds, like speech and bird sound, may exist in some classes, such as *airport*, *metro station*, *park*, *shopping mall*, *street pedestrian*, and *street traffic*, so that the time-frequency bins concentrate on high-frequency areas in these environments.

To make an understandable decision with machines, the visualisation of the CNNs is essential in real-life applications, especially in security-sensitive fields. Moreover, since the contribution of each bin in the heat maps to the predictions may be relative to the time and frequency of sound events, the visualisation of the heat maps can offer the potential to enhance the performance of other tasks, such as sound event detection [211].

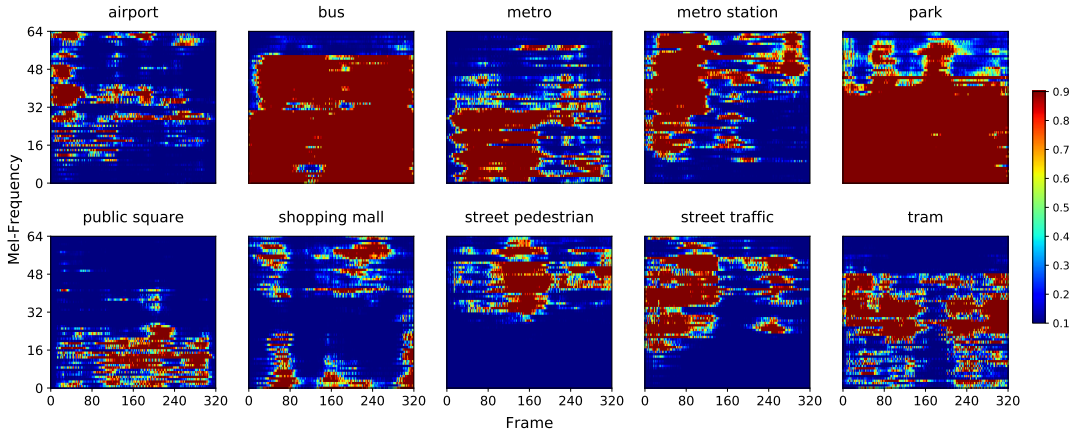
4.6.3 Summary

In summary, the CAA-Net was proposed for visualising and understanding the CNNs on multi-device data. Particularly, in the multi-task CAA-Net, log mel spectrograms were extracted from the audio signals, and fed into two CNN models to predict the acoustic scenes and the device information, respectively. The produced device information was represented by one-hot encoders, and then used to condition the CNN model for ASC, which consists of four dilated convolutional layers and a global attention pooling layer. The dilated convolutional layers were employed to preserve the size of the feature maps, and the attention mechanism was used to estimate the contribution of each time-frequency bin to the predictions. The proposed conditional training outperformed single-device training and joint training. The attention mechanism was visualised and analysed in this study.

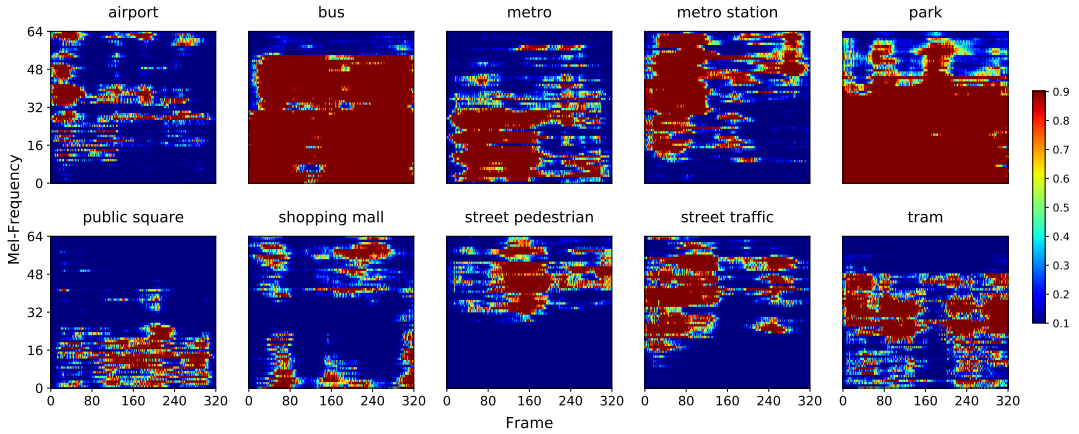
In future efforts, learning the value of λ in Equation (3.14) will be investigated, since the fixed value of λ is not flexible to apply the proposed approach to a new database. A hierarchical structure to classify both the types of scenes and device classes will be considered due to the aforementioned effect of scene types. Further-



(a) Heat maps of CNNs on data from device A



(b) Heat maps of CNNs on data from device B



(c) Heat maps of CNNs on data from device C

Figure 4.15: Heat maps with a size of 64×320 are obtained by visualising the attention tensors in the multi-task CAA-Net which works on the DCASE 2018 dataset. The horizontal and vertical axes represent the time frames and frequency bins, respectively.

more, training of cross-database CNNs will be explored through involving data from more recording devices, so that the CNNs are effective on audio data recorded with more devices. In practice, to mitigate the burden from the computation complexity caused by an amount of multi-device data, the training strategies, such as transfer learning [212] and lifelong learning [213], will be considered. The heat maps of the attention mechanism will also be utilised for other tasks, such as audio event detection.

4.7 Protecting DL Models against White-box Adversarial Attacks

To achieve the proposed adversarial training in Section 3.3.2, the white-box adversarial attacks are generated, and the SER models are protected by adversarial training on the DEMoS database. The experimental setup will be introduced in Section 4.7.1, and the results will be presented and discussed in Section 4.7.2. Finally, the conclusion of this experiment will be given in Section 4.7.3.

4.7.1 Experimental Setup

The log mel spectrograms are firstly extracted from the speech samples resampled from 44.1 kHz to 16 kHz, since the sampling rate of 16 kHz may lead to faster progress, and the approaches in our early experiments achieved similar results on the data with the two sampling rates. During the extraction of log mel spectrograms, the length of the window size is set to 512, the length of the overlap is 256, and 64 mel bins are extracted for each time frame in a log mel spectrogram. Then, the log mel spectrograms are broadcasted to have a unified time length of 373, therefore all log mel spectrograms have a size of 373×64. Next, the log mel spectrograms are fed into CNNs for the SER task. Three CNN architectures are employed in this experiment: CNN-4, ResNet-50 [214, 215], and VGG-16 [100]. Specifically, the CNN-4 model consists of four convolutional layers with the number of output channels 64, 128, 256, 512 and a kernel size of 5×5, a global max pooling layer, a fully connected layer, and a softmax layer. Each convolutional layer is followed by a local max pooling layer with a kernel size of 2×2.

During the training procedure, each CNN model is optimised by an *Adam* optimiser with a learning rate of 0.001. To improve the stabilisation of the training procedure, the learning rate is reduced with a factor of 0.9 at every 100-th training iteration. The training procedure is finally stopped at the $10,000$-th iteration. The performances on both adversarial and real data are evaluated by UAR.

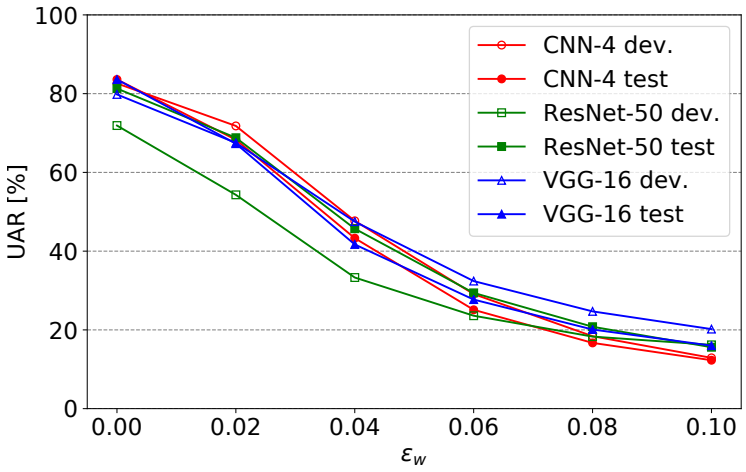


Figure 4.16: The performance (UAR [%]) of the CNN models obtained by single training on the adversarial (dev)elopment and test data generated from the DEMoS database. The adversarial development/test data is equal to the real data when $\epsilon_w = 0.00$.

4.7.2 Results and Discussion

To verify the effectiveness of the adversarial data generated by FGSM, the three CNN models are firstly trained on the real training data as the baseline (called single training). The performances of the single training on the adversarial development and test data are shown in Figure 4.16. Particularly, the performances are evaluated on the real data when $\epsilon_w = 0.00$. All of the six models perform well on the real data with UAR values of around 80 %. The UAR values decrease when ϵ_w increases, indicating that the generated adversarial data can successfully attack the CNN models when the perturbations become bigger.

Furthermore, the performances of the vanilla and similarity-based adversarial training frameworks are presented in Table 4.18. Due to the data augmentation with fake data, the performances on the real data are mostly improved. Both of the two training approaches perform well on the fake data, although their performances are slightly worse than those on the real data. Especially, the performance on the fake data becomes worse when the ϵ_w increases, perhaps because that a bigger value of ϵ_w can affect the data distribution. When comparing the two training approaches, the vanilla adversarial training mostly outperforms the similarity-based adversarial training on the real data, and the similarity-based adversarial training performs better on the fake data than the vanilla one. This indicates that the proposed similarity-based adversarial training can effectively defend against the attacks. The loss function in the similarity-based adversarial training reduces the difference between the features learnt from the real and fake data, but it affects the representations learnt from the real data.

Table 4.18: Performance (UAR [%]) comparison of the three CNN topologies in the three training strategies: single training, (van)illa (adv)ersarial training, and (sim)ilarity-based (adv)ersarial training. The CNN models are validated on both the real data and the fake (i. e., adversarial) data in the (dev)elopment and test sets of the DEMoS Corpus.

		CNN-4				**ResNet-50**				**VGG-16**			
	UAR [%]	Real		Fake		Real		Fake		Real		Fake	
NN	ϵ_w	*Dev.*	*Test*	*Dev.*	*Test*	*Dev.*	*Test*	*Dev.*	*Test*	*Dev.*	*Test*	*Dev.*	*Test*
Single Training	.00	82.6	83.6	–	–	71.9	81.3	–	–	79.8	83.6	–	–
Van. Adv. Training	.02	82.5	85.6	74.4	80.0	69.9	81.7	62.0	77.4	85.0	84.7	79.4	80.6
Van. Adv. Training	.04	81.7	87.1	65.7	74.9	81.3	85.0	68.5	75.5	84.9	85.5	74.3	78.3
Van. Adv. Training	.06	85.4	86.9	57.6	67.1	77.4	83.9	59.5	67.2	87.1	87.8	74.1	77.0
Van. Adv. Training	.08	85.3	85.8	52.0	54.0	81.3	85.5	60.7	71.0	**87.5**	**86.7**	70.9	75.6
Van. Adv. Training	.10	86.6	88.0	45.7	57.0	82.3	84.5	60.2	67.8	84.2	87.0	63.8	71.6
Sim. Adv. Training	.02	84.4	79.7	82.7	78.4	74.3	79.8	73.2	77.1	84.7	82.3	**83.9**	**82.1**
Sim. Adv. Training	.04	82.2	82.5	77.2	76.9	70.8	79.8	65.2	75.9	81.4	84.2	80.6	82.0
Sim. Adv. Training	.06	77.5	82.4	67.5	73.1	72.3	78.8	63.0	72.8	78.6	83.9	75.3	81.5
Sim. Adv. Training	.08	73.9	80.6	61.0	67.4	63.1	80.3	45.0	71.4	79.2	65.3	73.0	55.0
Sim. Adv. Training	.10	72.7	75.2	52.6	58.5	40.7	73.4	31.6	49.8	80.4	83.3	71.6	76.9

The VGG-16 model performs the best among the three CNN models. However, ResNet-50 performs the worst on both real and fake data, perhaps because ResNet-50 consists of more convolutional layers in the Inception architecture than the other two CNN models. The performances of the CNN models are highly related to the number of layers, so that too many convolutional layers may slow down the convergence. Moreover, VGG-16 is more stable and more robust than CNN-4. We think there are two possible reasons: i) more convolutional layers can help to extract higher-level representations, and ii) it may be easier for a VGG-16 model to be optimised on the fake data, as more convolutional layers can increase the difference between the representations from the real and fake data. Finally, the best performances of the adversarial training on the real data (development: 87.5 %, test: 86.7 %) significantly outperform the performances of single training (development: 82.6 %, test: 83.6 %) ($p < 0.001$ in a one-tailed z-test).

The best results on the real data (development: 87.5 %, test: 86.7 %) are compared to the state-of-the-art methods for data augmentation in Table 4.19. We can see that WaveNet achieves a good performance, especially on the development set. However, this WaveNet-based method was used for a binary classification task, instead of classifying seven emotional classes [216]. When comparing the methods for seven-class classification, the proposed approach performs significantly better than the data (raw audio waves and log mel spectrograms) augmentation methods using random noise ($p < 0.001$ in a one-tailed z-test).

Table 4.19: Performance (UAR [%]) comparison between the proposed approach and other data augmentation methods on the DEMoS corpus.

UAR [%]	*Dev.*	*Test*
WaveNet (two classes) [216]	85.7	74.1
Raw audio augmentation by random noise	79.5	83.3
Spectrogram augmentation by random noise	80.8	83.3
The proposed approach	**87.5**	**86.7**

4.7.3 Summary

A training framework was proposed to train a robust SER model against adversarial attacks. The vanilla and similarity-based adversarial training approaches were employed to work on the three CNNs, including CNN-4, ResNet-50, and VGG-16. In the experiments, adversarial training performed better on the real data than single training due to the data augmentation with the adversarial data. The similarity-based adversarial training helped the CNN models perform better on the fake data than the vanilla adversarial training.

In future work, since only the white-box adversarial attacks are generated in this experiment, the black-box adversarial attacks will be investigated to approach the practical situation that the parameters of the target model are unknown for an attacker. Furthermore, transferring the fake data across multiple deep learning models might be helpful to validate models' robustness. Training a detector to recognise the fake data is also promising to improve the performances of the models on the fake data.

4.8 Improving Transferability of Black-box Adversarial Attacks

To improve the transferability of black-box adversarial attacks for deceiving SER models, the proposed lifelong learning approach in Section 3.3.3 is achieved in this experiement. In the following, the experimental setup will be introduced in Section 4.8.1, and the results will be presented and analysed in Section 4.8.2, followed by a summary of this study in Section 4.8.3.

4.8.1 Experimental Setup

Similar to the experimental setup in Section 4.7.1, the 373×64 log mel spectrograms are extracted from the speech signals with a sampling rate of 16 kHz. An atrous CNN model (cf. Section 3.3.1.2) is trained to generate the adversarial data. This

atrous CNN model consists of four convolutional layers with the number of output channels 64, 128, 64, and 1, the dilation rates of 1, 2, 4, and 8, and a kernel size of 5×5. Each convolutional layer is followed by a batch normalisation and a ReLU (for the first three convolutional layers) or a sigmoid (for the final convolutional layer) activation function. To generate the minimal perturbations using the attack model, a sigmoid function, which produces data values in $[0, 1]$, is assumed to be more helpful for the optimisation than a ReLU function which produces data values in $[0, +\infty]$. Furthermore, to validate the effectiveness of the generated adversarial data, the three pre-trained CNN models in Section 4.7, including CNN-4, VGG-16, and ResNet-50, are employed in this experiment. A final log-softmax layer is used in all of the three CNN models.

During the training procedure, the attack model is optimised by an *Adam* optimiser with an initial learning rate of 0.0001. To stabilise the training procedure, the learning rate is reduced by a factor of 0.9 at every $1,000$-th iteration. The training procedure is finally stopped at the $10,000$-th iteration. In particular, the attacker is trained for $1,000$ iterations to deceive a new defence in the lifelong learning framework. The hyperparameter α (cf. Equation (3.17)) is experientially set to 0.02, and λ (cf. Equation (3.21)) is optimised from $\{1e4, 1e5, 1e6, 1e7\}$ on the development set.

To explore the effect of the defence orders, an attacker is trained to deceive two types of defence sequences: one is going deeper (CNN-4 $\rightarrow$ VGG-16 $\rightarrow$ ResNet-50) (i. e., clockwise), and the other is going shallower (ResNet-50 $\rightarrow$ VGG-16 $\rightarrow$ CNN-4)) (i. e., counterclockwise). In the experiment on each sequence, the attackers are trained to fool either the first two defence models or the whole sequence, and then utilised to attack all of the three models. Finally, the UAR is used to evaluate the classification performance.

4.8.2 Results and Discussion

The performances of the generated fake data are evaluated on the development set, as shown in Figure 4.17. A strong attack model can lead to a low UAR value. We can see that, in each defence sequence, a bigger value of λ means more constraint to the parameters of the attack model, therefore more prior knowledge from the previous target model is remembered. For instance, in the clockwise sequence, the attack model can better deceive the CNN-4 model if using $\lambda = 1e7$ rather than $\lambda = 1e4$. While comparing the performances on the two kinds of sequences, the attacker in the clockwise sequence performs better on the third model ResNet-50 than the attack model in the counterclockwise sequence working on CNN-4. The reason might be that a shallower target model is helpful to train a more transferable attack model than a deeper one. Therefore, to improve the transferability of an attacker, it is necessary to remember more prior knowledge learnt from a shallower defence than that from a deeper one. In the experiment, the attacker trained on the clockwise

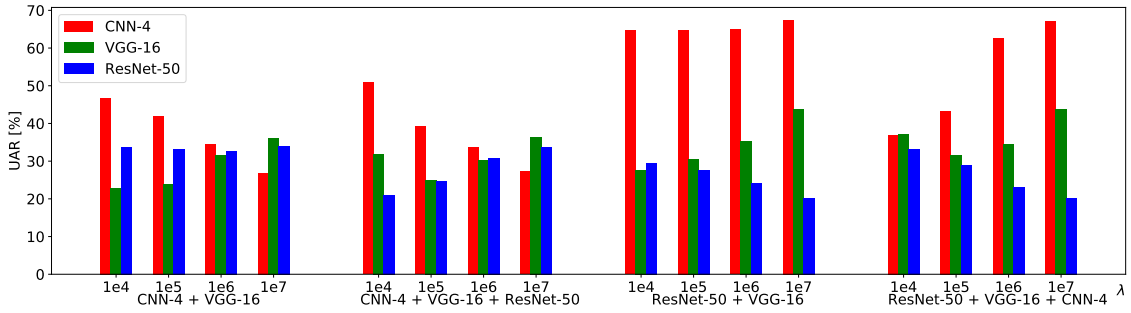


Figure 4.17: The performances (UAR [%]) of lifelong learning with various λ values on the development set. The attack models are trained on the four sequences of target models (bottom), and the transferability is tested on the three target models (CNN-4, VGG-16, and ResNet-50).

sequence requires a bigger value of λ ($1e6$ on the two-defence sequence, and $1e5$ on three), while the attack model on the counterclockwise needs a smaller λ (($1e4$ on the two-defence sequence, and $1e5$ on three). Finally, the attack models trained on both of the whole sequences can effectively deceive the three targets models.

In Table 4.20, the results of lifelong learning are compared to the other three learning methods, including single-task learning, multi-task learning, and transfer learning. Single-task learning performs the worst on new target models, since an attacker learnt on one target model has a low transferability. Multi-task learning trains the most transferable attacker, yet, it is time- and space-consuming. It is challenging to train a highly transferable attack model using transfer learning, as the attack model forgets the prior knowledge during the fine-tuning procedures, particularly in the results of the clockwise sequences. In contrast, as lifelong learning trains the attack model by remembering the prior knowledge, the attack model can fool target models which have been attacked in the training procedure.

When comparing the results of transfer and lifelong learning on the two types of sequences, lifelong learning outperforms transfer learning on the clockwise sequences, whereas transfer learning performs comparably with lifelong learning on the counterclockwise sequences. This is consistent with the aforementioned analysis, which pointed out that the attack models trained on the clockwise sequences need more constraint than those on the counterclockwise sequences, since a shallower target model can lead to a more transferable attacker. Transfer learning can be considered as a special case of lifelong learning without constraint. Therefore, the attack model is adapted to fool shallow defence models in the counterclockwise sequences by remembering less prior knowledge in transfer learning than in typical lifelong learning. In practice, it is difficult to know how deep a new defence model is. In this regard, transfer learning is not able to train a transferable attack model when the new target model is deeper than the existed defences. With the capability

Table 4.20: The performance (UAR [%]) comparison of the four learning methods on the (dev)elopment and test sets. The attackers are learnt on different defence sequences. Then, the generated fake data are utilised to attack the three target models (CNN-4, VGG-16, and ResNet-50).

		CNN-4		VGG-16		ResNet-50	
Method	UAR [%]	Dev.	Test	Dev.	Test	Dev.	Test
Single-task learning	CNN-4	28.5	21.9	47.5	46.1	39.3	54.5
	ResNet	68.8	56.0	49.7	42.1	20.0	20.7
Multi-task learning	CNN-4 + VGG-16	29.4	23.0	21.8	17.9	31.5	49.1
	ResNet-50 + VGG-16	63.5	52.6	24.0	19.8	21.2	22.9
	CNN-4 + VGG-16 + ResNet-50	33.1	28.0	22.9	18.5	21.0	23.9
Transfer learning	CNN-4 + VGG-16	45.2	48.6	19.3	16.1	30.8	54.7
	CNN-4 + VGG-16 + ResNet-50	52.7	41.4	30.7	28.8	18.5	18.9
	ResNet-50 + VGG-16	59.3	53.0	22.9	18.1	25.1	37.6
	ResNet-50 + VGG-16 + CNN-4	24.9	22.0	30.2	29.0	24.0	34.2
Lifelong learning	CNN-4 + VGG-16	34.6	30.1	31.4	29.7	32.5	46.9
	CNN-4 + VGG-16 + ResNet-50	39.2	38.5	25.0	26.4	24.5	33.4
	ResNet-50 + VGG-16	64.7	54.1	27.4	21.7	29.5	36.3
	ResNet-50 + VGG-16 + CNN-4	43.3	33.9	31.6	29.4	28.8	30.9

of remembering the prior knowledge, lifelong learning can help to train an attack model which can deceive the new target models and the target models which have been fooled during training.

4.8.3 Summary

In this study, an atrous CNN model was trained as the black-box adversarial attacker, and was improved to be highly transferable by lifelong learning. In the proposed lifelong learning framework, the attacker was trained to successfully deceive the three CNN models, including CNN-4, VGG-16, and ResNet-50. Moreover, after analysing the effect of different defence orders, a stricter constraint is required to train an attack model on a sequence from shallow to deep models than an inverse one.

Although the attack models' transferability has been improved with EWC by updating the parameters in the lifelong learning framework, it is still restricted by the limited parameters. In future efforts, more lifelong learning approaches, such as training dynamically expandable networks [217], will be explored to further improve the transferability of the attack models. For verification, more defence models will be used. As the attack model in this study generated a specific adversarial perturbation for each real speech sample, it is worthwhile to focus on universal black-box adversarial attacks, which generate a universal perturbation for all real data. Lifelong

learning will be further applied to enhance the transferability of universal black-box adversarial attacks.

5 Conclusions and Outlook

Audio signals are an essential part of interactions among humans and perceptions of the world. Computer audition techniques teach machines to perceive audio signals, and in its development, automatic audio signal classification, which attempts to predict a label for an audio signal, plays a crucial role. Consequently, audio signal classification has become increasingly popular in a number of applications in the real world, such as ASC, HSC, SER, etc. Considerable efforts, especially in deep learning, have been made with the aim of training accurate, robust, and reliable neural networks for audio signal classification. However, the development of audio signal classification techniques is still limited by many open challenges, including extracting high-level deep representations, explaining DNNs, and training robust DNNs against external attacks.

To tackle these challenges, this thesis investigates several novel deep-learning-based algorithms to classify audio signals. In particular, this work presents and evaluates a variety of approaches to achieve the three major objectives: i) learning advanced and effective representations, ii) improving the interpretability of DNNs, and iii) learning robust DNNs through adversarial training approaches.

In the following, this chapter summarises the presented approaches and results in Section 5.1. Then, the ethics in computer audition is discussed in Section 5.2. Finally, the limitations of the current work and suggestions of the future research directions are given in Section 5.3.

5.1 Achievements

In this section, the achievements of this thesis are summarised as follows.

In the first research contribution of extracting high-level representations (cf. Section 1.2), transfer learning from pre-trained image classification models was developed to process the time-frequency representations of audio signals. In particular, with the introduction of the pre-trained CNN models in Section 3.1.1, the state-of-the-art transfer learning frameworks were proposed in Section 3.1.2 to extract deep

spectrum representations. Moreover, to improve the performance, a novel transfer learning framework was proposed in Section 3.1.3 by dealing with multiple types of time-frequency representations.

For deep learning models explanation as the second contribution, the attention mechanism was applied to visualise DNNs by estimating the contributions of each bin to the predictions. Firstly, an attention mechanism was introduced in Section 3.2.1, and attentions at the frame level and the time-frequency level were proposed in Section 3.2.2. The attention at the frame level explains the contribution of each time frame in the time-frequency representations to the predictions, whereas the goal of the attention at the time-frequency level is to estimate each time-frequency bin's contribution. Furthermore, to visualise the attention mechanism with a high resolution, atrous CNNs were proposed to retain the size of the feature maps to be the same as that of the input (cf. Section 3.2.3).

To extend the second contribution, i. e., explaining deep learning models in multi-device conditions, this thesis introduced novel conditionally trained atrous CNNs with attention (cf. Section 3.2.4). In the framework of conditional training, the device information is fed into the atrous CNNs with attention for classification in order to train a device-robust CNN model. In detail, two sub-approaches were deliberated in conditional training: one is teacher forcing conditional training, which uses the known device information, and the other is multi-task conditional training, in which the device information is predicted via a separate CNN model. The experimental results demonstrated that the proposed conditional training can learn more robust models on multi-device data, yielding appealing performance improvements compared with single-device training and joint training.

Another contribution of this thesis is to train robust deep learning models against adversarial attacks (cf. Section 3.3). The approaches of generating white-box and black-box adversarial attacks were introduced in Section 3.3.1. Furthermore, to improve the robustness of deep learning models, the adversarial training approaches were employed to process both real and generated adversarial data in one training procedure (cf. Section 3.3.2). In particular, the similarity-based adversarial training was proposed to protect the CNNs against white-box adversarial attacks better than vanilla adversarial training in the experiments.

As an extension of the above contribution, this thesis is further dedicated to improving the transferability of adversarial attacks, especially black-box adversarial attacks (cf. Section 3.3.3). For this purpose, a lifelong learning framework was proposed to train an attack model in order to deceive multiple targeted defence models. The experiments indicated that lifelong learning performs better than single-task learning and transfer learning.

All in all, the research works presented in this thesis have demonstrated that the proposed deep learning approaches bear the potential to contribute to the development of audio signal classification applications, delivering good performances as well as interpretable and robust models.

5.2 Ethics in Computer Audition

During collecting and processing data for computer audition, a series of ethical considerations are demanded. In this section, the existing ethical topics in computer audition are listed and discussed.

Applications. In pure research experiments, there might be no harm because of guaranteed privacy for subjects [218]. Applications should be crucially required to ensure privacy for using (or monitoring) massive data. For example, in the current COronaVIrus Disease 2019 (COVID-19) pandemic, applications for COVID-19 diagnosis based on respiratory sounds (e. g., coughing and breathing) should always protect the users' privacy to aid in the prevention of COVID-19 transmission. Further concerns will be related to legal and societal implications [219].

Anonymisation and personalisation. Because of various experimental targets, subject information is not needed in some research studies that focus on specific phenomena, e. g., ASC [218]. However, subject information is required in some other research domains, such as therapy and teaching. Applications of computer audition can be either anonymous or personalised. For instance, personal information is necessary when we are only interested in the general emotional reactions of people to a movie. In contrast, personalisation is strictly enforced when a subject is monitored to analyse his or her emotional changes.

Data collection and usage. The forms of data collection can be either experimental or crowd-sourcing. When personal information is recorded, ethical considerations are required during data storage, transmission, and usage [220]. To protect the collected data, firewalls, encryption, and authentication are often used.

Multiple modalities. Apart from audio signals, other modalities (e. g., facial expressions, body gestures, etc) can benefit many computer audition tasks, such as emotion recognition. As personalisation is easier based on multi-modal data, ethical considerations are essential to ensure the anonymisation is stricter when multiple modalities are analysed [218].

The above descriptions discussed general ethical considerations in computer audition. For more details in specific research domains, such as computational paralinguistics, the readers are suggested to refer to the related studies [218, 220, 221].

5.3 Limitations and Future Perspectives

Despite that a set of deep learning techniques have been presented in this thesis to address some challenges of audio signal classification, several potential research directions are still worthwhile to be investigated in the future.

Small-scale datasets processing. Although a large-scale dataset is helpful to train deep learning models, a recent trend in machine learning is to train models based on small-scale data, since there are still a number of small databases in real

life. A potential solution is to augment the training data through synthesising new data using GANs [216, 222]. Additionally, to train a DNN model on small quantities of data only, it is promising to apply the recently developed machine learning algorithms, including few-shot learning [223], one-shot learning [224], and zero-shot learning [225].

Explainable AI. Investigating approaches for explainable AI can boost the applications of audio signal classification in practice, yielding transparency and reliable deep learning systems. Apart from the attention mechanism presented in this thesis, it will be interesting and useful to explore other approaches in explainable AI. Several approaches of correlating the regions in the input to each prediction are potential to be considered in audio signal classification, e. g., LIME [105], layer-wise relevance propagation [226], etc. Furthermore, constructing a decision tree [29] is helpful to clarify the specific reason for each prediction, and building a graph [106] can better reveal the knowledge hierarchy hidden inside a pre-trained classifier.

Adversarial audio data generation. This thesis presented generating adversarial attacks from the log mel spectrograms only, and most studies focused on adversarial attacks of images. Therefore, it is worthwhile to generate adversarial audio signals in the future [146]. Two investigation directions in adversarial attacks for audio signals are essential: one is to explore algorithms for generating adversarial data, such as one-pixel attack [227] which generates effective adversarial data with very small changes of the original data, and the other is to improve the robustness of the deceived models by several state-of-the-art machine learning algorithms, e. g., GANs.

In summary, all algorithms and approaches developed in this thesis reached the goals described in the introduction, that is, to improve the effectiveness, explainability, and robustness of audio signal classification models. Hopefully, the work presented herein can inspire other researchers in this community, and expedite the pace of developing deep learning systems in computer audition and the relative applications in real life.

Acronyms

AAD ..... Aalborg University

AI ..... Artificial Intelligence

ASC ..... Acoustic Scene Classification

AUC ..... Area under the ROC Curve

AUTH ..... Aristotle University of Thessaloniki

BGRU ..... Bi-directional Gated Recurrent Unit

BLSTM ..... Bi-directional Long Short-Term Memory

BoAW ..... Bag-of-Audio-Words

BRNN ..... Bi-directional Recurrent Neural Network

CAA-Net ..... Conditional Atrous Convolutional Neural Network with Attention

CinC ..... Computing in Cardiology

CNN ..... Convolutional Neural Network

ComParE ..... Computational Paralinguistic ChallengE

CRNN ..... Convolutional Recurrent Neural Network

CVD ..... Cardiovascular Disease

CWT ..... Continuous Wavelet Transform

DAP ..... Duessel-dorf Acute Pain

DCASE ..... Detection and Classification of Acoustic Scenes and Events

DCT ..... Discrete Cosine Transform

DEMoS ..... Database of Elicited Mood in Speech

DLUT ..... Dalian University of Technolog

DNN . . . . . Deep Neural Network
eGeMAPS . . . . . extended Geneva Minimalistic Acoustic Parameter Set
EWC . . . . . Elastic Weight Consolidation
FCN . . . . . Fully Connected Network
FGSM . . . . . Fast Gradient Sign Method
FNN . . . . . Feedforward Neural Network
FPR . . . . . False Positive Rate
GAN . . . . . Generative Adversarial Network
GRU . . . . . Gated Recurrent Unit
HCI . . . . . Human-Computer Interaction
HSC . . . . . Heart Sound Classification
HSS . . . . . Heart Sounds Shenzhen
ILSVRC . . . . . ImageNet Large Scale Visual Recognition Competition
LIME . . . . . Local Interpretable Model-agnostic Explanations
LLD . . . . . Low-Level Descriptor
LSTM . . . . . Long Short-Term Memory
MAcc . . . . . Mean Accuracy
MFB . . . . . Mel-Frequency Band
MFCC . . . . . Mel-Frequency Cepstral Coefficient
MIT . . . . . Massachusetts Institute of Technology
MSE . . . . . Mean Squared Error
MSV . . . . . Margin Sampling Value
NPRS . . . . . Numeric Pain Rating Scale
PLE . . . . . Pain Level Evaluation
ReLU . . . . . Rectified Linear Unit
RGB . . . . . Red, Green, and Blue
RNN . . . . . Recurrent Neural Network
ROC . . . . . Receiver Operating Characteristic
ROI . . . . . Region of Interest
SER . . . . . Speech Emotion Recognition

SGD ..... Stochastic Gradient Descent
STFT ..... Short-Time Fourier Transform
SUA ..... Shiraz University Adult
SVM ..... Support Vector Machine
TNR ..... True Negative Rate
TPR ..... True Positive Rate
UAR ..... Unweighted Average Recall
UHA ..... University of Haute Alsace
WAR ..... Weighted Average Recall

List of Symbols

N ..... number of samples

$\boldsymbol{x}$ ..... feature vectors

x_i ..... feature vector, indexed by i

y ..... labels

y_i ..... label, indexed by i

$P(\boldsymbol{x})$ ..... prior distribution of features

$P(y)$ ..... prior distribution of labels

$P(\boldsymbol{x}, y)$ ..... joint probability distribution

$P(y|\boldsymbol{x})$ ..... posterior probability

$P(\boldsymbol{x}|y)$ ..... likelihood probability

$\boldsymbol{w}$ ..... weight values in a classification model

b ..... biases in a classification model

C ..... complexity parameter

α_i ..... slack variable, indexed by i

s ..... audio signal

f ..... spectrogram

w ..... window function of STFT

ω ..... angular frequency

t_s ..... translation distance of the sliding window in STFT

f_{mel} ..... mel spectrogram

f_{mfcc} ..... MFCCs

N_{mel} ..... dimensional number of mel frequency

N_{mfcc} ..... dimensional number of MFCCs

Ψ_{bump} ..... wavelet basis of *bump* wavelet transform

ϵ ..... wavelet scale

μ, σ ..... constant wavelet parameters

Ψ_{morse} ..... wavelet basis of *morse* wavelet transform

u ..... unit step

β ..... normalising constant

a^2 ..... time-bandwidth product

γ ..... symmetry parameter

N_l ..... number of layers in a DNN model

$\boldsymbol{h}^{n-1}$ ..... input activation/feature map of the n-th layer

$\boldsymbol{h}^{n}$ ..... output activation/feature map of the n-th layer

$\boldsymbol{w}^{n}$ ..... weights of the n-th layer

b^{n} ..... biases of the n-th layer

C^{n-1} ..... channel number of the outputted feature map at the $(n-1)$-th layer

P^{n-1}, Q^{n-1} ..... size of the outputted feature map at the $(n-1)$-th layer

$\boldsymbol{h}_j^{n}$ ..... the j-th channel of the outputted feature map at the n-th layer

$\boldsymbol{w}_{ij}^{n}$ ..... the i,j-th convolutional kernel of the n-th convolutional layer

b_j^{n} ..... biases at the j-th channel of the n-th convolutional layer

T ..... number of time steps in the input data of a recurrent layer

x_t ..... input data, indexed by the time step t

h_t ..... hidden state, indexed by the time step t

$\boldsymbol{w}_h$ ..... weights of the inputs in a recurrent layer

$\boldsymbol{u}_h$ ..... weights of the hidden states in a recurrent layer

b_h ..... bias in a recurrent layer

σ_h ..... an activation function in a recurrent layer

c_t ..... cell state, indexed by the time step t

$\tilde{c_t}$ ..... candidate cell state, indexed by the time step t

i_t . . . . . input gate, indexed by the time step t

f_t . . . . . forget gate, indexed by the time step t

o_t . . . . . output gate, indexed by the time step t

$\boldsymbol{w}_i, \boldsymbol{w}_f, \boldsymbol{w}_o, \boldsymbol{w}_c$ . . weights of the inputs in an LSTM memory block

$\boldsymbol{u}_i, \boldsymbol{u}_f, \boldsymbol{u}_o, \boldsymbol{u}_c$ . . . weights of the hidden states in an LSTM memory block

b_i, b_f, b_o, b_c . . . . . biases in an LSTM memory block

σ_r . . . . . logistic sigmoid function

r . . . . . reset gate

z . . . . . update gate

$\tilde{h_t}$ . . . . . candidate hidden state

$\boldsymbol{w}_r, \boldsymbol{w}_z, \boldsymbol{w}_g$ . . . . . weights of the inputs in a GRU memory block

$\boldsymbol{u}_r, \boldsymbol{u}_z, \boldsymbol{u}_g$ . . . . . weights of the hidden states in a GRU memory block

b_r, b_z, b_g . . . . . biases in a GRU memory block

N_c . . . . . number of classifiers

$\tilde{y}$ . . . . . predicted label

$\tilde{y}_j$ . . . . . predicted label, indexed by j

p_j . . . . . probability, indexed by j

p_j^1 . . . . . the highest predicted probability produced by the j-th classifier

p_j^2 . . . . . the second highest predicted probability produced by the j-th classifier

c . . . . . context vector in a speech-to-text task

c_t . . . . . context vector in a speech-to-text task, indexed by the time step t

ρ_{it} . . . . . the normalised attention vector in a speech-to-text task, indexed by i and the time step t

e_{it} . . . . . attention vector in a speech-to-text task, indexed by i and the time step t

f_a . . . . . alignment model in a speech-to-text task to evaluate a matching score between the hidden state and the output

s_{i-1} . . . . . the $(i-1)$-th predicted word in a speech-to-text task

r_j . . . . . produced tensor by global pooling, indexed by j

C . . . . . classification tensor

C_k . . . . . classification tensor, indexed by k

A . . . . . attention tensor

A^* . . . . . product of applying an activation function to attention tensor

A^*_{kt} . . . . . product of applying an activation function to attention tensor, indexed by k and t

P . . . . . probability tensor

P_k . . . . . probability tensor, indexed by k

p_k . . . . . probability, indexed by k

F . . . . . length of a hidden state in a recurrent layer

K . . . . . number of classes

H . . . . . number of channels in a feature map

P_w, Q_w . . . . . size of a feature map

P_s, Q_s . . . . . size of each small area in global ROI pooling

s_r, s_r . . . . . size of the receptive field at the first convolutional layer

N_d . . . . . number of audio recording devices

σ_u . . . . . ReLU activation function

$\boldsymbol{v}_{ij}$ . . . . . the i, j-th convolutional kernel of the convolutional layer which processes the device information

$\boldsymbol{e}_i$ . . . . . the i-th channel of the expanded device one-hot encoder

$\mathcal{L}_{caa}$ . . . . . loss function in the multi-task conditional training approach

$\mathcal{L}_s$ . . . . . loss function of an audio classification model

$\mathcal{L}_d$ . . . . . loss function of a device prefiction model

$\boldsymbol{\theta}$ . . . . . perturbation

$\boldsymbol{x'}$ . . . . . adversarial data

$\nabla_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{w}, \boldsymbol{x}, y)$ . . . gradient of the loss function $\mathcal{L}(\boldsymbol{w}, \boldsymbol{x}, y)$

ϵ_w . . . . . perturbation factor

η . . . . . a hyperparameter to control the interval of the perturbation

f^D . . . . . defence model

L_A . . . . . loss function of a black-box attack model

L_{cla} . . . . . loss function leading to misclassification on the fake data

L_{MSE} . . . . . MSE loss function

$f^D(x')_{gt}$ ..... predicted probability of the fake data on the ground truth

$f^D(x')_{other}$ ..... maximum predicted probability of the fake data on the other classes except the ground truth

α ..... a constant factor to balance the gradients of the two loss functions

$\mathcal{L}_{vat}$ ..... loss function of vanilla adversarial training

β_v ..... a constant factor to balance the gradients of loss functions in vanilla adversarial training

$\mathcal{L}_{sat}$ ..... loss function of similarity-based adversarial training

ζ_1, ζ_2 ..... constant factors in the loss function of similarity-based adversarial training

$\boldsymbol{h'}$ ..... high-level representation of the adversarial data

f^A ..... attack model

K^D ..... learnt knowledge from a defence model

N_m ..... number of defence models

λ ..... constant factor to balance the global importance of parameters

$\boldsymbol{\delta}$ ..... parameters of an attack model

$\boldsymbol{\delta^*}$ ..... parameters of an attack model learnt from the previous target model

$\boldsymbol{F}^A$ ..... Fisher information matrix

$\mathcal{L}_{ewc}$ ..... EWC loss function

N_{TP} ..... numer of true positive samples

N_{FN} ..... numer of false negative samples

N_{TN} ..... numer of true negative samples

N_{FP} ..... numer of false positive samples

N_k ..... number of samples, indexed by k

$\tilde{N}_k$ ..... number of correctly predicted samples, indexed by k

$\hat{N}_k$ ..... number of predicted samples, indexed by k

C_s ..... codebook size

Bibliography

[1] Z. Ren, K. Qian, Z. Zhang, V. Pandit, A. Baird, and B. Schuller, "Deep scalogram representations for acoustic scene classification," *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 3, pp. 662–669, May 2018.

[2] 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 *Proc. ITG*, Oldenburg, Germany, 2018, pp. 56–60.

[3] Z. Ren, N. Cummins, V. Pandit, J. Han, K. Qian, and B. Schuller, "Learning image-based representations for heart sound classification," in *Proc. DH*, Lyon, France, 2018, pp. 143–147.

[4] D. Michie, D. Spiegelhalter, and C. Taylor, "Machine learning," *Neural and Statistical Classification*, vol. 13, no. 1994, pp. 1–298, 1994.

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[6] S. Chen, C. Yao, G. Xiao, Y. Ying, and W. Wang, "Fault detection and prediction of clocks and timers based on computer audition and probabilistic neural networks," in *Proc. IWANN*, Barcelona, Spain, 2005, pp. 952–959.

[7] S. Chu, S. Narayanan, C.-C. Kuo, and M. Mataric, "Where am I? Scene recognition for mobile robots using audio features," in *Proc. ICME*, Toronto, Canada, 2006, pp. 885–888.

[8] T. Reed, N. Reed, and P. Fritzson, "Heart sound analysis for symptom detection and computer-aided diagnosis," *Simulation Modelling Practice and Theory*, vol. 12, no. 2, pp. 129–146, May 2004.

[9] R. Kirby, J. Forlizzi, and R. Simmons, "Affective social robots," *Robotics and Autonomous Systems*, vol. 58, no. 3, pp. 322–332, Mar. 2010.

[10] H. Harb and L. Chen, "A general audio classifier based on human perception motivated model," *Multimedia Tools and Applications*, vol. 34, no. 3, pp. 375–395, Mar. 2007.

[11] 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, May 2015.

[12] G. Clifford, C. Liu, B. Moody, D. Springer, I. Silva, Q. Li, and R. Mark, "Classification of normal/abnormal heart sound recordings: The PhysioNet/-Computing in Cardiology Challenge 2016," in *Proc. CinC*, Vancouver, Canada, 2016, pp. 609–612.

[13] B. W. Schuller, "Speech emotion recognition: Two decades in a nutshell, benchmarks, and ongoing trends," *Communications of the ACM*, vol. 61, no. 5, pp. 90–99, Apr. 2018.

[14] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," in *Proc. ICML*, Pittsburgh, PA, 2006, pp. 161–168.

[15] J. Geiger, B. Schuller, and G. Rigoll, "Large-scale audio feature extraction and SVM for acoustic scene classification," in *Proc. WASPAA*, New Paltz, NY, 2013, pp. 1–4.

[16] J. Wang, Q. Wu, H. Deng, and Q. Yan, "Real-time speech/music classification with a hierarchical oblique decision tree," in *Proc. ICASSP*, Las Vegas, NV, 2008, pp. 2033–2036.

[17] K. Michael and K. Miller, "Big data: New opportunities and new challenges [guest editors' introduction]," *Computer*, vol. 46, no. 6, pp. 22–24, June 2013.

[18] A. L'heureux, K. Grolinger, H. Elyamany, and M. Capretz, "Machine learning with big data: Challenges and approaches," *IEEE Access*, vol. 5, pp. 7776–7797, Apr. 2017.

[19] S. Hershey, S. Chaudhuri, D. Ellis, J. Gemmeke, A. Jansen, R. Moore, M. Plakal, D. Platt, R. Saurous, B. Seybold, M. Slaney, R. Weiss, and K. Wilson, "CNN architectures for large-scale audio classification," in *Proc. ICASSP*, New Orleans, LA, 2017, pp. 131–135.

[20] 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. 6340–6344, Jan. 2017.

[21] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Convolutional recurrent neural networks for music classification," in *Proc. ICASSP*, New Orleans, LA, 2017, pp. 2392–2396.

[22] S. Amiriparian, M. Gerczuk, S. Ottl, N. Cummins, M. Freitag, S. Pugachevskiy, A. Baird, and B. Schuller, "Snore sound classification using image-based deep spectrum features," in *Proc. INTERSPEECH*, Stockholm, Sweden, 2017, pp. 3512–3516.

[23] J. Zhao, X. Mao, and L. Chen, "Speech emotion recognition using deep 1D & 2D CNN LSTM networks," *Biomedical Signal Processing and Control*, vol. 47, pp. 312–323, Jan. 2019.

[24] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.

[25] M. Schmitt and B. Schuller, "End-to-end audio classification with small datasets – Making it work," in *Proc. EUSIPCO*, A Coruna, Spain, 2019, pp. 1–5.

[26] S. Shi, Q. Wang, P. Xu, and X. Chu, "Benchmarking state-of-the-art deep learning software tools," in *Proc. CCBD*, Macau, China, 2016, pp. 99–104.

[27] R. Goebel, A. Chander, K. Holzinger, F. Lecue, Z. Akata, S. Stumpf, P. Kieseberg, and A. Holzinger, "Explainable AI: The new 42?" in *Proc. CD-MAKE*, Hamburg, Germany, 2018, pp. 295–303.

[28] W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen, and K.-R. Müller, *Explainable AI: Interpreting, explaining and visualizing deep learning.* Springer Nature, 2019, vol. 11700.

[29] Q. Zhang, Y. Yang, H. Ma, and Y. N. Wu, "Interpreting cnns via decision trees," in *Proc. CVPR*, Long Beach, CA, 2019, pp. 6261–6270.

[30] L. Chen, H. Zhang, J. Xiao, L. Nie, J. Shao, W. Liu, and T.-S. Chua, "SCA-CNN: Spatial and channel-wise attention in convolutional networks for image captioning," in *Proc. CVPR*, Honolulu, HI, 2017, pp. 5659–5667.

[31] Y. Xu, Q. Kong, Q. Huang, W. Wang, and M. Plumbley, "Attention and localization based on a deep convolutional recurrent model for weakly supervised audio tagging," in *Proc. INTERSPEECH*, Stockholm, Sweden, 2017, pp. 3083–3087.

[32] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *Proc. ICLR*, Banff, Canada, 2014, pp. 1–10.

[33] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access*, vol. 6, pp. 14 410–14 430, Feb. 2018.

[34] Y. Gong and C. Poellabauer, "Crafting adversarial examples for speech paralinguistics applications," in *Proc. DYNAMICS*, San Juan, PR, 2017, 8 pages.

[35] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. CVPR*, Miami, FL, 2009, pp. 248–255.

[36] J. Schlüter and S. Böck, "Improved musical onset detection with convolutional neural networks," in *Proc. ICASSP*, Florence, Italy, 2014, pp. 6979–6983.

[37] J. Wagner, D. Schiller, A. Seiderer, and E. André, "Deep learning in paralinguistic recognition tasks: Are hand-crafted features still relevant?" in *Proc. INTERSPEECH*, Hyderabad, India, 2018, pp. 147–151.

[38] B. Schuller, S. Steidl, A. Batliner, P. Marschik, H. Baumeister, F. Dong, S. Hantke, F. Pokorny, E.-M. Rathner, K. 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 *Proc. INTERSPEECH*, Hyderbad, India, 2018, pp. 122–126.

[39] M. Swain, A. Routray, and P. Kabisatpathy, "Databases, features and classifiers for speech emotion recognition: A review," *International Journal of Speech Technology*, vol. 21, no. 1, pp. 93–120, Jan. 2018.

[40] N. Cummins, Z. Ren, A. Mallol-Ragolta, and B. Schuller, *Artificial Intelligence in Precision Health*. Elsevier, 2020, ch. Chapter 5 – Machine learning in digital health, recent trends, and ongoing challenges, pp. 121–148.

[41] F. Eyben, *Real-time speech and music classification by large audio feature space extraction*. Springer, 2016.

[42] F. Eyben, K. Scherer, B. Schuller, J. Sundberg, E. André, C. Busso, L. Devillers, J. Epps, P. Laukka, S. Narayanan, and K. 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, Apr.–June 2016.

[43] S. Latif, J. Qadir, and M. Bilal, "Unsupervised adversarial domain adaptation for cross-lingual speech emotion recognition," in *Proc. ACII*, Cambridge, UK, 2019, pp. 732–737.

[44] F. Haider, S. De La Fuente, and S. Luz, "An assessment of paralinguistic acoustic features for detection of Alzheimer's dementia in spontaneous speech," *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, no. 2, pp. 272–281, Feb. 2019.

[45] F. Eyben, M. Wöllmer, and B. Schuller, "OpenSMILE: the Munich versatile and fast open-source audio feature extractor," in *Proc. ACM Multimedia*, Firenze, Italy, 2010, pp. 1459–1462.

[46] 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 *Proc. INTERSPEECH*, San Francisco, CA, 2016, pp. 495–499.

[47] M. Schmitt and B. Schuller, "OpenXBOW: Introducing the Passau open-source crossmodal bag-of-words toolkit," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 3370–3374, Jan. 2017.

[48] 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 *Proc. INTERSPEECH*, Hyderbad, India, 2018, pp. 3082–3086.

[49] S. Bhakre and A. Bang, "Emotion recognition on the basis of audio signal using Naive Bayes classifier," in *Proc. ICACCI*, Jaipur, India, 2016, pp. 2363–2367.

[50] D. Ververidis and C. Kotropoulos, "Emotional speech classification using gaussian mixture models and the sequential floating forward selection algorithm," in *Proc. ICME*, Amsterdam, Netherlands, 2005, pp. 1500–1503.

[51] Y.-T. Peng, C.-Y. Lin, M.-T. Sun, and K.-C. Tsai, "Healthcare audio event classification using hidden markov models and hierarchical hidden markov models," in *Proc. ICME*, New York, NY, 2009, pp. 1218–1221.

[52] K. Qian, Z. Ren, V. Pandit, Z. Yang, Z. Zhang, and B. Schuller, "Wavelets revisited for the classification of acoustic scenes," in *Proc. DCASE*, Munich, Germany, 2017, pp. 108–112.

[53] N. Cummins, Y. Pan, Z. Ren, J. Fritsch, V. Nallanthighal, H. Christensen, D. Blackburn, B. Schuller, M. Magimai.-Doss, H. Strik, and A. Härmä, "A comparison of acoustic and linguistics methodologies for Alzheimer's dementia recognition," in *Proc. INTERSPEECH*, Shanghai, China, 2020, pp. 2182–2186.

[54] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, Aug. 2002.

[55] S. Küçükbay and M. Sert, "Audio-based event detection in office live environments using optimized MFCC-SVM approach," in *Proc. ICSC*, Anaheim, CA, 2015, pp. 475–480.

[56] L. Yang, D. Jiang, L. He, E. Pei, M. Oveneke, and H. Sahli, "Decision tree based depression classification from audio video and language information," in *Proc. AVEC*, Amsterdam, Netherlands, 2016, pp. 89–96.

[57] A. Milton, S. Roy, and S. Selvi, "SVM scheme for speech emotion recognition using MFCC feature," *International Journal of Computer Applications*, vol. 69, no. 9, pp. 34–39, May 2013.

[58] K. Qian, Z. Ren, F. Dong, W.-H. Lai, B. Schuller, and Y. Yoshiharu, "Deep wavelets for heart sound classification," in *Proc. ISPACS*, Taipei, Taiwan, 2019, 2 pages.

[59] H. Nussbaumer, *Fast Fourier Transform and Convolution Algorithms*. Springer, 1981, vol. 2, ch. The fast Fourier transform, pp. 80–111.

[60] K. Prabhu, *Window functions and their applications in signal processing*. Taylor & Francis, 2014.

[61] S. Stevens, J. Volkmann, and E. Newman, "A scale for the measurement of the psychological magnitude pitch," *The Journal of the Acoustical Society of America*, vol. 8, no. 3, pp. 185–190, 1937.

[62] O. Douglas and O. Shaughnessy, "Speech communications: Human and machine," *IEEE press*, pp. 367–433, 2000.

[63] G. Fant, "Analysis and synthesis of speech processes," *Manual of phonetics*, vol. 2, pp. 173–277, 1968.

[64] Y. Sakashita and M. Aono, "Acoustic scene classification by ensemble of spectrograms based on adaptive temporal divisions," DCASE Challenge, Surrey, UK, Tech. Rep., 2018, 5 pages.

[65] S. Amiriparian, M. Schmitt, N. Cummins, K. Qian, F. Dong, and B. Schuller, "Deep unsupervised representation learning for abnormal heart sound classification," in *Proc. EMBC*, Honolulu, HI, 2018, pp. 4776–4779.

[66] Y. Xu, Q. Kong, W. Wang, and M. Plumbley, "Large-scale weakly supervised audio classification using gated convolutional neural network," in *Proc. ICASSP*, Calgary, Canada, 2018, pp. 121–125.

[67] T. Koike, K. Qian, Q. Kong, M. Plumbley, B. Schuller, and Y. Yamamoto, "Audio for audio is better? An investigation on transfer learning models for heart sound classification," in *Proc. EMBC*, Montreal, Canada, 2020, pp. 74–77.

[68] 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, 2009.

[69] P. Olver, "Topics in Fourier analysis: DFT & FFT, wavelets, Laplace transform," University of Minnesota, 2018, 41 pages.

[70] Y. Meyer, *Wavelets and operators*. Cambridge University Press, 1989.

[71] L. Barford, R. Fazzio, and D. Smith, *An introduction to wavelets*. CiteSeer, 1992.

[72] C. Torrence and G. P. Compo, "A practical guide to wavelet analysis," *Bulletin of the American Meteorological society*, vol. 79, no. 1, pp. 61–78, Jan. 1998.

[73] I. Daubechies, *Ten lectures on wavelets*. SIAM, 1992.

[74] S. C. Olhede and A. T. Walden, "Generalized morse wavelets," *IEEE Transactions on Signal Processing*, vol. 50, no. 11, pp. 2661–2670, Nov. 2002.

[75] R. Hecht-Nielsen, *Neural networks for perception*. Elsevier, 1992, ch. III.3 - Theory of the backpropagation neural network, pp. 65–93.

[76] A. Gomez, M. Ren, R. Urtasun, and R. Grosse, "The reversible residual network: Backpropagation without storing activations," in *Proc. NIPS*, Long Beach, CA, 2017, pp. 2214–2224.

[77] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. COMPSTAT*, Paris, France, 2010, pp. 177–186.

[78] A. Graves, "Generating sequences with recurrent neural networks," arXiv preprint arXiv:1308.0850, 2013, 43 pages.

[79] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, San Diego, CA, 2014, 15 pages.

[80] K. Kamnitsas, C. Ledig, V. Newcombe, J. Simpson, A. Kane, D. Menon, D. Rueckert, and B. Glocker, "Efficient multi-scale 3D CNN with fully connected CRF for accurate brain lesion segmentation," *Medical image analysis*, vol. 36, pp. 61–78, Feb. 2017.

[81] E. Cakır and T. Virtanen, "Convolutional recurrent neural networks for rare sound event detection," in *Proc. DCASE*, Munich, Germany, 2017, pp. 27–31.

[82] E. Chai, M. Pilanci, and B. Murmann, "Separating the effects of batch normalization on CNN training speed and stability using classical adaptive filter theory," in *Proc. ACSSC*, Pacific Grove, CA, 2020, pp. 1214–1221.

[83] H. Ide and T. Kurita, "Improvement of learning for CNN with ReLU activation by sparse regularization," in *Proc. IJCNN*, Anchorage, AK, 2017, pp. 2684–2691.

[84] T. Kobayashi, "Global feature guided local pooling," in *Proc. ICCV*, Seoul, Korea, 2019, pp. 3365–3374.

[85] J. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, Mar. 1990.

[86] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 2, pp. 107–116, 1998.

[87] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[88] R. Zhao, D. Wang, R. Yan, K. Mao, F. Shen, and J. Wang, "Machine health monitoring using local feature-based gated recurrent unit networks," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 2, pp. 1539–1548, Feb. 2017.

[89] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *Proc. NIPS Workshop on Deep Learning*, Montréal, Canada, 2014, 9 pages.

[90] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997.

[91] M. Valenti, A. Diment, G. Parascandolo, S. Squartini, and T. Virtanen, "DCASE 2016 acoustic scene classification using convolutional neural networks," in *Proc. DCASE*, Budapest, Hungary, 2016, pp. 95–99.

[92] J. Wang, Y. Yang, J. Mao, Z. Huang, C. Huang, and W. Xu, "CNN-RNN: A unified framework for multi-label image classification," in *Proc. CVPR*, Las Vegas, NV, 2016, pp. 2285–2294.

[93] K. Pasupa and W. Sunhem, "A comparison between shallow and deep architecture classifiers on small dataset," in *Proc. ICITEE*, Yogyakarta, Indonesia, 2016, pp. 1–6.

[94] S. Bimorogo and G. Kusuma, "A comparative study of pretrained convolutional neural network model to identify plant diseases on android mobile device," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 9, no. 3, pp. 2824–2833, May–June 2020.

[95] Z. Ren, V. Pandit, K. Qian, Z. Yang, Z. Zhang, and B. Schuller, "Deep sequential image features on acoustic scene classification," in *Proc. DCASE*, Munich, Germany, 2017, pp. 113–117.

[96] A. El-Sawy, E.-B. Hazem, and M. Loey, "CNN for handwritten arabic digits recognition based on LeNet-5," in *Proc. AISI*, Cairo, Egypt, 2016, pp. 566–575.

[97] Z. Zhong, L. Jin, and Z. Xie, "High performance offline handwritten Chinese character recognition using GoogLeNet and directional feature maps," in *Proc. ICDAR*, Tunis, Tunisia, 2015, pp. 846–850.

[98] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[99] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. NIPS*, Stateline, NV, 2012, pp. 1097–1105.

[100] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. ICLR*, San Diego, CA, 2015, 14 pages.

[101] N. Tajbakhsh, J. Shin, S. Gurudu, R. Hurst, C. Kendall, M. Gotway, and J. Liang, "Convolutional neural networks for medical image analysis: Full training or fine tuning?" *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1299–1312, May 2016.

[102] T. Scheffer, C. Decomain, and S. Wrobel, "Active hidden markov models for information extraction," in *Proc. IDA*, Porto, Portugal, 2001, pp. 309–318.

[103] K. Qian, Z. Zhang, A. Baird, and B. Schuller, "Active learning for bird sound classification via a kernel-based extreme learning machine," *Journal of the Acoustical Society of America*, vol. 142, no. 4, pp. 1796–1804, Oct. 2017.

[104] D. Wang, Q. Yang, A. Abdul, and B. Lim, "Designing theory-driven user-centric explainable AI," in *Proc. CHI*, Glasgow, UK, 2019, pp. 1–15.

[105] M. Ribeiro, S. Singh, and C. Guestrin, ""Why should I trust you?": Explaining the predictions of any classifier," in *Proc. KDD*, San Francisco, CA, 2016, pp. 1135–1144.

[106] Q. Zhang, R. Cao, F. Shi, Y. Wu, and S.-C. Zhu, "Interpreting CNN knowledge via an explanatory graph," in *Proc. AAAI*, New Orleans, LA, 2017, pp. 4454–4463.

[107] J. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," in *Proc. ICLR*, San Diego, CA, 2015, 14 pages.

[108] A. Dosovitskiy and T. Brox, "Inverting visual representations with convolutional networks," in *Proc. CVPR*, Las Vegas, NV, 2016, pp. 4829–4837.

[109] Q. Zhang, Y. Wu, and S.-C. Zhu, "Interpretable convolutional neural networks," in *Proc. CVPR*, Salt Lake City, UT, 2018, pp. 8827–8836.

[110] Z. Ren, Q. Kong, K. Qian, M. Plumbley, and B. Schuller, "Attention-based convolutional neural networks for acoustic scene classification," in *Proc. DCASE*, Surrey, UK, 2018, pp. 39–43.

[111] Z. Ren, Q. Kong, J. Han, M. Plumbley, and B. Schuller, "Attention-based atrous convolutional neural networks: Visualisation and understanding perspectives of acoustic scenes," in *Proc. ICASSP*, Brighton, UK, 2019, pp. 56–60.

[112] ——, "CAA-Net: Conditional atrous CNNs with attention for explainable device-robust acoustic scene classification," *IEEE Transactions on Multimedia*, vol. 23, pp. 4131–4142, Nov. 2020.

[113] Z. Ren, K. Qian, F. Dong, Z. Dai, Y. Yamamoto, and B. Schuller, "Deep attention-based representation learning heart sound classification," arXiv:2101.04979v1, 2021, 8 pages.

[114] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. ICLR*, San Diego, CA, 2015, 15 pages.

[115] L. Duong, A. Anastasopoulos, D. Chiang, S. Bird, and T. Cohn, "An attentional model for speech translation without transcription," in *Proc. NAACL-HLT*, San Diego, CA, 2016, pp. 949–959.

[116] A. Bérard, O. Pietquin, C. Servan, and L. Besacier, "Listen and translate: A proof of concept for end-to-end speech-to-text translation," in *Proc. NIPS*

Workshop on End-to-end Learning for Speech and Audio Processing, 2016, 5 pages.

[117] N. Akhtar and U. Ragavendran, "Interpretation of intelligence in CNN-pooling processes: A methodological survey," *Neural Computing and Applications*, vol. 32, p. 879–898, 2020.

[118] Y.-L. Boureau, J. Ponce, and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition," in *Proc. ICML*, Haifa, Israel, 2010, pp. 111–118.

[119] V. Vukotić, S.-L. Pintea, C. Raymond, G. Gravier, and J. Gemert, "One-step time-dependent future video frame prediction with a convolutional encoder-decoder neural network," in *Proc. ICIAP*, Catania, Italy, 2017, pp. 140–151.

[120] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. MICCAI*, Munich, Germany, 2015, pp. 234–241.

[121] M. Frid-Adar, I. Diamant, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan, "GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification," *Neurocomputing*, vol. 321, pp. 321–331, Dec. 2018.

[122] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," arXiv preprint arXiv:1706.05587, 2017, 14 pages.

[123] H. Phan, F. Andreotti, N. Cooray, Y. Chèn, and M. De Vos, "DNN filter bank improves 1-max pooling CNN for single-channel EEG automatic sleep stage classification," in *Proc. EMBC*, Honolulu, HI, 2018, pp. 453–456.

[124] A. Mesaros, T. Heittola, and T. Virtanen, "A multi-device dataset for urban acoustic scene classification," in *Proc. DCASE*, Surrey, UK, 2018, pp. 9–13.

[125] P. Brezina and S. Jeseničová, "Sound recording technologies and music education," *Ad Alta: Journal of Interdisciplinary Research*, vol. 8, no. 2, pp. 13–18, 2018.

[126] A. Goyal, S. Shukla, and R. Sarin, "Identification of source mobile hand sets using audio latency feature," *Forensic Science International*, vol. 298, pp. 332–335, May 2019.

[127] P. Primus, H. Eghbal-zadeh, D. Eitelsebner, K. Koutini, A. Arzt, and G. Widmer, "Exploiting parallel audio recordings to enforce device invariance in CNN-based acoustic scene classification," in *Proc. DCASE*, New York City, NY, 2019, pp. 204–208.

[128] A. Kumar, M. Khadkevich, and C. Fügen, "Knowledge transfer from weakly labeled audio using convolutional neural network for sound events and scenes," in *Proc. ICASSP*, Calgary, Canada, 2018, pp. 326–330.

[129] T. Nguyen and F. Pernkopf, "Acoustic scene classification with mismatched recording devices using mixture of experts layer," in *Proc. ICME*, Shanghai, China, 2019, pp. 1666–1671.

[130] D. Fourure, R. Emonet, E. Fromont, D. Muselet, N. Neverova, A. Trémeau, and C. Wolf, "Multi-task, multi-domain learning: Application to semantic segmentation and pose regression," *Neurocomputing*, vol. 251, pp. 68–80, Aug. 2017.

[131] V. Sindagi and V. Patel, "CNN-based cascaded multi-task learning of high-level prior and density estimation for crowd counting," in *Proc. AVSS*, Lecce, Italy, 2017, pp. 1–6.

[132] K. He and J. Sun, "Convolutional neural networks at constrained time cost," in *Proc. CVPR*, Boston, MA, 2015, pp. 5353–5360.

[133] C. Xiong, X. Zhao, D. Tang, K. Jayashree, S. Yan, and T.-K. Kim, "Conditional convolutional neural network for modality-aware face recognition," in *Proc. ICCV*, Santiago, Chile, 2015, pp. 3667–3675.

[134] A. Gibiansky, S. Arik, G. Diamos, J. Miller, K. Peng, W. Ping, J. Raiman, and Y. Zhou, "Deep voice 2: Multi-speaker neural text-to-speech," in *Proc. NIPS*, Long Beach, CA, 2017, pp. 2962–2970.

[135] A. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A generative model for raw audio," arXiv preprint arXiv:1609.03499, 2016, 15 pages.

[136] Z. Zhang, J. Han, E. Coutinho, and B. Schuller, "Dynamic difficulty awareness training for continuous emotion prediction," *IEEE Transactions on Multimedia*, vol. 21, no. 5, pp. 1289–1301, May 2019.

[137] M. Valenti, S. Squartini, A. Diment, G. Parascandolo, and T. Virtanen, "A convolutional neural network approach for acoustic scene classification," in *Proc. IJCNN*, Anchorage, AK, 2017, pp. 1547–1554.

[138] D. Snyder, D. Garcia-Romero, D. Povey, and S. Khudanpur, "Deep neural network embeddings for text-independent speaker verification," in *Proc. INTERSPEECH*, Stockholm, Sweden, 2017, pp. 999–1003.

[139] A. Goyal, A. Lamb, Y. Zhang, S. Zhang, A. Courville, and Y. Bengio, "Professor forcing: A new algorithm for training recurrent networks," in *Proc. NIPS*, Barcelona, Spain, 2016, pp. 4601–4609.

[140] R. Cipolla, Y. Gal, and A. Kendall, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proc. CVPR*, Salt Lake City, UT, 2018, pp. 7482–7491.

[141] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. ICLR*, San Diego, CA, 2015, 11 pages.

[142] H. Chen, H. Zhang, P.-Y. Chen, J. Yi, and C.-J. Hsieh, "Attacking visual language grounding with adversarial examples: A case study on neural image captioning," in *Proc. ACL*, Melbourne, Australia, 2018, pp. 2587–2597.

[143] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. Yuille, "Adversarial examples for semantic segmentation and object detection," in *Proc. ICCV*, Venice, Italy, 2017, pp. 1369–1378.

[144] Z. Ren, A. Baird, J. Han, Z. Zhang, and B. Schuller, "Generating and protecting against adversarial attacks for deep speech-based emotion recognition models," in *Proc. ICASSP*, Barcelona, Spain, 2020, pp. 7184–7188.

[145] Z. Ren, J. Han, N. Cummins, and B. Schuller, "Enhancing transferability of black-box adversarial attacks via lifelong learning for speech emotion recognition models," in *Proc. INTERSPEECH*, Shanghai, China, 2020, pp. 496–500.

[146] N. Carlini and D. Wagner, "Audio adversarial examples: Targeted attacks on speech-to-text," in *Proc. SPW*, San Francisco, CA, 2018, pp. 1–7.

[147] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," in *Proc. ICLR*, Toulon, France, 2017, 14 pages.

[148] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *Proc. CVPR*, Salt Lake City, UT, 2018, pp. 9185–9193.

[149] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. SP*, San Jose, CA, 2017, pp. 39–57.

[150] A. Bose and P. Aarabi, "Adversarial attacks on face detectors using neural net based constrained optimization," in *Proc. MMSP*, Vancouver, Canada, 2018, pp. 1–6.

[151] L. Wu, Z. Zhu, C. Tai, and W. E, "Understanding and enhancing the transferability of adversarial examples," arXiv preprint arXiv:1802.09707, 2018, 15 pages.

[152] Z. Zhang, X. Zhu, Y. Li, X. Chen, and Y. Guo, "Adversarial attacks on monocular depth estimation," arXiv preprint arXiv:2003.10315, 2020, 8 pages.

[153] W. Zhou, X. Hou, Y. Chen, M. Tang, X. Huang, X. Gan, and Y. Yang, "Transferable adversarial perturbations," in *Proc. ECCV*, Munich, Germany, 2018, pp. 452–467.

[154] C. Xie, Z. Zhang, Y. Zhou, S. Bai, J. Wang, Z. Ren, and A. Yuille, "Improving transferability of adversarial examples with input diversity," in *Proc. CVPR*, Long Beach, CA, 2019, pp. 2730–2739.

[155] B. Wang, Y. Yao, B. Viswanath, H. Zheng, and B. Zhao, "With great training comes great vulnerability: Practical attacks against transfer learning," in *Proc. SEC*, Baltimore, MD, 2018, pp. 1281–1297.

[156] G. Parisi, R. Kemker, J. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, vol. 113, pp. 54–71, May 2019.

[157] S. Barnett and S. Ceci, "When and where do we apply what we learn?: A taxonomy for far transfer," *Psychological Bulletin*, vol. 128, no. 4, pp. 612–637, 2002.

[158] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, Mar. 2017.

[159] G. Desjardins, K. Simonyan, R. Pascanu, and K. Kavukcuoglu, "Natural neural networks," in *Proc. NIPS*, Montréal, Canada, 2015, pp. 2071–2079.

[160] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M. Plumbley, "Detection and classification of acoustic scenes and events," *IEEE Transactions on Multimedia*, vol. 17, no. 10, pp. 1733–1746, Oct. 2015.

[161] K. Qian, C. Janott, Z. Zhang, J. Deng, A. Baird, C. Heiser, W. Hohenhorst, M. Herzog, W. Hemmert, and B. Schuller, "Teaching machines on snoring: A benchmark on computer audition for snore sound excitation localisation," *Archives of Acoustics*, vol. 43, no. 3, pp. 465–475, Feb. 2018.

[162] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: A survey," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 414–454, May 2013.

[163] A. Harma, J. Jakka, M. Tikander, M. Karjalainen, T. Lokki, and H. Nironen, "Techniques and applications of wearable augmented reality audio," in *Proc. AES*, Amsterdam, The Netherlands, 2003, 20 pages.

[164] E. Martinson and A. Schultz, "Robotic discovery of the auditory scene," in *Proc. ICRA*, Roma, Italy, 2007, pp. 435–440.

[165] B. Schuller, E. Marchi, S. Baron-Cohen, A. Lassalle, H. O'Reilly, D. Pigat, P. Robinson, I. Davies, T. Baltrusaitis, and M. Mahmoud, "Recent developments and results of ASC-Inclusion: An integrated internet-based environment for social inclusion of children with autism spectrum conditions," in *Proc. IDGEI Workshop as part of IUI*, Atlanta, GA, 2015, 8 pages.

[166] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, "DCASE 2017 challenge setup: Tasks, datasets and baseline system," in *Proc. DCASE*, Munich, Germany, 2017, 8 pages.

[167] Wolrd Health Organisation (WHO). (2017) Cardiovascular diseases (CVDs) Key facts. [Online]. Available: https://www.who.int/en/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)

[168] D. Mozaffarian, E. Benjamin, A. Go, D. Arnett, M. Blaha, M. Cushman, S. Das, S. Ferranti, J.-P. Després, H. Fullerton, V. Howard, M. Huffman, C. Isasi, M. Jiménez, S. Judd, B. Kissela, J. Lichtman, L. Lisabeth, S. Liu, R. Mackey, D. Magid, D. McGuire, E. Mohler, C. Moy, P. Muntner, M. Mussolino, K. Nasir, R. Neumar, G. Nichol, L. Palaniappan, D. Pandey, M. Reeves, C. Rodriguez, W. Rosamond, P. Sorlie, J. Stein, A. Towfighi, T. Turan, S. Virani, D. Woo, R. Yeh, and M. Turner, "Heart disease and stroke statistics – 2016 update: A report from the American Heart Association," *Circulation*, vol. 133, no. 4, pp. e38–e360, Jan. 2016.

[169] J. Hu, X. Cui, Y. Gong, X. Xu, B. Gao, T. Wen, T. Lu, and F. Xu, "Portable microfluidic and smartphone-based devices for monitoring of cardiovascular diseases at the point of care," *Biotechnology Advances*, vol. 34, no. 3, pp. 305–320, May–June 2016.

[170] L. Schwamm, N. Chumbler, E. Brown, G. Fonarow, D. Berube, K. Nystrom, R. Suter, M. Zavala, D. Polsky, K. Radhakrishnan, N. Lacktman, K. Horton, M.-B. Malcarney, J. Halamka, and A. Tiner, "Recommendations for the implementation of telehealth in cardiovascular and stroke care: A policy statement from the American Heart Association," *Circulation*, vol. 135, no. 7, pp. e24–e44, Feb. 2017.

[171] A. Dwivedi, S. Imtiaz, and E. Rodriguez-Villegas, "Algorithms for automatic analysis and classification of heart sounds – A systematic review," *IEEE Access*, vol. 7, pp. 8316–8345, Dec. 2018.

[172] S. Mangione, "Cardiac auscultatory skills of physicians-in-training: A comparison of three English-speaking countries," *The American Journal of Medicine*, vol. 110, no. 3, pp. 210–216, Feb. 2001.

[173] S. Ari, K. Hembram, and G. Saha, "Detection of cardiac abnormality from PCG signal using LMS based least square SVM classifier," *Expert Systems with Applications*, vol. 37, no. 12, pp. 8019–8026, Dec. 2010.

[174] R. Saraçoğlu, "Hidden Markov model-based classification of heart valve disease with PCA for dimension reduction," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 7, pp. 1523–1528, Oct. 2012.

[175] D. Springer, L. Tarassenko, and G. Clifford, "Support vector machine hidden semi-Markov model-based heart sound segmentation," in *Proc. CinC*, Cambridge, MA, 2014, pp. 625–628.

[176] C. Liu, D. Springer, Q. Li, B. Moody, R. Juan, F. Chorro, F. Castells, J. Roig, I. Silva, A. Johnson, Z. Syed, S. Schmidt, C. Papadaniil, L. Hadjileontiadis, H. Naseri, A. Moukadem, A. Dieterlen, C. Brandt, H. Tang, M. Samieinasab, M. Samieinasab, R. Sameni, R. Mark, and G. Clifford, "An open access database for the evaluation of heart sound algorithms," *Physiological Measurement*, vol. 37, no. 12, pp. 2181–2213, Nov. 2016.

[177] F. Dong, K. Qian, Z. Ren, A. Baird, X. Li, Z. Dai, B. Dong, F. Metze, Y. Yamamoto, and B. Schuller, "Machine listening for heart status monitoring: Introducing and benchmarking HSS–the heart sounds Shenzhen corpus," *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 7, pp. 2082–2092, July 2020.

[178] A. Apkarian, M. Bushnell, R.-D. Treede, and J.-K. Zubieta, "Human brain mechanisms of pain perception and regulation in health and disease," *European Journal of Pain*, vol. 9, no. 4, pp. 463–484, Aug. 2005.

[179] A. Craig, "A new view of pain as a homeostatic emotion," *Trends in Neurosciences*, vol. 26, no. 6, pp. 303–307, June 2003.

[180] P. Mantyh, "Cancer pain and its impact on diagnosis, survival and quality of life," *Nature Reviews Neuroscience*, vol. 7, no. 10, pp. 797–809, Oct. 2006.

[181] E. Scherder, A. Bouma, M. Borkent, and O. Rahman, "Alzheimer patients report less pain intensity and pain affect than non-demented elderly," *Psychiatry*, vol. 62, no. 3, pp. 265–272, Fall 1999.

[182] S. Arner and B. Meyerson, "Lack of analgesic effect of opioids on neuropathic and idiopathic forms of pain," *Pain*, vol. 33, no. 1, pp. 11–23, Apr. 1988.

[183] J. Lee, M. Lee, J. Kim, H. Kim, S. Park, J. Tae, and S. Choi, "Pain relief scale is more highly correlated with numerical rating scale than with visual analogue scale in chronic pain patients." *Pain Physician*, vol. 18, no. 2, pp. E195–200, Mar. 2015.

[184] F. Tsai, Y. Hsu, W. Chen, Y. Weng, C. Ng, and C. Lee, "Toward development and evaluation of pain level-rating scale for emergency triage based on vocal characteristics and facial expressions," in *Proc. INTERSPEECH*, San Francisco, CA, 2016, pp. 92–96.

[185] P. Lucey, J. Cohn, K. Prkachin, P. Solomon, and I. Matthews, "Painful data: The UNBC-McMaster shoulder pain expression archive database," in *Proc. AFGR*, Santa Barbara, CA, 2011, pp. 57–64.

[186] J. Kappesser and A. de C. Williams, "Pain and negative emotions in the face: Judgements by health care professionals," *Pain*, vol. 99, no. 1–2, pp. 197–206, Sep. 2002.

[187] M. Aung, S. Kaltwang, B. Romera-Paredes, B. Martinez, A. Singh, M. Cella, M. Valstar, H. Meng, A. Kemp, M. Shafizadeh, A. Elkins, N. Kanakam, A. Rothschild, N. Tyler, P. Watson, A. de C. Williams, M. Pantic, and N. Bianchi-Berthouze, "The automatic detection of chronic pain-related expression: Requirements, challenges and the multimodal EmoPain dataset," *Transactions on Affective Computing*, vol. 7, no. 4, pp. 435–451, July 2015.

[188] T. Olugbade, M. Aung, N. Bianchi-Berthouze, N. Marquardt, and A. Williams, "Bi-modal detection of painful reaching for chronic pain rehabilitation systems," in *Proc. ICMI*, Istanbul, Turkey, 2014, pp. 455–458.

[189] B. Schuller, F. Friedmann, and F. Eyben, "Automatic recognition of physiological parameters in the human voice: Heart rate and skin conductance," in *Proc. ICASSP*, Vancouver, Canada, 2013, pp. 7219–7223.

[190] N. Cummins, S. Scherer, J. Krajewski, S. Schnieder, J. Epps, and T. Quatieri, "A review of depression and suicide risk assessment using speech analysis," *Speech Communication*, vol. 71, pp. 10–49, July 2015.

[191] Y. Oshrat, A. Bloch, A. Lerner, A. Cohen, M. Avigal, and G. Zeilig, "Speech prosody as a biosignal for physical pain detection," in *Proc. SP*, Boston, MA, 2016, pp. 420–424.

[192] S. Ramakrishnan and I. El Emary, "Speech emotion recognition approaches in human computer interaction," *Telecommunication Systems*, vol. 52, pp. 1467–1478, 2013.

[193] M. Lech, M. Stolar, C. Best, and R. Bolia, "Real-time speech emotion recognition using a pre-trained image classification network: Effects of bandwidth reduction and companding," *Frontiers in Computer Science*, vol. 2, no. 14, pp. 1–14, May 2020.

[194] E. Stepanov, B. Favre, F. Alam, S. Chowdhury, K. Singla, J. Trione, F. Béchet, and G. Riccardi, "Automatic summarization of call-center conversations," in *Proc. ASRU*, Scottsdale, AZ, 2015, 2 pages.

[195] J. Liu and X. Wu, "Prototype of educational affective arousal evaluation system based on facial and speech emotion recognition," *International Journal of Information and Education Technology*, vol. 9, no. 9, pp. 645–651, Sep. 2019.

[196] S. Harati, A. Crowell, H. Mayberg, and S. Nemati, "Depression severity classification from speech emotion," in *Proc. EMBC*, Honolulu, HI, 2018, pp. 5763–5766.

[197] E. Parada-Cabaleiro, G. Costantini, A. Batliner, M. Schmitt, and B. Schuller, "DEMoS: An Italian emotional speech corpus," *Language Resources and Evaluation*, vol. 54, p. 341–383, Feb. 2019.

[198] 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 *Proc. INTERSPEECH*, Lyon, France, 2013, pp. 148–152.

[199] F. Eyben, F. Weninger, F. Groß, and B. Schuller, "Recent developments in openSMILE, the Munich open-source multimedia feature extractor," in *Proc. ACM Multimedia*, Barcelona, Spain, 2013, pp. 835–838.

[200] A. Vedaldi and K. Lenc, "MatConvNet: Convolutional neural networks for MATLAB," in *Proc. ACM Multimedia*, Brisbane, Australia, 2015, pp. 689–692.

[201] R. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," in *Proc. ICML Deep Learning Workshop*, Lille, France, 2015, 6 pages.

[202] S. Mun, S. Park, D. Han, and H. Ko, "Generative adversarial network based acoustic scene training set augmentation and selection using SVM hyperplane," in *Proc. DCASE*, Munich, Germany, 2017, pp. 93–97.

[203] B. Schuller, S. Steidl, A. Batliner, J. Hirschberg, J. Burgoon, A. Baird, A. Elkins, Y. Zhang, E. Coutinho, and K. Evanini, "The INTERSPEECH 2016 computational paralinguistics challenge: Deception, sincerity & native language," in *Proc. INTERSPEECH*, San Francisco, CA, 2016, pp. 2001–2005.

[204] Y. Pan, P. Shen, and L. Shen, "Speech emotion recognition using support vector machine," *Journal of Smart Home*, vol. 6, no. 2, pp. 101–108, Apr. 2012.

[205] R. Vempada, B. Kumar, and K. Rao, "Characterization of infant cries using spectral and prosodic features," in *Proc. NCC*, Kharagpur, India, 2012, pp. 1–5.

[206] 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 *Proc. NIPS*, Vancouver, Canada, 2019, pp. 8026–8037.

[207] Z. Zhang and B. Schuller, "Active learning by sparse instance tracking and classifier confidence in acoustic emotion recognition," in *Proc. INTERSPEECH*, Portland, OR, 2012, pp. 362–365.

[208] B. Schuller, S. Steidl, A. Batliner, P. Marschik, H. Baumeister, F. Dong, S. Hantke, F. Pokorny, E.-M. Rathner, K. 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 *Proc. INTERSPEECH*, Hyderabad, India, 2018, pp. 122–126.

[209] A. Humayun, M. Khan, S. Ghaffarzadegan, Z. Feng, and T. Hasan, "An ensemble of transfer, semi-supervised and supervised learning methods for pathological heart sound classification," in *Proc. INTERSPEECH*, Hyderabad, India, 2018, pp. 127–131.

[210] G. Gosztolya, T. Grósz, and L. Tóth, "General utterance-level feature extraction for classifying crying sounds, atypical & self-assessed affect and heart beats," in *Proc. INTERSPEECH*, Hyderabad, India, 2018, pp. 531–535.

[211] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, "DCASE 2017 challenge setup: Tasks, datasets and baseline system," in *Proc. DCASE*, Munich, Germany, 2017, 8 pages.

[212] X. Yang, T. Zhang, and C. Xu, "Cross-domain feature learning in multimedia," *IEEE Transactions on Multimedia*, vol. 17, no. 1, pp. 64–78, Jan. 2015.

[213] J. Qi, Y. Peng, and Y. Zhuo, "Life-long cross-media correlation learning," in *Proc. ACM Multimedia*, Seoul, Korea, 2018, pp. 528–536.

[214] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, Las Vegas, NV, 2016, pp. 770–778.

[215] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, Inception-ResNet and the impact of residual connections on learning," in *Proc. AAAI*, San Francisco, CA, 2017, pp. 4278–4284.

[216] A. Baird, S. Amiriparian, and B. Schuller, "Can deep generative audio be emotional? Towards an approach for personalised emotional audio generation," in *Proc. MMSP*, Kuala Lumpur, Malaysia, 2019, 5 pages.

[217] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, "Lifelong learning with dynamically expandable networks," in *Proc. ICLR*, Vancouver, Canada, 2018, 11 pages.

[218] A. Batliner and B. Schuller, "More than fifty years of speech and language processing-the rise of computational paralinguistics and ethical demands," in *Proc. ETHICOMP*, Paris, France, 2014, 11 pages.

[219] B. W. Schuller, D. M. Schuller, K. Qian, J. Liu, H. Zheng, and X. Li, "COVID-19 and computer audition: An overview on what speech & sound analysis could contribute in the SARS-CoV-2 corona crisis," *Frontiers in Digital Health*, vol. 3, no. 564906, pp. 1–10, Mar. 2021.

[220] S. Hantke, A. Batliner, and B. Schuller, "Ethics for crowdsourced corpus collection, data annotation and its application in the web-based Game iHEARu-PLAY," in *Proc. ETHI-CA*, Portoroz, Slovenia, 2016, pp. 54–59.

[221] A. Baird and B. Schuller, "Considerations for a more ethical approach to data in AI: On data representation and infrastructure," *Frontiers in Big Data*, vol. 3, no. 25, pp. 1–11, Sep. 2020.

[222] G. Rizos, A. Baird, M. Elliott, and B. Schuller, "Stargan for emotional speech conversion: Validated by data augmentation of end-to-end emotion recognition," in *Proc. ICASSP*, Barcelona, Spain, 2020, pp. 3502–3506.

[223] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *Proc. ICLR*, Toulon, France, 2017, 11 pages.

[224] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "One-shot learning with memory-augmented neural networks," in *Proc. NIPS Deep Learning Symposium*, Barcelona, Spain, 2016, 13 pages.

[225] E. Kodirov, T. Xiang, and S. Gong, "Semantic autoencoder for zero-shot learning," in *Proc. CVPR*, Honolulu, HI, 2017, pp. 3174–3183.

[226] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PLoS ONE*, vol. 10, no. 7, pp. 1–46, July 2015.

[227] J. Su, D. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 828–841, Oct. 2019.