Dissertation

Automatic Generation of Natural Language Descriptions of Visual Data

Describing Images and Videos Using Recurrent and Self-Attentive Models

Philipp Harzig

Department of Computer Science
University of Augsburg

Advisor:	Prof. Dr. rer. nat. Rainer Lienhart
Reviewers:	Prof. Dr. rer. nat. Rainer Lienhart
	Prof. Dr. rer. nat. Jörg Hähner
Thesis defense:	March 4$^{\text{th}}$, 2022
Examiners:	Prof. Dr. rer. nat. Rainer Lienhart
	Prof. Dr. rer. nat. Jörg Hähner
	Prof. Dr.-Ing. habil. Björn Schuller

Abstract

Humans are faced with a constant flow of visual stimuli, e.g., from the environment or when looking at social media. In contrast, visually-impaired people are often incapable to perceive and process this advantageous and beneficial information that could help maneuver them through everyday situations and activities. However, audible feedback such as natural language can give them the ability to better be aware of their surroundings, thus enabling them to autonomously master everyday's challenges. One possibility to create audible feedback is to produce natural language descriptions for visual data such as still images and then read this text to the person. Moreover, textual descriptions for images can be further utilized for text analysis (e.g., sentiment analysis) and information aggregation. In this work, we investigate different approaches and techniques for the automatic generation of natural language of visual data such as still images and video clips.

In particular, we look at language models that generate textual descriptions with recurrent neural networks: First, we present a model that allows to generate image captions for scenes that depict interactions between humans and branded products. Thereby, we focus on the correct identification of the brand name in a multi-task training setting and present two new metrics that allow us to evaluate this requirement. Second, we explore the automatic answering of questions posed for an image. In fact, we propose a model that generates answers from scratch instead of predicting an answer from a limited set of possible answers. In comparison to related works, we are therefore able to generate rare answers, which are not contained in the pool of frequent answers. Third, we review the automatic generation of doctors' reports for chest X-ray images. That is, we introduce a model that can cope with a dataset bias of medical datasets (i.e., abnormal cases are very rare) and generates reports with a hierarchical recurrent model. We also investigate the correlation between the distinctiveness of the report and the score in traditional metrics and find a discrepancy between good scores and accurate reports.

Then, we examine self-attentive language models that improve computational efficiency and performance over the recurrent models. Specifically, we utilize the Transformer architecture. First, we expand the automatic description generation to the domain of videos where we present a video-to-text (VTT) model that can easily synchronize audio-visual features. With an extensive experimental exploration, we verify the effectiveness of our video-to-text translation pipeline. Finally, we revisit our recurrent models with this self-attentive approach.

Acknowledgments

Hereby I want to pay credit to all the people, who influenced me and therefore made this work possible. First of all, I want to thank Prof. Lienhart for all the opportunities he gave me as well as for a lot of wonderful ideas, positive criticism, and discussions about work and life in general. Furthermore, I want to express my gratitude towards Prof. Hähner for taking his valuable time to review this thesis.

Besides, I want to acknowledge my colleagues Daniel Kienzle, Julian Lorenz, Katja Ludwig, Robin Schön, especially Moritz Einfalt for his commitment to proofreading, his valuable suggestions and countless discussions. Furthermore, I want to thank Stephan Brehm, who accompanied me on my long way and helped me with our long discussions, his alternative views, and different approaches on countless topics. Also, this extends to my esteemed former colleagues Dr. Dan Zecha, and Dr. Christian Eggert, whose doors still stands open for conversations of all kind.

I also want to express my deepest thanks towards the German non-profit market research association (GfK Verein; Nuremberg Institute for Market Decisions), particularly Carolin Kaiser, René Schaller, Holger Dietrich, Raimund Wildner, and Andreas Neus, for a great collaboration and for the creation of a dataset, on which my first papers were based, and therefore, helped to shape the foundation of this thesis. Francine Chen and Yan-Ying Chen from FX Palo Alto Laboratory (FXPAL) also need to be mentioned and thanked for giving me the chance to do an internship in California and for providing me some more practical views and insights on the medical part of our field, while always giving me a helping hand.

Then, a big thanks goes to my extraordinary family, in particular, my parents Manuela and Christian Harzig. They have been and are by my side no matter what and whenever I need them, and I hope I can make them as proud as I am to have them as parents. Finally, I want to thank my wonderful girlfriend and partner Tizia Kretzler, who has walked down this road with me, and although she was a severe critic, she showed me her love and affection every day and even allowed me to decorate our whole apartment with all kinds of exotic plants to see me happy.

Contents

Glossary of Abbreviations and Symbols

List of Abbreviations

BLEU-n	Bilingual evaluation understudy. Precision-based metric to assess the quality of generated sentences against a set of human sentences. 31
CAM	Class activation map. Probability distribution over a DCNN's output feature map that highlights areas of interest which contributed most to a classification outcome. 46
CIDEr	Consensus-based image description evaluation. Specific metric developed for the task of image captioning in order to assess the quality of generated sentences against a set of human sentences. 35
DCNN	Deep convolutional neural network. 13
FFN	Feed-forward network. 27
FPE	Fractional positional encoding. A variant of the traditional positional encoding which we present in Chapter 7. 138
GI	The human gastrointestinal (GI) tract. 43
GLU	Gated linear unit. A unit consisting of two convolutions with a gate that can control the amount ($\in [0, 1]$) of features forwarded to the next layer. 89
GRU	Gated recurrent unit. Simpler alternative to an LSTM cell. 85
HLSTM	Hierarchical LSTM. A model that stacks multiple LSTMs on top of each other hierarchically. E.g., the output of an LSTM is input to the second hierarchy level. 110
IDF	Inverse document frequency. 35
LCA	Less common answers. Answers contained in a VQA dataset, but not represented by the top-k most probable answers. These answers cannot be generated by a classification model, but a recurrent text generation model. 98
LCS	Longest common subsequence. 33

LE	LogosExtended dataset. The GfK-Dataset combined with the MSCOCO dataset. 60
LM	Language model. A probabilistic model for representing language on computers, e.g., a model that assigns probabilities to words of sentences is called a LM. 17
LS	LogosSimple dataset. Just the GfK-Dataset. 60
LSTM	Long short-term memory cell. A specialized version of an RNN cell that allows to generate language recurrently. 20
MA	Mean accuracy metric. 67
MCA	Most common answers. Top-k most common answers in the VQA task. These are the answers which can be generated by a classification model. 98
MCC	Matthews correlation coefficient. 48
METEOR	Metric for evaluation of a translation with explicit ordering. Specific metric for machine translation to assess the quality of generated sentences against a set of human sentences. 33
MHA	Multi-head attention. A special attention layer used in the Tranformer architecture. 25
MSCOCO	The Microsoft common objects in context dataset. 58
MSR-VTT	Microsoft research video-to-text large-scale video captioning dataset. 129
MSVD	Microsoft research video description corpus dataset. 129
MTI	Medical text indexer encodings. MTI encodings are automatically extracted keywords from the indication and findings of a medical report. 106
MTL	Multi-task learning. A learning objective that considers multiple learning tasks at the same time. 76
NLP	Natural language processing. An active area of research in the field of linguistics, computer science and machine learning. 17
NMT	Neural machine translation. Task that deals with the automatic translation of languages with artificial neural networks. 24
OA	Overall accuracy metric. 67
PA	Posteroanterior view of a chest X-ray image. When taking an x-ray shot, the x-ray beam enters through the posterior (back) aspect of the chest and exits out of the anterior (front) aspect. 107

PE	Positional encoding. An artificial signal mostly consisting of sine and cosine waves that encode indices within a sequence. 29
RNN	Recurrent neural network. In our work an RNN mostly consists of a single cell that is calculated recurrently. 19
ROUGE	Recall-oriented understudy for gisting evaluation. Recall-based metric to assess the quality of generated sentences against a set of human sentences. 33
SCA	Sentence classification accuracy. An accuracy metric that evaluates whether the correct brand of a class is contained in a generated sentence. 66
SCST	Self-critical sequence training. 139
SGDR	Stochastic gradient descent with warm restarts learning rate schedule. 135
SPO	Subject-predicate-object metrics. A set of accuracy metrics that check whether the correct subject, predicate, object, or combinations of these are contained in a generated sentence. 64
TF	Term frequency. 35
VATEX	Video And TEXt large-scale multilingual video captioning dataset. 129
VQA	Visual question answering. A task that tries to automatically answer questions about a given image. 79
VTT	Video-to-text. Task that tries to automatically infer short descriptions for videos. 126

List of Symbols

$\boldsymbol{\alpha}$	Relative importances of elements of a vector within an attention mechanism. 24
b	Beam size. Number of sentences that should be sampled with the beam search algorithm. 23
$\mathbf{c}$	Context vector $\mathbf{c}$. Typically used as initialization for an LSTM decoder when raw image features are not sufficient. For example, when attention is applied to an image given some query, we call the attended features context features, i.e., a context vector. Also denotes a topic vector, which is an intermediate representation for a single sentence in a hierarchical LSTM. 86
$\tilde{c}$	Denotes the distinct sentence count, i.e., absolute number of times a distinct sentence occurs in a dataset. 107
d	Size of a dimensionality. For example, d is the number of channels of a feature map while d_{model} is the dimensionality of a Transformer model. 25
$\mathbb{E}[X]$	The expectation of a random variable X, i.e., $\mathbb{E}[X] = \sum_x p(x)x$. 139
$f(\cdot)$	An arbitrary function with input $\cdot$. Projects the input to an output. Can also stand for an arbitrary layer in a neural network. For example, f_{att} is a nonlinear layer on page 92. 84
$\mathcal{F}$	Feature Map. Usually an output produced by a Deep Convolutional Neural Network (e.g., *Inception-v3*) for a given input image. 46
$\hat{\mathcal{F}}$	Attended Feature Map, i.e., a spatial feature map that is modified by some attention query and reweighted to represent the relative importance of image regions given a query. 92
$\tilde{\mathcal{F}}$	Average-pooled Feature Map. Usually computes the average of an input feature map by averaging across the spatial dimensions, i.e., the height and width dimension. 44
$\mathcal{F}_e$	Embedded feature map. For example, $\mathcal{F}_e^I$ stands for an embedded image feature map. 108
$\mathbf{h}_t$	Hidden state of an RNN cell at iteration step t. For example, $\mathbf{h}_m^{\text{sent}}$ is the hidden state of the sentence LSTM for sentence m (see page 110). 20

I	Image used as an input to a model. $\mathbf{I}$ then denotes the set of all images in a dataset. 54
K	Number of feature maps. 87
κ	Kernel size. 89
L	Loss function. L represents a generic loss function, i.e., an error function for a task. For example, L^r represents the loss for an image rating r. 44
λ	A weighting factor. For example, we weight different losses $L_\cdot$ with respective weighting factors $\lambda_\cdot$. 113
M	Number of sentences in a paragraph. 108
$\mathbf{m}$	Mask vector. Defines a mask for words within the vocabulary $\mathbf{V}$. For example, a mask that filters out classwords from the vocabulary. 62
μ	Dimension of the word embedding. 88
N	Number of words in a sentence. Also used to denote the number of image frames (N^I) or the number of audio frames (N^A) in a video. 23
$\mathcal{N}$	Number of repetitions for building blocks of our models. 27
$\mathbf{n}_{t+1}$	Projected LSTM output at iteration step t. The hidden state, i.e., the output of the LSTM is projected to the length of the Vocabulary $\lvert\mathbf{V}\rvert$ with a fully-connected layer. 54
ν	Number of output neurons. 89
Φ	Set of generated candidate sentences. 67
ϕ	Softmax activation function. 24
π	A policy for a reinforcement learning agent. 139
$\mathbf{p}_{t+1}$	Probability distribution over the whole vocabulary of the LSTM's output at iteration step t, i.e., $\mathbf{p}_{t+1} = \phi(\mathbf{n}_{t+1}) = \text{Softmax}(\mathbf{n}_{t+1})$. 139
$r(\cdot)$	A reward function which is used for self-critical learning. 139
ρ	A regular fully-connected/dense layer. 44
σ	Sigmoid activation function. 21
t	Defines the index of a word within a sentence/sequence, i.e., $\mathbf{y}_t^S$ is the t-th word within the ground-truth sentence $\mathbf{y}^S$. We also use t to denote the iteration step during the unrolling of an RNN cell. Related works often denote t as the time step. 20
τ	Duration of a video clip. Durations of a single audio frame or a video frame are denoted by τ^A or τ^I, respectively. 138
θ	The whole parameter set of a neural network model. 55

$\mathbf{V}$	Vocabulary of words that can be used to construct sentences. 17
$\mathbf{W}^{\cdot}$	Weight matrix for some layer/operation $\cdot$. 20
$\mathbf{x}/\mathbf{X}$	Input data in form of a vector or matrix. 14
$\mathbf{y}^{\cdot}$	Ground-truth vector for a machine-learning task. The superscript $^{\cdot}$ denotes the kind of task, e.g., $\mathbf{y}^{S}$ is a ground-truth sentence. 17
$\cdot$	Regular product of two scalars or a matrix product.
$\bullet$	In this work, we denote a dot-product between two vectors with a slightly bigger product symbol.
$\circ$	Element-wise product or hadamard product.

1 Introduction

1.1 Motivation and Applications

When humans are asked to describe an image that is shown, they can come up with appropriate descriptions in a short amount of time. These descriptions may vary greatly but all describe the image or parts of it correctly. Even young kids are able to describe scenes depicted in children's books with natural language. This task may seem trivial at first but it consists of multiple sub-tasks, which are complicated on their own. First, persons need to see and understand the scene. Furthermore, they need to identify objects and relations between them to make sense of the given image. Also, they might need to interpret things that are not directly visible in the scene but are important for an accurate description. Besides all these steps, they need to have prior knowledge, e.g., know properties of objects or know what objects, persons, or things can do. Finally, persons need to have the ability to transform all this information into a natural language sentence.

Significant leaps have been made in the field of machine learning and computer vision that has allowed computers to automatically classify images with better-than-human accuracy (i.e., the human top-5 ImageNet classification error rate is 5.1 % [122] whereas a ResNet ensemble has a top-5 classification error of 3.57 % [58]). These big advances were led by the advent of end-to-end trainable *Deep Convolutional Neural Networks* (DCNNs) that could be trained on a corpus of multiple million images. These DCNNs have then been applied to other tasks such as object detection or segmentation. In addition, *Recurrent Neural Networks* (RNNs) have made breakthroughs in the domain of *Natural Language Processing* (NLP), especially in the related subdomain of machine translation. However, RNNs have been mostly replaced lately by the newer and more powerful *Transformer* architecture.

Image Captioning, as we also call the problem stated above, is a complex - yet seemingly easy - task for humans. But to implement this technique in a way that computers can perform it automatically faces challenges of its own. In this thesis, we investigate the automatic generation of descriptions for visual data with models that make use of the aforementioned DCNNs and RNNs/Transformers to interpret visual data (e.g., images or videos) and generate text.

The ability to accurately describe everyday scenes can help visually-impaired people to understand what is happening in their environment. However, there may not always be another person around to help out the person in need. Having an algorithm or app that can describe the surrounding captured by a mobile phone camera can support these people in everyday activities. Moreover, this technique can be extended to video clips,

i.e., moving images. This allows for more applications such as video summarization or automatic tagging of YouTube videos.

Generated textual descriptions for images can also be used for retrieval tasks such as a textual search for images. Datasets consisting of millions of pictures can be indexed by automatically generating captions and storing them in a database. By using existing text search techniques, we can then quickly search this database for matches and return related images. Furthermore, other text analysis tools such as sentiment analysis can be applied to a collection of texts and then associated with the images.

1.2 Problem Definition and Challenges

When we speak of *Image Description Generation* or *Image Captioning*, we refer to the task of understanding an image and then giving a short concise natural language description of the image's contents. This task may seem trivial at first, but even humans struggle in coming up with a comprehensive description for a given image. Different people may prefer to describe different aspects of that image. Nevertheless, humans mostly suggest a reasonable sentence when they are asked to describe a given image.

However, when we look at the complexity of this task, we realize that we need to understand the image, recognize relations and links between objects and then transfer this intermediate information into a natural language sentence. Even much simpler tasks like only classifying an image are hard to solve in an algorithmic way. With the advent of end-to-end learnable computer vision models, the focus has shifted from manually engineering a mapping function to data-driven models that solely learn on image-label pairs in order to deduce a mapping function through optimization techniques.

In this thesis, we address the problem of automatically generating short textual descriptions for images and videos with data-driven models. Furthermore, we examine the area of *Visual Question Answering* (VQA) which is somewhat remotely related to image description generation. In the following, we discuss common problems and challenges of generating textual descriptions for visual data.

Modeling Language Before even trying to describe an image with natural language, we have to think about how natural language can be generated. This is a huge problem on its own, but a model somehow needs to find out how language works and how to generate a sentence word by word in a way that a human can understand it. This includes using a vocabulary to generate grammatically and semantically correct sentences.

Another problem in language generation are *long-term dependencies* (see Chapter 2.2.5) where a connection between two words cannot be established if the words are far apart in a sentence. For example, a language generation model may have forgotten the gender of the subject by the time it has to emit personal pronouns.

Recognizing and Understanding Images Even if our model is able to produce reasonable captions, i.e., language that is sound and correct, we first need to tell that language model *what* to describe. Therefore, a deep understanding of the input image is crucial

for a model that tries to infer captions for it. In particular, objects need to be found, relations between these objects have to be identified and the common setting (e.g., outdoors or in a supermarket) has to be detected. This information has to be summarized, correlated, and then transformed into a representation that aids our language model.

Paragraphs of Sentences The problem of language generation manifests even more if we try to generate more than one sentence as a description. That is, multiple consecutive sentences which we call *paragraphs* in this work. It is even harder to identify and connect long-range dependencies in a paragraph than in a single sentence. That is, if language models have a hard time connecting words that have a distance of a few words, it is inherently more difficult to make connections between words that have a distance of multiple sentences to each other.

In addition, richer contextual information has to be passed to a language model in order to generate a more detailed description that spans multiple sentences. Also, the language model has to have a sense of temporal succession if the individual sentences depend on each other.

Evaluation Criteria Another problem that comes with natural language is to automatically assess the quality of a generated sentence. Even for humans, this is quite subjective, i.e., humans tend to describe different aspects of an image or have different emphases they personally like to describe. For example, different parts of the images could be described or we could lay focus on the mood or emotional context of the image. Furthermore, we could describe the setting of the picture, e.g., location, or only give a description of what we see but not what happens in the image.

As we see, language is very complex and not even humans can agree on what is a good description or a bad one. Furthermore, there are many correct descriptions of an image: First, there can be multiple synonymous descriptions, which are formulated differently. Second, the description could focus on different parts or aspects of the image. Nonetheless, humans can say whether a given caption does match an image correctly to a certain extent or not. However, assigning humans to evaluate a number of sentences is both time-consuming and expensive. Therefore, a quick, reliable, and automatic way of evaluating generated captions is indispensable. In contrast to standard accuracy-based classification metrics, there is no straightforward or easy way to quantitatively measure the correctness of a sentence. Parts of this thesis try to find alternative metrics that expand the capabilities of traditional metrics in order to measure caption quality regarding different criteria.

Correctly Answering Questions about Images The VQA task goes one step beyond image captioning. It not only gives a short description of an image but has to give a precise answer to a posed question. Thus, in comparison to common challenges in image captioning, the resulting text has to be more appropriate given a question. Answers can be just as complex as natural language descriptions. However, most VQA models are

designed to select an answer out of a small pool of predefined answers. Therefore, new and unexpected answers can not be given with such a model.

Generating answers from scratch - just as in the task of image captioning - is a possible solution, which we tackle in this work. However, datasets with more extensive answers are not readily available. Furthermore, we cannot evaluate the positive or negative effects of newly generated answers as the official metric for VQA is based on accuracy. Generated answers may be more diverse or different but still correct and cannot easily be matched against a predefined set of answers.

Dataset Biases As language models are only probabilistic models, they generate the most probable word given a history of previously generated words. Thus, they tend to generate the most likely sentence given some preconditioning, e.g., an input image. As such, language models are very prone to dataset biases such as repeating formulations for different dataset samples. This presents a huge problem as some datasets consist of repeating samples. For example, in a dataset of doctors' reports, most of the depicted cases are normal cases and only a few cases show abnormalities. As a consequence, the ground-truth reports are the same for the majority of the dataset samples. Thus, normal descriptions are far more likely than descriptions for abnormal cases. However, this is not the desired behavior as we want to generate fitting descriptions for the respective input images. Furthermore, it is difficult to quantify this behavior as common metrics also reward generating normal descriptions. This is because the abnormality is usually only described in just a few words within an entire paragraph.

Video Inputs When going from still images to short video clips, more challenges arise during description generation. First, the added dimension along the time-axis leads to a higher computational complexity as more images have to be analyzed. Second, important information which we want correctly described may only be visible for a few frames. In contrast, most frames of a video could show the same thing or some things which are unimportant for the final description. Third, some videos come with audio information, which also might be important for the video clip's content. Fourth, in comparison to image captioning, it is harder for a machine learning model to generalize as training data grows while the number of annotations does not. Specifically, one video that can consist of hundreds of frames still comes with the same number of ground-truth captions.

1.3 Contributions

Classification-aware Image Description Generation We present an adapted image captioning model, which is based on the Show and Tell [139] model. Our model is aimed at generating answers with a focus on images depicting persons that interact with branded products, e.g., a Coca-Cola can. Our model operates on a new dataset created in collaboration with the GfK-Verein e.V. (Nuremberg Institute for Market Decisions). Particularly, we introduce the *classification-aware loss* [50], which forces the model to generate at least one word in the sentence that is of the brand name. We find that this extra

loss function improves the overall caption scores and ensures that the generated caption contains the correct brand name.

In combination with the classification-aware loss, we present the *sentence classification accuracy* (SCA) metric, which allows to calculate the accuracy of certain classes within a generated caption. More specifically, we want to know if a brand name is contained in a generated sentence or not. Thus, we search for the brand names in sentences and can, therefore, calculate the accuracy over generated sentences whether they contain the correct brand name.

SPO metrics As traditional machine translation metrics are mostly precision-based or recall-based, they do not necessarily check for the semantical correctness of a sentence. That means that few - yet important - words may change the whole meaning of a sentence. For instance, if a model correctly generates a caption word by word but misinterprets a female hand as a male hand, the BLEU-4 score is still very high with 0.827 for the ground-truth sentence *"A **female** hand holds a can of cocacola above a tiled floor."*. However, in our setting, identifying the subject correctly is crucial. Therefore, we introduce the *subject-predicate-object* (SPO) [57] metrics that analyzes a sentence if the correct subject, predicate, and object are found. We annotate our whole dataset with those SPO annotations and define eight different accuracy metrics derived from the subject, predicate, and object accuracies. Additionally, we collect synonym tables specific to our problem domain to allow the usage of synonyms for the subject and object. For the predicates, we create conjugation tables.

Finally, we collect the SPO automatically for the MSCOCO dataset. By doing so, we can show the effectiveness and the feasibility of the SPO metrics on other datasets.

Multi-Task Learning with Image Ratings Human-product interactions can often be assessed in a different way as only generating a textual description. For this purpose, the GfK-Verein e.V. (Nuremberg Institute for Market Decisions) re-annotated the whole dataset with integer-valued ratings that measure the human-product interaction depicted in three ways: (1) whether a branded product appears in a positive, neutral or negative light, (2) whether a person is involved with a branded product and (3) whether the branded product is used in an emotional or functional way. We propose a multi-task learning objective [50] for our image captioning model such that we train the captioning task in parallel with the image ratings subtask and the classification-aware loss. Furthermore, we train image ratings with three different training objectives including a soft-target training objective. Finally, we are able to improve traditional sentence evaluation metrics, the SCA metric, and SPO metrics by employing the multi-task training strategy.

Diverse Answers for VQA In the task of visual question answering, a model tries to answer a question about an image. Traditionally, this task is modeled as a classification problem. This means, there is a fixed catalog of possible answers and a model has to choose the most probable answer given a question and an image. We argue that a fixed

number of possible answers (e.g., 3000) does not cover the vast space of natural language answers to questions about images. Therefore, we propose a new VQA architecture [52] that allows to dynamically generate answers - even including previously unseen ones. Also, we introduce a convolution-based feature extraction for questions instead of employing RNNs. Judging from our experiments, this change further improves the question feature extraction and in turn, our final scores.

With numerous experiments, we compare different approaches and justify our parameter and architectural decisions. We show that - even though our accuracies are not better than related works - the generated answers make a lot of sense and are sometimes more detailed than the predefined answers. In contrast to other works, we are able to generate new answers and *less common answers* (LCA).

Addressing Data Bias Problems for Chest X-ray images In the medical domain, high-quality and well-annotated data is scarce. Due to regulations and privacy concerns, most data is kept in hospitals and cannot be released to the public. But there is one public dataset for chest X-ray images, which are annotated with natural language doctors' reports. However, this dataset has some undesirable biases: For instance, there are few abnormalities and many normal cases. Furthermore, the textual descriptions for normal cases are highly repetitive. Following the theme of our thesis, we want to generate these doctors' reports automatically given a chest X-ray image [51]. We first annotate every sentence of the dataset with abnormality labels, i.e., labels stating whether a sentence describes an abnormality or not. Second, we implement a hierarchical language generation module that can distinguish between abnormalities and use two different individual LSTMs to generate descriptions for abnormalities and normal cases. Finally, we discuss the correlation between traditional metrics and the variability in generated doctors' reports. We also find that the repetition of the same sentence can yield a high score but does not necessarily mean that a generated report is of high quality.

Video-to-Text and Synchronization of Audio-Visual Frames Video clips carry considerably more information than single images. Thus, generating a description for short video clips is way more complex than just trying to understand one static scene. A model has to simultaneously understand single images, the temporal correlation between those frames, and optimally extract information from the corresponding audio track. For instance, a video clip with 300 frames has 300 times the raw visual information compared to a single image. Yet, we want to generate a single sentence that correctly describes the video clip's content. Transformers [137] which operate on sequences are a natural choice for this problem as video clips are - by definition - sequential data, i.e., the time dimension defines the sequence. Thus, we develop a straightforward and easy-to-use video-to-text (VTT) model [54] that can be trained on huge corpora of videos (e.g., YouTube videos) while still being reasonably efficient. Our model utilizes state-of-the-art ideas from image captioning and is easy to implement. Furthermore, we do not train an ensemble of dozens of models as competing models do. Moreover, we introduce an fractional positional encoding which allows synchronizing audio frames and visual frames

that are input to a Transformer's encoder. This allows the model to attend more easily to audio and vision information while generating a description for a video clip. Finally, we evaluate our VTT model on the default datasets for video-to-text generation and generate descriptions for the TRECVID-VTT challenge [55, 56].

Validating Ideas on the Transformer Architecture Because of its success, the Transformer architecture has lately found its way into many areas of computer vision and NLP. Especially in the machine translation task and the image captioning task, recurrent models are more and more replaced with Transformers. Motivated by this success, we first port our image captioning model for describing human-product interactions to the Transformer architecture. We validate whether the Transformer also yields better results in our setting. Furthermore, we check if our extensions such as the classification-aware loss and the multi-task learning objective with image ratings are also applicable to the Transformer-based model.

Second, we show that the Transformer can also be used to generate answers for visual questions. In the same fashion as before, we want to generate answers instead of selecting one out of a predefined set of answers. Particularly, we implement a Transformer that can generate answers at once when observing an image and a question. This method is in contrast to autoregressive decoding which is usually employed in language generation models but perfectly suited for the nature of the VQA task where an image and a complete question are available at the inference case.

For both models, we are able to improve the scores in contrast to the respective recurrent models. However, we observe some disadvantages because other metrics such as the SCA decrease. Moreover, the Transformer-based VQA model does not generate as many new answers as its recurrent counterpart.

1.4 List of Publications

Parts of this thesis have been published in the academic literature and have been presented at international conferences:

Automatic Disease Detection and Report Generation for Gastrointestinal Tract Examination [53], Philipp Harzig, Moritz Einfalt and Rainer Lienhart, ACM International Conference on Multimedia 2019, Nice, France, October 2019.

Multimodal Image Captioning for Marketing Analysis [50], Philipp Harzig, Stephan Brehm, Rainer Lienhart, Carolin Kaiser and René Schallner, IEEE Conference on Multimedia Information Processing and Retrieval 2018, Miami, FL, April 2018.

Image Captioning with Clause-Focused Metrics in a Multi-Modal Setting for Marketing [57], Philipp Harzig, Dan Zecha, Rainer Lienhart, Carolin

Kaiser and René Schallner, IEEE Conference on Multimedia Information Processing and Retrieval 2019, San José, CA, March 2019.

Visual Question Answering with a Hybrid Convolution Recurrent Model [52], Philipp Harzig, Christian Eggert and Rainer Lienhart, ACM International Conference on Multimedia Retrieval 2018, Yokohama, Japan, June 2018.

Addressing Data Bias Problems for Chest X-ray Image Report Generation [51], Philipp Harzig, Yan-Ying Chen, Francine Chen and Rainer Lienhart, 30th British Machine Vision Conference 2019, Cardiff, Wales, September 2019.

Synchronized Audio-Visual Frames with Fractional Positional Encoding for Transformers in Video-to-Text Translation [54], Philipp Harzig, Moritz Einfalt and Rainer Lienhart, submitted to: IEEE International Conference on Image Processing, ICIP, 2022.

Transforming Videos to Text (VTT Task) Team: MMCUniAugsburg [55], Philipp Harzig, Moritz Einfalt, Katja Ludwig and Rainer Lienhart, TRECVID Workshop, 2020, virtual.

Extended Self-Critical Pipeline for Transforming Videos to Text (VTT Task 2021) – Team: MMCUniAugsburg, to be published [56], Philipp Harzig, Moritz Einfalt, Katja Ludwig and Rainer Lienhart, TRECVID Workshop, 2021, virtual.

1.5 Thesis Outline

This thesis consists of three parts:

Part I introduces foundations important for this work. We start by presenting DCNN backbone feature extractors, which we included in our models. Next, we introduce the concept of language adapted to the domain of machine learning. That is, we define common concepts like words, tokens, and n-grams, that are the basis of language models. Additionally, we introduce attention mechanisms and the Transformer architecture that are used in Part III of this work. Finally, we introduce metrics that can automatically assess whether a generated sentence is good in comparison to ground-truth sentences written by humans.

In Part II, we discuss models which generate natural language descriptions for images in a recurrent way. We will first look at a dataset for image captioning with a focus on branded products. This dataset was created in conjunction with the GfK-Verein e.V. (Nuremberg Institute for Market Decisions). We implement a model that specifically focuses on branded products and creates natural language captions for these images. Second, we explore the task of VQA in the light of generating new answers whereas traditional VQA models are limited to a fixed set of answers. At the end of this part, we

consider the task of automated generation of doctors' reports for chest X-Ray images. As reports consist of more than a single sentence, we examine a hierarchical RNN in order to generate paragraphs of sentences.

Part III deals with the more recent Transformer architecture. These networks yield significantly better results than RNNs. First, we extend the task of image captioning to video-to-text, i.e., we try to generate reasonable descriptions for short video clips. We present a straightforward Transformer architecture that allows us to reliable generate short descriptions for these video clips and extend it with the ability to synchronize audio and vision frames. This increases caption quality even further. Then, we revisit two models from Part II in the light of Transformers. Particularly, we include the Transformer into the image captioning model from the second part and present a Transformer-based VQA model that can generate answers at once instead of generating them word by word.

Ultimately, we conclude this thesis and we illustrate interesting and compelling ideas for future research in the domain of automatic description generation of images and video clips.

Part I

Foundations

2 Base Models and Metrics

In this chapter, we present important foundations such as basic concepts that we make use of intensively in this thesis. First, we give a short overview of *Deep Convolutional Neural Networks* (DCNNs) that extract representative features from input images or videos and are a vital part in order to generate textual descriptions for said inputs. We follow this with a quick introduction to language models that allow the processing and generation of natural language in a machine learning setting. Furthermore, we explore attention mechanisms in the realm of *Natural Language Processing* (NLP) and transfer these concepts to the task of *image captioning*. Based on attention mechanisms, we introduce the *Transformer* [137] architecture which we apply to our models in Part III of this thesis. At the end of this chapter, we investigate different metrics that allow us to evaluate generated textual descriptions against reference sentences written by humans.

2.1 Backbone Architectures

For extracting visual features, we include different DCNNs into our models. We only give a short overview of the main feature extractor networks used in this thesis. In the task of image description generation or video-to-text, a good understanding of the image or video is needed in order to create reasonable captions. Thus, feature extractors like DCNNs play an important role in all our models. In Part II, we process image features from Inception-v3 [132] and ResNet [59] networks. In Part III, we utilize image and video features from ResNet [59] and the Inflated 3D ConvNet (I3D [15]). We briefly explore features from MobileNet-v2 [124] and DenseNet-121 [67] in Chapter 3. But as these DCNNs are not employed in other models, we refer the reader to the respective papers for more details.

2.1.1 Inception-v3

Inception-v3 [132] is based on the famous GoogLeNet [131] architecture. GoogLeNet aka Inception-v1 both increased width and depth compared to DCNNs of that time. However, it is computationally more efficient than the VGG-16 [125] while performing a little bit better. Inception-v3 implements multiple tricks to increase the computational efficiency again. First, the authors split 5×5 convolutions into two subsequent 3×3 convolutions. Therefore, they can reduce the number of parameters by 28 %. Second, they factorize some 3×3 convolutions into one 3×1 convolution followed by a 1×3 convolution, which reduces parameters by 33 %. In total, Szgedy et al. present three Inception modules that reduce the number of parameters in the network. By using this technique, the network can have more layers while being less prone to overfitting.

Furthermore, they make changes to the auxiliary classifier from the original GoogLeNet and change its purpose to be a regularizer. In addition, they replace traditional max pooling operations (as used in AlexNet [79] and VGG [125]) with the efficient grid size reduction. That is, they split a max pool operation into two halves: convolutions with stride 2 and max pooling. This new operation is computationally less expensive and keeps the network efficient. Finally, the whole DCNN is normalized with batch normalizations [68]. Note that the Inception-v3 has a different input resolution than most other DCNNs (299×299 vs. 224×224). We depict the Inception-v3 DCNN in Figure 2.1.

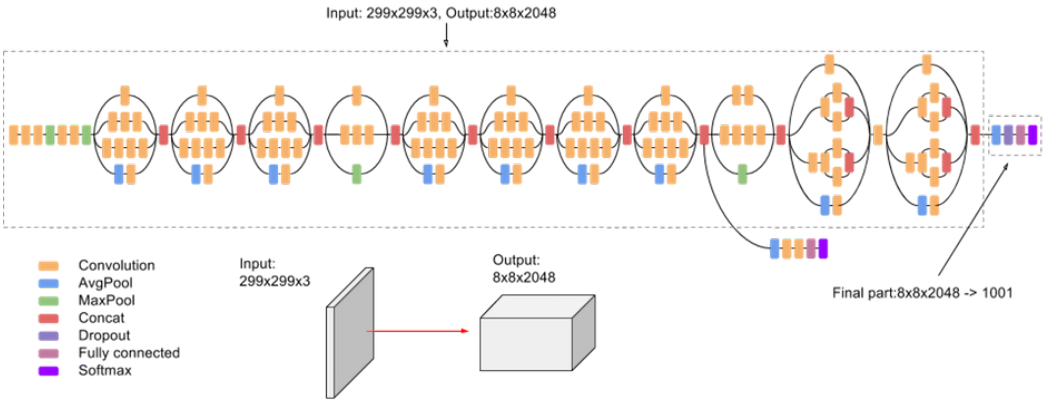


Figure 2.1: Network architecture of the Inception-v3 [132] DCNN. This particular Inception-v3 is trained on the 1000 classes from ImageNet and a background class, hence, it has 1001 output neurons. Image from the TensorFlow Cloud TPU documentation homepage [31].

2.1.2 ResNet

He et al. [59] presented *ResNet*s as an alternative to VGG [125] which are both deeper but with lower computational complexity (i.e., the number of FLOPs is reduced). They motivate their new network architecture that deeper networks should have at least the same expressiveness as shallower ones. However, they show that a 34-layer network has a higher error rate than an 18-layer network, for instance. Residual Networks (i.e., ResNets) follow a very simple idea: They introduce shortcut connections between every two convolutional layers. The shortcut connection is merely an identity mapping. Thus, the layers in between are able to learn the residual. We depict a residual block in Figure 2.2, where $\mathbf{x}$ is the input to a residual block and the output is calculated as $f(\mathbf{x}) + \mathbf{x}$. Note that $f(\mathbf{x})$ can be an arbitrary operation. In a classical residual block this operation consists of two convolutional layers with a ReLU non-linearity after the first layer. The shortcut connection is then added channel-wise on top of the second convolutional layer's output.

Coming back to the previous example, the authors claim "that if the added layers [of the

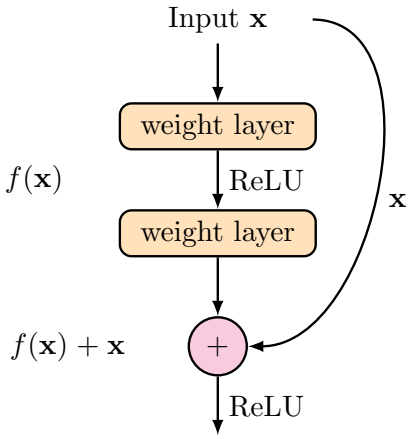


Figure 2.2: Visualization of an arbitrary residual block in the ResNet.

34-layer network] can be constructed as identity mappings, a deeper model should have training error no greater than its shallower counterpart [e.g., an 18-layer network]" [59]. Additionally, the authors add batch normalization [68] layers after each convolution and before every activation. Furthermore, every time the resolution is halved, i.e., when the number of channels is doubled, we need to adopt the number of channels of the shortcut connection as well. The authors implement this with a strided 1×1 convolution. In Figure 2.3, we show a 34-layer ResNet with residual connections. Note that dashed lines represent a residual connection combined with a convolution layer with a kernel size of 1×1, a stride of 2 and double the number of output channels as input channels. Therefore, this convolution reduces the resolution while doubling the number of channels. This ensures that both the number of channels and spatial resolution matches the other computation stream in the add operation.

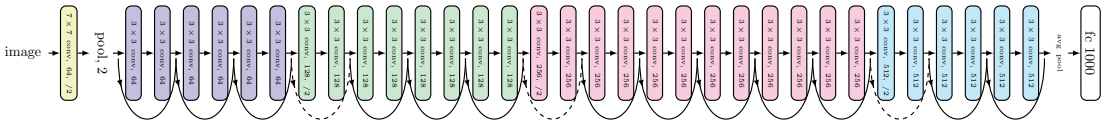


Figure 2.3: Network architecture of the ResNet-34 [59] DCNN. Figure redrawn from [59].

2.1.3 I3D

In Chapter 7, we switch from using still images as inputs to videos. Traditional DCNNs trained on the task of image classification have shown good results on image-related tasks such as object detection [119] or image captioning (see Chapters 4 and 6). Yet, these network architectures can still be used to extract image features on a frame-per-frame basis and produce tolerable results in the task of video-to-text (see Chapter 7). However, traditional DCNNs trained on the task of image classification are not optimal to catch temporal dependencies on an inter-frame level. To remedy this problem, Carreira and

Zisserman [15] introduced the *Two-Stream Inflated 3D ConvNet (I3D)* that is trained on the Kinetics Human Action Video [74] dataset.

The I3D network is based on the Inception-v1 [131] DCNN. First, the authors *inflate* the 2D operations into 3D, that is, they inflate all filters and pooling layers into an additional time dimension by expanding square filters ($N \times N$) into cubic ones ($N \times N \times N$). Second, they bootstrap the newly created 3D filters from 2D filters pre-trained on ImageNet [122] by repeating the weights N times along the time dimension and scaling the weights by $\frac{1}{N}$. Third, they make some other small architectural design choices such as not performing temporal pooling in the first two max-pooling layers. Also, the final pooling layer uses a $2 \times 7 \times 7$ max-pooling operation. Finally, as the name suggests, they introduce two separate streams, i.e., one RGB stream as described above and one stream that utilizes optical flow information. In Chapter 7, we only extract features with the RGB stream as two-stream features did not yield better results. In Figure 2.4, we depict the modified Inception-v1 network that allows for spatio-temporal feature extraction.

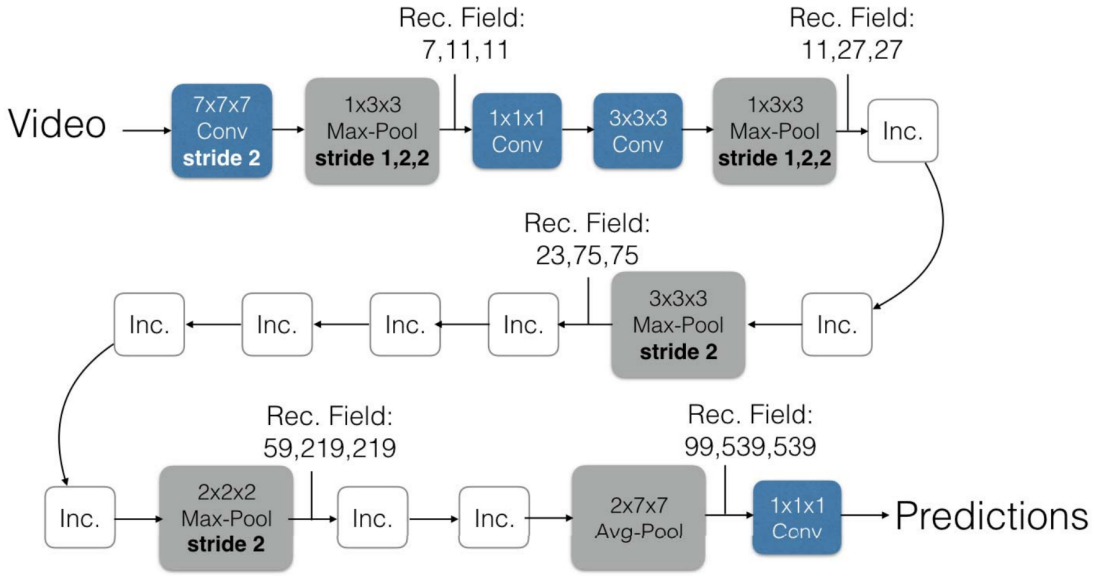


Figure 2.4: Network architecture of the RGB stream from the I3D [15] spatio-temporal DCNN. "Inc." stands for an Inception-v1 [131] block, "Rec. Field" is short for receptive field. Figure taken from the original paper [15].

2.1.4 Data Layout

All our models in this thesis are implemented in the machine learning framework TensorFlow [2]. In particular, we implemented our models in both major versions 1 and 2. Inputs, intermediate results and outputs are commonly represented in mulidimensional arrays in this work. These multidimensional arrays are also called tensors. For example, a matrix is represented as a tensor with rank 2 within TensorFlow. Scalars in TensorFlow have rank 0 and vectors rank 1.

The input images to DCNNs are represented by 4-rank tensors with $\mathbb{R}^{B\times H\times W\times C}$. The first dimension is the **B**atch dimension, which refers to the number of samples passed to any operation (i.e., the DCNN in this case). In this work, we usually omit the batch dimension but discuss the batch size in the training details for each model. The batch dimension is followed by two spatial dimensions (i.e., the **H**eight and **W**idth of each image). The final dimension represents the number of **C**hannels (e.g., for an RGB image $C = 3$). We often refer to this last dimension as depth of the tensor. Different layer types such as fully-connected layers and convolution layers often output tensors with a new last dimension. For example, a fully-connected layer with 5 output neurons projects inputs to outputs with a channel dimension size of 5. Same goes for the number of filters in a convolution layer. Layers can also operate on other dimensions such as the spatial dimensions. For example, pooling layers can reduce the spatial dimensions of an input tensor.

There are some exceptions to the typical tensor rank of 4: In this thesis, we often operate on sequences of words. If looking at the nature of text, we see that the two spatial dimensions that are typical for images are not present in text. A sentence is usually represented by a sequence of words. More specifically, sentences have a sequence length dimension instead of spatial dimensions. Thus, we represent a sentence with N words as a 3-rank tensor with dimensions $B \times N \times C$.

2.2 Language Models

In this thesis, we combine the domain of understanding the contents of images with the domain of Natural Language Processing (NLP). NLP is a widely studied topic [9, 99, 23, 108] and an essential part of this thesis. Therefore, in this chapter, we give a short overview of the concepts for working with language.

2.2.1 Words, Tokens, and n-grams

Sentences and Words When working with language, we first need to define basic entities and building blocks before introducing models for language processing. Second, when we speak of language we mostly mean a single sentence, e.g., in the task of image captioning, we generate a single sentence for a given input image. However, there is one exception in this thesis: In Chapter 6, our goal is to generate a paragraph of text that consists of multiple sentences. A sentence can be seen as a *sequence* of N words. We denote a ground-truth sentence with

$$\mathbf{y}^S = \left[\mathbf{y}_0^S, \mathbf{y}_1^S, \ldots, \mathbf{y}_{N-1}^S\right] \tag{2.1}$$

in this work. We represent a word $\mathbf{y}_t^S$ with a one-hot vector with depth $|\mathbf{V}|$ (the number of words in the vocabulary $\mathbf{V}$). Furthermore, $\mathbf{y}_0^S$ and $\mathbf{y}_{N-1}^S$ are the special start-of-sequence and end-of-sequence words (see Section 2.2.2), respectively. As the name suggests, we also call sentences *captions* in the task of image captioning.

Tokens As outlined before, a sentence is a sequence of words or tokens. A *token* is a general building block for natural language and can stand for different things depending on the context. If we tokenize a text fragment, we split it up into smaller units such as words, subwords, or even characters. In this work, a token mostly is a word. However, in Chapter 7.3 we make use of subword tokenization. Architectures like Transformers [137] and RNNs/LSTMs [64] process natural language on a token level. Thus, multiple tokens $\{\mathbf{y}_i^S\}, i \in \{0, \ldots, N-1\}$ form a sentence. We usually represent a token $\mathbf{y}_i^S$ with a one-hot encoded vector.

n-grams In this paragraph, we adapt some of the notations and definitions from Jurafsky and Martin [72, Chapter 4]. An *n-gram* is a sequence of n words. We call sequences of 1 word, 2 words, and 3 words with the terms unigram, bigram, and trigram. In comparison to a token, a unigram is always a single word in our context (character n-grams are sequences of n characters but we only consider word n-grams). When we speak of n-grams, we mostly mean n-gram *language models* (LMs). A language model assigns probabilities to sequences of words. Even though a n-gram LM is a much simpler model than an RNN or LSTM language model, n-gram LMs have an important role in language processing and are a foundation for language modeling. For example, we make use of different length n-grams during the definition of the evaluation metrics for image description generation in Chapter 2.4.

2.2.2 Language Preprocessing

In this thesis, we present multiple models and applications for translating images or videos into a textual description. Accompanying datasets come with textual ground-truth annotations for these images and videos. However, before we can utilize these annotations for training, we need to preprocess the raw text data.

The first step is to change each ground-truth sentence to lowercase and then to tokenize the sequence into word tokens. Second, these tokens are accumulated for the whole dataset and sorted top-down by their frequency, i.e., we get a vocabulary with the most common words having the smallest index. In all our use cases, we limit the vocabulary $\mathbf{V}$ to have a maximum number of words or restrict the maximum number of words by enforcing a minimum frequency for each word of the vocabulary. We replace words in the ground-truth sentences that do not appear in the vocabulary with the unknown token (`<UNK>`). As we mentioned earlier, for some experiments in the video-to-text task, we employ subword tokenization. In this case, we do not build a vocabulary like described above. For more details, we refer the reader to Chapter 7.3, where we describe the concept of subword tokenization. Furthermore, we preprocess each ground-truth sentence and add special tokens at the beginning and end of each sentence. We prepend the start-of-sequence token (`<S>`) and append the end-of-sequence token (`</S>`) at the beginning and end of each sentence, respectively. The special tokens `<S>`, `</S>`, and `<UNK>` are also part of our vocabulary. First, we encode each individual token $\mathbf{y}_t^S$ as a one-hot vector with depth $|\mathbf{V}|$. Second, $\mathbf{y}_0^S$ is the start-of-sequence token and $\mathbf{y}_{N-1}^S$ is the end-of-sequence token. We do this to tell our LMs to start generating a sentence and

on the other hand, the LM tells us to discard tokens after it emits an end-of-sequence token. Thus, we represent each sentence as a sequence of one-hot vectors starting with the special start-of-sequence token and ending with the special end-of-sequence token.

2.2.3 Word Embeddings

One-hot vectors have a fundamental flaw: Each word of the vocabulary is represented with a vector that has a one at the index of the word and zeros otherwise. First, these vectors are huge and do not carry a lot of information about the word itself, i.e., they are maximally sparse and not dense. Second, each word has the same distance to every other word. This leads to a problem since LMs cannot easily make connections between similar or related words without having huge datasets to train on such as the Wikipedia. Mikolov et al. [101] argue that continuous space LMs can have several advantages and learn vector-space word representations. In this paper, they argue that their model can learn the female/male relationship such that calculating the result of the vectors for "King - Man + Woman" results in a vector that is very close to "Queen". Later Mikolov et al. [100] present the well-known *Word2Vec* model, which is able to learn even better vector representations for words. They construct two models that learn to predict a missing word given context words or predict preceding and posterior words given a single word. These tasks contain an *embedding layer* that transforms one-hot encodings into a continuous vector space that has a significantly lower dimension d than the length of the vocabulary. These continuous vector representations also have the advantage that words have different distances to each other. Synonymous words (e.g., *dog* and *puppy*) are closer to each other than unrelated words. Depending on the dataset, use-case, and training objective the words can get clustered by other similarity concepts than synonyms.

Word embeddings are just linear projections that can be learned in parallel to some surrogate tasks such as predicting a missing word. In our thesis, we employ word embeddings to project an input one-hot vector word $\mathbf{y}_i^S$ into a vector-space word representation

$$\mathbf{W}^e \mathbf{y}_i^S, \tag{2.2}$$

where $\mathbf{W}^e \in \mathbb{R}^{d \times |\mathbf{V}|}$ with $d \ll |\mathbf{V}|$ is the word embedding matrix that we learn from scratch.

However, in our case, the surrogate task is the main task. For instance, for the task of image captioning, we embed our one-hot vectors into a lower dimension with a word embedding layer before feeding them to our LSTM language model. Vinyals et al. [139] found that learning word embeddings from scratch yields a similar performance as initializing them with pre-trained embeddings learned on a text-only dataset.

2.2.4 Basic Recurrent Neural Network Cell

In Part II of this thesis, we work with recurrent language generation models. These models are constructed using *Recurrent Neural Network* (RNN) cells. As the name suggests, these cells are applied in a recursive manner. This means that an RNN cell

shares its parameters along multiple calculation steps in a row. A basic RNN cell has two learnable parameter matrices $\mathbf{W}^h$ and $\mathbf{W}^x$ and calculates an output hidden state $\mathbf{h}_t$ given an input $\mathbf{x}_t$ and a hidden state $\mathbf{h}_{t-1}$ from the previous step. We index these calculation steps, i.e., iteration steps with the variable t and calculate the new hidden state for the current iteration step t as follows:

$$\mathbf{h}_t = \tanh\left(\mathbf{W}^h\mathbf{h}_{t-1} + \mathbf{W}^x\mathbf{x}_t\right). \tag{2.3}$$

A recurrent neural network computes the RNN cell for every iteration step given an embedded input sequence. Thus, we say we can unroll an RNN cell along the time dimension. Unrolling means that the same cell with the same weights is repeated, but has different inputs for every iteration step. We denote the outputs as hidden states and utilize them in two ways. First, we need them as an input for the computation of the RNN cell in the next iteration step. Second, we use the hidden state as an intermediate value to further process the result, e.g., we project the hidden state back to the dimensionality of the vocabulary to predict a word. We depict a basic RNN cell in Figure 2.5.

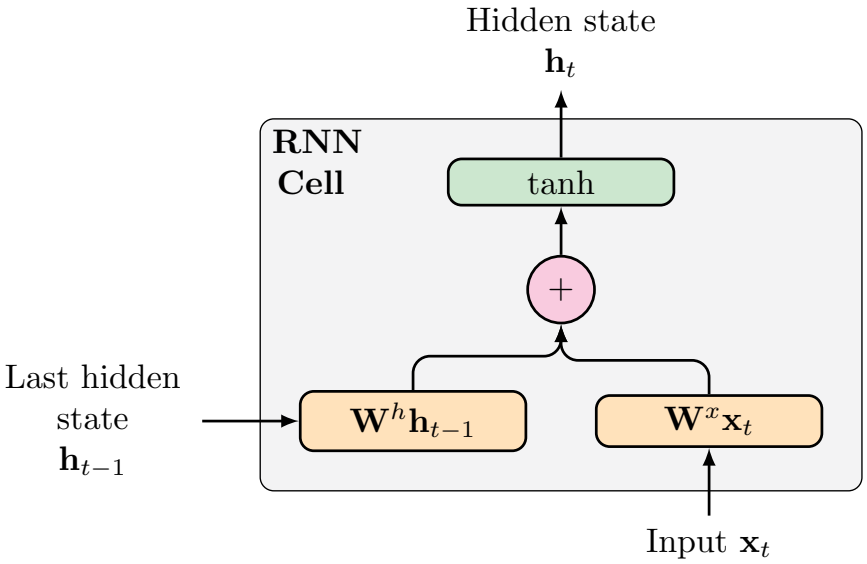


Figure 2.5: Schematic illustration of a basic RNN cell. Arbitrary inputs and the hidden state of the last iteration are both fed through a linear projection, summed up, and then activated with a tanh non-linearity that yields a new hidden state.

2.2.5 Long Short-Term Memory Cells

The name RNN is a generic term for all kinds of recurrent cells. One example is the basic RNN cell, which we introduced before. However, RNNs have some downsides that

make them unpractical to use in LMs. If we look at the sentence "The grass is *green*", one can easily conclude the sentence with the word "green" *given* the context "The grass is". With their internal hidden state, RNN cells are also able to solve this task because the distance between the information and the position where it is needed to predict the word is short. However, as this distance grows, RNNs tend to forget contextual information from previous iteration steps. Although in theory, basic RNN cells are able to memorize this information, they have the tendency to forget contextual information after a few iteration steps, i.e., they are not able to make a connection between words that are far away from each other (*long-range dependencies*). Second, basic RNNs can easily cause vanishing or exploding gradients during training if the number of iteration steps is too big. When a gradient is small or big in magnitude and is propagated back through multiple identical RNN cells, it may die out or explode and lead to numerical instabilities.

To counteract these issues, Hochreiter et al. [64] presented the Long Short-Term Memory (LSTM) cell. The LSTM cell is also an RNN cell as it computes its outputs recurrently. To deal with exploding and vanishing gradients, the LSTM cell contains multiple gates that control the flow of information. In Figure 2.6, we show a schematic model of an LSTM cell. In particular, the LSTM cell contains a forget gate, an input gate, and an output gate that control whether to forget the previous cell value, whether it should consume its input, and whether to output a new cell value, respectively. These gates are multiplicative layers that can keep the previous value or discard it. We calculate the multiplicative factor of a gate with the sigmoid σ activation function. Thus, the gate can forward any fraction ($\in [0, 1]$) of its input. The calculation rule of an LSTM cell is more difficult than the basic RNN cell as we need to calculate the gate values individually. Furthermore, the LSTM cell has an internal cell state (c_t) which memorizes information. We can remove or add information to this cell state with the results of the cell gates. The output $\mathbf{h}_t$ and cell state c_t for the iteration step t and input $\mathbf{x}_t$ can be calculated as follows:

$$i_t = \sigma\left(\mathbf{W}^{ix}\mathbf{x}_t + \mathbf{W}^{ih}\mathbf{h}_{t-1}\right) \tag{2.4}$$

$$f_t = \sigma\left(\mathbf{W}^{fx}\mathbf{x}_t + \mathbf{W}^{fh}\mathbf{h}_{t-1}\right) \tag{2.5}$$

$$\tilde{c}_t = \tanh\left(\mathbf{W}^{cx}\mathbf{x}_t + \mathbf{W}^{ch}\mathbf{h}_{t-1}\right) \tag{2.6}$$

$$o_t = \sigma\left(\mathbf{W}^{ox}\mathbf{x}_t + \mathbf{W}^{oh}\mathbf{h}_{t-1}\right) \tag{2.7}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \tag{2.8}$$

$$\mathbf{h}_t = o_t \circ \tanh\left(c_t\right), \tag{2.9}$$

where $\mathbf{W}^{\cdot}$ are learned parameter matrices of the LSTM cell and $\circ$ is the element-wise product.

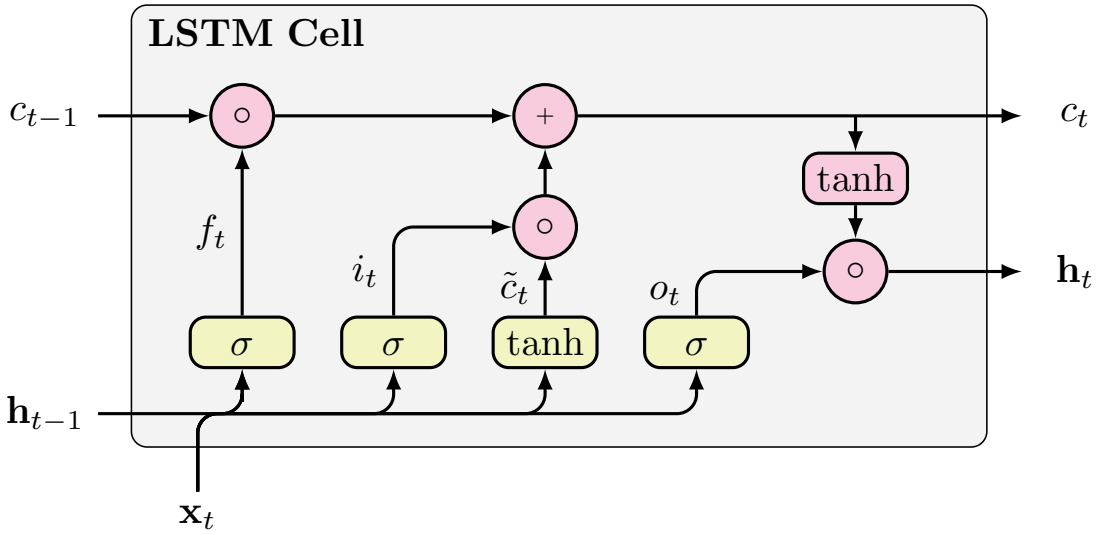


Figure 2.6: Schematic illustration of an LSTM cell. Figure adapted to our nomenclature from [102]. Multiple gates control whether to forget parts (f_t) from the cell state, whether to add parts (i_t) of the input to it, or which parts of the cell state should be outputted (o_t) as the new hidden state.

.

2.2.6 Combining Cells into Networks

In this thesis, we integrate recurrent neural networks into the decoder of our encoder-decoder architectures. In all our cases, we generate language with a decoder network that is fed with visual features from an encoder network. The name RNN suggests that many RNN/LSTM cells are combined into a network. RNN cells typically are only defined by their respective input and output dimension which we set to 512 in most cases. However, in our models from Chapter 4 and Chapter 5, we only include a single LSTM cell in the decoder network. Yet, we unroll (see Chapter 2.2.4) the LSTM cell multiple times for every input word such that we consecutively re-evaluate the LSTM with the same parameters but different inputs. Additionally, the decoder network contains a word embedding and a fully-connected layer that projects hidden states back to the vocabulary subspace.

In Chapter 6, we do not generate single sentences but multiple sentences, i.e., paragraphs, at once. Therefore, we implement multiple LSTM cells in a hierarchical way, i.e., outputs from one hierarchy level (sentence-level) are inputs to the second hierarchy level (word-level). Specifically, we unroll a sentence LSTM to generate vectors representing individual sentences which are then initialization states for a word LSTM that unrolls for every sentence to generate words. In addition, we can also stack RNN cells vertically so that the output of one LSTM cell is input to another cell. Note that this is different from the hierarchical way employed in Chapter 6. In hierarchical LSTM, we implement

two layers of LSTM cells. In particular, we unroll an LSTM in the second layer for every iteration step of the first LSTM layer.

2.2.7 Beam Search

In the inference case, i.e., when we try to predict a sentence for an unseen image, we generate the sentence word by word. Normally, a LM is trained to maximize the probability of a word given previous words and an input image. Thus, during inference one way to generate a sentence word per word is to always predict the most likely word. This method is called greedy sampling as the algorithm greedily predicts the most likely word in every iteration.

Beam search on the other hand is an algorithm that tries to predict the b most likely sentences. However, it is computationally infeasible to compute the probabilities for every word and then subsequently compute all word probabilities for every possible word from the previous iteration step. More specifically, for a sentence of length N, we would need to evaluate the network $|\mathbf{V}|^{N-1}$ times, where $|\mathbf{V}|$ is the cardinality of the vocabulary.

Beam search on the other hand always keeps the b most likely sentences up to that point in memory. The probability of a sentence $\mathbf{y}^S = [\mathbf{y}_0^S, \ldots, \mathbf{y}_t^S]$ up to an iteration step t is calculated by multiplying each individual word probability:

$$p_{0:t} = \prod_{i=0}^{t} p(\mathbf{y}_t^S | \mathbf{y}_0^S, \ldots, \mathbf{y}_{t-1}^S). \tag{2.10}$$

In reality, we calculate the log probabilities $\log(p_{0:t}) = \sum_{i=0}^{t} \log(p(\mathbf{y}_t^S | \mathbf{y}_0^S, \ldots, \mathbf{y}_{t-1}^S))$ to improve numerical stability. Beam search then keeps a list of the top b sentences up to iteration step $t-1$ and re-evaluates the sentence probability ($\log(p_{0:t}) = \log(p_{0:t-1}) + \log(p(\mathbf{y}_t^S | \mathbf{y}_0^S, \ldots, \mathbf{y}_{t-1}^S))$) for all words of the vocabulary at every iteration step for each of the b sentences. The sentence probabilities are then sorted top-down and only the b most likely sentences are kept for generating the next word. Beam search can help to explore the space of possible answers while keeping the computational overhead reasonably low ($1 + (N-1) \cdot b \ll |\mathbf{V}|^{N-1}$).

2.3 Attention Mechanisms and Transformers

In Part III of our thesis, we introduce models based on self-attention mechanisms by using a Transformer architecture. This architecture has been a big leap in the machine translation task, as it solves the problem of long-range dependencies with attention mechanisms. Furthermore, the architecture has set new state-of-the-art scores for multiple tasks [29, 14, 24] and outperforms many LSTM systems.

2.3.1 Multiplicative Attention

Attention is a basic concept that was early introduced for the *Neural Machine Translation* (NMT) task [99]. Luong et al. presented an attention mechanism for NMT architectures that is often referred to as *multiplicative attention* in the literature. In the following, we explain the basic concept of the Luong attention, which we use as top-down attention in the VQA task (see Chapter 5). Furthermore, the multiplicative attention forms a basic building block for the scaled dot-product attention (see Chapter 2.3.2) that is utilized in the Transformer architecture.

The motivation for the multiplicative attention in the machine translation task was to attend to different parts of the source sentence while generating each word of the translation. This is a huge advantage for the NMT task as sentences with the same meaning have different structures for different languages. Thus, the model can focus on different parts of the source sentence that correlate with the translation.

Transferred to more generic terms, we want to select elements of some matrix $\mathbf{V}$ by importance given some query vector $\mathbf{Q}$. For example, we want to weight parts of an image differently *given* a question in the task of VQA. This can be done by calculating weights $\boldsymbol{\alpha}$ and subsequently weighting the elements of the value matrix.

Let the value matrix $\mathbf{V} \in \mathbb{R}^{K \times d_V}$ be an arbitrary feature map and the query vector $\mathbf{Q} \in \mathbb{R}^{d_Q}$. Then, we can calculate an attention score for every spatial location $\mathbf{V}_i, i \in \{1, \dots, K\}$ of a feature map

$$\mathbf{a}_i = \mathbf{w}^{\text{att}} \cdot [\mathbf{V}_i, \mathbf{Q}], \tag{2.11}$$

where $[\cdot, \cdot]$ concatenates two vectors, $\mathbf{w}^{\text{att}} \in \mathbb{R}^{d_Q + d_V}$ is a learnable parameter vector, and $\cdot$ is the dot-product. The attention scores $\mathbf{a}$ are normalized into a probability distribution over all spatial locations $\{1, \dots, K\}$ with the softmax function $\phi(\cdot)$:

$$\boldsymbol{\alpha} = \phi(\mathbf{a}), \boldsymbol{\alpha} \in \mathbb{R}^K. \tag{2.12}$$

This yields attention weights $\boldsymbol{\alpha}$ that sum up to 1, which allow us to re-weight the original feature map according to the relative importance of each spatial location i

$$\hat{\mathbf{V}} = \sum_{i=1}^{K} \boldsymbol{\alpha}_i \mathbf{V}_i = \boldsymbol{\alpha} \cdot \mathbf{V}. \tag{2.13}$$

This is a re-weighting of elements in the value matrix according to some query vector. That is, the value matrix $\mathbf{V} \in \mathbb{R}^{K \times d_V}$ is reduced to $\hat{\mathbf{V}} \in \mathbb{R}^{d_V}$. This operation can replace any final average-pooling operation in a CNN and can improve scores drastically because the network "looks" at different parts of an image.

2.3.2 Scaled Dot-Product Attention

Vaswani et al. [137] introduced the *scaled dot-product attention* that is a variant of multiplicative attention. In addition to query ($\mathbf{Q} \in \mathbb{R}^{N_Q \times d_K}$) and value ($\mathbf{V} \in \mathbb{R}^{N_K \times d_V}$), this operation also has a key ($\mathbf{K} \in \mathbb{R}^{N_K \times d_K}$) as input. $N_\cdot$ is the sequence length of the inputs, e.g., of the query, key, and value inputs to the scaled dot-product attention.

Note that the sequence length of keys and values must be the same (N_K). Typically $N_Q = N_K$ in the encoder (see Section 2.3.4, Encoder) and $N_Q \neq N_K$ in the decoder (see Section 2.3.4, Decoder). Furthermore, the depth dimension of queries and key matrices need to match (d_K). One can imagine this attention as a lookup table that looks for the keys nearest to a query and then returns the corresponding values weighted by their relative importance. We depict the scaled dot-product attention in Figure 2.7. The scaled dot-product attention can be computed as follows:

$$\hat{\mathbf{V}} = \text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \phi\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_K}}\right) \cdot \mathbf{V}. \tag{2.14}$$

In similar fashion to Section 2.3.1, $\phi\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_K}}\right)$ calculates attention weights that sum up to 1. These attention weights are then used to weight the values, which yields attended values $\hat{\mathbf{V}} \in \mathbb{R}^{N_Q \times d_V}$. Vaswani et al. [137] argue that multiplicative attention has a drawback in contrast to additive attention [9]. In particular, they suspect that larger dimensions d_K of the key $\mathbf{K}$ lead to an unstable behavior in the softmax function as gradients become extremely small. As a consequence, they introduce a scaling factor $\sqrt{d_K}$, hence, the name *scaled* dot-product attention.

2.3.3 Multi-Head Attention

Vaswani et al. [137] extend the single scaled dot-product into a *multi-head attention* (MHA). That is, they divide the dimension of the attention into multiple heads to perform multiple attentions at once. Furthermore, the vanilla Transformer model [137] operates on a single dimension d_{model} rather than having different dimensions for d_K and d_V. For the multi-head attention, we split d_{model} into h parallel attention heads of size $\frac{d_{\text{model}}}{h} \in \mathbb{N}$. The interested reader might have noticed that the scaled dot-product attention from Chapter 2.3.2 does not have any learnable parameters. In contrast to simple multiplicative attention (Chapter 2.3.1), we project queries, keys, and values into the dimension $\frac{d_{\text{model}}}{h}$ for every attention head. This projection happens *before* feeding them into scaled dot-product attention mechanism for every attention head.

After we performed this parallel scaled dot-product attention, we concatenate the attended values and project them back to the model dimension d_{model}. If we use $h = 8$ attention heads for a model dimension of $d_{\text{model}} = 512$, we get $d_K = d_V = \frac{d_{\text{model}}}{h} = \frac{512}{8} = 64$. We depict a multi-head attention block in Figure 2.8. The authors claim that the multi-head attention allows the model to attend to different representation subspaces, which would not be possible with a single scaled dot-product attention as averaging effects would prevent this. We can calculate the multi-head attention as follows:

$$\text{MultiHeadAttention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = [\text{head}_1, \text{head}_2, \ldots, \text{head}_h] \cdot \mathbf{W}^{\text{MHA}}, \tag{2.15}$$

where

$$\text{head}_i = \text{Attention}(\mathbf{Q} \cdot \mathbf{W}_i^{\mathbf{Q}}, \mathbf{K} \cdot \mathbf{W}_i^{\mathbf{K}}, \mathbf{V} \cdot \mathbf{W}_i^{\mathbf{V}}). \tag{2.16}$$

Note that the actual number of columns in the input matrices $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ is equal to the model dimension. Therefore, for the MHA the input matrices are of following dimensions: $\mathbf{Q} \in \mathbb{R}^{N_Q \times d_{\text{model}}}$, $\mathbf{V} \in \mathbb{R}^{N_K \times d_{\text{model}}}$, and $\mathbf{K} \in \mathbb{R}^{N_K \times d_{\text{model}}}$. The projections

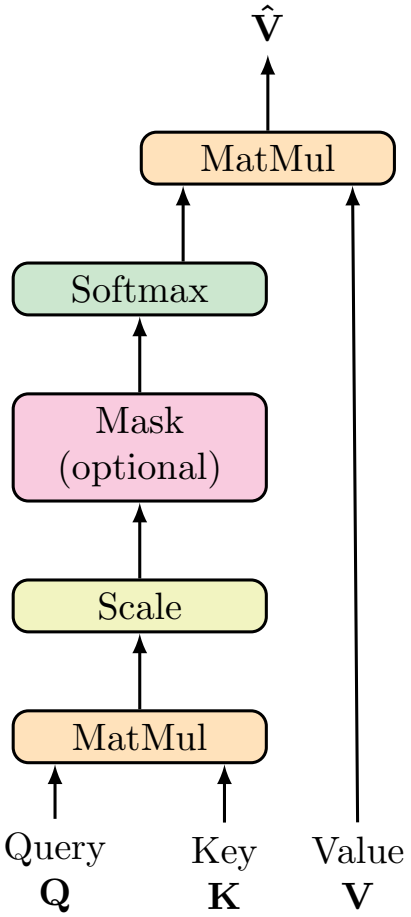


Figure 2.7: Graphical representation of the scaled dot-product attention operation. The mask operation can hide inputs by effectively reducing the softmax attention score to zero. The mask is only used in the decoder (see Chapter 2.3.4). Figure redrawn from [137].

are fully-connected layers without biases and $\mathbf{W}_i^{\mathbf{Q}} \in \mathbb{R}^{d_{\text{model}} \times d_K}$, $\mathbf{W}_i^{\mathbf{K}} \in \mathbb{R}^{d_{\text{model}} \times d_K}$, $\mathbf{W}_i^{\mathbf{V}} \in \mathbb{R}^{d_{\text{model}} \times d_V}$ and $\mathbf{W}^{\text{MHA}} \in \mathbb{R}^{h \cdot d_V \times d_{\text{model}}}$. These parameters are the only learned parameters in the multi-head attention module while the attention mechanism itself is merely a calculation rule.

2.3.4 Transformers

The Transformer model is based on an encoder-decoder structure, which is commonly used in NMT and image captioning tasks. In Part III, all our models and architectures are based on the Transformer. In the following, we give a short introduction into the Transformer architecture by describing its main components. We depict a vanilla Transformer in Figure 2.9.

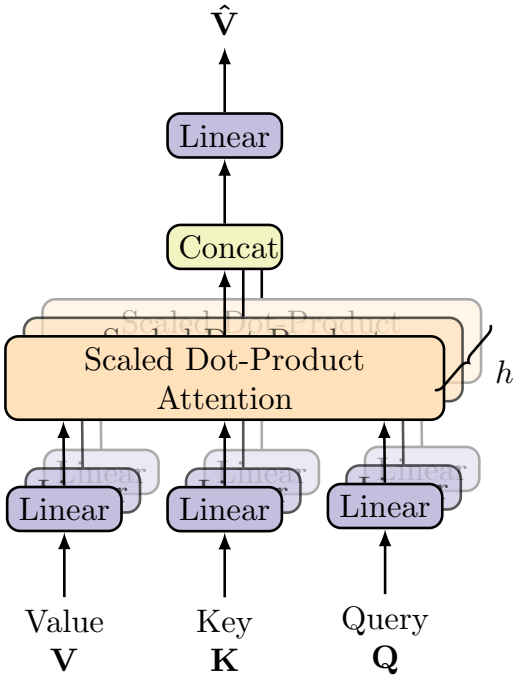


Figure 2.8: Schematic representation of the multi-head attention operation. Figure redrawn from [137].

Encoder The encoder of a Transformer consists of $\mathcal{N}$ encoder blocks, which are all built in the same way: A multi-head attention layer with subsequent layer normalization, followed by a small feed-forward network (FFN), again with layer normalization. Additionally, there is a skip connection before the multi-head attention layer to the first normalization layer and before the FFN network and the second layer normalization. We depict an encoder block within the gray box on the left side in Figure 2.9. Note that an *Add & Norm* layer consists of the element-wise addition of the skip connection, a layer normalization, and a dropout operation.

Decoder We depict the decoder of the Transformer on the right side in Figure 2.9. The decoder also consists of $\mathcal{N}$ decoder blocks which are built in a similar manner as encoder blocks. However, there are some key differences. First, the input to the decoder (named as outputs in the Transformer model in Figure 2.9) is shifted to the right. This means, that we prepend a start-of-sequence token and, thus, shift the sequence to the right. The input to the decoder then goes through a masked multi-head attention layer, which means that parts of the inputs can be masked out. This is important for text generation as the decoder should not be able to look at words in the future during training. That means, when processing a unigram with index i, the decoder can only look at unigrams of the same sequence with index $< i$. Second, we feed the multi-head attention the key and value from the encoder and the query from the masked multi-head attention. That

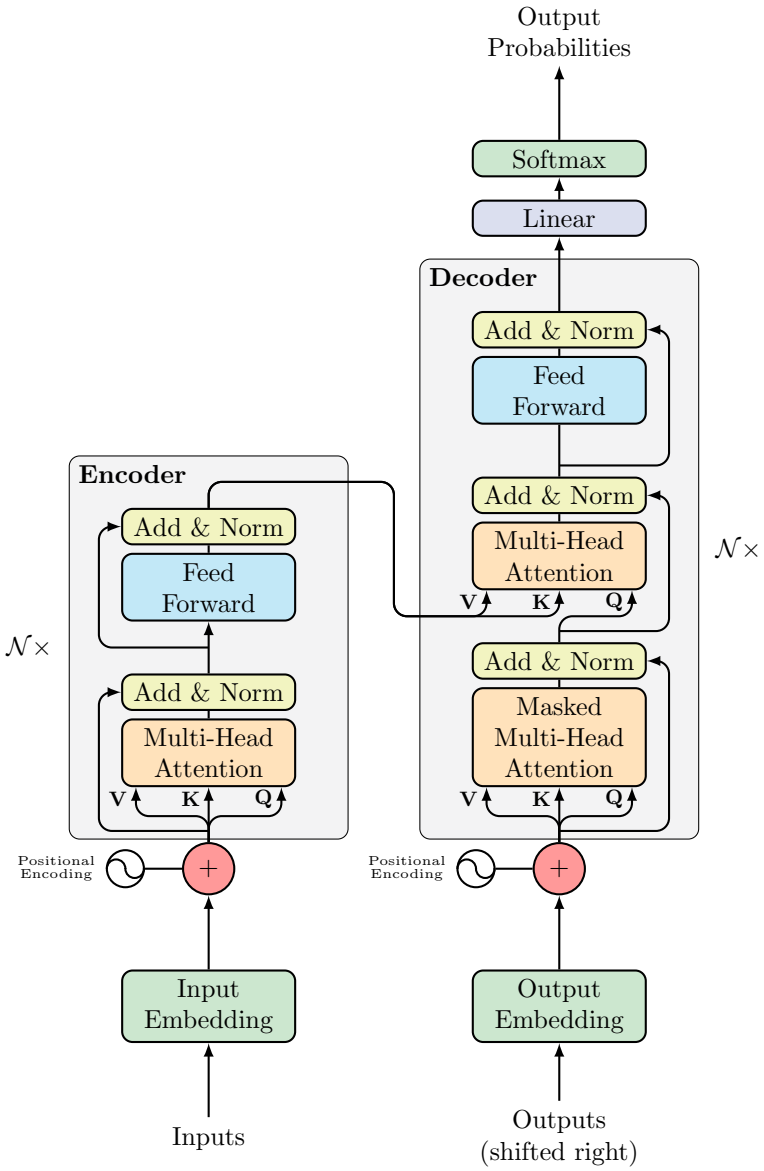


Figure 2.9: Full Transformer architecture, which was introduced by Vaswani et al. Figure redrawn from [137]. On the left side, we see encoder blocks while on the right side are the decoder blocks. The input to the decoder (named as outputs) is shifted to the right. For example, a sentence without a start-of-sequence token is shifted to the right while a start-of-sequence token is prepended. The output probabilities on the top of the decoder then represent the sentence without the start-of-sequence token. We optimize these probabilities with a softmax cross-entropy loss during training phase. In Chapter 7.4, we re-purpose this architecture for the task of video-to-text translation.

is, the decoder then weights features from the encoder given a query generated by the attended decoder inputs. Finally, the decoder then outputs a sequence of vectors with dimension $\in \mathbb{R}^{d_{\text{model}}}$ which are fed through a linear layer that projects these vectors into a vector with depth of the vocabulary's size ($|\mathbf{V}|$). A softmax function transforms these outputs into probabilities, which are optimized with a softmax cross-entropy loss during training. Note that the targets are the same as the decoder's inputs but not shifted to the right, i.e., without a start-of-sequence token.

Long-Range Dependencies The Transformer tries to solve one serious drawback of RNNs, i.e., the inability to cope well with long-range dependencies. Even LSTMs, whose main motivation was to solve this issue, still cannot resolve long-range dependencies over many words. With the Tranformer's MHA module, we can easily correlate any input word with any other input word independently of their respective distance to each other. Within the MHA, every input word is able to "see" every other input word.

Embedding of Tokens Similar to other models in the realm of NLP, we embed words/tokens with a word embedding $\mathbf{W}^e$ into a lower-dimensional space to better seize word properties such as described in Chapter 2.2.3.

Positional Encoding We already explored the advantage of Transformers, which allows each word of a sequence to see each other word because of the properties of the MHA. However, attention layers cannot get a sense of ordering as the input is a set of vectors with no explicit ordering. Because of that, Vaswani et al. add a so-called positional encoding onto the embedded sequences (see the bottom of Figure 2.9). The positional encoding is a vector that is simply added on top of the embedding vector. Vaswani et al. [137] present two types of positional encoding: a learned one and a fixed one. Most modern works make use of the fixed positional encoding which consists of sine and cosine functions of different frequencies:

$$\mathbf{PE}_{\text{pos},2i} = \sin\left(\frac{\text{pos}}{10000^{2i/d_{\text{model}}}}\right) \tag{2.17}$$

$$\mathbf{PE}_{\text{pos},2i+1} = \cos\left(\frac{\text{pos}}{10000^{2i/d_{\text{model}}}}\right), \tag{2.18}$$

where pos is the position index to encode and i the dimension as can be seen on the y-axes in Figure 2.10. This figure shows the sine and cosine parts for all positional encodings with $d_{\text{model}} = 64$ (i.e., $i \in [0, 31]$) up to a maximum position of 31.

2.4 Metrics

As this thesis resides in the domain of NLP, we need a measure to assess whether sentences (i.e., sequences of unigrams) are "good" or "bad". The best way to evaluate if a generated image description fits a given image is to ask humans to rate the caption

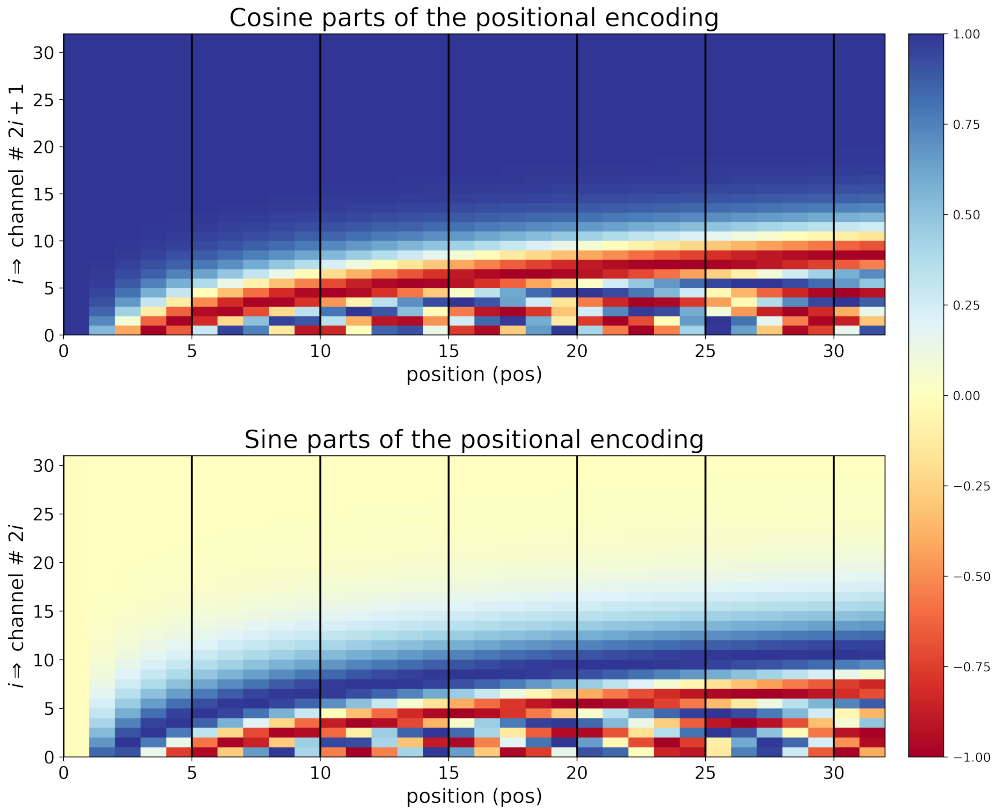


Figure 2.10: Visual representation of the positional encoding (PE) for sequences up to length 32. On top, we show cosine parts of the positional encoding according to Equation 2.18, while we depict the sinus parts according to Equation 2.17 of the PE on the bottom. We inserted the black vertical bars to help to see the differences between the cosine and sine parts of the PE. Visualization generated with code adapted from [1].

quality. However, this method is time-consuming and very cost-intensive and cannot be applied in an automatic way to a huge set of generated sentences. Yet, this problem is not new to the machine learning community: In different tasks such as machine translation [107] and text summarization [123] a system also needs to automatically evaluate whether a generated sentence is good in comparison to some reference sentences which were written by humans. Thus, even before the image captioning task was first proposed, the machine translation community has had to come up with some possible solutions to automatically evaluate candidate translations against some reference (i.e., ground-truth) sentences.

In the following, we give a short overview of metrics that we utilize to analyze the performance of various models. This overview of metrics is based on their original papers and adapted to our setting. We present metrics that were developed for machine

[1] `https://www.tensorflow.org/text/tutorials/transformer#positional_encoding`

translation tasks (i.e., BLEU [108] in Chapter 2.4.1 and METEOR [10] in Chapter 2.4.3) and for text summarization (i.e., ROUGE [89] in Chapter 2.4.2). Furthermore, we explain the CIDEr [138] metric (see Chapter 2.4.4), which was developed for the image captioning task. In addition, this metric has been shown to correlate better with human judgments than the other metrics. Finally, we present the VQA accuracy metric, which falls a little bit out of line. In contrast to the other metrics, this metric merely calculates a modified form of the accuracy as the VQA task is modeled as a classification task.

2.4.1 BLEU-N

BLEU (*Bilingual Evaluation Understudy*) was proposed by Papineni et al. [108] to allow for a quick way to evaluate machine translations. This evaluation metric is designed to be cost-effective, thus, be automatic and not involve humans, nevertheless, correlate with human judgement. At its essence, this metric merely calculates the precision of n-grams occurring in reference sentences. The authors present the *modified n-gram precision*, a modification to the traditional precision measure. If we look at the example from [108] in Figure 2.11, we can easily compute precision by first counting the number of words in the candidate sentence that occur in any of the reference sentences ($Count(word)$). Second, we need to count the total number of words in the candidate sentence and calculate the ratio between the two. However, this leads to a problem as the precision for the candidate sentence in Figure 2.11 is $\frac{7}{7} = 1$, which is clearly no good score for the given candidate sentence. Note that the precision is the same regardless of choosing the first or the second reference sentence for comparison.

Candidate:	the the the the the the the.
Reference 1:	The cat is on the mat.
Reference 2:	There is a cat on the mat.

Figure 2.11: Example candidate sentence and two reference sentences.

In contrast, the modified n-gram precision disallows the over-counting of words, i.e., words found in a reference sentence can be used up. Thus, we modify the first step by clipping the count of found words by the maximum number of times a word is contained within any reference sentence:

$$Count_{clip}(word) = \min(Count(word), Max\_Ref\_Count(word)). \tag{2.19}$$

As a consequence, the modified n-gram precision for the example above becomes $\frac{2}{7}$. In particular, the word *the* is contained twice in the first reference sentence and once in the second reference sentence. Therefore, $Max\_Ref\_Count(the) = 2$ and the modified n-gram precision evaluates to $\frac{2}{7}$.

It is not difficult to extend this approach from words to n-grams, i.e., n subsequent words. Instead of counting occurrences of single words, we count whether each sequence of n words is contained in any reference sentence. Furthermore, we only considered a

single candidate so far. However, we want to calculate the BLEU scores over a whole corpus of samples. That means that for a single sample (e.g., an image from a dataset), we generate one candidate sentence C. We denote all candidate sentences (one for each sample) with *Candidates*. Thus, if we want to compute the modified n-gram precision, we need to evaluate $Count_{clip}(n\text{-}gram)$ for each distinct n-gram within each candidate sentence C and sum these counts up. Second, to compute the precision, we need to count all n-grams of every candidate sentences. To summarize, we can define the modified precision for n-grams of length n as

$$P_n = \frac{\sum\limits_{C \in \{Candidates\}} \sum\limits_{n\text{-}gram \in C} Count_{clip}(n\text{-}gram)}{\sum\limits_{C' \in \{Candidates\}} \sum\limits_{n\text{-}gram' \in C'} Count(n\text{-}gram')}. \tag{2.20}$$

Note that this formula counts all matching n-grams over all candidate sentences and divides these matches by the unclipped count of all n-grams contained in the set of candidate sentences. If the same n-gram occurs at multiple places within a candidate sentence it is only counted once, i.e., $n\text{-}gram \in C$ selects all distinct n-grams contained in C. For example, the candidate sentence from Figure 2.11 contains the single unigram (1-gram) *the*. Thus, $Count_{clip}(the)$ evaluates to 2 and $Count(the) = 7$. As Equation 2.20 describes, we sum up all counts of matches for every n-gram for all candidate sentences $C \in \{Candidates\}$ in the nominator. In the denominator, we sum up the counts of the distinct n-grams within every candidate sentence $C \in \{Candidates\}$.

We can compute the BLEU-N scores by first computing the geometric average[2] of each modified n-gram precision P_n up to N:

$$\begin{aligned} \sqrt[N]{\prod_{n=1}^{N} P_n} = \exp\left(\log\left(\left[\prod_{n=1}^{N} P_n\right]^{\frac{1}{N}}\right)\right) &= \exp\left(\frac{1}{N} \cdot \log\left(\prod_{n=1}^{N} P_n\right)\right) \\ &= \exp\left(\frac{1}{N} \cdot \sum_{n=1}^{N} \log P_n\right). \end{aligned} \tag{2.21}$$

Note that this transformation is only valid for $P_n > 0$. Finally, the BLEU-N score is defined by

$$\text{BLEU-N} = \text{BP} \cdot \exp\left(\frac{1}{N} \sum_{n=1}^{N} \log P_n\right), \tag{2.22}$$

whereas BP is a sentence brevity penalty that penalizes candidate sentences which are shorter than the reference sentence. The brevity penalty is defined as follows:

$$\text{BP} = \begin{cases} 1, & \text{if } c > r \\ e^{1-r/c}, & \text{if } c \leq r. \end{cases} \tag{2.23}$$

[2]The geometric average of N data samples $\{a_1, \ldots, a_N\}$ is defined as $\sqrt[N]{\prod_{i=1}^{N} a_i}$.

Here c is the length of the all candidate sentences and r is the effective reference corpus length. We can obtain the effective reference corpus length by adding up all lengths (i.e., number of words) of the reference sentences that best match the length of the candidate sentences. In this work, we mostly report the BLEU-4 score from all BLEU-N scores.

2.4.2 ROUGE-L

In contrast to BLUE, the ROUGE [89] (*Recall-Oriented Understudy for Gisting Evaluation*) metric is recall-focused as the name suggests. Originally, ROUGE was developed for evaluating summaries. In their paper, the authors presented four variants of ROUGE, i.e., ROUGE-N, ROUGE-L, ROUGE-W, and ROUGE-S. As it is common practice in the image captioning community, we only evaluate the ROUGE-L score.

The main idea behind the ROUGE-L score is to count the longest common subsequence (LCS) between a candidate sentence and any reference sentence. With a candidate sentence $\mathbf{n}$ and a reference sentence $\mathbf{y}^S$, we can define the LCS-based recall and precision:

$$R_{\text{LCS}} = \frac{\text{LCS}(\mathbf{n}, \mathbf{y}^S)}{|\mathbf{y}^S|} \tag{2.24}$$

$$P_{\text{LCS}} = \frac{\text{LCS}(\mathbf{n}, \mathbf{y}^S)}{|\mathbf{n}|}, \tag{2.25}$$

whereas $|\cdot|$ is the length of the corresponding sequence. We can compute the ROUGE-L score by combining the LCS-based recall and precision with the F_β [121, 22] score:

$$\text{ROUGE-L} = F_\beta = \frac{(1+\beta^2) \cdot R_{\text{LCS}} \cdot P_{\text{LCS}}}{R_{\text{LCS}} + (\beta^2 \cdot P_{\text{LCS}})}. \tag{2.26}$$

In particular, the harmonic mean[3] is a special case of the F_β score . If we replace β for 1, we see that F_1 evaluates to the definition of the harmonic mean. Chinchor [22, p. 25] argues that β is the relative importance which is given to recall over precision. In fact, the F_β score is based on Van Rijsbergen's effectiveness measure E. In particular, he derives the E measure such that it "... measures the effectiveness of retrieval with respect to a user who attaches β times as much importance to recall as precision." [121, p. 133f.] In the MSCOCO evaluation script (see Chapter 2.4.5), β is set to 1.2 as default. Thus, as the name of ROUGE-L implies, we weight recall slightly more than precision in this metric.

2.4.3 METEOR

The authors of the METEOR [10] (*Metric for Evaluation of Translation with Explicit ORdering*) metric specifically designed the metric to address a weakness of the BLEU

[3] The harmonic mean is defined as the reciprocal of the arithmetic mean of the reciprocals of given samples. For example, the harmonic mean of precision P and recall R is defined as $\left(\frac{P^{-1}+R^{-1}}{2}\right)^{-1} = \frac{2}{\frac{1}{P}+\frac{1}{R}} = \frac{2}{\frac{R+P}{RP}} = \frac{2RP}{R+P}$.

metric. They argue that BLEU lacks the recall measure. Furthermore, BLEU only considers higher-order n-grams ($n > 2$) as an indirect measure of grammatical correctness instead of respecting word ordering. Also, BLEU does not pay attention to explicit word-to-word matching (i.e., it does not consider simple morphological variants of the word which have the same word stem). Furthermore, the geometric mean results in a score of zero if any of the precision scores $P_n = 0$, i.e., the product $\prod_{n=1}^{N} P_n$ in Equation 2.21 evaluates to zero. Also, in the transformed Equation 2.22, $\log(P_n)$ is undefined for $P_n = 0$. Thus, in such cases, we set the BLEU score to zero if any of the precision scores is zero. Because of this behaviour, the authors of METEOR [10] argue that BLEU scores on the sentence level are meaningless.

In METEOR, for a candidate sentence, we compute a score against all reference sentences, and only the best score is reported. In contrast to the aforementioned metrics, METEOR creates an alignment between the two sequences, i.e., each unigram from the candidate sequence is mapped to a unigram in a reference sentence or *zero*. Note that in contrast to ROUGE, we align unigrams instead of the LCS. This alignment is done in three stages subsequently, i.e., if we are unable to map a unigram in the first stage, we may try again in the second stage and so on.

In the first phase of a stage a module tries to map the unigrams based on different criteria. The "exact" module maps two unigrams in case of a direct match, e.g., *computers* maps to *computers*, but not to *computer*. The second module is called "porter stem". This module only maps unigrams if they match after they were stemmed with the porter stemmer [114] (e.g., in this case *computers* maps to *computer*). The last module maps synonymous unigrams and we call it "WN synonymy". Note that each stage uses a different module, i.e., the first stage, the second stage and the third stage map unigrams with the "exact" module, "porter stem" module, and "WN synonymy" module, respectively.

The second phase of each stage selects the best alignment based on two criteria. Note that there could be multiple alignments for a unigram, i.e., the unigram *computer* in a candidate sentence can map to multiple instances of *computer* in the reference sentence. However, we only want that a unigram maps to at most *one* unigram in the reference sentence. Therefore, we select the largest subset of alignments. This subset is chosen such that it only contains alignments abiding to the following: If there are more subsets with the same cardinality, we choose the subset with the fewest *crossings*, i.e., literal line crossings if we connect each unigram alignment between candidate and reference sentence with a drawn line.

Similar to ROUGE-L (see Chapter 2.4.2), we can then calculate precision P_{align} and recall R_{align}. However, in contrast to other metrics and as stated above, we count the number of unigram alignments. Banerjee et al. [10] then calculate a harmonic mean with most of the emphasis on recall, i.e., they calculate F_β with $\beta = 3$:

$$F_3 = \frac{10 \cdot P_{\text{align}} \cdot R_{\text{align}}}{R_{\text{align}} + 9 \cdot P_{\text{align}}} \tag{2.27}$$

In addition, METEOR introduces a penalty that is different from BLEU's brevity penalty. In a nutshell, METEOR favors sentences whose alignments can be grouped

into the fewest amount of chunks. A chunk is a group of unigrams that are adjacent to each other. For example, if the candidate sentence matches the reference translation one-to-one, there is only one chunk, thus, there is a lower penalty. The upper bound for the penalty is set to be 0.5, whereas the lower bound converges to zero, but is ultimately decided by the number of unigrams matched. METEOR's penalty is then defined as:

$$\text{Penalty} = 0.5 \cdot \left(\frac{\#\text{ chunks}}{\#\text{ unigrams matched}} \right)^3 \tag{2.28}$$

To conclude, we can compute the METEOR score by

$$\text{METEOR} = F_3 \cdot (1 - \text{Penalty}) \tag{2.29}$$

2.4.4 CIDEr

Vedantam et al. [138] proposed a metric for image description evaluation and called it CIDEr (*Consensus-based Image Description Evaluation*). They argue that metrics like BLEU and ROUGE seem to correlate weakly with human judgement [34, 80]. Their metric measures the *consensus*, that is, the similarity of a candidate sentence (i.e., a generated image caption) against a set of reference sentences that were written by humans. Furthermore, they show that CIDEr "has a high agreement with consensus as assessed by humans.". Basically, CIDEr is a metric based on *Term Frequency Inverse Document Frequency* (TF-IDF). In order to calculate a TF-IDF weighting over the whole corpus for an n-gram k, we first calculate the *term frequency* (TF) for a reference sentence $\mathbf{y}_{i,j}^S$ as

$$\text{TF}_k(\mathbf{y}_{i,j}^S) = \frac{\xi_k(\mathbf{y}_{i,j}^S)}{\sum_{l \in \Omega} \xi_l(\mathbf{y}_{i,j}^S)}, \tag{2.30}$$

whereas $\xi_k(\mathbf{y}_{i,j}^S)$ is the number of times a n-gram k occurs in a reference sentence $\mathbf{y}_{i,j}^S$. Note that i is the index of an image in the image corpus $\mathbf{I}$ and j denotes the j-th ground-truth sentence of the corresponding image $\mathbf{I}_i$. Furthermore, Ω is the vocabulary of all distinct n-grams (contrary to the vocabulary of words/unigrams which we denote by $\mathbf{V}$ in this work). Therefore, l is an n-gram from the vocabulary of all distinct n-grams Ω. Thus, the TF assigns higher weights to n-grams that frequently appear in a reference sentence for a given image. The *inverse document frequency* (IDF) then calculates to

$$\text{IDF}_k = \log \left(\frac{|\mathbf{I}|}{\sum_{I_p \in \mathbf{I}} \min(1, \sum_q \xi_k(\mathbf{y}_{p,q}^S))} \right), \tag{2.31}$$

where $\mathbf{I}$ is the set of all images in the dataset. Similar to the traditional definition of the IDF, we calculate the logarithm of the number of images (i.e., documents) in the dataset divided by the number of images for which an n-gram is contained in any of its ground-truth sentences. Intuitively, the IDF gives a higher weight to n-grams that are rarer and a lower weight to n-grams that are more frequent in the corpus of reference sentences. Therefore, less representative words are not seen as important as rare words which might be more noticeable in the visual domain.

With these two measures, we can define the TF-IDF weighting for each n-gram contained in a reference sentence $\mathbf{y}_{i,j}^S$ as

$$g_k(\mathbf{y}_{i,j}^S) = \mathrm{TF}_k(\mathbf{y}_{i,j}^S) \cdot \mathrm{IDF}_k. \tag{2.32}$$

Based on the TF-IDF weighting $g_k(\cdot)$, Vedantam et al. [138] define the CIDEr_n score that calculates the average cosine similarity between a generated candidate sentence $\mathbf{n}_i$ and all ground-truth sentences $\mathbf{y}_i^S$ for an image $\mathbf{I}_i$ from the dataset as

$$\mathrm{CIDEr}_n(\mathbf{n}_i, \mathbf{y}_i^S) = \frac{1}{\#\text{Reference Sentences for } I_i} \sum_j \frac{\mathbf{g}^n(\mathbf{n}_i) \bullet \mathbf{g}^n(\mathbf{y}_{i,j}^S)}{\|\mathbf{g}^n(\mathbf{n}_i)\| \cdot \left\|\mathbf{g}^n(\mathbf{y}_{i,j}^S)\right\|}. \tag{2.33}$$

Here, $\mathbf{g}^n(\cdot)$ is a vector consisting of the TF-IDF values g_k of all n-grams of length n. $\|\cdot\|$ denotes the magnitude of a vector and $\bullet$ is the dot-product.

Finally, we define the CIDEr score as the weighted average over all CIDEr_n metrics up to N

$$\mathrm{CIDEr}(\mathbf{n}_i, \mathbf{y}_i^S) = \sum_{n=1}^{N} w_n \cdot \mathrm{CIDEr}_n(\mathbf{n}_i, \mathbf{y}_i^S). \tag{2.34}$$

The weights are uniform with $w_n = \frac{1}{N}$ and CIDEr considers n-grams up to a length of $N = 4$.

2.4.5 MSCOCO Captions Evaluation Script

We evaluate our generated image and video descriptions extensively with the BLEU, ROUGUE, METEOR, and CIDEr metrics. This is the same for all tasks (with the exception of VQA, see Chapter 2.4.6) discussed in this thesis. We want to be (1) comparable to related works and (2) correctly evaluate our results. Therefore, we use an evaluation script commonly used in image captioning, video-to-text, and related tasks. This official evaluation script was released by Chen et al. [19] in conjunction with the release of the Microsoft COCO captions dataset. On their GitHub page [4] they release a Python package that allows to easily evaluate generated captions against multiple reference captions. This script automatically calculates the aforementioned metrics (BLEU-N, ROUGE-L, METEOR and CIDEr) for generated sentences (i.e., candidate sentences) against a set of ground-truth annotations.

2.4.6 VQA Accuracy

Previously, we discussed metrics that can automatically assess whether generated candidate sentences are "similar" to a set of reference sentences written by humans. In most parts of this thesis, we use these metrics to compare generated sentences of models against other models or related works. However, one part of this thesis marks an exception to this evaluation protocol. For the Visual Question Answering (VQA) task,

[4] `https://github.com/tylin/coco-caption`

the official evaluation method is the VQA accuracy, which is a variant of the traditional accuracy measure. In contrast to methods that explicitly generate image descriptions, the VQA task is modeled as a classification problem, i.e., the n most probable answers of a dataset are considered classes. This is an important fact, as most VQA models (except ours) do not generate answers word by word, but only choose one answer.

The VQA-v2 dataset ([45]; discussed in more detail in Chapter 5.3) is annotated with multiple questions per image and ten ground-truth answers for each question. In the first version of the dataset, Antol et al. [5] defined that a generated answer is considered 100 % accurate if at least three ground-truth answers match this answer. Otherwise, the accuracy of a single answer is reduced proportionally

$$\text{accuracy}_{\text{VQA}}(\text{answer}) = \min(\frac{\#\text{ humans that provided that answer}}{3}, 1). \tag{2.35}$$

Following this variant of the accuracy, an answer can also be partially correct, i.e., $\frac{1}{3}$ or $\frac{2}{3}$. With this definition, we can easily calculate the accuracy over the whole dataset by summing up the accuracies for each question and dividing by the number of questions asked.

Part II

Recurrent Language Generation Models for Image Description Generation

Overview

After the enormous success of deep convolutional neural networks (DCNNs) on the task of image classification [79, 125], other applications in the domain of computer vision quickly adopted DCNNs for extracting image features. In addition, recurrent neural networks (RNNs) became popular in the machine translation community [23, 129, 9]. Inspired by these big leaps, Vinyals et al. [139] combined these two network types into an *image translation network*, i.e., an architecture that allows generating natural language descriptions for a given input image.

In the second part of this thesis, we address the task of generating textual descriptions for images with recurrent language models, which exploit RNNs. RNNs, for example, can take visual features as input and can be trained to *maximize* the probability of a word *given* previous words and some conditioning. In our case, we condition the RNN on the visual feature input. We consider different aspects of preconditioning (traditional image captioning vs. visual question answering) and different types of RNNs (RNNs that consist of a single layer of RNN cells as well as a hierarchical RNN).

To begin with, however, we make an exception and investigate more traditional ways of generating language instead of employing RNNs: Templates. For a dataset consisting of colonoscopy videos without written annotations, we create textual descriptions by inserting detections into a template.

Second, we expand the vanilla image captioning approach by Vinyals et al. [139] to improve description generation for images that contain certain objects, i.e., instances of branded products. We introduce a new metric and in combination with multi-task learning, we find that our model describes these branded products in almost all instances.

Third, we visit the task of visual question answering that answers questions *about* images. One may think that this task is closely related to image captioning, however, related works in this area do not generate answers recurrently. Rather, they try to solve this problem with a classification approach. In contrast, we explore a model that generates answers in a recurrent way and is even capable of generating new answers.

Finally, we attempt to generate paragraphs of text for medical images, i.e., chest X-ray images. In contrast to the task of image captioning, we now need to generate multiple sentences that form a paragraph. Particularly, we conduct experiments on medical images with a hierarchical recurrent network and address data bias problems.

3 Template-Based Language Generation

This chapter falls out of line with the common theme of this part of our thesis. Particularly, in this chapter, we present a way of generating text for medical videos without descriptive ground-truth annotations. To put it in other words, as we lack annotations for the dataset used in this chapter, we will, nevertheless, try to produce plausible captions. Template-based approaches have shown success [36] before RNNs have been successfully applied to the task of language generation. Thus, we present such an approach for generating paragraphs that describe a video of a colonoscopy.

Furthermore, we explore the concept of transfer learning in the medical domain that proves to be effective and, therefore, can be applied to other tasks. For example, in Chapter 6, we initialize one of our hierarchical language generation models with pretrained weights from a chest X-Ray dataset.

This chapter is based on our Grand Challenge paper at ACM Multimedia 2019:

> **Automatic Disease Detection and Report Generation for Gastrointestinal Tract Examination** [53], Philipp Harzig, Moritz Einfalt and Rainer Lienhart, ACM International Conference on Multimedia 2019, Nice, France, October 2019.

3.1 Motivation

In this section, we present our method for the automatic identification of diseases and anatomical landmarks in the human digestive system. Gastroscopy and colonoscopy are real-time examinations of the human gastrointestinal (GI) tract by using digital endoscopes. Analyzing these videos is both time-consuming and needs a domain expert. Supporting these domain experts by machine learning techniques is one promising approach to reduce costs.

This chapter gives a technical overview of our first place submission to the 2019 ACM Multimedia Grand Challenge *BioMedia: Multimedia in Medicine* [63]. This challenge focuses on automatically detecting normal findings, abnormalities and anatomical landmarks in GI tract images composed from two different datasets. Pogorelov et al. [113] presented a multi-class dataset consisting of GI tract images, which has greater variability than other publicly available datasets [133, 13]. In addition, their dataset contains other classes than only focusing on polyps, i.e., classes related to polyp removal and three anatomical landmarks of the GI tract. Pogorelov et al. [112] also presented a dataset consisting of classes that allow to assess whether the bowel was cleansed sufficiently before colonoscopy.

3.2 Dataset

We use the Medico 2018 dataset provided by the challenge organizers. The dataset includes images for 16 classes (see first column of Table 3.3) and consists of a development split and a test split with 5,293 and 8,740 images, respectively. The classes represent anatomical landmarks, pathological findings or endoscopic procedures in the GI tract. The Medico 2018 dataset comprises parts of the Kvasir [113] and the Nerthus [112] dataset. The different classes are quite balanced with the exception of the *out-of-patient* and *instruments* class, which only account to 0.076% (4 images) and 0.680% (36 images) of the development split, respectively. For our models, we create a train and validation split from the development set with a ratio of 3 : 1.
In addition, we use the Kvasir-v2 dataset [113] for two of our models to improve detection accuracy. The Kvasir-v2 dataset provides 8000 additional images covering eight classes. For the report generation subtask, an additional dataset consisting of six videos has been provided by the BioMedia challenge organizers.

3.3 Method

We use two different CNNs to extract features from the input images: The MobileNet-V2 [124] and the DenseNet-121 [67]. In order to detect the class of the input image, we append two single fully-connected layers in parallel to the average-pooled feature map $\tilde{\mathcal{F}} \in \mathbb{R}^{1\times1\times d}$ of the CNN with d being the depth of the respective CNN's feature map. The development dataset is labeled as a single-classification problem, i.e., one correct class per image. Thus, we train the first fully-connected layer ρ_1 with a softmax (ϕ) cross-entropy loss function

$$L(I, \mathbf{y}^{\text{cls}}) = -\left[\log \phi(\rho_1(\tilde{\mathcal{F}}))\right] \cdot \mathbf{y}^{\text{cls}} \tag{3.1}$$

with $\mathbf{y}^{\text{cls}}$ being the one-hot encoded ground-truth label for training sample I while $\cdot$ denotes the dot-product. Nevertheless, a single image could still have multiple correct classifications, which we try to model with a second fully-connected layer. For instance, an image could depict a pathological finding and an instrument. We train the second fully-connected layer with a sigmoid cross-entropy loss, which allows us to output the likelihood of every class instead of predicting the most probable class only. However, predicting probabilities for independent classes does not impose a ranking on those classes. Hence, we always use the prediction of the fully-connected layer trained with the softmax cross-entropy loss first. We select the class-specific thresholds for the predictions trained with the sigmoid cross-entropy loss by using the thresholds that yield the highest F1-score on the validation set. If there are additional predictions after applying the class-specific thresholds, we output those according to the softmax ranking.

Training details While training, we resize every image to a size of 256×256 and extract a random crop of size 224×224. Additionally, for every image, we randomly

decide whether to rotate the image by 90°, to flip it horizontally, or to flip it vertically, which results in eight possible configurations for every input image.

We employ a two-stage training. In the first stage, we freeze the weights of the feature extractor CNN and only train each of the two separate fully-connected layers with the softmax loss and sigmoid loss, respectively. We use the Adam [75] optimizer with a learning rate $\eta = 0.001$ and train for up to 100 epochs with early stopping. In the second stage, we unfreeze the weights of the CNN and train with $\eta = 0.0001$ and an exponential learning rate schedule, which decays the learning rate by 0.5 every 20 epochs. In the fine-tuning stage, we also use an L2 regularization loss with a multiplier of $1 \cdot 10^{-4}$ for all convolution and fully-connected layer weights. We select our final model with an early stopping strategy, i.e., we choose the model with the best accuracy on the validation set.

3.4 Generating Language Without Ground-Truth Data

Medical report generation is a task with which is dealt in other areas as well. For instance, with the release of the Indiana University X-ray dataset [28], many works deal with connecting natural language and chest X-ray images. In particular, Jing et al. [70] use a hierarchical Long Short-Term Memory (HLSTM) [77] model, which allows to generate multiple sentences to form a paragraph that reflects a doctor's report. However, in contrast to our problem, this dataset [28] contains chest X-ray images combined with natural language reports of doctors. We, however, cannot use a language model like a RNN, trained on natural language paragraphs.

The provided dataset contains six videos which have a duration between 1 s–311 s (39–7783 frames) and from which we extract each frame with the FFmpeg library. Then, we use our trained model to predict the class for each frame. Given a frame index i, we smooth the predictions over 30 frames in the past and future using a simple algorithm: Given the future and past frames (i.e., frames with indices $\in [\max(0, i - 30), \ldots, i, \ldots \min(\text{num\_frames} - 1, i + 30]$), we determine the most probable prediction with majority voting on outlier-free class predictions for the past window and future window consisting of 31 frames each (each window includes the current frame index i): First, we remove outliers by normalizing each prediction[1] of both the past and future windows. We then remove all predictions from the windows, which have an absolute normalized score of ≥ 3. After we removed the outliers, we can determine the most probable prediction for each window of 31 frames. Given these two windows, we can determine the smoothed prediction of the current frame i based on following rules:

- If the prediction for frame i is the same as the most probable future prediction and different from the most probable past prediction, we change the classification result of our smoothed prediction to the prediction for frame i.

[1] We normalize all raw predictions $\mathbf{X} := \rho_1(\tilde{\mathcal{F}})$ by calculating $\frac{\mathbf{X} - \mathbb{E}[\mathbf{X}]}{\sigma(\mathbf{X})}$, i.e., we enforce zero mean and a standard deviation $\sigma(\cdot)$ for all predictions. $\mathbb{E}[\cdot]$ and $\sigma(\cdot)$ are computed for the current window of predictions.

- Otherwise, we keep the prediction of the last frame $i-1$ for the current frame i.

For each continuous sequence of frames for which the same classification result was predicted, we create a so-called video section. For example, we identified 19 consecutive video sections in Figure 3.3.

In addition, we use class activation maps (CAM) [160] to localize class-specific image regions, i.e., we infer the regions that contributed most to the classification outcome. The CAM can be seen as a probability distribution over the DenseNet-121's unpooled output feature map $\mathcal{F} \in \mathbb{R}^{8\times8\times d}$ (our input frames during video evaluation are resized to 256×256). We average these probabilities over each video section that we identified in the video. By using these averaged probability distributions, we then identify the area (one of top-left, top-right, bottom-left, bottom-right or center) that seems to be mostly responsible for the classification. We visualized this process in Figure 3.1.

Our final report for each video consists of three sections, (1) the main findings, (2) a brief summary and (3) a detailed summary. In the main findings, we provide the two most probable classifications over the whole video sequence together with their respective frequency of occurrence. Second, we provide a brief summary which explains all consecutive classifications of video sections in a chronological order. Finally, we give a detailed summary that describes every event within the video sequence with an exact time span, the classification result and the spatial location in which the event has been detected with the highest probability.

Table 3.1: Results of all our models on the detection (*detection-ver?*) and efficient detection (*speed-ver?*) subtasks. *detection-ver2* is our model with the best performance according to the MCC score. In Table 3.2, we list the individual times for the efficient detection subtasks. We report the true positives (TP), true negatives (TN), false positives (FP), false negatives (FN), precision, recall, specificity, F1 and MCC metrics. The specificity is also known as the true negative rate (TNR) and defined as $\frac{\mathrm{TN}}{\mathrm{TN+FP}}$.

model	TP	TN	FP	FN	precision	recall	specificity	F1	MCC
detection-ver1	8291	130609	446	446	**0.94442**	**0.90053**	0.99664	**0.91054**	0.94332
detection-ver2	**8419**	**130737**	**318**	**318**	0.89897	0.88458	**0.99752**	0.88471	**0.95974**
speed-ver1	8108	130426	629	629	0.86063	0.85300	0.99514	0.85142	0.92009
speed-ver2	8375	130693	362	362	0.89534	0.88129	0.99713	0.87993	0.95429
hardware	8108	130426	629	629	0.86063	0.85300	0.99514	0.85142	0.92009

3.5 Results

For our submitted models, we use two different training datasets. For the *-ver1* models, we use 75% (3969 images) of the provided Medico development dataset for training while keeping 25% for validating our model and selecting the best performing one. We extend the training split by the Kvasir-v2 [113] dataset ($3969 + 8000 = 11969$ images) for our *-ver2* models.

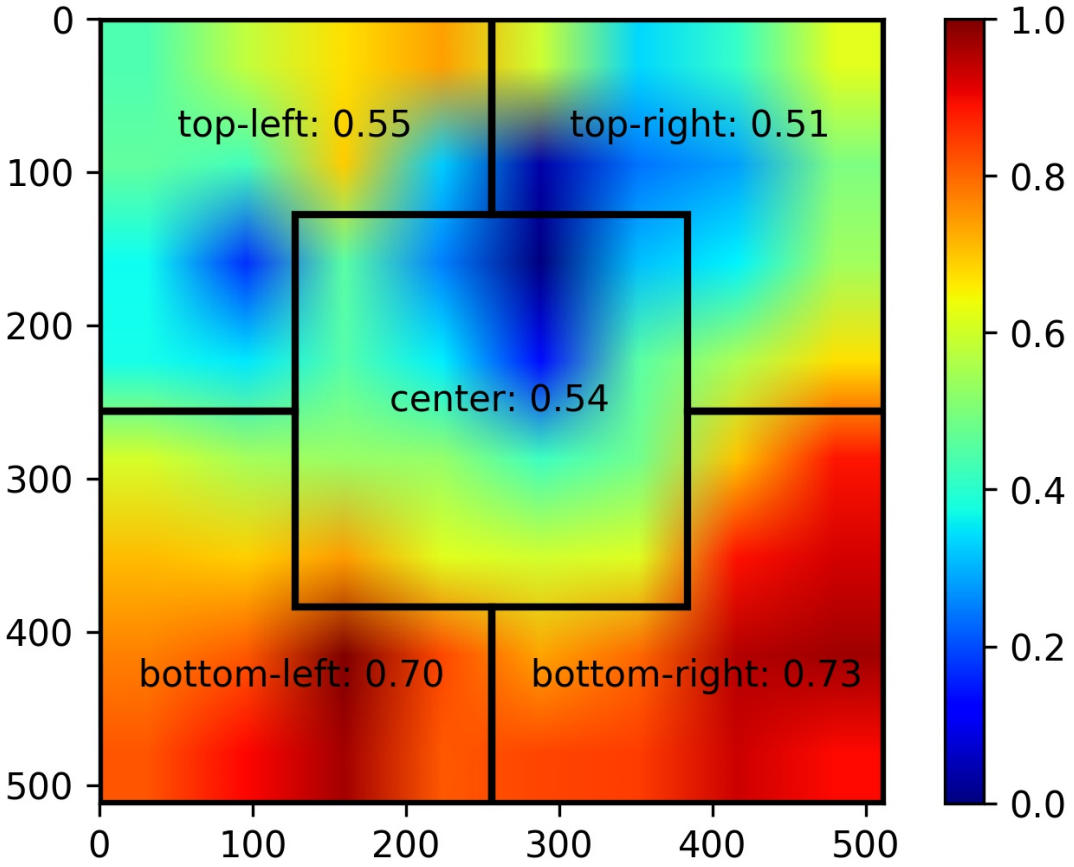


Figure 3.1: Average class activation map (CAM) for a video segment together with an overlay describing our five areas of interest. This CAM is of size 8×8 and upscaled to the original image size of 512×512. The area with the highest attention score is chosen as the most interesting area within the current video segment. Note that the region scores for each area of interest are the mean of activations over the respective area.

Table 3.2: Official timings for our models. All times t are measured in milliseconds (ms). fps stands for frames per second. The times for the model *hardware* were measured by the workshop organizers and the machine configuration is unknown. *speed-1* and *speed-2* times were measured on a single NVIDIA TITAN X (Pascal) GPU.

model	t_{avg}	t_{min}	t_{max}	fps_{avg}	fps_{min}	fps_{max}
speed-ver1	**0.3087**	0.1219	18.1812	**3238.87**	55.00	8204.27
speed-ver2	0.3100	0.1342	17.3601	3226.04	57.60	7453.23
hardware	0.7862	**0.1090**	**9.3824**	1271.98	**106.58**	**9175.40**

3.5.1 Detection Subtask

We submit two models for the detection subtask, namely *detection-ver1* and *detection-ver2*. We train *detection-ver1* with a MobileNetV2 with a width multiplier of 1.4. Even though the MobileNetV2 is designed as a mobile architecture, we find it to perform better than a DenseNet-121 and a DenseNet-201 when only using the train split of the Medico development dataset. For *detection-ver2*, we use the DenseNet-121 CNN that achieves better results on the validation split. We depict our results in Table 3.1 and see that *detection-ver2* performs better for almost every metric except precision, recall and the F1-score. As we can see in Table 3.3 this is caused by the underrepresented *out-of-patient* class, which does not get detected by the *detection-ver2* model.

Our models are able to output multiple detections, e.g., there might be cases where a finding and an instrument is detected. However, evaluation for multiple classes is constrained as there are no multi-class annotations as of now.

Table 3.3: Main metrics listed by class. We report the results of models *detection-ver1* and *detection-ver2* seperated by /. We report the true positives (TP), true negatives (TN), false positives (FP), false negatives (FN), precision, recall, specificity and F1 metrics. The specificity is also known as the true negative rate (TNR) and defined as $\frac{\mathrm{TN}}{\mathrm{TN+FP}}$.

	TP	TN	FP	FN	precision	recall	specificity	F1
blurry-nothing	**37**/35	**8698**/**8698**	**0**/2	**2**/**2**	**0.949**/0.946	**1.000**/0.946	**1.000**/**1.000**	**0.974**/0.946
colon-clear	**1065**/**1065**	**7660**/7634	**0**/**0**	**12**/38	**0.989**/0.966	**1.000**/**1.000**	**1.000**/**1.000**	**0.994**/0.982
dyed-lifted-polyps	520/**540**	8101/**8130**	36/**16**	80/**51**	0.867/**0.914**	0.935/**0.971**	0.996/**0.998**	0.900/**0.942**
dyed-resection-margins	535/**564**	8122/**8142**	29/**0**	51/**31**	0.913/**0.948**	0.949/**1.000**	0.996/**1.000**	0.930/**0.973**
esophagitis	462/**543**	8132/**8180**	94/**13**	49/**1**	0.904/**0.998**	0.831/**0.977**	0.989/**0.998**	0.866/**0.987**
instruments	**131**/125	**8464**/**8464**	**142**/148	**0**/**0**	**1.000**/**1.000**	**0.480**/0.458	**0.984**/0.983	**0.649**/0.628
normal-cecum	570/**582**	8136/**8149**	14/**2**	17/**4**	0.971/**0.993**	0.976/**0.997**	0.998/**1.000**	0.974/**0.995**
normal-pylorus	560/**561**	8171/**8176**	1/**0**	5/**0**	0.991/**1.000**	0.998/**1.000**	**1.000**/**1.000**	0.995/**1.000**
normal-z-line	512/**562**	8082/**8162**	51/**1**	92/**12**	0.848/**0.979**	0.909/**0.998**	0.994/**1.000**	0.877/**0.989**
out-of-patient	**1**/0	**8735**/**8735**	**1**/2	**0**/**0**	**1.000**/0.000	**0.500**/0.000	**1.000**/**1.000**	**0.667**/0.000
polyps	365/**373**	8261/**8295**	9/**1**	102/**68**	0.782/**0.846**	0.976/**0.997**	0.999/**1.000**	0.868/**0.915**
retroflex-rectum	**184**/179	8535/**8544**	**8**/13	10/**1**	0.948/**0.994**	**0.958**/0.932	**0.999**/0.998	0.953/**0.962**
retroflex-stomach	**394**/**394**	**8338**/8336	**3**/**3**	**2**/4	**0.995**/0.990	**0.992**/**0.992**	**1.000**/**1.000**	**0.994**/0.991
stool-inclusions	**494**/468	**8221**/8181	**12**/38	**10**/50	**0.980**/0.903	**0.976**/0.925	**0.999**/0.995	**0.978**/0.914
stool-plenty	**1956**/1886	6771/**6772**	**9**/79	1/**0**	0.999/**1.000**	**0.995**/0.960	**0.999**/0.988	**0.997**/0.979
ulcerative-colitis	505/**542**	**8182**/8139	37/**0**	**13**/56	**0.975**/0.906	0.932/**1.000**	0.996/**1.000**	**0.953**/0.951

3.5.2 Efficient Detection Subtask

Similar to the detection subtask, we submitted two models for the efficient detection subtask, which make use of the two different dataset variants, which we proposed in Section 3.5. We use a MobileNetV2 with a width multiplier of 1.0 for our efficient detection models, which allows for faster detection times while sacrificing a bit of accuracy. However, the Matthews correlation coefficient[2] (MCC) score for the model *speed-ver2* is almost on par with the *detection-ver2* model. In contrast, when using the smaller

[2]The Matthews correlation coefficient is a specific application of the Pearson correlation coefficient to a confusion matrix.

Main findings:
==========
The video mostly shows esophagitis (70.85%), followed by blurry-nothing (14.88%).
Brief summary:
==========
The video sequence shows the following events in this chronological order: colon-clear, blurry-nothing, esophagitis, normal-z-line, esophagitis.
Detailed summary:
==========

FROM - TO	Description of current time period within the video.
00:00-00:00	An inflammation of the esophagus is visible mostly in the center (Esophagitis).
00:00-00:01	A clear colon can be seen mostly in the center.
00:01-00:02	An inflammation of the esophagus is visible mostly in the center (Esophagitis).
00:02-00:04	A clear colon can be seen mostly in the center.
00:04-00:09	The image is blurry and it is hard to identify what currently can be seen.
00:09-00:09	Instruments are visible within the current section of the video mostly in the bottom-left.
00:09-00:09	The image is blurry and it is hard to identify what currently can be seen.
00:09-00:10	retroflex-rectum mostly in the top-left.
00:10-00:11	The image is blurry and it is hard to identify what currently can be seen.
00:11-00:15	An inflammation of the esophagus is visible mostly in the center (Esophagitis).
00:15-00:15	A normal z-line can be seen mostly in the top-right.
00:15-00:31	An inflammation of the esophagus is visible mostly in the center (Esophagitis).
00:31-00:32	The image is blurry and it is hard to identify what currently can be seen.
00:32-00:44	An inflammation of the esophagus is visible mostly in the center (Esophagitis).
00:44-00:46	A normal z-line can be seen mostly in the center.
00:46-00:47	An inflammation of the esophagus is visible mostly in the top-right (Esophagitis).
00:47-00:48	Dyed resescected margins can be seen mostly in the top-left.
00:48-00:49	Instruments are visible within the current section of the video mostly in the bottom-left.
00:49-00:51	An inflammation of the esophagus is visible mostly in the center (Esophagitis).

Figure 3.2: Generated report for 3e3a7ac0-4244-46cc-89a1-44ce84dd1ccf.avi. This report matches with the smoothed prediction (bottom bar) of Figure 3.3.

dataset, i.e., only the train split of the Medico development dataset, the performance decreases by over two percent when using the MobileNetV2 with the smaller width multiplier. In Table 3.2, we list the times our models take to classify a single image. *speed-ver1* and *speed-ver2* are our submitted models with an average detection time for a single image of 0.3087 ms and 0.3100 ms, respectively. We measured those times on a dual-CPU workstation with 48 threads and a single NVIDIA TITAN X (Pascal) GPU. The *hardware* model is the same as *speed-ver1*, but was submitted to the organizers of the challenge as a Docker image to be comparable with other submissions in terms of hardware configuration. In this disclosed hardware setup, the average processing time per image takes longer with 0.7862 ms, but the minimal processing time for an image is shorter with 0.1090 ms compared to 0.1219 ms for model *speed-ver1*.

3.5.3 Report Generation

For generating reports, we used the *detection-ver1* model from Section 3.5.1. As we already described in Section 3.4, we extracted all frames for each given video and predicted their most probable class label. We depict such a classification result for one video in Figure 3.3, where the top bar shows the raw classifications for every frame. The bottom bar shows the predictions which were smoothed over a window of 30 frames in the future and past. In the figure, we also depict one image representative for each extracted video section. Together with a text template library and identifying the region that mostly contributed to the classification outcome, we generated a detailed summary of each video. We depict one such report combined with main findings and a brief summary in Figure 3.2.

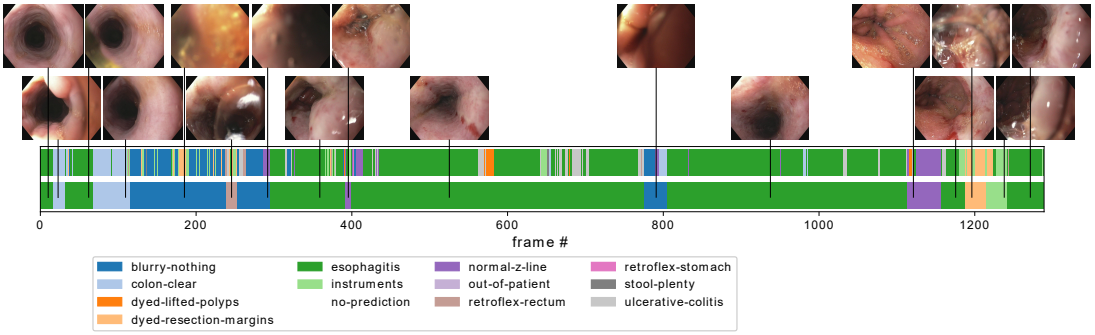


Figure 3.3: Resulting classifications on a per frame basis for 3e3a7ac0-4244-46cc-89a1-44ce84dd1ccf.avi. The upper bar shows the raw classifications for every frame within the video. The bottom bar shows the classification smoothed over a time period of 30 frames. We also depict one example frame for each smoothed section within the video.

Note that the reports were not evaluated by the challenge organizers because no ground-truth annotations were available. Also, humans did not rate the reports in the challenge. For the report generation subtask, the challenge organizers did not impose rules or define the desired form of the textual reports. Rather, we did define the form of the textual report generated in Figure 3.2.

3.6 Summary

In this chapter, we presented an architecture using a DCNN to predict abnormalities and diseases from GI tract images. To improve our classifications, we employed augmentation and examined different CNN feature extractors to find models that perform best given two constraints: Inferring the best possible predictions and to return the predictions as fast as possible while not sacrificing too much detection accuracy. In addition, we expanded our architecture to automatically generate a detailed report for a given video of a gastroscopy or colonoscopy. This report also describes in which spatial location of the video the findings were observed.

4 Automatic Description of Images with Branded Products in Natural Language

After we introduced a model that is able to generate a report from a video by inserting detections into text templates, we will now focus on the task of generating descriptions from scratch. First, we limit the number of sentences to one, i.e., we want to generate a single sentence for a given image. This task is often referred to as image captioning. Second, we want to learn a model that is able to generate sentences from scratch in an end-to-end model. That is, similar to other tasks working with DCNNs, we do not want to implement a model that acts on predefined rules (as we did in Chapter 3), but learns to generate language on its own.

This chapter is based on the following two publications:

> **Multimodal Image Captioning for Marketing Analysis** [50], Philipp Harzig, Stephan Brehm, Rainer Lienhart, Carolin Kaiser and René Schallner, IEEE Conference on Multimedia Information Processing and Retrieval 2018, Miami, FL, April 2018.
>
> **Image Captioning with Clause-Focused Metrics in a Multi-Modal Setting for Marketing** [57], Philipp Harzig, Dan Zecha, Rainer Lienhart, Carolin Kaiser and René Schallner, IEEE Conference on Multimedia Information Processing and Retrieval 2019, San José, CA, March 2019.

Automatically generating descriptive captions for images is a well-researched area in computer vision. However, the focus lies on describing everyday situations rather than specific situations such as persons interacting with branded products. Furthermore, existing evaluation approaches focus on measuring the similarity between two sentences disregarding fine-grained semantics of the captions. In our setting of images depicting persons interacting with branded products, the subject, predicate, object, and the name of the branded product are important evaluation criteria of the generated captions. Generating image captions with these constraints is a new challenge, which we tackle in this chapter. By simultaneously predicting integer-valued ratings that describe attributes of the human-product interaction, we optimize a deep neural network architecture in a multi-task learning setting, which considerably improves the caption quality. Furthermore, we introduce a novel metric that allows us to assess whether the generated captions meet our requirements (i.e., subject, predicate, object, and product name) and describe a series of experiments on caption quality and how to address annotator disagreements for the image ratings with an approach called soft-targets. We also show that our novel

clause-focused metrics are applicable to other image captioning datasets, such as the popular MSCOCO dataset.

4.1 Related Work

Traditional Approaches for Image Description Generation The machine-learning community has put quite a lot of research in ranking descriptions for given images [44, 65]. These methods usually embed images and descriptions into the same vector space, such that matching captions lie close to their corresponding image. Socher et al. [127] use convolutional neural networks and dependency trees to embed images and sentences into the same space. Early attempts which try to generate captions from images have not incorporated RNN networks. Farhadi et al. [36] try to infer triplets (object, action, scene) from image features which are then converted into text. Li et al. [85] also use image features to compose sentences from scratch by using n-grams collected from a web-scale text corpus.

Recurrent Image Captioning The generation of image descriptions by recurrent neural networks is part of recent research. Vinyals et al. [139, 140] use this approach in form of an LSTM network. Furthermore, Karpathy et al. use a bidirectional RNN [73] for generating captions. In contrast to Vinyals et al., Karpathy et al. gradually extend their technique to not only describe a whole image, but parts of the image which they call dense captioning [71]. For the task of activity recognition, image captioning, and video description, Donahue et al. [32] provide another example of successfully applying recurrent sequencing models. Kiros et al. [76] also use an encoder-decoder approach, where they embed images and sentences in the same common space and use an LSTM for encoding the sentences. Hendricks et al. [61] go one step further and describe novel concepts not contained in the training images by incorporating information from image datasets and text corpora independently from each other.

Soft-Targets for Classification Using soft-targets in a classification setting has also been explored by Teney et al. [135]. In a different task called visual question answering, they face a classification problem, where multiple annotators gave an answer to a question regarding the content of an input image. Several thousand unique answers given by the annotators constitute the possible classes. Obviously, annotators do not always agree on the same answer, hence, this classification problem was modeled using soft-targets (probability distributions of each answer) given the relative frequency of given answers.

Multi-Task Learning Multi-task learning (MTL) is a long-studied domain in machine learning (cf. Caruana [16]). Luong et al. [98] work on MTL with focus on sequence-to-sequence learning. They introduce three different MTL settings for sequence-to-sequence models, i.e., the one-to-many setting, the many-to-one setting, and the many-to-many setting using multiple decoders and encoders. Our work falls into the category of one-to-many MTL setting, i.e., we use the same encoder (CNN) and multiple decoders to

compute three image ratings and a caption from image features. Dong et al. [33] introduce another work that employs multi-task learning in a natural language processing setting. In particular, they present a model that translates a source sentence into multiple languages.

4.2 Show and Tell Model

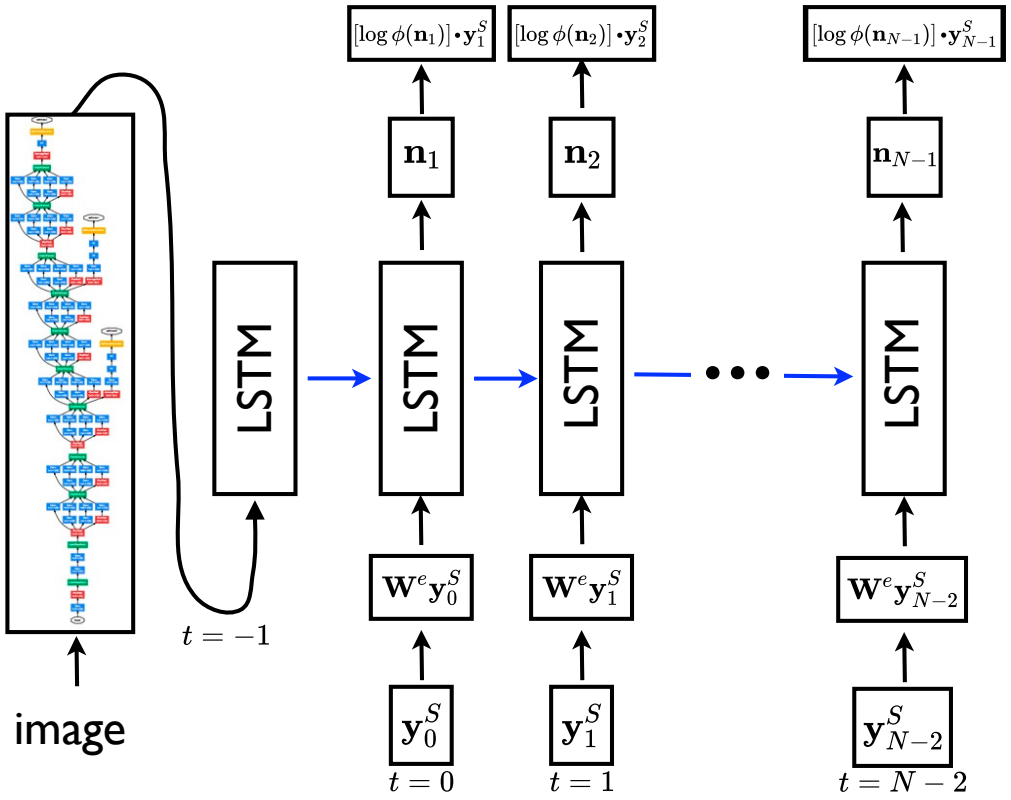


Figure 4.1: The architecture of the original Show and Tell model. Image taken from [139] and adapted to our nomenclature. At iteration step $t = -1$, we precondition the LSTM cell with the embedded image features from the encoder DCNN. Then, at iteration step $t = 0$, we feed the LSTM with $\mathbf{y}_0^S$, which is the special start-of-sequence token `<S>`. Given this special token and the image preconditioning, we want to predict the first real word of the sentence (see output $\mathbf{n}_1$ for input token $\mathbf{y}_0^S$). A softmax cross-entropy loss (depicted on the top) then tries to maximize the probability of the output $\mathbf{n}_1$ given the ground-truth token $\mathbf{y}_1^S$. The same LSTM cell is repeatedly fed with the embedded sentence tokens $\mathbf{W}^e\mathbf{y}_t^S$ until we feed the second to last token $\mathbf{y}_{N-2}^S$. Given this token, we want to predict the last token of the sentence, which is always the special end-of-sequence token `</S>`.

Our model is based on the popular Show and Tell model by Vinyals et al. [139]. We base the following explanation on their work. Similar to their architecture, our model follows an encoder-decoder design pattern. For image captioning, this architecture was

inspired by prior works in machine translation [23, 129, 9], i.e., translating text from one language into another language. On an abstract level, the encoder extracts information from one modality (e.g., image or video) into an abstract feature vector, while the decoder decodes this information back into the desired target modality (e.g., language). We visualize an overview of this architecture in Figure 4.1.

Encoder In the case of image captioning, Vinyals et al. replace the LSTM encoder network with a traditional CNN network, i.e., the Inception-v3 CNN. We employ a pretrained Inception-v3 CNN. Like other CNNs, this network does perform well on the task of image classification, i.e., it produces a rich image representation in order to accurately classify images. Thus, we remove the fully-connected classification layer because we want to utilize this representation. As a result, this encoder CNN encodes the contents of an image $I \in \mathbb{R}^{299\times299\times3}$ into a feature map $\mathcal{F} \in \mathbb{R}^{8\times8\times2048}$ which contains semantic information about the depicted scene. We average-pool the feature map over the spatial dimensions, i.e., we average over the first and second dimension. This yields a feature vector of size 2048 which we embed into a space with dimension 512 with a fully-connected layer. This so-called image embedding maps image features into a joint multi-modal embedding space. This embedding space is shared with the language part of the model (see the following decoder paragraph).

Decoder Prior works in machine translation use RNN networks to model the encoding and decoding of natural language. RNNs are an obvious choice for modeling sequences of tokens (i.e., natural language) of varying length. For the task of image captioning, we want to generate a short description (i.e., caption) that matches the contents of a given image I. For every image there are one or more ground-truth sentences

$$\mathbf{y}^S = \left[\mathbf{y}_0^S, \ldots, \mathbf{y}_{N-1}^S\right], \tag{4.1}$$

where N is the variable number of words within the sentence. Each word $\mathbf{y}_i^S$ is represented by a one-hot vector encoding, i.e., we construct a vector with zeroes except for a one at the index of the word within the vocabulary. We embed each word with a word embedding matrix $\mathbf{W}^e$ into the same joint multi-modal embedding space as our image. Note however, that the word embedding and image embedding do not share weights as their respective source spaces are substantially different.

In particular, Vinyals et al. [139] model the decoder with an LSTM network [64]. The LSTM network yields a hidden state/output for every embedded word. We forward this resulting hidden state through a fully-connected layer which yields a vector $\mathbf{n}_{t+1}$ for the ground-truth word $\mathbf{y}_t^S$ at iteration step t. This fully-connected layer has $|\mathbf{V}|$ neurons, i.e., it produces a score for every word within the vocabulary $\mathbf{V}$. Therefore, the vectors $\mathbf{n}$ are of size $|\mathbf{V}|$. Note that the image embedding from the encoder network serves as an initialization vector for the LSTM cell. As the embedded image features have the same dimensions as embedded word vectors, we can call the LSTM cell once with the embedded image features at iteration step $t = -1$ before feeding the ground-truth sentence or before generating the caption word by word.

Preprocessing of Inputs Before training the model, we need to modify our ground-truth sentence so we can both optimize our model to generate the most probable sentence and later predict a sentence for an unseen image. To do so, we prepend a special start-of-sequence token `<S>` and append a special end-of-sequence token `</S>` (see also Section 2.2.2). Note that as a consequence the sentence length N is now increased by 2. Thus, when optimizing the model for an image-caption pair, $\mathbf{y}_0^S$ is *always* the start-of-sequence token and $\mathbf{y}_{N-1}^S$ is *always* the end-of-sequence token. This has the advantage that we can feed the model with an image and the start-of-sequence token in order to predict a sentence word by word in the inference case (see subsection predicting captions).

Training the model During training, we feed the ground-truth words at every word index of the sentence. We then try to maximize the probability of the correct description given an image I:

$$\theta^* = \arg\max_\theta \sum_{(I,\mathbf{y}^S)} \log p(\mathbf{y}^S|I;\theta), \tag{4.2}$$

where θ are the parameters of the model. As our sentence has an arbitrary length N, we can apply the chain rule to model the joint probability

$$\log p(\mathbf{y}^S|I) = \sum_{t=0}^{N-1} \log p(\mathbf{y}_t^S|I, \mathbf{y}_0^S, \ldots, \mathbf{y}_{t-1}^S) \tag{4.3}$$

for our ground-truth sentence $\mathbf{y}^S$. Note that we dropped θ for convenience. Because, we always prepend the special start-of-sequence token to our ground-truth sentences at index $t = 0$, the probability of $p(\mathbf{y}_0^S|I)$ is always 1. Thus, the joint probability does not change regardless of whether we set the starting point of t to 0 or 1 in the sum in Equation 4.3. We already explored a way to implement this joint probability for our model, namely an RNN network with LSTM cells.

Before predicting captions for unseen images, we need to optimize our model for each image-caption pair. As we already outlined, we first initialize the decoder's LSTM cell with the embedded image from the encoder DCNN at iteration step $t = -1$. Then, at iteration step $t = 0$, we feed the LSTM cell the embedded start-of-sequence token `<S>`, which outputs a hidden state that is fed through a fully-connected layer which outputs $\mathbf{n}_1$. In the next step, we feed the LSTM cell the embedded ground-truth token $\mathbf{W}^e\mathbf{y}_1^S$, which then outputs $\mathbf{n}_2$ after the fully-connected layer. We repeat this until iteration step $t = N - 2$ that outputs $\mathbf{n}_{N-1}$. Therefore, we optimize our model's outputs for iteration steps $t \in \{0, \ldots, N-2\}$ with a softmax cross-entropy loss

$$L(I, \mathbf{y}^S) = -\sum_{t=0}^{N-2} \left([\log \phi(\mathbf{n}_{t+1})] \cdot \mathbf{y}_{t+1}^S \right). \tag{4.4}$$

where $\mathbf{n}_{t+1}$ is the output of the LSTM for the input $\mathbf{W}^e\mathbf{y}_t^S$ at iteration step t and $\phi(\cdot)$ is the softmax activation function. More specifically, $\mathbf{n}_{t+1}$ is the output of the fully-connected layer that *follows* the LSTM cell. $\mathbf{y}_{t+1}^S$ is the corresponding ground-truth

word one-hot vector, $\cdot$ is the dot-product, and $\mathbf{W}^e$ is a word embedding matrix. We minimize Equation 4.4 w.r.t. all parameters of the LSTM, the image embedding layer, and the word embedding matrix $\mathbf{W}^e$.

Predicting captions When generating descriptions for unknown images during inference, we employ a greedy sampling strategy. First, we forward an image through the encoder CNN network and the subsequent image embedding layer to compute an image embedding. We feed the decoder network the resulting feature vector at iteration step $t = -1$, i.e., before unrolling the LSTM cell. Then, we feed the LSTM cell the first word token at iteration step $t = 0$. The first token is the same for all sentences, i.e., it is the special `<S>` (start-of-sequence) token. Following this preconditioning, we can predict the most probable word. For the next iteration step we feed the previously predicted word and sample another word. We repeat this greedy sampling method until the most probable output prediction is the `</S>` (end-of-sequence) token.

However, in our setting greedily sampling captions has a severe disadvantage. If we compare the behaviour of the model in the prediction stage, we notice that it is substantially different from the training stage: We feed the network predicted words at subsequent iteration steps during inference unlike the ground-truth words during training. That is, during inference the network sees sentences which are built differently than the ground-truth sentence. This training strategy for RNNs is called *teacher forcing* [149]. Teacher forcing uses the ground-truth value at iteration step t rather than the output generated by the decoder at iteration step $t-1$. Teacher forcing allows us to train our model in an effective and fast way, however, it can lead to a limited model, i.e., the model does not necessarily know what to make of a sequence of previously predicted words and behave unexpectedly during inference. There have been some proposals like *curriculum learning* [12] that try to circumvent these limitations.

A second disadvantage of the greedy sampling algorithm is that sentence candidates which could potentially yield a better sentence both in terms of metrics and overall probability are eliminated early on during greedy sampling. For example, the most probable first token of a sentence could be *A*, whereas *The* is less likely. However, the sentence could ultimately be of better quality or yield a higher score if we generated subsequent tokens for the *The* token. Still, as we argued in Section 2.2.7, exploring all possible sentences is computationally infeasible. Though, we discussed a technique called beam search which allows us to explore a smaller space of possible sentences. Beam search then yields a set of probable sentences instead of only generating the sentence with the highest individual word probabilities. For some of our final models, we utilize beam search in order to generate more diverse sentences.

4.3 Motivation

Marketing companies have decades of experience in analyzing text fragments in order to investigate sentiment and consumer-brand-relationships. However, large collections of images in social media rarely come with a description connected to them. By transcribing

Table 4.1: Distribution for the train and test split of the GfK-Captions dataset. The dataset contains different modalities (images, captions and image ratings). For reference, we list statistics for the MSCOCO dataset [91] at the bottom. Note that we created a custom split (i.e., we split the train and validation part into train and test) for the MSCOCO dataset.

Dataset	Modality	# Train	# Test	# Total
GfK-Captions	Images	9469	1060	10,529
GfK-Captions	Captions	47,345	5300	52,645
GfK-Captions	Image ratings (×3)	12,175	1415	13,590
MSCOCO [91]	Images	117,211	4051	121,262
MSCOCO [91]	Captions	586,368	20,267	606,635

images into text, we want to enable marketing companies to rely on text analysis tools that capture two important aspects in descriptions: (1) Occurrence of words that identify a certain brand and (2) attributes and verbs that allow to detect sentiment and affection to a brand and other relevant properties depicted by an image.

Encoder-decoder networks, like the one presented by Vinyals et al. [139] are promising, when generating captions for input images. Inspired by their work, we build a model with a multi-task objective which simultaneously predicts image ratings. Image ratings describe three different attributes of interactions between humans and branded products in our case. In other words, our model processes three modalities: images, textual descriptions, and image ratings.

In particular, we look at images that contain an object which is related to a brand by depicting a logo of this brand. Such an image, for instance, could depict persons drinking out of a Coca-Cola bottle. Figure 4.3 shows one example image from our test set.

It is of particular interest to us to correctly identify the brand contained in the image, but state-of-the-art models like [139, 140] tend to produce rather generalized descriptions, i.e., the model could just leave out the brand, because it has generalized from pictures of persons holding bottles from different brands. For example, the caption generated by [139, 140] for the image in Figure 4.3 is *"a close up of a person holding a cell phone"*. In contrast, we want our model to correctly mention the name of the brand contained in the image within the sentence.

Our second goal is to simultaneously predict attributes that describe the involvement of the human with the brand, whether the branded product appears in a positive or negative context, and whether the interaction is functional or emotional. For example, in a functional interaction a person eats a product, while an emotional interaction might depict people taking selfies with branded products. We encode these attributes by ratings, which are integer values from 0 to 4 encoding how much the rating attribute holds. Our goal is to simultaneously predict ratings and, thus, improve the overall quality of the generated sentences in a joint multi-task optimization. This leads to a unique problem we aim to address in this chapter: We want our model to create reasonable captions,

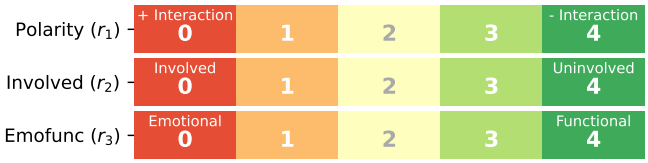


Figure 4.2: A visualization of our three different kinds of image ratings. Each rating can have an integer value between 0 and 4. Annotators created image ratings for a subset of images (2718/10,529) for the GfK-Captions dataset.

while at least mentioning the brand of the logo contained in the image. In addition to determining the standard scores (BLEU [108], METEOR [10], CIDEr [138]), we measure the accuracy of correctly generated brand names in our sentences, i.e., we check if the generated captions contain a certain class. With this metric, we can evaluate our model with a focus on correct classification within the generated captions. Furthermore, we propose a constituent sensitive metric to assess the quality of the subject, predicate, and object of generated sentences. These metrics can also be applied to popular image captioning datasets like MSCOCO [91].

4.4 GfK-Captions Dataset

We use a real-world dataset, which was created by a German non-profit market research association (GfK Verein[1]). The GfK-Captions dataset consists of images that depict scenarios of persons interacting with branded products, e.g., a man holding a can with a Coca-Cola logo on it. Altogether, this dataset covers 26 different brand classes and consists of 10,529 images.

In Table 4.1 we depict the number of examples in the train set, test set and total examples for our dataset. Our dataset covers three different modalities. The first row summarizes the images contained in our dataset, the second row reports the total number of captions associated with the images. Five annotators created a caption for each image. For a subset of all images (2718) five annotators created image ratings with five possible values $(0-4)$. There are three kinds of image ratings, which we visualize in Figure 4.2. These image ratings express the following properties of human product interactions:

- The *Polarity* (r_1) rating states whether the person interacts with the branded product in a positive (0) or negative (4) way.
- *Involved* (r_2) says if the person in the image is involved (0) with the branded product or uninvolved (4).
- *Emofunc* (r_3) describes if there is an emotional (0) or a functional (4) interaction with the branded product.

[1]We would like to thank Carolin Kaiser, René Schallner, Holger Dietrich, Raimund Wildner and Andreas Neus for the great collaboration and creation of the dataset.

Table 4.2: Dataset statistics for the GfK-Captions dataset. The dataset includes 26 brand classes. However, the dataset is distributed unequally among classes, e.g., cocacola and nutella together make up nearly 40 % of all images while meggle only makes up 0.21 %. Furthermore, there are multiple underrepresented classes which each make up 0.95 % of the dataset.

class	train	test	total	% of ds
bacardi	309	35	344	3.27 %
barilla	90	10	100	0.95 %
becks	524	59	583	5.54 %
campari	90	10	100	0.95 %
cocacola	1879	209	2088	19.83 %
dallmayr	90	10	100	0.95 %
dove	90	10	100	0.95 %
gerolsteiner	90	10	100	0.95 %
granini	90	10	100	0.95 %
haribo	598	67	665	6.32 %
heinz	89	10	99	0.94 %
jacobs	90	10	100	0.95 %
kinderriegel	168	19	187	1.78 %
krombacher	306	34	340	3.23 %
rittersport	369	42	411	3.90 %
loreal	489	55	544	5.17 %
maggi	90	10	100	0.95 %
meggle	19	3	22	0.21 %
milka	860	96	956	9.08 %
nimm2	90	10	100	0.95 %
nivea	216	25	241	2.29 %
nutella	1888	210	2098	19.93 %
pantene	90	10	100	0.95 %
pepsi	617	69	686	6.52 %
syoss	90	10	100	0.95 %
volvic	148	17	165	1.57 %
total	9469	1060	10,529	100.00 %

Because of the small number of images in our dataset, we split it into a training and test dataset with a ratio of 9 to 1. We use 10-fold cross-validation to select the best performing model due to the small number of training images. Figure 4.3 depicts a sample image from this dataset. Two problems arise with our dataset. First, images are distributed unequally among the different classes. In Table 4.2, we outline the 26 brand classes of our dataset. We see that classes like *cocacola* and *nutella* make up a big portion of all images within the dataset while half of the classes (13) only make up 11.60 %.

In addition, we have extremely few training images compared to other datasets like

Figure 4.3: One image from the test partition of the GfK-Captions dataset. We see that this image represents the logo class *cocacola*.

MSCOCO [91] (see Table 4.1). Hence, training a Show and Tell model from scratch is not the best idea as MSCOCO contains more than 10× the images and captions than our dataset.

As a consequence, we create two different variants of our dataset. First, we only use images, image ratings and captions from the GfK-Captions dataset and call this variant *LogosSimple*. For the extended variant of our dataset, we append the MSCOCO dataset to *LogosSimple*. To compensate for the fewer examples in *LogosSimple* compared to the MSCOCO dataset (see Table 4.1), we multiply *LogosSimple* 8 times. We call this variant *LogosExtended*.

Still, generating high-quality captions for this special application of image captioning is no easy job. Thus, in the following, we propose solutions in order to generate high-quality captions for images that contain person-product interactions.

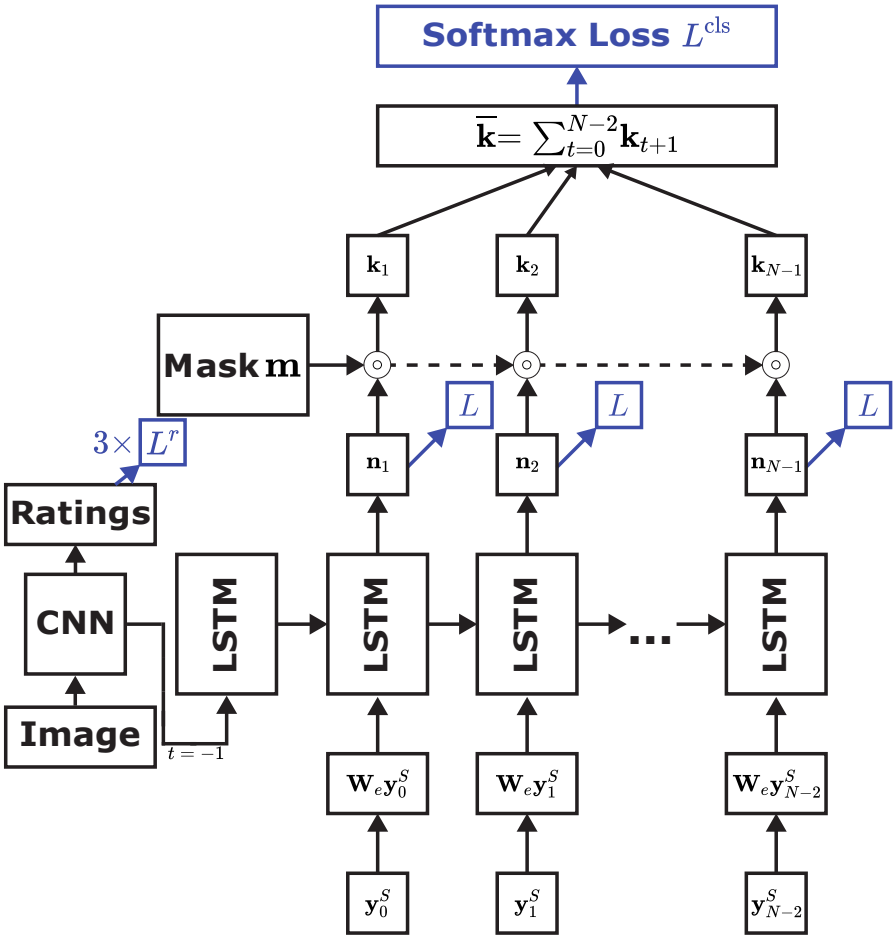


Figure 4.4: Our modified Show and Tell model that introduces the classification-aware loss function L^{cls} for describing brand names within image captions. The classification-aware loss is added on top of the standard LSTM decoder network. Furthermore, we add three losses L^r for the image ratings.

4.5 Describing Brand Names through Multi-Task Training

As our baseline architecture, we choose the Show and Tell [139] model by Vinyals et al. for generating short descriptions that match the images with branded products. In order to fully adapt to the different requirements imposed by the marketing environment, we extend the baseline architecture with additional objectives to allow for more appropriate captions by use of multi-task training. We depict our modified model in Figure 4.4.

4.5.1 Classification-Aware Loss

From a marketing point of view, it is crucial to correctly identify and describe the brand of a product with which a human interacts. The vanilla Show and Tell model is aimed at generating descriptions of everyday activities and interaction. All images of the GfK-

Captions dataset depict everyday activities. However, these interactions are between a person and branded products. The Show and Tell model is optimized to create captions that are similar to sentences created by humans and does not necessarily focus on naming the exact brand name of an object, i.e., it tends to confuse different brands. For example, when describing a person holding a can of soda the model might not care if it is a can, a can of *Coca-Cola* or a can of *Pepsi*.

We need to tell our model, whether or not a generated sentence contains the correct brand. Therefore, we introduce an additional objective which encourages our model to include the correct brand name within the generated sentence. On the contrary, this loss punishes the model if the correct brand name is not contained within a generated sentence. Thus, we want to urge the LSTM decoder to sample at least one word that is of the desired brand for every input image, i.e., each caption should include one of the 26 brand classes (see Section 4.4) contained in the dataset. Each of our brand names is part of the vocabulary and we call these words *classword*s. That is, a *classword* is a word in the vocabulary that clearly identifies a main brand logo (e.g., the word 'cocacola' is a *classword*.). We construct a constant binary mask vector $\mathbf{m}$ of size $|\mathbf{V}|$ which has a one at every index of a *classword*. To put it another way, every element $\mathbf{m}_i$ of the mask vector $\mathbf{m}$ is given by

$$\mathbf{m}_i = \begin{cases} 1, & \text{if } \mathbf{V}_i \text{ is a classword} \\ 0, & \text{otherwise.} \end{cases} \tag{4.5}$$

Next, we multiply the projected outputs $\mathbf{n}_{t+1}$ for every iteration step t with the mask vector. As result, we get a filtered view

$$\mathbf{k}_{t+1} = \mathbf{n}_{t+1} \circ \mathbf{m}, \tag{4.6}$$

on the scores for the *classwords* only. A *classword* should only occur once in any sentence. However, it could be generated at any index within this sentence. Therefore, we sum over all iteration steps of the filtered view ($\overline{\mathbf{k}}$)

$$\overline{\mathbf{k}} = \sum_{t=0}^{N-2} \mathbf{k}_{t+1}. \tag{4.7}$$

By applying the softmax (ϕ) function on $\overline{\mathbf{k}}$

$$\widehat{\overline{\mathbf{k}}} = \phi(\overline{\mathbf{k}}), \tag{4.8}$$

we get a probability distribution over the *classwords* for the predicted sentence. Note that $\overline{\mathbf{k}}$ and $\widehat{\overline{\mathbf{k}}}$ still have the same size as the vocabulary ($\in \mathbb{R}^{|\mathbf{V}|}$).

Finally, we can define the classification-aware loss for the LSTM decoder. The objective acts as a classification loss for brand names that occur within a generated sentence. Hence, we add a softmax cross-entropy loss

$$L^{\text{cls}}(I, \mathbf{y}^{\text{cls}}) = -\log \left[\widehat{\overline{\mathbf{k}}}\right] \cdot \mathbf{y}^{\text{cls}}, \tag{4.9}$$

whereas $\mathbf{y}^{\text{cls}}$ is the one-hot encoded ground-truth vector for the classword included within the image I and $\bullet$ is the dot-product. This introduces a second modality and changes the model's total loss to

$$L^{\text{total}} = L(I, \mathbf{y}^S) + L^{\text{cls}}(I, \mathbf{y}^{\text{cls}}). \tag{4.10}$$

We visualize the classification-aware loss in Figure 4.4, which we integrated on top of the LSTM decoder network. The bottom of the figure (without the ratings module) is identical to the vanilla Show and Tell model from [139].

4.5.2 Image Ratings

The GfK-Captions dataset contains another modality, i.e., three integer-valued image ratings in the range of $[0, 4]$. For a subset of images, we have three annotations from five different annotators each, rating the interactions between the person and product in the following three dimensions: sentiment (r_1: positive vs. negative), involvement (r_2: high vs. low), and motive (r_3: emotional vs. functional). In the following r represents one of the three ratings ($r \in \{r_1, r_2, r_3\}$), e.g., ρ_r is a fully-connected layer for rating r, thus, we have three separate fully-connected layers.

In order to correctly describe these image ratings, we add an image ratings module for predicting image ratings in addition to our LSTM decoder module. In the following, we present three different ways to train our image ratings module.

4.5.2.1 Linear Regression

First, we add a linear regression model for predicting the image ratings. These image ratings are loosely and only indirectly related to the image captions. However, we want to infer the ratings from the image as well as indirectly optimize the caption quality by employing this objective for multi-task training. We append a fully-connected layer

$$\rho^r = \tilde{\mathcal{F}} \cdot \mathbf{W}^r \tag{4.11}$$

with one output neuron and weight matrix $\mathbf{W}^r$ for every image rating r at the last layer ($\tilde{\mathcal{F}} \in \mathbb{R}^{1\times 2048}$) of the Inception-v3 CNN. Thus, we can define the loss for an image rating r by

$$L^r(I, \mathbf{y}^r) = (\rho^r - \mathbf{y}^r)^2, \tag{4.12}$$

where $\mathbf{y}^r \in \mathbb{R}^1$ is the corresponding label for rating r.

4.5.2.2 Classification Task for Majority Ratings

The sentiment of the person-product interactions is perceived differently by different annotators. When giving annotators the task to objectively judge these ratings, they do not always agree with each other. In order to account for this issue, we determine a majority rating and define a classification problem. Every image rating can have an integer value between 0 and 4. We automatically determined the majority rating, i.e.,

the rating that most of the five annotators agree on. If there was no majority, we asked an additional annotator to determine this rating based on the image shown. We reuse the same fully-connected layer ρ^r for predicting majority ratings. However, as we change the problem formulation from a linear regression to a classification problem with five classes, we increase the number of output neurons to 5. In this case we train our model with a softmax (ϕ) cross-entropy loss

$$L^r(I, \mathbf{y}^r) = -\log\left[\phi(\rho^r)\right] \cdot \mathbf{y}^r, \tag{4.13}$$

where $\mathbf{y}^r \in \mathbb{R}^5$ again is the corresponding ground-truth class for rating r but encoded as a one-hot vector.

4.5.2.3 Soft-Targets for Annotator Disagreements

In order to account for the annotator disagreements, we additionally propose soft-targets for learning to predict the image ratings. With soft-targets, we model the occasional uncertainty between the 5 annotators, e.g., if 4 annotators chose 4 as rating and one annotator chose 3 as rating, our ground-truth signal $\mathbf{y}^r$ would be $[0, 0, 0, 0.2, 0.8]$, which describes the probability distribution of the 5 possible values of each rating. We use the sigmoid (σ) cross-entropy as loss function

$$L^r(I, \mathbf{y}^r) = -\sum_{i=0}^{4} \left(\mathbf{y}_i^r \cdot \log\left[\sigma(\rho_i^r)\right] + (1 - \mathbf{y}_i^r) \cdot \log\left[1 - \sigma(\rho_i^r)\right]\right). \tag{4.14}$$

This can be seen as logistic regression, which predicts the probability of each rating value. Note that $\mathbf{y}^r$ is also $\in \mathbb{R}^5$ but encoded as a probability distribution. Thus, $\mathbf{y}_i^r$ and ρ_i^r are the i-th element of the ground-truth distribution and i-th element of the fully-connected layer's output, respectively.

4.5.2.4 Total Loss

If we optimize our final model with the classification-aware loss and one of the above image ratings losses, the objective now changes to

$$\begin{aligned} L^{\text{total}} =& L(I, \mathbf{y}^S) + L^{\text{cls}}(I, \mathbf{y}^{\text{cls}}) + \\ & L^{r_1}(I, \mathbf{y}^{r_1}) + L^{r_2}(I, \mathbf{y}^{r_1}) + L^{r_3}(I, \mathbf{y}^{r_1}). \end{aligned} \tag{4.15}$$

4.5.3 SPO Captioning Metrics

Common metrics developed for the task of machine translation are BLEU [108] and METEOR [10]. CIDEr [138] is a metric developed specifically for image captioning and designed to correlate well with human judgment [138]. All these metrics have shown to score higher for machine-generated sentences than for human-generated sentences for the MSCOCO captioning challenge [19] in some cases. We also find that captions generated by our models score higher in comparison to the ground-truth sentences (see

Section 4.6.3.2 and Table 4.3). Common machine translation metrics also have the downside that they do not capture tiny important pieces of generated sentences like the object of interest or the predicate. For example, the generated sentence *"A **male** hand holds a can of cocacola above a tiled floor."* for the ground-truth sentence *"A **female** hand holds a can of cocacola above a tiled floor."* has a BLEU-4 score of 0.827, which is very high. In our setting, such minor differences are very important and, thus, we introduce novel metrics and make the assumption that popular metrics may disregard the semantics of the captions.

To allow for a more fine-grained evaluation of generated captions than existing methods do, we introduce *subject-predicate-object* (SPO) accuracies. We show in Section 4.6.3.4 that we can also use this metric on the MSCOCO [91] image captioning dataset. Therefore, it is presenting itself as an alternative to the common metrics that try to measure the quality of generated sentences. For our dataset, we manually collect the subject, predicate, and object of each of our ground-truth sentences. Since the brand names of the objects on the images are already known, we made sure that the annotators do not choose the brand name as object, but the actual object, i.e., the caption "A hand is holding a Coca Cola can in a car." results in the SPO triple *(hand, hold, can)*. While encoding the ground-truth annotations, our annotators faced several challenges, some of which we list here:

- Some sentences contained predicates associated with a noun (e.g., "a man is taking a selfie with a heinz bottle."), which we encode as *(man, take selfie, bottle)*.
- We code the grammatical number/numerus of the subject, e.g., "Three girls ..." will be annotated as *(three girls, ..., ...)*.
- We also faced sentences written in the passive case, which we also annotate in our ground-truths, e.g., the SPO triple *(woman, be covered, can)* results from the ground-truth caption "A young woman stands indoors and her chin is covered by a Coca Cola can.".
- Sentences containing an object describing a location, e.g., *table*, which is not the object of interest (in our case), were challenging to annotate. For example, we encode "A boy sits at a table with a heinz ketchup on it." as *(boy, sit, ketchup)*.
- We had difficulties annotating sentences with wrong spelling or grammar, but fixed these by letting the additional annotators re-annotate the corresponding image.

Because we have five sentences from different annotators per image, we also get 5 ground-truth SPO triples per image. We require the LSTM generated sentence to only match one of the SPO triples. We first define three different matching criteria $m_?$ (? is a wildcard, which stands for different kinds of matching criteria) per generated sentence:

(1) we set $m_{\text{subj}} = 1$ if the subject of the generated sentence matches, (2) $m_{\text{pred}} = 1$ if the predicate matches and (3) $m_{\text{obj}} = 1$ if the object matches. $m_{\text{subj}}, m_{\text{pred}}, m_{\text{obj}}$ are set to 0, otherwise. By combining those matching criteria, we define eight derived matchings $m_0 := \neg m_{\text{pred}} \wedge \neg m_{\text{obj}} \wedge \neg m_{\text{subj}}$, $m_1 := m_{\text{obj}}$, $m_2 := m_{\text{subj}}$, $m_3 := m_{\text{obj}} \wedge m_{\text{subj}}$,

$m_4 := m_{\text{pred}}$, $m_5 := m_{\text{pred}} \wedge m_{\text{obj}}$, $m_6 = m_{\text{pred}} \wedge m_{\text{subj}}$ and $m_7 := m_{\text{pred}} \wedge m_{\text{obj}} \wedge m_{\text{subj}}$. Note that $m_0 = 0$ if at least one of subject, predicate, and object matches. For example, m_4 describes whether the predicate was generated correctly and m_7 equals 1 if subject, predicate, and object were generated correctly. We define the accuracies $\text{SPO}_?$ to be the fraction of generated captions over the test set which satisfy the matching criterion $m_?$. Thus, we have a number of different accuracies which tell us how often we generated a sentence with the correct subject, predicate, or object, and combinations of those. Different matching accuracies have different evaluation emphases. For example, for a captioning task that focuses on the interactions between persons and objects, the SPO_4 accuracy may be of special interest, while for a task that specializes on correctly identifying the actor, the SPO_2 accuracy may be suited best.

Since two different words can literally have the same meaning (e.g., *ad* and *advertisement*), we use synonym tables for our subjects and objects. Furthermore, we use conjugation tables for the predicates (e.g., *take*, *takes*, *take*, *is taking*, and *are taking*). Based on an analysis of captions collected by our annotators, we found that annotators tend to avoid repeating words (e.g., they alternate between pack, package and packaging). Hence, we created synonym tables consisting of manually encoded bidirectional and unidirectional synonyms. Bidirectional synonyms are of equal meaning. Unidirectional synonyms cannot be used synonymous in all contexts, e.g., the words *man* and *boy* can be replaced by *guy*, but we cannot implicitly infer the age of a *guy*.

4.6 Experiments

In the following, we verify changes made to the original Show and Tell architecture by conducting several experiments that justify our architectural choices. But first, we introduce the new sentence classification accuracy that allows us to interpret generated sentences with different accuracy measures. Furthermore, we explain our experimental setting and the training configuration.

4.6.1 Sentence Classification Accuracy

The sentence classification accuracy (SCA) is an essential metric to assess if our model generates correct captions. The images of our dataset always show an interaction between a person and an object belonging to a certain company logo (e.g., in Figure 4.3 a person is holding a Coca-Cola can). While humans may think that a generated sentence like 'A woman is holding a Pepsi can' sounds reasonable, this sentence fails to classify the correct company logo. Since we describe interactions between humans and branded products, it is essential that a generated sentence mentions the correct brand logo. Our metric measures the accuracy of correctly classified company logos within the generated captions.

For each brand logo, we have a special word (we call these brand words *classwords*) in the vocabulary that identifies this brand, e.g., "cocacola". We generate three captions for each image using beam search (see Section 2.2.7) with a beam size of 3. This heuristic considers the 3 best sentences up to the iteration step t as candidate sentences for the

generation of sentences of length $t + 1$. If the classification of an image is *cocacola*, we consider the set of three generated sentences (Φ) a match if the classword is contained in any of them, i.e., $\text{match}(I, \Phi, \mathbf{y}^{\text{cls}}) = 1$ if the classword for the ground-truth class $\mathbf{y}^{\text{cls}}$ is contained in any sentence in Φ. If the ground-truth class is not contained in any of the generated sentences, then $\text{match}(I, \Phi, \mathbf{y}^{\text{cls}}) = 0$. The correct classification of an image I is given by $\mathbf{y}^{\text{cls}}$, i.e., $\mathbf{y}^{\text{cls}}$ is the one-hot encoded class label.

We then calculate the accuracy metric for each individual classword C with

$$\text{accuracy}(C) = \frac{1}{|\mathbf{I}_C|} \sum_{I \in \mathbf{I}_C} \text{match}(I, \Phi, C), \tag{4.16}$$

where $\mathbf{I}_C$ is the set of all test images belonging to classword C. For evaluating our model, we use the *mean accuracy* (MA) and the *overall accuracy* (OA) metrics. Because our dataset is distributed unequally among classes (see Section 4.4), we introduce the mean accuracy metric which calculates the average accuracy over all classwords

$$\text{MA} = \frac{\sum_{C \in \mathbf{C}} \text{accuracy}(C)}{|\mathbf{C}|}, \tag{4.17}$$

where $\mathbf{C}$ is the set of all classwords. The overall accuracy (OA) is given by

$$\text{OA} = \frac{1}{|\mathbf{I}|} \sum_{C \in \mathbf{C}} \sum_{I \in \mathbf{I}_C} \text{match}(I, \Phi, C) = \frac{1}{|\mathbf{I}|} \sum_{C \in \mathbf{C}} |\mathbf{I}_C| \cdot \text{accuracy}(C), \tag{4.18}$$

where $\mathbf{I}$ is the set of all test images. We calculate the overall accuracy and mean accuracy and analyze them depending on the use case. In our experiments, we commonly call both metrics under the term of sentence classification accuracy (SCA).

4.6.2 Training Configuration

4.6.2.1 Feature Extractor CNN

Similar to Vinyals et al. [139], we extract image features with a pretrained Inception-v3 CNN [132]. For our baseline model, we take a vanilla Inception-v3 pretrained on the 1000 ImageNet classes [122]. For further experiments, we also fine-tune the CNN to our logo classes. The network is fine-tuned in two stages. First, we append a fully-connected classification layer to the feature extractor of the CNN and train it for 50,000 iterations with a learning rate $\eta = 0.01$ while keeping the weights of the feature extractor frozen. Second, we also adapt the weights of the feature extractor with a learning rate of $\eta = 5 \cdot 10^{-4}$ for 20,000 iterations. This allows for adaption of the parameters in the early layers to generate activations specific to our logo classes.

4.6.2.2 Show and Tell Model

The remaining parameters of the Show and Tell model match the configuration of Vinyals et al. [139], i.e., we set the number of hidden units of the LSTM cell to 512, the dimension

of the word embedding $\mathbf{W}^e$ to 512 and train on image batches of size 32. We employ a stochastic gradient descent optimizer with a learning rate $\eta = 2.0$ and clip gradients to a maximum magnitude of 5.0. The learning rate is decayed by 0.5 every 8 epochs. We train our models on the *LogosSimple* dataset for 68 epochs and decay the learning rate 8 times by a factor of 0.5. For the extended models with classification-aware loss and ratings prediction, we optimize L^{total} according to Equation 4.15.

4.6.2.3 Image Ratings

As we described in Section 4.5.2, we train three different variants of the image ratings classification. We denote the linear regression ratings models (Section 4.5.2.1) with *LIN*, the classification models (Section 4.5.2.2) with *SCE* and the soft-targets models (Section 4.5.2.3) with *ST* in Table 4.3. A learning rate of $\eta = 2.0$ is too high for training a vanilla DCNN with a fully-connected layer. Thus, we reduce the learning rate by a factor of 2000 to $\eta = 0.001$ for training the fully-connected layers for the image ratings. For each of the image ratings $r \in \{r_1, r_2, r_3\}$, we train the corresponding fully-connected layer with the same learning rate of $\eta = 0.001$. We train the models with linear regression (denoted by LIN in column IR in Table 4.3) with the squared error loss defined in Equation 4.12. For the vanilla classification model *ratings-cls* (denoted by SCE in column IR in Table 4.3), we train a softmax cross-entropy loss (see Equation 4.13). For the soft-target models (denoted by SCE in column IR in Table 4.3), we optimize the ratings according to a sigmoid cross-entropy loss (see Equation 4.14).

4.6.2.4 Multi-Task Training

The only way for the image ratings to influence the generated sentences is to fine-tune the Inception-v3 feature extractor network. In other words, the training signal of the loss function is backpropagated into the layers of the CNN encoder, thus, allowing the CNN to influence the LSTM decoder network. Hence, we unfreeze the Inception-v3 encoder network for 3,000,000 more iterations with a reduced learning rate of $\eta = 5 \cdot 10^{-4}$. We employ this training strategy in Section 4.6.3 for our models with suffix *+ft*.

4.6.3 Results

In the following, we analyze and discuss results of our models for

- the sentence classification accuracy (SCA),
- commonly used metrics for image captioning,
- predicted image ratings and
- the subject-predicate-object metric for various model configurations.

The baseline model is the Show and Tell model trained on our *LogosSimple* dataset (*base*) with an Inception-v3 initialized with ImageNet weights (IC-v3). Furthermore, we

initialize our models with suffix *+L* (models #2, 3, 4, 8, 9) with the Inception-v3 model fine-tuned to our 26 logo classes. For models trained with the classification-aware loss (see Section 4.4), we append the suffix *+CA* to the model name, e.g., *base+CA*.

Table 4.3: Results for our models. The first column states the model number and the second column the model name, columns 3–5 state the dataset used (DS), whether we trained with the classification-aware loss (CA) and the kind of image ratings loss (IR). Columns 6–8 show the BLEU-4 (B-4), METEOR (Met) and CIDEr (Cid) scores. The last two columns show the overall accuraccy (OA) and mean accuracy (MA).

#	Method (Initialization)	DS	CA	IR	B-4	Met	Cid	OA	MA
1	**base (IC-v3)**	LS	—	LIN	54.80	34.10	177.10	75.66	58.72
2	**base+L (IC-v3 Logos)**	LS	—	LIN	56.30	35.10	192.50	90.94	81.34
3	**base+CA+L (IC-v3 Logos)**	LS	✓	LIN	44.40	28.60	146.40	91.04	81.86
4	**fuse+L (IC-v3 Logos)**	LE	—	LIN	54.20	34.40	185.80	88.87	79.63
5	**fuse+L+ft (#4)**	LE	—	LIN	58.50	36.40	201.20	88.02	79.43
6	**LIN+CA+ft (#5)**	LS	✓	LIN	61.20	37.10	**210.70**	90.75	83.20
7	**base+CA (IC-v3)**	LS	✓	ST	53.09	32.65	155.93	74.72	56.25
8	**base+CA+L (IC-v3 Logos)**	LS	✓	ST	53.62	33.38	174.64	91.42	81.26
9	**fuse+CA+L (IC-v3 Logos)**	LE	✓	ST	54.30	34.40	178.01	88.17	77.31
10	**fuse+CA+L+ft (#9)**	LE	✓	ST	51.45	32.69	162.04	77.10	46.39
11	**soft-targets+CA (#10)**	LS	✓	ST	60.82	37.23	204.26	90.25	78.76
12	**soft-targets+CA+ft (#10)**	LS	✓	ST	61.30	**37.56**	207.25	91.11	80.45
13	**ratings-cls (#10)**	LS	✓	SCE	60.39	37.28	204.10	90.68	81.96
14	**ratings-cls+ft (#10)**	LS	✓	SCE	**61.43**	37.34	206.80	**91.53**	**83.55**
—	**gt**	—	—	—	52.00	38.00	194.00	—	—
—	**NICv2 [140] on MSCOCO**	—	—	—	31.00	25.00	97.00	—	—

We also conduct experiments in which we merge the *LogosSimple* dataset with the MSCOCO dataset (*LogosExtended*) to allow for learning a wider variety of sentences. We give models trained on the extended dataset the prefix *fuse*. To compensate for the fewer images in our dataset, we replicate our dataset eight times to balance both datasets. The resulting ratio is about $0.65 : 1$ for our dataset vs. the MSCOCO dataset. For example, *fuse+L* is the model where both the LSTM part and the image ratings part are trained on the larger dataset for 1,700,000 iterations. *fuse+L+ft* continues training *fuse+L* with the parameters of the Inception-v3 networks unfreezed and a learning rate of $\eta = 5 \cdot 10^{-4}$ until iteration 3,000,000. We do not use classification-aware training for both *fuse+L* and *fuse+L+ft*.

For models that predict image ratings via the soft-targets strategy, we enable the classification-aware training, as our preliminary experiments with linear regression (*LIN+CA+ft*) show that enabling the classification-aware loss benefits models with fine-tuning enabled for the encoder network. For a final performance boost, we initialize our final models (#6, 11, 12, 13, 14) with models trained on *LogosExtended* and train again on images that only contain person-product interactions (*LogosSimple*).

In Table 4.3, we list our models with their respective BLEU-4, METEOR, CIDEr and SCA scores. In Table 4.5 and Table 4.6, we report the results for the image ratings and the SPO metrics, respectively.

4.6.3.1 Sentence Classification Accuracy

In Table 4.3 we also report SCA scores (i.e., mean accuracy (MA) and overall accuracy (OA)) for our models. We see that our problem mainly profits from fine-tuning the base network to the specific classification labels of our dataset, i.e., models #1 and #7 which were initialized with a vanilla Inception-v3 perform significantly worse. At first glance, the classification-aware loss does only marginally improve the SCA scores (see model #3 vs. #2). However, when using the *LogosExtended* dataset (#4 vs. #2 and #9 vs. #8), we notice a small degradation of the SCA scores, which is due to the fact that the dataset now not only consists out of branded product images. When allowing changes to the Inception-v3 encoder network (models $\#5, 10$), the SCA scores decrease even more. However, models #6 and $\#11 - \#14$ use all three modalities, allow changes to the encoder network and are fine-tuned on the *LogosSimple* dataset. As a consequence the SCA scores improve because the different tasks can influence each other through the encoder network. In total, our multi-task architectures improve the mean accuracy (MA) by an absolute of 24.5 % (model #6 vs. #1), 24.2 % (model #12 vs #7) and 27.3 % (model #14 vs #7), respectively. The models also improve the overall accuracies (OA) by 15.1 %, 16.4 % and 16.8 %.

4.6.3.2 BLEU-4, METEOR and CIDEr Scores

In Table 4.3, we also list how our models perform according to the well-known BLEU-4 [108], METEOR [10] and CIDEr [138] scores. We use the official script provided by the authors of the MSCOCO dataset (see Section 2.4.5) to calculate these metrics. Note that the scores for the ground-truth (*gt*) were computed by comparing each annotator against the other four and then averaging over all scores per annotator. For the other models, we compared the generated sentence against the five reference sentences from the *LogosSimple* dataset, i.e., even when trained on *LogosExtended* we evaluate the performance of our model on images from *LogosSimple* only. When training with the classification-aware loss, all scores decrease compared to the baseline models (see models $(\#3, \#8)$ vs. #2, and #7 vs. #1). BLEU-4, which is the standard in machine-translation, only decreases slightly in contrast to the baseline. When updating parameters in the encoder network (#5), the parallel training of the three modalities improve the BLEU-4, METEOR and CIDEr scores in comparison to #4. Initializing the *LIN+CA+ft* model with model #5 yields the best results for CIDEr. The models which train the image ratings with soft-targets and with a vanilla classification loss (*ratings-cls*) reach similar performance, i.e., model #12 has the highest METEOR score.

Because the *LogosExtended* training data contains a great number of sentences that do not only focus on interactions of humans and branded products (i.e., the MSCOCO dataset), our models learn to generate more diverse sentences. Usually, the *fuse* models

profit form this extended dataset (see models #4 vs. #3 and #9 vs. #8). However, if we train too long, the model tends to generate sentences that stem from the MSCOCO dataset, thus, decreasing performance on *LogosSimple*, which we test on. For example, we see this behaviour for model #10 that looses CIDEr performance in comparison to the initialization model (#9). For final optimization, we initialize our best performing models with model #10 and fine-tune on the *LogosSimple* dataset. We see that model *soft-targets+CA+ft* performs best for METEOR and performs second best for CIDEr and BLEU-4 metrics. Thus, we conclude that using soft-targets instead of linear regression (*LIN+CA+ft*) has no negative effect on the overall sentence quality. Using the traditional classification approach instead of soft-targets (*ratings-cls-ft*) delivers similar results.

4.6.3.3 Image Ratings

In the following, we present the results for predicting our three different image ratings r_1, r_2 and r_3. For each rating, we use a separate fully-connected layer $\rho^r, r \in \{r_1, r_2, r_3\}$ and model it (1) with a linear regression (*LIN*), (2) with a softmax cross-entropy loss as a classification problem (*SCE*) and (3) with a sigmoid cross-entropy loss as a logistic regression problem (*ST*), i.e., we predict the probability distribution among the different rating values (an integer-value between 0 and 4). For (1), we set the number of output neurons of the fully-connected layer to 1, while (2) and (3) have 5 output neurons. We evaluate the performance by using the accuracy measures accuracy^{r_1}, accuracy^{r_2} and accuracy^{r_3}, which tell us how often our model predicts the correct rating out of all evaluation examples. For *LIN*, we also measure the performance with a deviation measure that describes the deviation from the ground-truth.

Linear Regression (LIN) We evaluate the *LIN* model with the mean deviation from the ground-truth. For the image depicted in Figure 4.3, *LIN+CA+ft* infers a rather positive interaction with the brand ($\rho^{r_1} = 0.92$), concludes the person is involved with the branded product ($\rho^{r_2} = 0.00$) and infers that the interaction is neither emotional nor functional ($\rho^{r_3} = 1.86$). We calculate the deviations from the ground-truth for a single example with

$$\text{deviation}^r(I) = \sqrt{(\rho^r - \mathbf{y}^r)^2} = |\rho^r - \mathbf{y}^r|, \tag{4.19}$$

where $\mathbf{y}^r \in \mathbb{R}^1$ as defined in Section 4.5.2.1. Note that this is the square root of the squared error as defined in Equation 4.12 and r can stand for any rating $\in \{r_1, r_2, r_3\}$. We calculate the mean deviation (MD) for an image rating r by averaging over all images $\mathbf{I}$ in the test set:

$$\text{MD}^r = \frac{1}{|\mathbf{I}|} \sum_{I \in \mathbf{I}} \text{deviation}^r(I) \tag{4.20}$$

In Table 4.4 we present the mean deviations from the ground-truth image ratings in our test set. We notice that MD^{r_2} increases, while MD^{r_3} decreases when initializing the model with Inception-v3 Logos in *base+L*. In *base+CA+L* the Inception-v3 encoder

Table 4.4: Mean deviations from the ground-truth image ratings on the test set for our different models with linear regression.

#	Method	MD^{r_1}	MD^{r_2}	MD^{r_3}
1	**base**	0.5703	0.8141	0.8504
2	**base+L**	0.5663	0.8656	0.8078
3	**base+CA+L**	0.5644	0.8613	0.8054
4	**fuse+L**	0.6071	0.9379	0.8576
5	**fuse+L+ft**	**0.5218**	**0.7827**	0.6974
6	**LIN+CA+ft**	0.5236	0.7876	**0.6968**
—	**gt**	0.2541	0.1894	0.3138

network is kept frozen. Thus, the model performs about the same for all ratings, because the classification-aware loss cannot influence the image ratings. All image rating deviations increase, when using the *LogosExtended* dataset and start to decrease below the values from *base, base+L* and base+CA+L when allowing adaptions of the parameters of the encoder network. Note that *fuse+L+ft* and *LIN+CA+ft* only differ slightly and we observe that training all modalities in parallel improves results for all experiments. Again, the scores for *gt* were computed by comparing each annotator against the other four and then averaging over all deviations per annotator.

Logistic Regression (soft-targets) *soft-targets* allows us to evaluate results from two different perspectives. First, we can predict one rating with the arg max and compare it to *ratings-cls*. Second, the sigmoid cross-entropy loss function allows us to predict a probability distribution, which matches the distribution of the ground-truth annotations.

Table 4.5: Results for our models. The first column states the model number and the second column the model name, columns 3–5 depict the ratings accuracies for ratings r_1, r_2, r_3.

#	Method	$accuracy^{r_1}$	$accuracy^{r_2}$	$accuracy^{r_3}$
7	**base+CA**	57.26	51.34	54.72
8	**base+CA+L**	63.28	59.34	58.28
9	**fuse+CA+L**	62.22	61.85	57.04
10	**fuse+CA+L+ft**	**76.30**	66.67	69.63
11	**soft-targets+CA**	75.58	69.95	**70.65**
12	**soft-targets+CA+ft**	74.60	**71.01**	**70.65**
13	**ratings-cls**	73.15	68.15	67.51
14	**ratings-cls+ft**	75.26	70.40	66.74

To allow comparison between *ratings-cls* and *soft-targets*, we use the majority rating as ground-truth during evaluation for both methods. From the predicted probability distribution, we choose the maximum argument as the predicted class. Thus, we predict a value between $0-4$ even though the fully-connected layer produces a probability distribution that mimics the occasional uncertainty between annotators.

As we see in Table 4.5, our best model *soft-targets+CA+ft* achieves 74.60 %, 71.01 % and 70.65 % for the accuracies accuracy^{r_1}, accuracy^{r_2} and accuracy^{r_3}, respectively. For the image in Figure 4.3 the image ratings predict that the interaction is neither positive nor negative ($r_1 = 2$), the person is rather involved with the product ($r_2 = 1$) and the interaction is more functional than emotional ($r_3 = 3$).

In addition to analyzing the image ratings with the accuracy measure, we also compare the predicted probability distribution against the distribution generated by the annotators, i.e., we want to examine if we can imitate the occasional uncertainty between our human annotators. We calculate the L2 distance from the predicted value to the majority rating for both the ground-truth values and predicted values. In Figure 4.5, we visualize the prediction distribution of model *soft-targets+CA+ft* compared to the test split ground-truth. The green bars show the fraction of predicted ratings which have an L2 distance to the ground-truth rating of $0-4$. The red bars show the L2 distance of each annotator to the majority rating. We can observe that the predicted distribution closely models the annotator disagreement on the hold-out test set. In addition, we also visualize the normal distributions of the predictions and ground-truth together with their respective mean μ and standard deviation σ. For r_1 the annotators deviate $\mu = 0.30$ from the majority rating on average and have a standard deviation of $\sigma = 0.50$. The distribution predicted by our model comes really close with $\mu = 0.28$ and $\sigma = 0.49$. Also, rating r_3 has a very similar distribution and only r_2 differs slightly from the ground-truth.

For the models with linear regression, we calculate the L2 distance (i.e., *deviation*) from the float value prediction to the mean of the ground-truth annotations. The best model that uses linear regression (see *fuse+L+ft* in Table 4.4) achieves mean L2 distances of 0.52, 0.78, and 0.70 for ratings r_1, r_2, and r_3, respectively. Furthermore, the mean deviations for the ground-truths (we compare one annotator against the mean of the other 4 and average over all 5 annotators) are 0.25, 0.19, and 0.31. Thus, the deviations of the predictions differ from the ground-truth by 0.27, 0.59, and 0.39, respectively. In comparison, the best model with soft-targets (*soft-targets+CA+ft*) only differs from the ground-truth mean by 0.02, 0.07, and 0.03 for ratings r_1, r_2, and r_3, respectively.

Classification (ratings-cls) For a more thorough analysis, we employed the classical classification approach, i.e., we used a softmax cross-entropy loss for the image ratings with the majority rating as the ground-truth.

In Table 4.5, we denote the models with the classification loss as *ratings-cls* and *ratings-cls+ft*. We see that neither *ratings-cls* nor *ratings-cls+ft* could surpass the classification accuracy of the soft-targets approach except for rating r_1 (75.26 %, 70.40 %, and 66.74 % vs. 74.60 %, 71.01 %, and 70.65 % for rating accuracies accuracy^{r_1}, accuracy^{r_2},

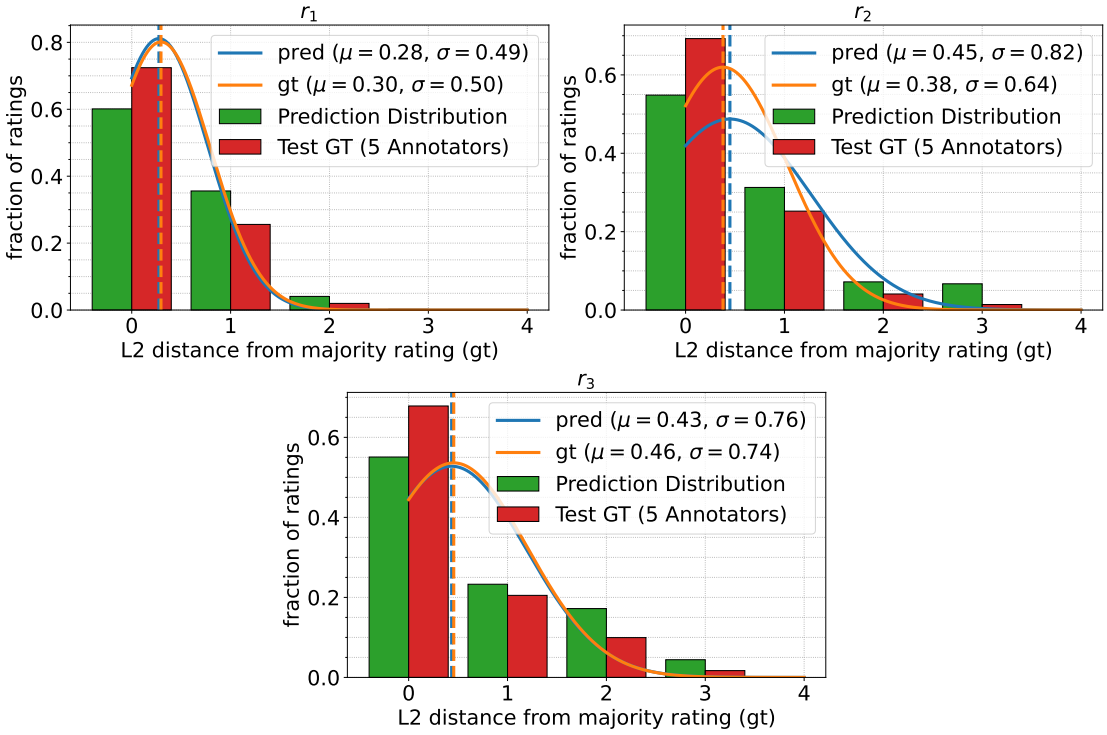


Figure 4.5: The predicted (green) L2 deviations from the ground-truth majority rating in comparison to the mean L2 deviations (red) of the annotator labels to the majority rating. (The predictions were generated by model *soft-targets+CA+ft*, i.e., model #12.)

and accuracy^{r_3}, respectively). However, the SCA scores (see Section 4.6.3.1 and Table 4.3) are the best for the *ratings-cls+ft* model.

4.6.3.4 SPO Accuracy Metrics

GfK-Captions In Table 4.6, we also report the scores our models achieve with our proposed SPO accuracy metrics. As we did for all other evaluation measures, we calculated the ground-truth accuracies for SPO_0 - SPO_7 by calculating the accuracies for every annotator against the other four annotators and averaging them over the five annotators. In our models we use beam search with a beam size of 3 during evaluation and when infering captions for images without annotations. During evaluation, we choose the caption that matches most of the three sentence clauses defined by us (m_{subj}, m_{pred}, m_{obj}). For example, if one of the three sentences yielded by beam search has a match for subject, predicate, and object and the other sentences match two of the sentence clauses, we choose the first caption. If multiple sentences have the same number of matches, we randomly select one caption. With the *soft-targets+CA+ft* model, we achieve the best scores, i.e., we identify the correct predicate in 85.66 % of all cases and generate completely correct sentences (according to the SPO metric) in 70.00 % of all cases. In

Table 4.6: Results for our models. The first column states the model number and the second column the model name, the following eight columns represent the SPO accuracies SPO_0–SPO_7.

#	Method	SPO_0	SPO_1	SPO_2	SPO_3	SPO_4	SPO_5	SPO_6	SPO_7
7	**base+CA**	**0.00**	77.55	82.83	67.55	80.38	66.04	68.30	58.68
8	**base+CA+L**	**0.00**	81.32	77.83	65.47	80.47	68.40	64.91	56.04
9	**fuse+CA+L**	**0.00**	81.27	82.51	68.60	80.03	69.56	67.63	59.78
10	**fuse+CA+L+ft**	**0.00**	76.17	80.30	63.22	80.58	66.67	66.25	56.34
11	**soft-targets+CA**	**0.00**	85.19	86.89	76.42	85.57	74.62	76.42	68.02
12	**soft-targets+CA+ft**	**0.00**	85.94	**88.87**	**78.49**	**85.66**	**75.75**	**77.74**	**70.00**
13	**ratings-cls**	**0.00**	84.15	87.64	75.19	83.96	72.36	75.28	65.47
14	**ratings-cls+ft**	**0.00**	**86.13**	88.21	77.36	84.62	74.53	75.94	67.45
—	**gt**	—	68.00	98.00	67.00	98.00	67.00	97.00	66.00
—	**NICv2 [140] on MSCOCO**	**0.00**	63.00	67.00	42.00	64.00	37.00	39.00	22.00

comparison, our base model only generates sentences with matching subjects, predicates and objects only in 58.68 % of all cases (*base+CA*). Note that $SPO_0 = 0$ represents the best case for the SPO_0 metric. A score of 0 means that at least one of subject, predicate, or object was matched in each generated caption of all images in the whole test set.

MSCOCO We want to show that our SPO accuracies measure can be used on other image captioning datasets. To do so, we generated SPO ground-truth triples from the human-annotated sentences of the MSCOCO validation split. We use the natural language processing library spaCy[2] to extract SPO triples from sentences similar to how our human annotators extracted the triples from our dataset and publish these annotations[3]. We then trained the Show and Tell [139, 140] implementation from the TensorFlow models repository on MSCOCO, which yielded scores slightly worse than those published by Vinyals et al. [140].

We found that the ground-truth captions of MSCOCO often contain no predicate (e.g., *A room with blue walls and a white sink and door.*). We were able to automatically extract SPO ground-truth triples (including the triples without a predicate) from 184,308 out of 202,654 image captions in total. Additionally, we did not collect synonym tables for the MSCOCO ground-truth SPO triples, which decreases the final SPO accuracies. In the bottom row of Table 4.6, we list the different accuracies the MSCOCO model achieves with our metric. We match at least one object in 63 %, the subject in 67 %, and the predicate in 64 % of all cases. However, as we see with the $\mathbf{SPO_7}$ accuracy, the Show and Tell model produces captions which contain the correct subject, predicate, and object in only 22 % of all cases. We hypothesize that manually annotated SPO triples would lead to a more accurate result.

[2] `https://spacy.io/`

[3] `https://github.com/philm5/mscoco-spo-triples`

4.6.3.5 Multi-Task Learning

To show the effectiveness of MTL, we can look at model *base+CA+L* (model #8). Note that we did not unfreeze the Inception-v3 encoder network, i.e., we did not allow fine adjustments to the parameters in the encoder. As the encoder network is the only shared part for the different modalities (i.e., image ratings and image captions), the image ratings cannot influence the image caption quality through the encoder network and vice versa. Therefore, we see in Tables 4.3, 4.5, 4.6 that all scores except the sentence classification accuracy decrease considerably when compared to a model trained in a multi-task setting (e.g., *soft-targets+CA* and *soft-targets+CA+ft*).

4.6.4 Visual Results

In Figure 4.6, we show three more images from the GfK-Captions dataset test split. Alongside the images, we present results generated by our model *soft-targets+CA+ft*. Directly below each image, we depict the three image ratings with their predicted integer value (class) on the scale of possible ratings. For example, the interaction of image (b) is neither positive nor negative, the person is rather involved with the branded product and the interaction is functional.

4.7 Summary

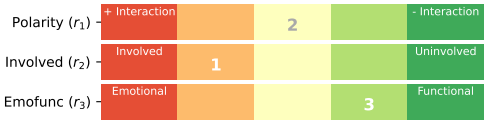
In this chapter, we presented an architecture capable of simultaneously generating image captions and ratings for images depicting scenes of interactions between humans and branded products. Our focus lies on generating captions that satisfy other constraints than traditional image captioning pipelines.

First, we extended an image captioning model to prioritize on describing the brands of products depicted in a scene by using a special loss function, which penalizes if a brand name is not contained within the caption.

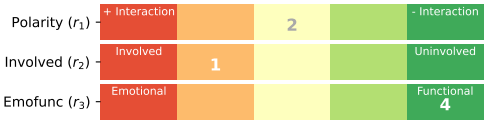
Second, we presented novel clause-focused accuracy metrics that focus on the correct transcription of the subject, object, and predicate. In addition to the ground-truth image captions, we annotated subject, object, and predicate (including synonyms) for our dataset to be able to measure the quality of our generated sentences in terms of these important clauses of the sentence. Furthermore, we automatically extracted subject, predicate, and object from the ground-truth annotations of the popular MSCOCO dataset to verify our approach by applying our metrics on a different dataset.

Third, with a multi-task training objective, we were also able to simultaneously predict ratings for the input image, which describe (1) if the person is involved with the brand, (2) whether the interaction is rather positive or negative, and (3) whether the interaction is functional or emotional.

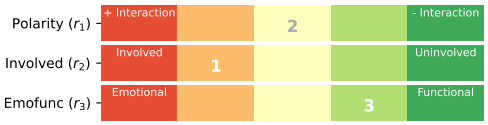
Fourth, by modeling the occasional uncertainty between annotators with a soft-target logistic regression, we were able to improve overall sentence quality in an end-to-end multi-task optimization.



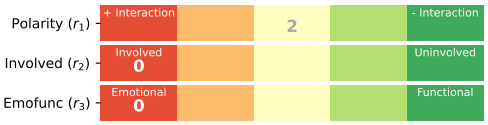
(a) *"a female hand holds a can of cocacola above a tiled floor."*



(b) *"a hand is holding a bar of kinderriegel."*



(c) *"a hand is holding a can of heinz."*



(d) *"a woman is holding a nutella jar in front of her face."*

Figure 4.6: More images from the test partition of the GfK-Captions dataset for the classes cocacola, kinderriegel, heinz and nutella. Alongside the images, we show predicted image ratings and generated captions by our model #12 (*soft-targets+CA+ft*).

5 Visual Question Answering

In this chapter, we address the problem of answering questions regarding an image. Instinctively, one might think that answering a question about an image falls into the category of image description generation similar to image captioning, which we discussed in the previous chapter. However, Antol et al. [5] defined the task of visual question answering (VQA) as a classification problem, i.e., there is a fixed number of possible answers for a dataset. This contradicts the common theme of this part of the thesis as language is not generated word by word. Particularly, we find that answering questions is not as simple as choosing an answer from a predefined set of answers. Therefore, we explore the possibility of generating answers from scratch, and do not choose one out of a set of given possible answers.

The following chapter is based on our publication:

> **Visual Question Answering with a Hybrid Convolution Recurrent Model** [52], Philipp Harzig, Christian Eggert and Rainer Lienhart, ACM International Conference on Multimedia Retrieval 2018, Yokohama, Japan, June 2018.

Visual question answering is a relatively new task, which infers answer sentences for an input image coupled with a corresponding question. Instead of dynamically generating answers, they are usually inferred by finding the most probable answer from a fixed set of possible answers. Previous works did not address the problem of finding all answers contained in the training set. In contrast, they only modeled the answering part of VQA as a classification task. The classification is trained on the top-k most common answers from the training set. Thus, by design, a fraction of the answers contained in the dataset cannot be generated by the model. Furthermore, other works did also not explore the generation of new, previously unseen, answers. To tackle this problem, we infer answer sentences by using an LSTM network that allows us to dynamically generate answers for image-question pairs. In a series of experiments, we have developed an end-to-end trainable deep neural network structure which allows us to dynamically answer questions referring to a given input image by using an LSTM decoder network. With this approach, we are able to generate both less common answers, which are not considered by classification models and more complex answers. This could be beneficial for datasets to come that contain more answers which are longer than three words.

5.1 Motivation

The goal of the VQA task is to answer a question that asks about something related to a given input image. This task is challenging, because a good understanding of both

(a) *Q: How many tomatoes are there?*
A: 3

(b) *Q: What vegetables are on the pizza?*
A: olives

(c) *Q: Is the trick the skateboarder is doing difficult?*
A: yes

(d) *Q: What is in the picture?*
A: cupcakes

Figure 5.1: Original questions, images and answers from the VQA-v2 dataset [45]. The answer is the most likely answer, i.e., the answer that was given by the majority of annotators.

images and natural language is essential to answer a question regarding the contents of the image. Its difficulty lies in fusing feature representations of the image and question into a joint representation, which then makes it possible to generate an answer for that very image-question pair. Unlike most other approaches, we use an LSTM network to generate the answers for the visual question. Contrary, the general practice is to collect all answers from the training set and select the top-k most common answers. Then, a classification model is trained that is only able to predict an answer from this predefined set of most common answers. Teney et al. [135] treat VQA as a multi-label classification task, i.e., for a given image and question, the answer is predicted from a fixed set of answers. This strategy delivers good scores as the official evaluation metric is the answer classification accuracy. Hence, a fixed set of the most common answers has a greater influence on the score than less common answers. We tackle the serious drawback of leaving out the less common answers altogether. We deal with this problem by constructing the answer with an LSTM network which can produce answers out of the entire vocabulary. In this chapter, we present the following new ideas for the VQA task:

1. We use a CNN network to extract features from the input question, which has been turned out to work better than an LSTM encoding in our case.
2. We compare different approaches that embed the question and image features into a joint semantic space.
3. We do not see the VQA task as a classification problem but rather as a more complex problem that can not be approximated by some 3000 possible answers. Natural language is more complex than a set of predefined answers. Thus, we replace the fully-connected classification layer with an LSTM network that can produce arbitrary sentences.
4. Finally, we find that our architecture is able to produce new answers not contained in the training set at all, some of which we show in a qualitative analysis.

5.2 Related Work

As we have outlined before, we need to understand the contents of an image, parse a question and then properly fuse this information in order to answer a visual question. This might sound simple to achieve but we have to understand and review different subfields of research in the domain of computer vision, linguistics and machine learning in order to build a model that is capable of generating answers for visual questions. Therefore, we give a short overview of related work in the areas of natural language processing, image captioning and the embedding of multiple different modalities. Finally, we review existing VQA architecturs.

Natural Language Processing Natural language processing is a long-studied topic in the domain of linguistics and computer science. Barnard et al. [11] present a statistical

model that learns relationships between visual information from images and text information coming with the images. Farhadi et al. [36] infer triplets consisting of object, action and scene from handcrafted image features and transfer them to text afterwards. Similarly, Li et al. [85] use image features to compose sentences from scratch by using a text corpus comprised of n-grams gathered from the web. Lately, Recurrent Neural Networks (RNNs) in form of Long Short-Term Memory (LSTM) [64] networks became very popular in both understanding text and generating text.

The encoder-decoder structure used by current image captioning approaches as well as our VQA model is heavily influenced by machine translation networks that transform a sentence from one language into another. For example, Google [150] uses LSTM-driven neural networks for their translation system. Facebook [43] goes into another direction and eliminates the LSTM layers by replacing them with ordinary convolutional layers resulting in better performance in speed and accuracy.

As of late, recurrent language generation models and convolution-based models have been mostly replaced by self-attention models. With the introduction of the Transformer architecure by Vaswani et al. [137], the community soon settled on employing this promising architecture: First, BERT [29] has set new state-of-the-art results on multiple language processing tasks. Then, the GPT-3 [14] model has surpassed BERT's performance in many NLP tasks even though GPT-3 is trained in a few-shot setting, i.e., a training setting where only few training examples with supervised information are available. We, however, base the language parsing and language generation part of our VQA model in this chapter on traditional CNNs and recurrent networks. But we re-evaluate the VQA task with a Transformer architecture in Section 8.3.

Image Captioning The image captioning task is closely related to the VQA task. The goal of image captioning is to create a sentence in natural language that describes a given input image. VQA can be seen as a special case of image captioning, where the input question modality is added to the model and answers instead of descriptions should be generated. Different approaches appeared over time which try to convert the content of images into text fragments. State-of-the-art approaches still use the idea of extracting features from an image and converting them into a natural language sentence. They use standard image classification DCNNs like [132] as feature extractors. One well-known architecture is the Show and Tell model by Vinyals et al. [139, 140], which makes use of recurrent neural networks for sequence modeling to convert image features into sentences (see also Chapter 4). Also, Karpathy et al. [73] utilize neural networks for sequence modeling in form of a bidirectional recurrent neural network to generate captions for images. They refine their model to additionally describe parts of the image instead of captioning the image as a whole and call this method dense captioning [71].

Joint Multi-Modal Embeddings There are a lot of approaches that deal with combining multiple modalities in a joint embedding space. Zhou et al. [161] propose a simple VQA architecture which concatenates image and question modalities. Other models use simple element-wise multiplication of question and image features [5, 135].

Fukui et al. [38] state that the outer product between question and image modalities are more expressive than a vector concatenation or an element-wise product and introduce a technique called multi-modal compact linear pooling. However, the outer product is infeasible due to its high dimensionality. Hence they project the outer product into a lower dimensional space. Yu et al. [156] also use a bilinear pooling model to project the image and question into a joint embedding space (see Section 5.6.2.3).

VQA Architectures For the original VQA [5] task, many architectures have been presented. For example, Zhou et al. [161] propose an architecture that uses a simple bag-of-words model as question features and image feature vectors extracted by a CNN. The creators of the original VQA dataset themselves introduced a model that trains a classification model on the 1000 most frequent answers. They use an LSTM model for the question embedding and the VGGNet [125] as the image feature extractor DCNN. Finally, Fukui et al. [38] won the VQA challenge by using multi-modal compact linear pooling.

The VQA-v2 [45] dataset (we depict examples in Figure 5.1) is a balanced version of the VQA dataset (in contrast to the first version, version 2 links the same question with two similar images that have differing answers, see Section 5.3 for details). In their analysis, Goyal et al. found that many VQA models take advantage of language priors to predict the target answers and do not necessarily consider visual information. Thus, it is harder to achieve high scores on VQA-v2 than on the first version of the dataset. Furthermore, Teney et al. [135] present an architecture that uses soft-scores as ground truth targets instead of one-hot targets as used in traditional classification. They also implement a bottom-up attention to attend to specific regions of the images. With those and other findings they won the 2017 VQA challenge. After the challenge has ended, Yu et al. [155] even surpassed Teney et al. on the VQA-v2 open ended challenge leaderboard by using the multi-modal factorized bilinear pooling (MFB) approach.

5.3 Dataset

We train and test our model on the VQA-v2 [45] dataset, which is a more balanced version of the original VQA [5] dataset. Goyal et al. [45] balanced the VQA dataset in such a way that for every question there are two similar images that have differing answers to that question and, therefore, containing about twice the number of questions. They also found that models trained on the first version of the dataset exploit language priors, thus, performing worse on the VQA-v2 dataset. The VQA dataset is built upon the MSCOCO [91] dataset and contains question-answer pairs for a subset of the MSCOCO images. Their dataset contains 443K, 214K and 453K questions in the train split, validation split and test split, respectively. The test split is divided into four subsplits: test-dev, test-standard, test-challenge and test-reserve. There is an online evaluation server[1] for generated answers on the test-dev and test-standard split known as the *VQA*

[1] https://evalai.cloudcv.org/web/challenges/challenge-page/1/overview, test-dev allows 10 submissions per day, while test-standard is limited to 5 submissions in total.

Real Image Challenge 2017 (Open-Ended). On average, the VQA-v2 dataset contains 5.34 questions per image and 53.4 answers per image, i.e., every question was answered by 10 different annotators. Every question has a ground-truth of 10 answers, where every answer is flagged whether the annotator was confident to answer the question correctly. There are 8.11 confident answers on average for every question in the whole dataset. We only use confident answers for training our models.

For our experiments we rebalance the train and validation splits: We use the train split and 90% of the validation split as training set and the remaining 10% of the validation split as the validation set. We do this to have a larger training corpus available. For evaluating the performance of our models, we use the standard VQA script by Antol et al. [5]. This script calculates the overall accuracy and the accuracies over three different answer categories (see Section 2.4.6). These are yes/no answers (Y/N), answers that are numerical and other answers.

For some of our experiments, we extend the dataset by the popular *Visual Genome* (VG) [78] dataset, which contains an additional 1.7 million Visual Question Answers on 108,077 images.

5.4 Base Model

Our VQA model (see Section 5.5) is inspired by the model introduced by Teney et al. [135] and somewhat similar to their model. Our model shares the same basic architecture and uses some of the original design elements. We depict their model architecture in Figure 5.2. In the following we describe each part of this architecture.

Non-linear layers Teney et al. [135] introduce multiple non-linear layers in their architecture. A common non-linear layer could be a linear projection that is followed by a non-linearity such as a ReLU. In their work, however, they make use of gated hyperbolic tangent layers. That is, a fully-connected layer followed by a tanh activation function which is gated by values produced by another fully-connected layer with a sigmoid activation function. This gating mechanism is similar to LSTMs and gated recurrent units (GRU). Teney et al. argue that that these gated tanh layers show some benefit over layers with simple ReLU activation, tanh activation or even gated ReLU activation. In contrast to those other activation types, they improve the accuracy by nearly 1 %. Furthermore, another benefit of the gated tanh layer is that it doubles the number of parameters without doubling the channel dimension. We define the gated tanh non-linear layer as a function $f_\theta : \mathbf{x} \in \mathbb{R}^a \rightarrow \mathbf{x}' \in \mathbb{R}^b$:

$$\tilde{\mathbf{x}} = \tanh(\mathbf{W}^\theta \mathbf{x} + \mathbf{b}^\theta) \tag{5.1}$$

$$\hat{\mathbf{x}} = \sigma(\mathbf{W}^{\theta'} \mathbf{x} + \mathbf{b}^{\theta'}) \tag{5.2}$$

$$\mathbf{x}' = \tilde{\mathbf{x}} \circ \hat{\mathbf{x}}, \tag{5.3}$$

where $\mathbf{W}^\theta, \mathbf{W}^{\theta'} \in \mathbb{R}^{b \times a}$ are learned weight matrices and $\mathbf{b}^\theta, \mathbf{b}^{\theta'} \in \mathbb{R}^b$ are learned biases. We can use the gated tanh non-linear layer as a replacement for a standard fully-

connected layer (including activation function). The subscript θ stands for a parameter set for the respective layer. Just as a regular fully-connected layer, the gated tanh non-linear layer can have inputs of arbitrary dimension $\mathbb{R}^a$ and project these into an output dimension of $\mathbb{R}^b$. In their architecture, Teney et al. make use of these non-linear layers at different positions. We denote these layers as "gated tanh" in Figure 5.2.

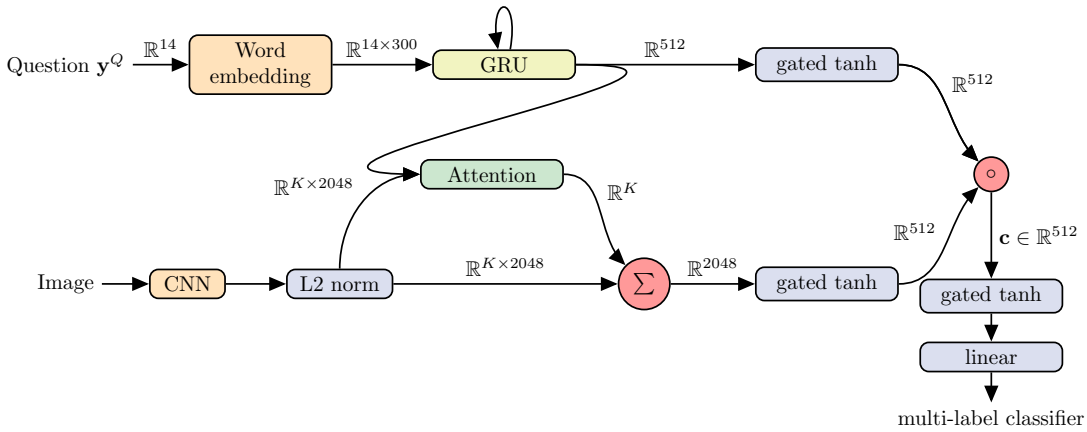


Figure 5.2: Architecture of the base model of Teney et al. [135]. Figure redrawn from their original paper. The question is embedded with an embedding layer and then processed with a gated recurrent unit. Image features are extracted with a CNN and then attended on with the question features. Both the question features and attended image's features are fed through a non-linear layer (i.e., a gated tanh unit, see Section 5.4). Finally, they fuse the results of these layers with an element-wise product, add another gated tanh layer and train the model on a multi-label classifier with a sigmoid cross-entropy loss.

Question Processing At the top left of Figure 5.2, we depict the question processing of the base VQA model. Teney et al. first preprocess the question by splitting the question into unigram tokens. In contrast to our models, they do not add a special start-of-sequence or end-of-sequence token. Then, they trim all questions to have a length of 14 tokens. About 99.75 % of the questions contained in the dataset have a length of 14 or less words. For questions that have more than 14 tokens, they simply discard these. All tokens are embedded with a word embedding that is initialized with the pre-trained *GloVe* (*Global Vectors for Word Representation*) [110] embeddings of dimension $\mathbb{R}^{300}$. Tokens which are not present in the pretrained embeddings are initialized with zero vectors and optimized during training. Questions that have less than 14 tokens are filled up with zero-only vectors that are not optimized during training. In contrast, we train these embeddings from scratch for our base model (see Section 5.5). Next, they pass the embedded tokens through a gated recurrent unit (GRU) [23]. A GRU is similar to an LSTM recurrent cell but its calculation process is slightly simpler. For instance, the forget and input gate are combined into a single gate and the cell state and hidden state

are combined into a single state. The last resulting hidden state is then utilized in the attention layer and forwarded through a gated tanh non-linearity.

Image Feature Extraction Teney et al. both use a standard DCNN [59] and a *Faster R-CNN* [119] for their model. As our model only uses a standard DCNN and does not rely on object-level features, we omit the R-CNN feature extractor in this description and Figure 5.2. In particular, the authors extract features with a ResNet-200 that yields feature maps of dimension $\mathbb{R}^{7\times7\times2048}$. Teney et al. found that normalizing the features with an L2 normalization was vital for yielding good results. These features are utilized for a multiplicative attention mechanism.

Attention The attention mechanism (see also Section 2.3.1) is visualized by the green attention layer in the center of Figure 5.2. More specifically, the spatial dimensions of the image features are reshaped to $\mathbb{R}^{K\times2048}$ and each spatial location $i \in \{1, \dots, K\}$ is concatenated with the question features from the GRU according to Equation 2.11 and forwarded through a gated tanh layer. This is done for every spatial location and the attention weights are calculated with the softmax function. Finally, the spatial locations of the feature map are re-weighted with the calculated attention weights and reduced by summation. This operation is visualized by the sum operation at the bottom center of Figure 5.2. The resulting attended feature map is passed through another gated tanh non-linear layer.

Fusion The results from the gated tanh layers of the question hidden state and the attended feature map are fused on the right side of Figure 5.2 (depicted by the red operation with the element-wise $\circ$ symbol). More specifically, the vectors are multiplied element-wise and yield a context vector $\mathbf{c}$. In their final model, Teney et al. also use pretrained linguistic and visual information before fusing the features. However, we only make use of simple element-wise fusion in our model, thus, we only discuss the element-wise fusion in this chapter.

Multi-Label Classification These fused features are fed through another gated tanh layer followed by a linear mapping to yield a fixed number of possible answers. More specifically, it is modeled as a multi-label classification task with soft-targets. As not every annotated answer is the same, the authors model the respective answers as a probability distribution (soft-targets). For example, if 7 out of 10 annotators answered with answer a) and 3 annotators answered with answer b), the ground-truth scores would be 0.7 and 0.3, respectively. The model is trained with a sigmoid-cross entropy loss that learns to predict the likelihood of each possible answer in the predefined set of answers.

5.5 Hybrid Convolution Recurrent Model

Our VQA model is composed of multiple components. We call it *Hybrid Convolution Recurrent Model* as the language processing and generation is a hybrid between convo-

lutional and recurrent neural networks. We depict our architecture in Figure 5.3. In similar fashion to the base model, we first need to make sense of the question asked. Therefore, we embed the question tokens and process these tokens with a CNN (denoted by Q-CNN on the top stream in Figure 5.3) which yields a single question feature vector. On the bottom, we forward the input image through a DCNN (Inception-v3 [132]

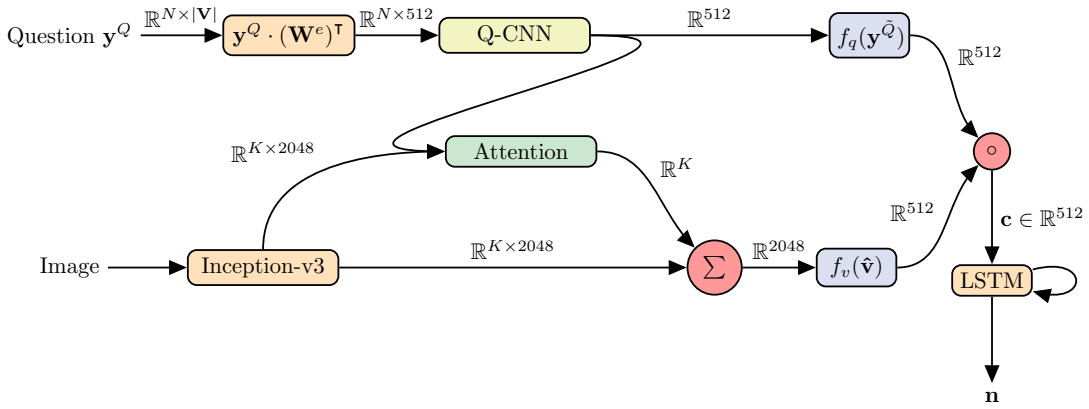


Figure 5.3: Architecture of our VQA model. We embed the question $\mathbf{y}^Q$ of length N via the word embedding matrix $\mathbf{W}^e$ and extract the question features with a residual CNN. We forward the input image through the Inception-v3 network that yields a feature map with K spatial locations. With an attention mechanism (depicted by the green attention layer), we produce K attention weights. With these attention weights, we then calculate a weighted sum over the spatial locations of our feature map (denoted by the red sum operation on the bottom center). Furthermore, we feed question features and attended image features through gated tanh layers f_q and f_v, respectively. Next, we extract a joint multi-modal feature vector $\mathbf{c}$ by element-wise multiplication which we use to initialize a regular LSTM cell with a fully-connected layer that produces outputs $\mathbf{n}$.

network in our case) that yields a feature map with $K = 8 \cdot 8$ spatial locations. We concatenate the spatial locations with the question feature vector from the Q-CNN and calculate attention weights. With these attention weights, we calculate a weighted sum of the spatial locations in the feature map. We then combine both the question feature vector and the attended image feature vector via multi-modal fusion into a single representation of the image-question pair.

Lastly, we aim to produce an answer that best fits the question given the input image. Contrary to other works in this area, we generate the answer with an LSTM network instead of modelling the answering part as a classification task.

We see that our architecture is built in a similar manner to the base model from Section 5.4. In the following, we describe each part of our architecture in more detail and also highlight the differences from the baseline model.

5.5.1 Understanding Questions

Question Preprocessing and Embedding One essential part of visual question answering is the question itself. Without understanding the question asked, we cannot properly answer things about the image. A natural approach for understanding a sentence (i.e., a question) is to use an RNN. However, in our model, we implement a CNN for question encoding, because it has shown better performance in our case (see Section 5.6.2.1). Similar to the explanation in Section 2.2.2 and to the preprocessing of ground-truth captions (see Chapter 4), we add a start word <S> at the beginning and an end word </S> to the end of each question. Additionally, we tokenize each question by splitting it up into single words. We allow questions to be of any length and construct our vocabulary $\mathbf{V}$ from all words contained in the questions and answers (we tokenize answers the same way as we do with the questions). Each word in the question $\mathbf{y}^Q = [\mathbf{y}_0^Q, \dots, \mathbf{y}_{N-1}^Q]$ of length N is represented by a one-hot vector $\mathbf{y}_t^Q$, where t is the index of the current word in the input question. We embed each word of the question into vectors

$$\hat{\mathbf{y}_t^Q} = \mathbf{W}^e \mathbf{y}_t^Q, t \in \{0 \dots N-1\} \tag{5.4}$$

of dimension $\mu = 512$, where $\mathbf{W}^e \in \mathbb{R}^{\mu \times |\mathbf{V}|}$. Technically, the embedding of a sentence or question is implemented as a single matrix multiplication instead of embedding each word at once. Furthermore, the question is represented as a matrix $\mathbf{y}^Q \in \mathbb{R}^{N \times |\mathbf{V}|}$. Therefore, the embedding operation in the top left of Figure 5.3 is written as $\mathbf{y}^Q \cdot (\mathbf{W}^e)^\intercal$.

Gated Linear Units We feed the question through a couple of convolutional layers, which then yield a feature representation of the whole sentence. Similar to Dauphin et al. [27], we use gated convolutions in a residual architecture. In particular, they state that CNNs do not suffer from vanishing gradients like RNNs do. In particular, they explain that ReLUs and gated convolutions have a linear path that allows the gradients to easily flow through. In contrast, a LSTM-style gating mechanism or a tanh activation can cut off the gradient as the activation functions can saturate. More specifically, the output $\mathbf{h}_t$ of an LSTM cell is the element-wise product of a sigmoid activation with a tanh activation (see Equations 2.9 and 2.7). As a consequence the gradients on these paths can easily die out because the derivatives of both tanh and σ approach 0 for positive and negative infinity:

$$\lim_{x \to \infty} \frac{\partial \tanh(x)}{\partial x} = 0 \tag{5.5}$$

$$\lim_{x \to -\infty} \frac{\partial \tanh(x)}{\partial x} = 0 \tag{5.6}$$

$$\lim_{x \to \infty} \frac{\partial \sigma(x)}{\partial x} = 0 \tag{5.7}$$

$$\lim_{x \to -\infty} \frac{\partial \sigma(x)}{\partial x} = 0 \tag{5.8}$$

Furthermore, Dauphin et al. find experimentally that gated convolutions do not require

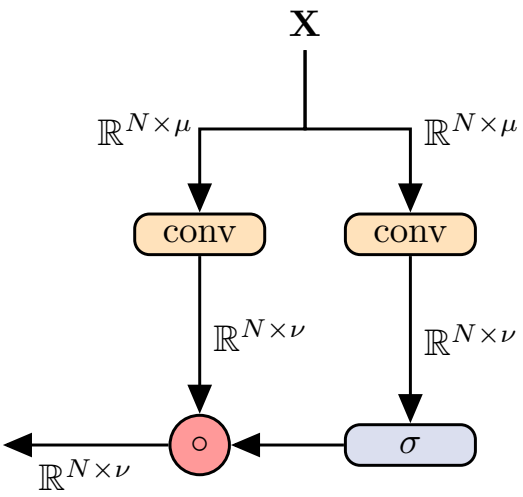


Figure 5.4: A gated linear unit (GLU). Inputs $\mathbf{X}$ are fed through a convolution layer twice with different parameter sets. One convolution layer has a sigmoid activation function while the other has no activation function. Their outputs are multiplied element-wise. N denotes the sequence length of the input, μ is the number of input channels, ν the number of output channels.

forget gates like LSTMs do. However, they discover that allowing the network to control what information should be propagated through the layers via output gates is useful for language modeling and introduce the gated linear unit (GLU). These units learn which information should be forwarded to the next layer and which should be discarded. For each GLU, we learn parameters for two convolution operations ($*$) with the same number of output feature maps μ and calculate its result as

$$\text{GLU}(\mathbf{X}) = (\mathbf{X} * \mathbf{W}^{c1} + \mathbf{b}^{c1}) \circ \sigma(\mathbf{X} * \mathbf{W}^{c2} + \mathbf{b}^{c2}), \tag{5.9}$$

where $\mathbf{X} \in \mathbb{R}^{N\times\mu}$ is the input of the GLU and $\mathbf{W}^{c1} \in \mathbb{R}^{\kappa\times\mu\times\nu}$, $\mathbf{b}^{c1} \in \mathbb{R}^{\nu}$, $\mathbf{W}^{c2} \in \mathbb{R}^{\kappa\times\mu\times\nu}$, $\mathbf{b}^{c2} \in \mathbb{R}^{\nu}$ are learned parameters. Note that κ is the kernel size of the convolution, $\nu = 512$ are the number of output channels and $\circ$ is the element-wise multiplication. Thus, the output of the GLU is of dimensionality $\mathbb{R}^{N\times\nu}$. In the following, we denote the element-wise multiplication with the sigmoid activation ($\circ\,\sigma(\mathbf{X} * \mathbf{W}^{c2} + \mathbf{b}^{c2})$) as "gated activation". We visualize the architecture of a GLU in Figure 5.4.

Gated Blocks We combine two GLUs in a residual-style block in Figure 5.5. More specifically, we depict a gated block in the big green rectangle. A gated block consists of two GLUs and a skip connection over these two GLU layers. The whole figure depicts the question embedding in the top left, which is followed by a GLU and two gated residual blocks. Finally, we use global average-pooling across the question length N to reduce the dimensionality of each question feature representation from $\mathbb{R}^{N\times\nu}$ to $\mathbf{y}^{\tilde{Q}} \in \mathbb{R}^{\nu}$. We set $\nu = 512$ in all GLUs and we change the kernel size κ for different experiments in

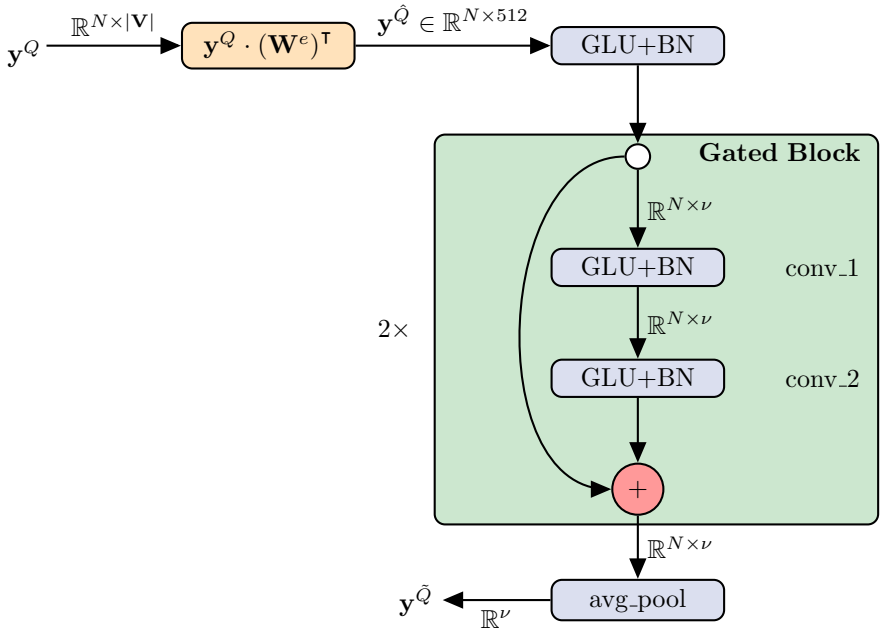


Figure 5.5: We extract a feature representation from the input questions with a convolutional architecture. First, we embed the question $\mathbf{y}^Q$ into a lower-dimensional representation, which we then feed through a GLU followed by two post-activation (the batch normalization BN is appended after each GLU) gated residual blocks. We depict the gated blocks with the green rectangle. After those blocks, we use global average-pooling to get one vector describing the question.

Section 5.6.2.1. Note that the first GLU, the gated blocks, and the average-pooling constitute the question CNN (Q-CNN) in Figure 5.3.

Post-Activation and Pre-Activation Furthermore, inspired by the ResNet-v2 [59] architecture, we utilize two different gated blocks in the experimental part of this chapter. In ResNet-v1 [58], He et al. add a batch normalization after the convolution, but before the ReLU activation. For the second version of ResNet, they change this architecture slightly by first feeding the inputs of a ResNet block through a batch normalization and a ReLU activation before the convolution. This is repeated for the second convolution in a ResNet block before adding the residual. They call these two variants post-activation and pre-activation, respectively.

Applied to our gated block, we add a batch normalization after each GLU for the post-activation Q-CNN just as depicted in Figure 5.5. For the pre-activation Q-CNN, we remove the gated activation (i.e., the convolution with sigmoid activation and element-wise product) from the GLU. Then we add a batch normalization before every GLU followed by the gated activation we just removed. We depict the Q-CNN with pre-activation in Figure 5.6.

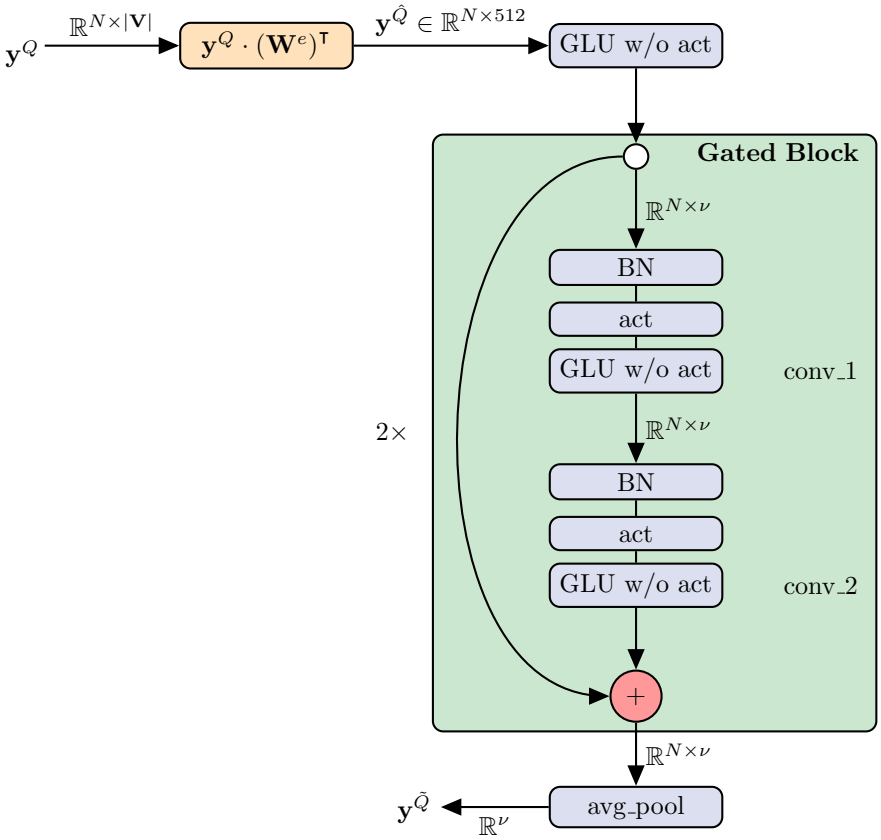


Figure 5.6: The pre-activation Q-CNN. In comparison to the post-activation gated blocks in Figure 5.5, we remove the gated activation (act) from the GLUs. Then we feed our inputs through a batch normalization (BN) and a standalone gated activation before each GLU (without gated activation).

5.5.2 Image Embedding

The second essential part of VQA is to perceive the input image itself. As we already explored in Chapter 4, DCNNs are one of the state-of-the-art architectures for extracting semantical information from an image. Of course, we also utilize a DCNN pretrained on the ImageNet [122] classification task. Because these networks were trained on a large set of everyday scenes, they have already encountered many examples of common objects and activities. The VQA-v2 dataset is based on the MSCOCO dataset and, thus, contains many such objects and scenes. Furthermore, as the Inception-v3 [132] architecture yielded good results for the task of image captioning (see Chapter 4), we also implement it as encoder for our VQA model.

In contrast to image captioning, we make an adaption for the case of image attention (see Section 5.5.3). As we attend to different areas of the image, we do not reduce the spatial dimensions in this case, i.e., we do not average-pool the resulting feature map,

but obtain a feature tensor of size $\mathbb{R}^{K\times 2048}$, where $K = 8 \cdot 8 = 64$. More specifically, this feature tensor is the output of the *Mixed_7c* layer of the Inception-v3 network.

Note that for experiments without image attention, we use a global average-pooled feature representation of size $\mathbb{R}^{2048}$. For experiments with image attention enabled, our feature tensor is of size $\mathbb{R}^{K\times 2048}$.

5.5.3 Image Attention

Similarly to Teney et al. [135], we use a simple form of image attention that attends to certain areas of the image. We do not use features from image regions extracted by a object detection pipeline like Faster-RCNN [119], but only attend to spatial locations on the feature map of the image. The feature map has a size of 8×8 with 2048 dimensions each, which corresponds to $K = 64$ image locations. We denote each location $i \in \{1 \dots K\}$ by $\mathcal{F}_i$ and concatenate it with the question representation $\mathbf{y}^{\tilde{Q}}$ from the Q-CNN. We forward the concatenated question-image feature for every location through a gated tanh layer f_{att} (see Section 5.4) and optimize a parameter vector $\mathbf{w}^{\text{att}} \in \mathbb{R}^{2560}$ that produces attention scores:

$$\mathbf{a}_i = \mathbf{w}^{\text{att}} \cdot f_{\text{att}}([\mathcal{F}_i, \mathbf{y}^{\tilde{Q}}]). \tag{5.10}$$

As we concatenate each location of the feature map ($\in \mathbb{R}^{2048}$) with the question representation ($\in \mathbb{R}^{512}$), we get a feature vector of size $\mathbb{R}^{2560}$ for each location i. Therefore, f_{att} has an input dimension $a = 2560$. We set the output dimension to the same, i.e., $b = 2560$. Note that the calculation of attention scores is essentially the same as in Equation 2.11 except we include the gated tanh layer f_{att}. After we evaluated $\mathbf{a}_i$ for every spatial location i, we can calculate the attention weights $\boldsymbol{\alpha}$ with the softmax ϕ function over all attention scores:

$$\boldsymbol{\alpha} = \phi(\mathbf{a}). \tag{5.11}$$

The previous two operations are depicted by the green attention layer in Figure 5.3. Finally, we can weight each spatial location $\mathcal{F}_i$ of the feature map:

$$\hat{\mathcal{F}} = \sum_{i=1}^{K} \boldsymbol{\alpha}_i \mathcal{F}_i. \tag{5.12}$$

Our attention weights $\boldsymbol{\alpha}_i$ sum up to 1 and we weight each location $\mathcal{F}_i$ by its respective importance, which yields the new feature representation $\hat{\mathcal{F}}$. Thus, this operation (depicted by the red sum symbol in Figure 5.3) reduces the features into the same embedding space as average-pooling does. However, it is designed to weight spatial image locations according to their relative importance instead of weighting each spatial location uniformly. For our experiments without image attention, we use the global average-pooled feature map as described in Section 5.5.2 ($\tilde{\mathcal{F}} = \text{avg\_pool}(\mathcal{F})$).

5.5.4 Multi-Modal Fusion

We need to project the question embedding and image embedding into a joint semantic space to be able to produce answers which are dependent both on the input image and

the question. In order to do so, we first apply the gated tanh layer from [135] (see Section 5.4) to the question features yielded by the Q-CNN. We denote this layer by $f_q(\cdot)$. Because this particular gated tanh layer operates on question features, we give it the subscript q. The question features from the Q-CNN have dimensionality $\nu = 512$, thus, we set the input dimension of $f_q(\cdot)$ to $a = 512$. We want the number of output dimensions to match the input dimension and set $b = a = 512$.

As is common practice in the VQA community [5, 135], we want to fuse our image and question features via the element-wise product into a context vector. To do this, we need to reduce the dimensionality of the image embedding to match the question feature vector's dimensionality of 512. Again, we use the gated tanh layer. In particular, we use $f_v(\cdot)$ with input dimension of $a = 2048$ and output dimension $b = 512$ to reduce the dimensionality of the attended image features $\hat{\mathcal{F}}$ to match the question feature vector. Consequently, we can compute the context vector $\mathbf{c}$ as follows:

$$\mathbf{c} = f_q(\mathbf{y}^{\tilde{Q}}) \circ f_v(\hat{\mathcal{F}}), \tag{5.13}$$

where $\circ$ is the element-wise product. We depict the element-wise fusion operation on the right side in Figure 5.3.

5.5.5 Output LSTM

As we pointed out, most other VQA models from the literature have the drawback of leaving out the less common answers and model the problem of predicting the answer to an image-question pair as a classification task. To do otherwise, we use a classical approach coming from image captioning models (see Chapter 4 and [139, 140]) and machine translation networks. Generally speaking, those kind of models use an encoder-decoder structure, which encodes inputs (e.g., source language sentence or input image) into a fixed-length vector representation and decodes this feature vector into the target sentence.

In contrast to our image captioning model, which we examine in greater detail in Chapter 4, the input is the combination of a question and an input image instead of *only* an image. We encode this combination in the context vector $\mathbf{c}$ and the target is the answer described by a sentence $\mathbf{y}^S = [\mathbf{y}_0^S, \ldots, \mathbf{y}_{N-1}^S]$, where $\mathbf{y}_t^S$ is the word at index t and N the length of the answer. Note that N is different from the question length, which we also denote by N. Also note that the following notations and steps to produce an answer works in the same way as generating an image caption, which we already highlighted in Section 4.2. Our model then maximizes the probability p of the correct answer $\mathbf{y}^S$, given the context vector $\mathbf{c}$:

$$\theta^* = \arg\max_{\theta} \sum_{(\mathbf{c}, \mathbf{y}^S)} \log p(\mathbf{y}^S | \mathbf{c}; \theta), \tag{5.14}$$

where θ are all variables of our model and θ^* is the optimal parameter set. Of course, each answer $\mathbf{y}^S$ consists of a variable number of words and the model is optimized for

each image-question-answer triple (i.e., question and image are contained within the context vector $\mathbf{c}$; the answer is represented by $\mathbf{y}^S$). Therefore, for each training triple, the joint log probability

$$\log p(\mathbf{y}^S|\mathbf{c}) = \sum_{t=0}^{N-1} \log p(\mathbf{y}_t^S|\mathbf{c}, \mathbf{y}_0^S, \ldots, \mathbf{y}_{t-1}^S) \tag{5.15}$$

is maximized. Note that we dropped θ for convenience. Identical to Chapter 4, the first word of each answer always is the start-of-sequence token. Thus, the probability for $p(\mathbf{y}_0^S|\mathbf{c})$ is always 1 and the joint probability in Equation 5.15 is the same for the starting point $t = 0$ or $t = 1$.

Again, we can model the joint probability by using an RNN, where $\mathbf{n}_{t+1}$ is the output of the fully-connected layer after the RNN cell for the input $\mathbf{y}_t^S$. Furthermore, we preprocess the ground-truth answer $\mathbf{y}^S$ by prepending a start-of-sequence (`<S>`) and end-of-sequence (`</S>`) token (identical to Sections 4.2 and 2.2.2). Thus, during optimization, the first token ($\mathbf{y}_0^S$) of each answer is the `<S>` token and the last token ($\mathbf{y}_{N-1}^S$) of the ground-truth answer is `</S>`. Therefore, the original answer length N is increased by 2 in our models. In particular, we use an LSTM network as the decoder network, where we initialize the hidden state at iteration step $t = -1$ (i.e., before we unroll the LSTM cell) with the multi-modal feature representation $\mathbf{c}$ of the question and image. For iteration step $t = 0$, we feed the embedded `<S>` token to the LSTM cell, which outputs $\mathbf{n}_1$. Note that we only feed $\mathbf{c}$ once into the LSTM network and $\mathbf{n}_{t+1}$ is the output of the LSTM cell at iteration step t fed through a fully-connected layer that inflates the hidden state vector to the vocabulary's dimension $|\mathbf{V}|$.

The loss that defines the error signal is then given by the sum over the log likelihoods of the correct word at each position starting at iteration step $t = 0$, where log is the natural logarithm applied element-wise to the vector $\mathbf{n}_{t+1}$:

$$L(\mathbf{c}, \mathbf{y}^S) = -\sum_{t=0}^{N-2} [\log \phi(\mathbf{n}_{t+1})] \cdot \mathbf{y}_{t+1}^S. \tag{5.16}$$

Here $\cdot$ is the dot-product and $\mathbf{y}_{t+1}^S$ is the one-hot encoded ground-truth vector for the correct word at iteration step t.

5.6 Experiments

5.6.1 Training Configuration

In all experiments, we use stochastic gradient descent (SGD) with a base learning rate of $\eta_b = 2.0$. The base learning rate applies to the decoder LSTM network, while we use a reduced learning rate for other parts of our model. For the question encoder CNN, the image attention and the multi-modal fusion, we use learning rates $\eta_{\text{Q-CNN}} = 0.005 \cdot \eta_b$, $\eta_{\text{att}} = 0.05 \cdot \eta_b$ and $\eta_{\text{fusion}} = 0.05 \cdot \eta_b$, respectively. At a varying number of epochs, we halve the learning rate. When fine-tuning the Inception-v3 network in an end-to-end

fashion, we set all learning rates to $\eta = 0.0005$. We use a batch size of 128 in most of our experiments and reduce the batch size to 32 when fine-tuning the Inception-v3 network. All experiments were conducted on a single NVIDIA Titan X Pascal GPU.

For training, we use each image-question-answer triple as one training example. Note that we sample multiple training examples from one image-question pair with multiple possible answers to model the uncertainty between the annotators. Thus, we get a slightly richer training signal because the model learns that multiple answers could be correct.

Depending on the model, we halve the learning rate 2 to 4 times and train for 2 to 4 epochs per learning rate. In the fine-tuning stage, we train until convergence for a total of up to 35 epochs. In addition, we increase the number of epochs per learning rate to 6. On our GPU, training in the first stage with pre-computed Inception-v3 features takes 2 to 4 days. The fine-tuning stage takes 8 to 10 additional days.

We implement our VQA model in TensorFlow 1.0 [3] and use the prebuilt *Inception-v3* DCNN from the *slim* module. Our model shares code with the Show and Tell [139] model, which is publicly available as a TensorFlow implementation.

5.6.2 Results

In the following section, we conduct a series of experiments to find an architecture producing best results for our scenario (see Table 5.1). We review every part of our architecture (see Section 5.5) and conduct extensive ablation experiments in order to justify our hyperparameter selection. All models build upon an Inception-v3 DCNN pretrained on the ILSVRC 2012 [122] corpus. We precompute the Inception-v3 features of the input images, because every image is used multiple times and, thus, training is sped up. Unless stated otherwise, we report accuracies on our validation split of the VQA-v2 dataset.

5.6.2.1 Question Feature Extraction

As we described in Section 5.5.1, we use a convolutional architecture to extract a single feature vector from a given question. In addition to the post-activation layers, we also conduct an experiment (see model #1) with a pre-activation layer (similar to ResNet-v2 [59], see Section 5.5.1), which performs a little worse in our case. In comparison to model #2, we loose 0.4 % in performance on all questions. Only the Y/N questions improve by 0.12 % while we loose 1.16 % and 0.67 % performance for number questions and other questions, respectively.

We also compare classification accuracies for different numbers of gated residual blocks (see the column *# blocks* in Table 5.1). Here we see that a larger number of residual blocks (4) does not improve performance in comparison to our choice of using only 2 blocks (compare model #2 vs. model #4, which achieve similar performance).

Furthermore, in the first column of the question feature extraction columns, we show our different choices for the kernel size κ of the question convolutions. As a comparison, we also extract the question features with a simple LSTM network (one LSTM cell with

Table 5.1: Studies on our validation split of the VQA-v2 dataset. We analyze different combinations of hyperparameters in our model. The first column states the model #, *Att* stands for Attention, *FT* for fine-tuning of all parameters (including the *Inception-v3* image feature extraction DCNN) and *VG* if we extended our dataset with the Visual Genome dataset. The columns under *Feature Extraction* $\mathbf{y}^Q$ state the configuration of the question feature encoder. *MM-Fuse* describes the kind of multi-modal fusion used and *Decoder* specifies whether we used an LSTM or a fully-connected (FC) layer for generating the answers. The columns for validation performance give the accuracies on our validation split (10 % of the VQA-v2 validation split).

				Feature Extraction $\mathbf{y}^Q$				MM-Fuse	Decoder	Validation Performance			
#	Att	FT	VG	κ	# blocks	act	$f_q(\tilde{\mathbf{Q}})$			All	Y/N	Num	Other
1	—	—	—	4	4	pre	—	eltwise	LSTM	52.02	71.69	34.82	41.38
2	—	—	—	4	4	post	—	eltwise	LSTM	52.42	71.57	35.98	42.05
3	—	—	—	4	1	pre	—	eltwise	LSTM	50.65	69.07	31.34	41.62
4	—	—	—	4	2	post	—	eltwise	LSTM	52.44	71.52	35.48	42.28
5	—	✓	—	4	2	post	—	eltwise	LSTM	54.67	72.63	37.04	45.55
6	—	—	—	5	2	post	—	eltwise	LSTM	53.00	72.57	35.44	42.53
7	—	—	—	—	LSTM	—	—	eltwise	LSTM	41.48	63.69	28.18	27.95
8	✓	—	—	5	2	post	—	eltwise	LSTM	53.54	73.33	34.81	43.34
9	✓	—	—	5	2	post	✓	eltwise	LSTM	53.71	73.62	35.12	43.38
10	✓	—	✓	5	2	post	✓	eltwise	LSTM	53.97	73.69	36.83	43.42
11	✓	✓	—	5	2	post	✓	eltwise	LSTM	**56.62**	**75.45**	**39.33**	**46.79**
12	✓	—	—	5	2	post	✓	eltwise	FC	49.16	72.82	33.41	35.26
13	✓	—	—	5	2	post	—	MFB	LSTM	51.94	72.44	33.54	41.13
14	✓	—	✓	5	2	post	—	MFB	LSTM	52.20	71.45	35.40	41.90

512 units), which decreases the validation accuracy about an absolute of 10 %. Note that we use gated linear units (GLU; see Section 5.5.1) for every model except model #7. We also see that model #6 with $\kappa = 5$ performs a little better than model #4 with $\kappa = 4$. Therefore, as a final choice of parameters for the question feature extraction part of our model, we choose to use 2 post-activation residual blocks (see Figure 5.5) and a kernel size of $\kappa = 5$ for the GLUs.

5.6.2.2 Image Attention

In the next step, we extend our model by implementing a simple image attention model (see Section 5.5.3). We already mentioned earlier that we precompute the feature maps ($\in \mathbb{R}^{8\times8\times2048}$) for the encoder *Inception-v3* network. In particular, we store the direct output of the *Mixed_7c* layer. For the models that make use of image attention, we do not average-pool those feature maps and attend to the output feature maps ($\in \mathbb{R}^{K\times2048}$, $K = 8 \cdot 8 = 64$).

In Table 5.1, we mark models with image attention by a checkmark in the *Att* column. In the validation results, we notice that our model benefits from adding this simple form of attention. More specific, if we compare models #6 and #8, we see that enabling image attention improves the overall accuracy by 0.54 %.

Finally, if we also optimize parameters of the *Inception-v3* DCNN, we get an additional improvement of about 3% on the overall accuracy (see model #11 vs. #9). We also see that fine-tuning has a larger impact on number questions and other questions than on Y/N questions.

5.6.2.3 Multi-Modal Fusion

We explored a simple and effective way of combining features from different modalities in Section 5.5.4. In particular, we combine the question and the image features into a joint semantic space before generating an answer. More specifically, in Equation 5.13, we fuse the image and question features via element-wise multiplication.

In addition, applying the non-linear layer $f_q(\mathbf{y}^{\tilde{Q}})$ (see Section 5.5.4) to the question features ($\mathbf{y}^{\tilde{Q}}$) before fusion also has a small positive effect. In comparison to model #8, model #9 improves the overall accuracy, Y/N question accuracy, number question accuracy and other question accuracy by 0.17 %, 0.29 %, 0.31 % and 0.04 %, respectively. These are minor improvements and could well be due to statistical uncertainty. However, as it does not hurt performance, we apply the non-linear layer to the question features of our final model. Besides fusing the question and image features simply by calculating the element-wise product, we also implemented the multi-modal factorized bilinear pooling (MFB) approach by Yu et al. [155], which was ranked second in the VQA-v2 open-ended challenge. They project the image features and question features into a larger embedding space, i.e., the embedding space is $\in \mathbb{R}^{5 \cdot 512}$. These larger embedding spaces are combined via an element-wise product and then reduced to $\mathbb{R}^{512}$ via sum-pooling, i.e., blocks of 5 values are summed up. We first calculate the element-wise signed square-root of this vector

$$z \leftarrow \text{sgn}(z) \cdot |z|^{0.5} \tag{5.17}$$

and then normalize it with the L2-norm ($\|\cdot\|_2$)

$$z \leftarrow \frac{z}{\|z\|_2}. \tag{5.18}$$

Yu et al. assume that MFB supports the exploitation of more complex correlations between multi-modal features. As we can see in the *MM-Fuse* column of Table 5.1, using the MFB approach (models #13 and #14) does not increase overall accuracy in our case and we stick with using the simple fusing approach described in Section 5.5.4.

5.6.2.4 Classical VQA Approach

As we use an LSTM network as the decoder instead of a fully-connected layer (FC) with a softmax cross-entropy loss function, we still want to compare our architecture to the classical VQA classification approach (see Section 5.4).

For this reason, we extended our model to be able to classify the 3000 most common answers of the train split. When we evaluate this approach (model #12) on our validation set, accuracies drastically decrease compared to the LSTM decoder (53.71 vs. 49.61). This may be an indicator that the classification approach performs worse than an LSTM decoder network.

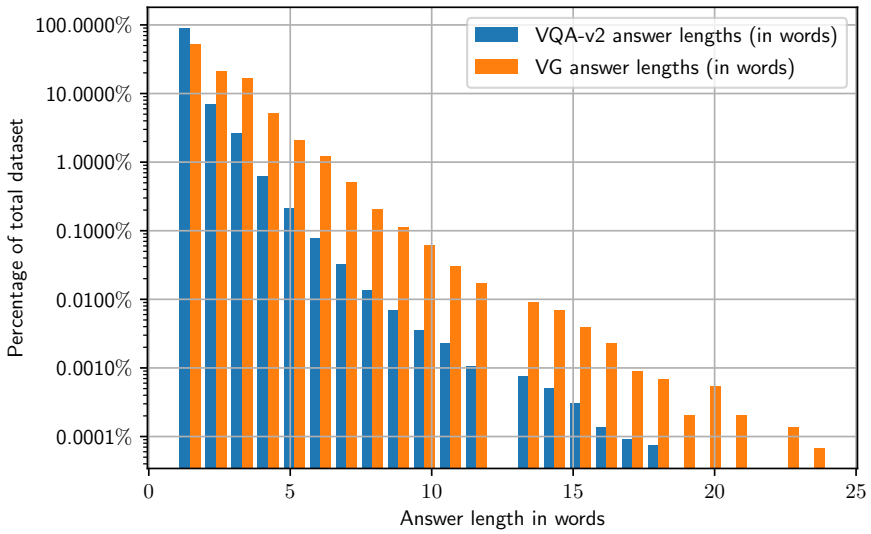


Figure 5.7: The VQA-v2 dataset contains many short answers (89.41% and 6.92% of all answers have a length of 1 and 2 words, respectively), while the VG dataset contains more answers with more words (52.47%, 21.86% and 16.70% of all answers are of length 1, 2 and 3 words, respectively).

5.6.2.5 Dataset Extension

We also examine whether the validation accuracy can be improved by extending the dataset with the Visual Genome (VG) dataset. Because we do not evaluate on the VG dataset, we include the whole dataset into our train split. Models trained with the extended dataset are indicated by the VG column in Table 5.1. In contrast to models that use the classification approach [135, 155], we include the whole VG dataset. These other works mostly exclude dataset samples that do not conform to the top answers of the VQA-v2 dataset, because the classification-based models are limited by a fixed number of possible answers (i.e., a fixed number of output neurons in the classification layer). In both experiments (models #10 and #14), in which we extended our dataset by VG, we notice a slight improvement in the scores. This could be attributed to the fact that the VG dataset contains longer answers than the VQA-v2 dataset, which mainly consists of one to two word answers. We hypothesize that our model is therefore able to produce richer answers. We depict the distribution of answer lengths in Figure 5.7.

5.6.3 Less Common Answers

We use an LSTM network to generate answers, thus, in contrast to classification models, we are able to generate answers which are not part of the 3000 most common answers. Our train set contains 190,677 unique answers and the 3000 most common answers (MCA) comprise 90.5 % of all answers in the dataset. Thus, classification models with 3000 logits are unable to produce 9.5 % of the less common answers (LCA) of the train

Table 5.2: Fractions of unique answers generated by our models. LCA are less common answers left out altogether by classification models. MCA are most common answers, which are the only answers generateable by classification models. New answers are newly generated sentences by our models not contained in the train split. The LCA accuracy describes the percentage of correctly generated answers out of the LCA set, i.e., these are correct answers given by our model that classification models are not able to produce.

#	LCA	LCA accuracy	MCA	new answers
10	14.97 %	11.91 %	62.73 %	22.30%
14	14.70 %	9.47 %	63.62 %	21.68 %
11	19.49 %	8.10 %	69.74 %	10.77 %
8	24.52 %	9.91 %	64.22 %	11.26 %

set. In contrast, our LSTM model is able to produce every answer and even new answers. In an experiment, we determined the fraction of MCA, LCA and new answers of all uniquely generated answers of some of our models. Our models generate between 14.70 % and 24.52 % LCA and between 11.26 % and 22.30 % new answers. With the official evaluation tool, we also find that the accuracy of LCA is up to 11.91 %. Note that these answers cannot be generated by classification-based models. We visualize these findings in Table 5.2. In a qualitative analysis we depict some of the new answers generated by our model in Figures 5.8 and 5.9. Figure 5.8 shows correct answers not detected by the official evaluation script. Figure 5.9 depicts incorrect answers, e.g., (a) and (b) show answers being too short (due to the short answer bias of the dataset) and (c) and (d) show wrong answers, even though the answers are plausible given the scene.

Table 5.3: One of our models (model 10) compared to other published models on the test-dev and test-std dataset splits.

Model / Dataset	VQA-v2 test-dev				VQA-v2 test-std			
	All	Y/N	Num	Other	All	Y/N	Num	Other
MCB [38, 45]	-	-	-	-	62.27	78.82	38.28	53.36
d-LSTM+n [97, 45]	-	-	-	-	54.22	73.46	35.18	41.83
Single Network [135]	**62.07**	**79.20**	**39.46**	**52.62**	**62.27**	**79.32**	**39.77**	**52.59**
#10 (Ours)	56.32	75.64	36.21	44.36	56.68	76.02	36.56	44.46

5.6.4 Performance on the Official Test Set

In Table 5.3, we compare our model to other models on the test-dev as well as the test-std dataset split. We compare our model to the MCB [38] approach as reported in [45] and the single network model by Teney et al. [135]. In comparison to those models, we did not train an ensemble of networks and did not implement attention on image regions with an

(a) *Q: Where is the man?*
A: in a restaurant

(b) *Q: Where are the trees?*
A: behind the elephants

(c) *Q: Where is the bus?*
A: on the street

(d) *Q: Where is the red carpet?*
A: on the ground

Figure 5.8: Images associated with questions from the VQA-v2 dataset [45] and generated answers by our model. All answers shown are new ones not contained in the training set. Figures (a) - (d) show correct answers not detected by the official evaluation script.

(a) *Q: Who wears a cowboy hat?*
A: the man on the

(b) *Q: What is the woman doing?*
A: feeding the

(c) *Q: Where is the photo taken?*
A: in a kitchen

(d) *Q: Where is the boy looking?*
A: at the beach

Figure 5.9: Images associated with questions from the VQA-v2 dataset [45] and generated answers by our model. All answers shown are new ones not contained in the training set. In this Figure, we show wrong answers. Especially, (a) and (b) show sentences, where the end-of-sentence token was generated to early (dataset bias of short answers). (c) and (d) show wrong answers.

object detection model. Nevertheless, our models surpass the *d-LSTM+n* approach but are worse than the best scoring approaches. However, as our focus lays on generating less common answers and new answers, our model cannot be fairly compared against these approaches, which are trained and evaluated on a classification task. Furthermore, our model can produce more complex answers due to the LSTM decoder network and can be seen as a viable approach for more complex VQA datasets to come.

5.7 Summary

In this chapter, we presented an end-to-end trainable architecture for the VQA task. In contrast to other approaches, we use an LSTM network to generate the answers for the image-question pairs. Furthermore, we conducted a series of experiments to find a good combination of hyperparameters for our architecture. Competing approaches model the VQA task as a classification task, where a fixed number of most common answers are the possible classes, hence, they are not able to produce less common answers. With our approach, we were able to generate such less common answers with an accuracy of 11.91% and also showed that our model generates new answers, which are not contained in the training set. Some of these produced sentences answer questions correctly, which the accuracy-based evaluation script cannot asses correctly. We also find out that the ground-truth answers of current VQA datasets are biased towards a short number of words, i.e., more than 90% of the answers have a maximum length of two words. For datasets containing more complex answers, our model is more suitable than models using the classification approach.

6 Hierarchical Language Generation of Doctors' Reports for Chest X-Ray Images

So far, we have discussed the image captioning and visual question answering tasks. We tackled both tasks with an RNN network that generates descriptions in a recurrent way. In this Chapter, we discuss the problem of generating doctors' reports for chest X-ray images. This task is quite similar to traditional image captioning (see Chapter 4). However, there are two obstacles: First, a report consists of a paragraph, i.e., multiple sentences instead of one. Second, in addition to longer textual descriptions, we have a lack of annotated data as medical data is highly regulated and not available on a large scale. In this chapter, we will deal with those two fundamental problems in automatic report generation for chest X-ray images.

We base this chapter on following publication:

> **Addressing Data Bias Problems for Chest X-ray Image Report Generation** [51], Philipp Harzig, Yan-Ying Chen, Francine Chen and Rainer Lienhart, 30th British Machine Vision Conference 2019, Cardiff, Wales, September 2019.

6.1 Motivation

Deep convolutional neural networks in combination with recurrent neural networks are a common architecture used to automatically generate descriptions of images. These recent advances in automatic description generation have not left other areas such as medical research untouched. For example, Wang et al. released a dataset [145] which contains thousands of chest X-ray images associated with up to 14 disease labels. Demner-Fushman et al. [28] released an anonymized dataset which contains 7470 chest X-ray images associated with doctors' reports and tag information specifying medical attributes. However, annotating these domain-specific datasets requires expert-knowledge and cannot be achieved cost-efficiently like more common datasets. In addition, medical data is connected with high privacy concerns and also regulated, e.g., by the Health Insurance Portability and Accountability Act (HIPAA).

Therefore, only a limited amount of data is publicly available. Especially, there is only one public dataset [28] that connects chest X-ray images with medical reports. In this dataset, there are far more sentences describing cases without medical conditions than cases with medical conditions. We refer to these two types as normal and abnormal cases, respectively. Thus, most machine learning models are biased towards generating normal results with a higher probability than abnormal results. However, abnormalities are more important and more difficult to detect given the small number of examples.

In this chapter, we address this issue with a new architecture, which can distinguish between generating sentences describing abnormal or normal findings.

Furthermore, common machine translation metrics such as BLEU [108] may not be the best choice, when even one word - such as 'no' - contained in a paragraph can make a huge difference for the indication and findings. Also, calculating these metrics over an imbalanced dataset raises the issue that sentences about normal cases are over-weighted and results in less diversity in the generated reports. We examine these issues of common machine translation metrics when used on a dataset of medical reports in this chapter.

Here, we address the data bias problems for chest X-ray image report generation by breaking the task down into the following sub-tasks:

1. We annotate each sentence of a public dataset with abnormal labels.

2. We use these labels to train a new hierarchical LSTM with a dual word LSTM. A dual word LSTM means that we implement two word LSTMs with different parameter sets. With an abnormality prediction module, we determine whether a sentence will describe an abnormal finding or a normal one. Then, we can choose one of the two word LSTMs to generate the respective sentence, i.e., one word LSTM is responsible to create sentences that describe abnormalities. The other one generates sentences that identify normal cases. This helps us to reduce problems associated with the data bias of the chest X-ray dataset.

3. We analyze the correlation between machine translation metrics and the variability in generated reports and find that a high score calculated over a dataset does not necessarily imply a result to rely on.

6.2 Related Work

Chest X-Ray Datasets In the field of combining computer vision and machine learning with medical chest X-ray images, Wang et al. [145] published the large Chest-Xray14 dataset, which includes a collection of over 100,000 chest X-rays annotated with 14 common thorax diseases. This dataset has been widely [88, 116, 146] used for predicting and localizing thorax diseases. The disease labels of this dataset were automatically extracted from the doctors' reports. However, the doctors' reports are not available publicly. Demner-Fushman et al. [28] are the first to release a rather large anonymized dataset called *Indiana University chest X-ray collection* consisting of chest X-rays paired with doctors' reports, indications and manually annotated disease labels. We use this dataset in our work.

Hierarchical Language Generation Automatically generating captions from images is a well-researched topic. Nowadays, most architectures use an encoder-decoder structure, where a DCNN is used to encode images into a semantic representation and a decoder generates the most likely sentence given this image representation. Older methods (e.g., [139, 73, 71], Chapters 4 and 5) utilize recurrent neural networks in the decoder. Krause

et al. [77] extended on these works by introducing a hierarchical LSTM structure to generate longer sequences for describing an image with a paragraph. In this chapter, we define a paragraph to be the concatenation of multiple sentences. For example, an image description consisting of 5 consecutive sentences is called a paragraph. In their work, Krause et al. argue that LSTM cells help to catch long-term dependencies with their gating mechanism, but present hierarchical recurrent networks as a suitable alternative for longer sequences like a paragraph. In this approach, different parts of the model operate on different parts within a textual description, i.e., each sentence of a paragraph is generated with a different precondition. This precondition vector is generated by another LSTM.

Language Generation for Doctors' Reports Jing et al. [70] use a hierarchical LSTM to generate doctors' reports with multiple sentences, and use a co-attention mechanism that attends to visual and semantic features, which are generated by medical tags annotated within the Indiana University chest X-ray collection [28]. Li et al. [87] describe a hybrid reinforced agent that decides during the process of creating every single sentence if it should be retrieved from a template library or generated in a hierarchical fashion. Instead of a hierarchical model, Xue et al. [153] use a bidirectional LSTM to encode semantic information of the previously generated sentence as guidance for an attention mechanism to generate an attentive context vector for the current sentence. The authors also state that standard sentence evaluation metrics like BLEU [108], METEOR [10], ROUGUE [89] and CIDEr [138] are not designed for medical report generation tasks and, therefore, present the keywords accuracy (KA) metric. This metric is the ratio of correctly generated keywords to all keywords within the ground-truth. Wang et al. [146] presented a joint framework, which simultaneously predicts one of 14 diseases and generates a report on the Chest-Xray14 dataset. However, the textual annotations are not available to the public as of yet. They use a single LSTM that produces a report conditioned on the previous hidden state, the previously generated words and image features extracted by a CNN. In a more recent work, Li et al. [82] use a graph transformer to decompose visual features into an abnormality graph, which is decoded as a template sequence and paraphrased into a generated report. Our method is based on a hierarchical LSTM structure [77, 70] and introduces an abnormal sentence predictor in combination with a dual word LSTM for separately generating sentences that describe abnormalities and normal cases. A dual word LSTM means that we implement two word LSTMs with different parameter sets. One of those two word LSTMs is selected if a sentence that describes an abnormality is more likely to be generated. If it is more likely that the sentence should describe a normal case, we choose the other LSTM. In contrast to [87, 82], we do not use any templates for sentence generation.

6.3 Dataset

We make use of the Indiana University chest X-ray collection [28] (IU chest X-ray dataset), which contains 7470 chest X-Ray images associated with multiple annotations.

Each annotation includes a textual report written by a domain expert (i.e., a doctor),

UID: CXR1001
Impression: Diffuse fibrosis. No visible focal acute disease.
Indication: dyspnea, subjective fevers, arthritis, immigrant from Bangladesh
Findings: Interstitial markings are diffusely prominent throughout both lungs. Heart size is normal. Pulmonary XXXX normal.
Problems: Markings; Fibrosis
MeSH: Markings/lung/bilateral/interstitial/ diffuse/prominent; Fibrosis/diffuse

Figure 6.1: An example from the IU chest X-ray dataset [28], which shows an abnormal case with findings. We highlight the sentences with our human abnormality annotation: In particular, sentences that represent normal observations are written in blue and sentences describing abnormalities are written in green. We define our ground-truth doctor's report to be the concatenation of the impression and the findings. In this case, our ground-truth doctor's report (i.e., paragraph) consists of 5 subsequent sentences.

which includes an indication, findings and an impression in a textual form. Therefore, we call these textual reports *doctors' reports*. Identical to other works [70, 146] in this area, we create our ground-truth doctors' reports by concatenating the sentences from the impression and findings. Both impression and findings can consist of multiple sentences. Thus, our final ground-truth doctors' reports also consist of multiple sentences. The original work that generates multiple subsequent sentences with a hierarchical model [77] for a non-medical dataset refers to multiple concatenated sentences as a paragraph. In order to keep the terminology conistent, we also refer to the concatenated ground-truth doctors' reports as paragraphs in this chapter.

Furthermore, the dataset is annotated with MTI (medical text indexer) encodings. The MTI encodings are automatically extracted keywords from the indication and findings. We identify 121 unique MTI labels in the dataset and use these labels for an

Table 6.1: Distinct sentences sorted top-down by their number of appearances in the dataset. We denote the absolute count of the corresponding distinct sentence with $\tilde{c}$.

rank	$\tilde{c}$	**sentence**
1	947	no acute cardiopulmonary abnormality
2	698	the lungs are clear.
3	523	no pneumothorax.
4	451	lungs are clear.
5	394	no acute cardiopulmonary findings.
...	...	...
8018	1	mild right basilar airspace consolidation may ...
8019	1	calcified granuloma is seen in the left medial...
8020	1	old rib fractures healed.
8021	1	negative for pneumothorax pneumomediastinum or...
8022	1	it is unchanged compared to a for the abdomen ...

additional training signal. Additionally, the authors manually annotated the images with MEDLINE® Medical Subject Headings® (MeSH®).

To summarize, this public dataset contains 3955 narrative reports, each associated with MeSH tags, MTI labels and two views of the chest, i.e., a posteroanterior (PA) and a lateral view. Most of the narrative reports are associated with those two views but some narrative reports only come with one view. Therefore, our dataset has nearly double the number (7470) of images as narrative reports. We show one example from this dataset in Figure 6.1.

It is very difficult for a machine learning model to properly learn the task of generating full paragraphs of doctors' reports from this small number of examples. Especially, we notice that most of the reports consist of repeating and very similar sentences, which are of descriptive nature and do not describe abnormalities and diseases. In Table 6.1, we list the count $\tilde{c}$ of distinct sentences within the doctors' reports, i.e., all sentences that appear at least once in the dataset sorted top-down with most frequent sentences listed on top. We notice a long-tail distribution with abnormal sentences often only occurring with an absolute count of $\tilde{c} = 1$ within the whole dataset. In fact, 6290 of the 8022 distinct sentences have an absolute count of $\tilde{c} < 3$. A machine learning model is optimized to generate the most probable doctor's report given the input image. However, most of the images in the dataset depict normal cases and it is difficult to generate accurate reports for abnormal cases. Considering that identifying abnormalities and diseases is the most crucial part in this problem domain, we want to address the data bias problems for chest X-ray image report generation.

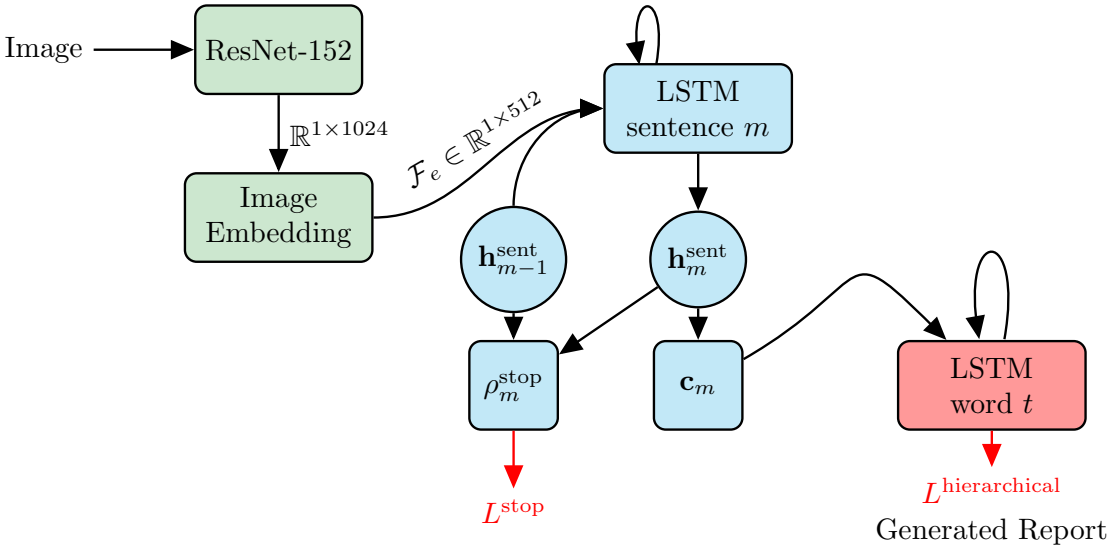


Figure 6.2: A basic HLSTM model for generating paragraphs of sentences. The green boxes show the CNN image encoder and the image embedding. The blue part and red parts show the hierarchical LSTM. Our final model (see Figure 6.3) is built upon this base model.

6.4 Hierarchical Base Model

One difficulty in generating doctors' reports for the IU chest X-ray dataset [28] lies in the nature of this dataset. That is, the doctors' reports are the concatenation of the impressions and findings and, therefore, consist of multiple sentences. Traditional RNNs and even LSTMs have the downside that they tend to not work well on longer sentences or even paragraphs as long-range dependencies (see Section 2.2.5) cannot be memorized well. Krause et al. [77] introduce a hierarchical LSTM that builds two hierarchy levels of LSTM cells on top of one another. Thus, many works that try to generate doctors' reports for the IU chest X-ray dataset include a hierarchical LSTM into their model. The hierarchical model separates the generation of different sentences in a paragraph from the generation of words: We depict a simple hierarchical LSTM baseline model for generating reports of chest X-ray images in Figure 6.2. On the top left, we forward the input image through a ResNet-152, which was pretrained on the ImageNet dataset [122]. We average-pool the resulting feature map over the spatial dimensions which yields a single feature map $\mathbb{R}^{1\times1024}$ per image. Next, we project the feature map into a lower dimension ($\mathcal{F}_e \in \mathbb{R}^{1\times512}$) with a fully-connected image embedding layer. This feature vector is used to initialize an LSTM cell at iteration step $m = -1$, i.e., we call an LSTM cell once with the embedded image features, before unrolling the cell. We call this first LSTM cell *sentence LSTM*, which is responsible to generate hidden states $\mathbf{h}^{\text{sent}}_m$ for every iteration step. An iteration step of the sentence LSTM represents a single sentence from the doctor's report paragraph. For example, if a doctor's report consists of $M = 6$ sentences, we unroll the sentence LSTM cell six times. Particularly, we iterate from

index $m = 0$ to $m = 5$. We visualize this unrolling by the loop arrow on top of the sentence LSTM cell on the center top in Figure 6.2.

We first feed the sentence LSTM's hidden state through a fully-connected layer to generate a topic vector $\mathbf{c}_m$ (see Section 6.5.1 for more details). The topic vector then represents the context of a particular sentence m. We then initialize an LSTM cell with this context/topic at iteration step $t = -1$, i.e., before we unroll this second LSTM cell. We denote the second LSTM cell by *word LSTM* because it generates the words for a sentence. We show the word LSTM in red on the right side of the figure. The word LSTM is trained with a softmax cross-entropy loss (see Section 6.5.1) in similar fashion to Chapters 4 and 5.

Second, we feed the sentence LSTM's hidden state and the hidden state from the previous iteration ($\mathbf{h}^{\text{sent}}_{m-1}$) through a second fully-connected layer with one output neuron. This layer is called stop prediction and its sole purpose is to predict, whether the current sentence m should be generated or not. In particular, the stop prediction signals whether we should generate another sentence and add it to the paragraph or if we should stop generating further sentences.

6.5 Dual Word LSTM Medical Image Report Generation

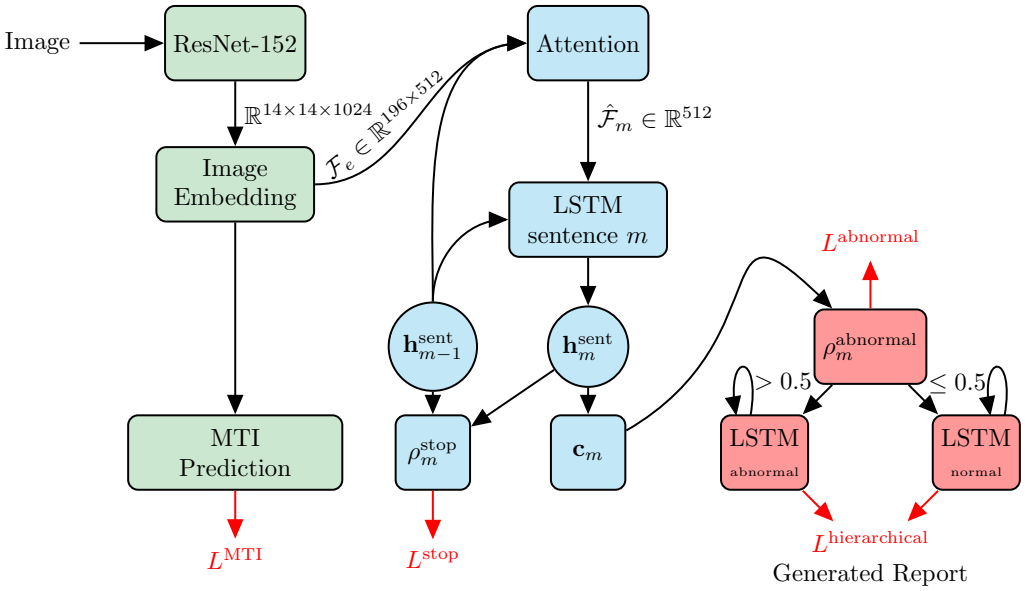


Figure 6.3: Our dual word LSTM model. The green boxes show the CNN image encoder, the image embedding and the MTI tag prediction. The blue part depicts the sentence LSTM, i.e., the topic generator $\mathbf{c}_m$ and the stop prediction ρ^{stop}_m. The red part shows an abnormal sentence predictor (ρ^{abnormal}_m) and dual word LSTMs for generating abnormal and normal sentences, respectively. Red font depicts the different loss functions.

We depict our extended model architecture in Figure 6.3. The input to our model are single images, i.e., either the lateral view or PA view of a chest X-ray image. We use the *res4b35* feature map $\mathcal{F} \in \mathbb{R}^{14\times14\times1024}$ of the ResNet-152 [59] as our image features. Similar to the base model, we include an image embedding layer that embeds the whole ResNet-152 feature map into a lower dimensional space $\mathcal{F}_e \in \mathbb{R}^{14\times14\times512}$ for further use. In contrast to the base model, we do not embed the average-pooled feature map but the full feature map with spatial information. Concurrently, we reduce the channel dimension of the feature map from 1024 to 512 in order to match the dimensionality of our embedded tokens. The embedded feature map is then reshaped into a feature map of shape $\mathbb{R}^{196\times512}$ allowing a soft attention mechanism to attend to 196 different spatial locations. Unless noted otherwise, all embedding and hidden dimensions are set to 512 in our model. We see that the dual word LSTM model in Figure 6.3 shares some resemblance with the base model. However, we added an attention mechanism, an MTI label prediction module, and extended the word LSTM by a second LSTM cell. These dual word LSTM cells are responsible for generating sentences that describe abnormalities and normal cases, respectively. In the following, we describe our model in more detail and highlight our additions to the base model.

6.5.1 Hierarchical Generation with Dual Word LSTMs

Even though LSTMs were designed to combat the issue of forgetting long-term dependencies, they still have problems keeping information for very long time-periods, e.g., over multiple sentences. Krause et al. [77] address this problem by splitting the generation into a hierarchical LSTM, which consists of two independent LSTMs. The sentence LSTM's sole purpose is to generate topic vectors, which in turn are used for the initialization of the word LSTM. The word LSTM then generates a single sentence conditioned on the topic vector. Jing et al. [70] extend the hierarchical LSTM for generating medical reports from chest X-ray images. We also use a hierarchical concept with an architecture that differs from Jing et al. [70]. For example, our model is trained in a multi-task setting: In particular, we add a multi-label classification objective on MTI tags (see Section 6.5.4) as a second task to our model. Furthermore, in contrast to Jing et al. [70], we do not use the co-attention mechanism in our model.

Sentence LSTM We initialize the sentence LSTM on image features extracted by the encoder CNN. However, we use a soft attention mechanism (see Section 2.3.1) $\hat{\mathcal{F}}_m = \text{Attention}(\mathcal{F}_e, \mathbf{h}^{\text{sent}}_{m-1})$ to attend to different spatial areas within the feature map conditioned on the sentence LSTM's hidden state $\mathbf{h}^{\text{sent}}_{m-1}$ of the preceding sentence. For the sentence with index $m = 0$, we set $\mathbf{h}^{\text{sent}}_{m-1}$ to be the zero state, i.e., a vector with only zeros. In subsequent sentences, we use the corresponding preceding hidden state, which we remember from the previous iteration of the sentence LSTM. In order to generate the topic vector for sentence m, we apply the sentence LSTM to the attentive image features $\hat{\mathcal{F}}_m$ to get an intermediate hidden state $\mathbf{h}^{\text{sent}}_m$ for the current sentence and feed

it through a fully-connected layer:

$$\mathbf{c}_m = \mathrm{ReLu}(\mathbf{W}^{\mathrm{sent}} \cdot \mathbf{h}_m^{\mathrm{sent}}). \tag{6.1}$$

The fully-connected layer has no bias terms, a ReLU activation function and generates a topic vector $\mathbf{c}_m$. $\mathbf{W}^{\mathrm{sent}} \in \mathbb{R}^{512\times512}$ is the learnable parameter matrix of the fully-connected layer.

Stop Prediction We also use the sentence LSTM's current and previous hidden state to predict if we should continue generating sentences ($\mathbf{y}_m^{\mathrm{stop}} = 0$) or stop generating them ($\mathbf{y}_m^{\mathrm{stop}} = 1$). The stop prediction ($\rho_m^{\mathrm{stop}}$) consists of three fully-connected layers without biases. Both the sentence LSTM's previous and current hidden state are forwarded through two parallel fully-connected layers without activation functions. The results are summed up and fed through the third fully-connected layer with tanh activation. The stop prediction (ρ_m^{stop}) can be calculated as follows:

$$\rho_m^{\mathrm{stop}} = \mathbf{W}^{\mathrm{stop}} \cdot \tanh(\mathbf{W}^{\mathrm{stop},m-1} \cdot \mathbf{h}_{m-1}^{\mathrm{sent}} + \mathbf{W}^{\mathrm{stop},m} \cdot \mathbf{h}_m^{\mathrm{sent}}), \tag{6.2}$$

where $\mathbf{W}^{\mathrm{stop},m-1} \in \mathbb{R}^{512\times512}$, $\mathbf{W}^{\mathrm{stop},m} \in \mathbb{R}^{512\times512}$ and $\mathbf{W}^{\mathrm{stop}} \in \mathbb{R}^{1\times512}$ are parameter matrices. We train the stop prediction with a sigmoid cross-entropy loss

$$L^{\mathrm{stop}} = -\sum_{m=0}^{M-1} \left(\mathbf{y}_m^{\mathrm{stop}} \cdot \log\left[\sigma(\rho_m^{\mathrm{stop}})\right] + (1 - \mathbf{y}_m^{\mathrm{stop}}) \cdot \log\left[1 - \sigma(\rho_m^{\mathrm{stop}})\right]\right), \tag{6.3}$$

where σ is the sigmoid function and M is the number of sentences in the current paragraph.

Dual Word LSTMs A word LSTM is trained to maximize the probability of predicting the ground-truth word $\mathbf{y}_{m,t+1}^S$ at iteration step t of sentence m. The hierarchical LSTM softmax cross-entropy loss is then defined by

$$L^{\mathrm{hierarchical}} = \sum_{m=0}^{M-1} \sum_{t=0}^{N_m-2} \left(\left[\log \phi(\mathbf{n}_{m,t+1}^{\mathrm{word}})\right] \cdot \mathbf{y}_{m,t+1}^S\right), \tag{6.4}$$

where N_m are the number of words in sentence m, $\phi(\cdot)$ is the softmax activation function, $\cdot$ is the dot-product, and $\mathbf{n}_{m,t+1}^{\mathrm{word}}$ is the output of the fully-connected layer appended to the word LSTM at iteration step t for sentence m. Note that during preprocessing we prepend the start-of-sequence token `<S>` and append the end-of-sequence token `</S>` to each sentence m within a paragraph. Therefore, as already described in Sections 2.2.2, 4.2, and 5.15, we only optimize the softmax cross-entropy for iteration steps $t = 0$ until $t = N - 2$. Note that the target sequence is shifted by one word, i.e., for the input word $\mathbf{y}_{m,t}^S$ in iteration step t we want to predict the ground-truth word $\mathbf{y}_{m,t+1}^S$. Therefore, we denote the output of the fully-connected layer after the LSTM cell in iteration step t by $\mathbf{n}_{m,t+1}^{\mathrm{word}}$. Because of that, we optimize the loss for the ground-truth words $[\mathbf{y}_{m,1}^S, \ldots, \mathbf{y}_{m,N_m-1}^S]$ of every sentence m.

During inference, we then need to feed the m-th Word LSTM with the respective topic vector $\mathbf{c}_m$ and the start-of-sentence token `<S>` to sample one word at a time. The input to the m-th word LSTM at iteration step t is the embedding of the predicted word $\arg\max(\phi(\mathbf{n}_{m,t}^{\text{word}}))$ from iteration step $t-1$.

Depending on whether the current sentence is of type abnormal or normal, we train two different sets of word LSTM parameters. In other words, we have an abnormal word LSTM and a normal word LSTM, which are trained when the label of the current sentence is abnormal and normal, respectively. In practice, we set the loss weights for the current sentence to 1 in the abnormal word LSTM and to 0 in normal word LSTM. In the case of a normal sentence, we set the loss weights inversely. In parallel, we optimize the abnormal sentence prediction module (see Section 6.5.2) to predict whether a topic vector represents a sentence describing an abnormality or a normality.

During inference phase, we use the prediction of the abnormal sentence prediction module to decide whether we want to use the generated sentence from the abnormal word LSTM or the normal word LSTM. We then concatenate sentences from both the abnormal word LSTM and the normal word LSTM into our final paragraph.

6.5.2 Abnormal Sentence Prediction

As we already argued in Section 6.3, the dataset consists of many distinct normal sentences, but only few different sentences exist that describe abnormalities. We integrate an abnormality prediction module, which infers whether the semantic meaning of topic vector $\mathbf{c}_m$ does describe an abnormality or not. We use a fully-connected layer ρ_m^{abnormal} with one output neuron and sigmoid (σ) activation function to predict the probability for a sentence to be abnormal or not:

$$\rho_m^{\text{abnormal}} = \sigma\left(\mathbf{W}^{\text{abnormal}} \cdot \mathbf{c}_m + \mathbf{b}^{\text{abnormal}}\right), \tag{6.5}$$

where $\mathbf{W}^{\text{abnormal}} \in \mathbb{R}^{1\times 512}$ is a learnable parameter matrix and $\mathbf{b}^{\text{abnormal}} \in \mathbb{R}^{1\times 1}$ is a learnable bias term. We train the fully-connected layer with a binary cross-entropy loss L^{abnormal}:

$$\begin{aligned} L^{\text{abnormal}} = -\sum_{m=0}^{M-1}\Bigg(&\mathbf{y}_m^{\text{abnormal}} \cdot \log\left[\rho_m^{\text{abnormal}}\right] + \\ &(1-\mathbf{y}_m^{\text{abnormal}}) \cdot \log\left[1-\rho_m^{\text{abnormal}}\right]\Bigg), \end{aligned} \tag{6.6}$$

where $\mathbf{y}_m^{\text{abnormal}}$ ($\in \{0,1\}$) indicates whether sentence m describes an abnormality (1) or not (0).

We manually annotated the IU chest X-ray dataset for every sentence within the ground-truth paragraph of every sample in the training dataset. Two annotators labeled whether a sentence is an abnormal case or not with the help of the provided MeSH tags.

6.5.3 Multi-Task Learning

We use the global average-pooled image embedding $\tilde{\mathcal{F}} \in \mathbb{R}^{1\times 512} = \text{avg\_pool}(\mathcal{F}_e)$ for predicting the MTI annotations. As it is common in multi-label classification, we utilize a fully-connected layer ρ^{MTI} with one output neuron and sigmoid (σ) activation function for every distinct MTI label:

$$\rho^{\text{MTI}} = \sigma\left(\tilde{\mathcal{F}} \cdot \mathbf{W}^{\text{MTI}} + \mathbf{b}^{\text{MTI}}\right), \tag{6.7}$$

where $\mathbf{W}^{\text{MTI}} \in \mathbb{R}^{512\times|\mathbf{C}|}$ is a learnable parameter matrix and $\mathbf{b}^{\text{MTI}} \in \mathbb{R}^{1\times|\mathbf{C}|}$ is a learnable bias term. $\mathbf{C}$ is the set of all MTI classes and $|\mathbf{C}| = 121$ is the number of unique MTI labels. We then optimize a multi-label binary cross-entropy loss function L^{MTI}

$$L^{\text{MTI}} = -\sum_{C\in\mathbf{C}} \left(\mathbf{y}_C^{\text{MTI}} \cdot \log\left[\rho_C^{\text{MTI}}\right] + (1 - \mathbf{y}_C^{\text{MTI}}) \cdot \log\left[1 - \rho_C^{\text{MTI}}\right]\right), \tag{6.8}$$

where $\mathbf{y}_C^{\text{MTI}}$ represents the ground-truth label for a single class C out of all possible MTI classes $\mathbf{C}$. We set $\mathbf{y}_C^{\text{MTI}} = 1$, if a particular MTI tag C is present for a training sample, otherwise $\mathbf{y}_C^{\text{MTI}} = 0$.

6.5.4 Learning Objective

For our experiments, we optimize the total loss

$$\begin{aligned} L^{\text{total}} =& \lambda^{\text{stop}} \cdot L^{\text{stop}} + \lambda^{\text{hierarchical}} \cdot L^{\text{hierarchical}} + \\ & \lambda^{\text{abnormal}} \cdot L^{\text{abnormal}} + \lambda^{\text{MTI}} \cdot L^{\text{MTI}}, \end{aligned} \tag{6.9}$$

where $\lambda^{(\cdot)}$ are the weighting factors for each loss. We set $\lambda^{\text{stop}} = 5$ and $\lambda^{\text{hierarchical}} = 1$ according to Krause et al. [77]. Furthermore, we choose $\lambda^{\text{MTI}} = 1$. We did not experiment with other values for these weighting factors.

When we enable the dual word LSTMs with the abnormal sentence prediction module, we set $\lambda^{\text{abnormal}} = 1$. Contrarily, we set $\lambda_{\text{abnormal}}$ to 0 for experiments in which we disable the dual LSTM approach, i.e., we only use a single word LSTM similar to Jing et al. [70]. In this case, we also calculate $L^{\text{hierarchical}}$ with only one word LSTM.

When using the abnormal and normal word LSTMs, $L^{\text{hierarchical}}$ is calculated with the same loss function from Equation 6.4. However, depending on the annotation that tells whether the ground-truth sentence with index m describes an abnormality or not, we choose a different word LSTM with its own set of parameters. More specifically, $\mathbf{n}_{m,t+1}^{\text{word}}$ from Equation 6.4 is the output of the abnormal or normal word LSTM, depending on whether the ground-truth annotation for sentence m is abnormal or normal.

We train the image embedding layer with both the hierarchical LSTM and the MTI predictor, so the captioning task can benefit from our multi-task loss function (see Equation 6.9). We use the Adam [75] optimizer with a base learning rate of $\eta = 5 \cdot 10^{-4}$ and do not use learning rate decay. We train for up to 250 epochs and use a batch size of 16.

Table 6.2: Results on the validation and test set calculated with common machine translation metrics. We denote the scores on the validation set first and the scores on the test set in brackets. B-n stands for BLEU-n, which uses up to n-grams. We selected the model configuration and hyperparameters based on the validation set. *HLSTM / HLSTM+att* are our hierarchical LSTM implementations similar to [77, 70], and are evaluated on our dataset splits. *... +Dual* are our new models. *Scores taken from [82], who used a different dataset split.

Model	B-1	B-2	B-3	B-4	CIDEr	METEOR	ROUGE-L
CNN-RNN [139]	31.9 (33.3)	19.8 (20.5)	13.3 (13.6)	9.4 (9.4)	29.1 (30.6)	13.5 (14.5)	26.8 (27.2)
CoAtt* [70]	— (45.5)	— (28.8)	— (20.5)	— (15.4)	— (27.7)	— (—)	— (36.9)
KERP* [82]	— (48.2)	— (32.5)	— (22.6)	— (16.2)	— (28.0)	— (—)	— (33.9)
HLSTM	36.4 (**37.6**)	23.2 (23.8)	16.1 (16.3)	11.4 (11.4)	29.1 (29.3)	15.5 (15.7)	30.6 (30.2)
HLSTM+att	35.1 (36.6)	22.8 (23.4)	16.1 (16.4)	11.6 (11.7)	34.3 (32.3)	14.9 (15.6)	29.7 (29.9)
HLSTM+Dual	35.2 (35.8)	22.8 (23.1)	15.9 (16.0)	11.3 (11.2)	34.8 (32.2)	14.6 (15.1)	29.5 (29.6)
HLSTM+att+Dual	35.7 (37.3)	23.3 (**24.6**)	16.5 (**17.5**)	11.8 (**12.6**)	34.0 (**35.9**)	15.6 (**16.3**)	31.3 (**31.5**)

6.6 Experiments

In the following, we present an evaluation study and results generated by our hierarchical models with dual word LSTMs: *HLSTM+Dual* and *HLSTM+att+Dual*. We choose to compare against the CNN-RNN [139] baseline which we trained ourselves on our train split. The CNN-RNN baseline is the baseline model from Chapter 4 and does not have a sense of multiple sentences. Specifically, the CNN-RNN model is trained to predict the whole paragraph consisting of multiple sentences at once. We also compared our model against the scores reported in CoAtt [70] and KERP [82]. These models were pretrained on a non-public dataset of chest X-ray images with Chinese reports, which were collected by a professional medical institution for health checking [87]. KERP [82] uses templates which is different in comparison to end-to-end generation approaches. However, since these methods were evaluated on a different dataset split, the scores are not directly comparable to ours. We therefore implemented hierarchical LSTM baselines similar to [77] which were referred to in CoAtt [70]. These hierarchical LSTM baselines with and without an attention mechanism are named *HLSTM* and *HLSTM+att* in this chapter, and are evaluated on our dataset split. As we do not have access to the CX-CHR dataset from [87], we did not employ any task-specific pretraining of the feature extractor network.

6.6.1 Model Selection

We choose our dataset split by randomly shuffling the dataset and splitting it into a train, validation and test set with a ratio of 0.9, 0.05 and 0.05, respectively. We make sure that images of an individual patient are only present in either one of train, validation or test set. We use the validation set for selection of hyperparameters and architectural decisions. In practice, we select the best model checkpoint based on two criteria. First,

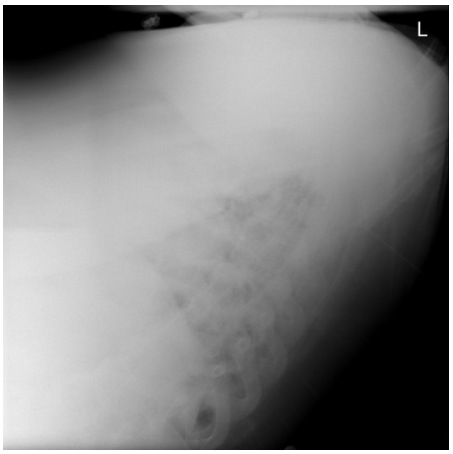
we calculate metrics such as BLEU-n twice per training epoch. Second, we also calculate the number of distinct sentences generated over the whole validation dataset for each sentence index m within a paragraph.

We then choose our final models with an early stopping method by calculating these criteria over the validation set:

1. We generate a doctor's report/paragraph for every sample in the validation set. We then count the number of distinct sentences for every sentence index within the generated doctors' reports. We only allow models that generate at least 4 distinct sentences for the first sentence (i.e., $m = 0$) within the paragraphs. In other words, we want to measure the variability of sentences generated by our model for all the first sentences of a paragraph.

2. We select the model with the highest BLEU-4 score. We depict the scores on the validation set in Table 6.2.

We report the scores on the held-out test set in brackets and show paragraphs generated by *HLSTM* and *HLSTM+Dual* in Figure 6.4.



Generated Caption *HLSTM*: exam quality limited by hypoinflation and rotation. the heart is normal in size. the lungs are clear. no focal consolidation suspicious pulmonary opacity large pleural effusion or pneumothorax is identified. no pneumothorax. no acute bony abnormalities. no pleural effusion
Generated Caption *HLSTM+Dual*: technically limited exam. basilar probable pulmonary fibrosis and scarring. the heart is mildly enlarged. there are low lung volumes with bronchovascular crowding. there is <unk >interstitial opacity and left basal platelike opacity due to discoid atelectasis scarring. there is no pneumothorax. no large pleural effusion
GT: Stable enlarged cardiomediastinal silhouette. Tortuous aorta. Low lung volumes and left basilar bandlike opacities suggestive of scarring or atelectasis. No overt edema. Question small right pleural effusion versus pleural thickening. No visible pneumothorax.

Figure 6.4: Examples of generated paragraphs with our model *HLSTM+Dual* vs. *HLSTM* in comparison with the ground-truth paragraph. Green sentences denote abnormal sentences, while blue sentences describe normalities. The colors for the generated sentence of model *HLSTM+Dual* indicate which of the two word LSTMs was responsible for generating the sentence. Ground-truth image taken from the IU chest X-ray dataset [28].

6.6.2 Analysis on Evaluation Scores and Distinct Sentences

We have observed a severe disadvantage in solely using scores such as BLEU-n as the evaluation criteria. As we mentioned before, we count the number of distinct sentences

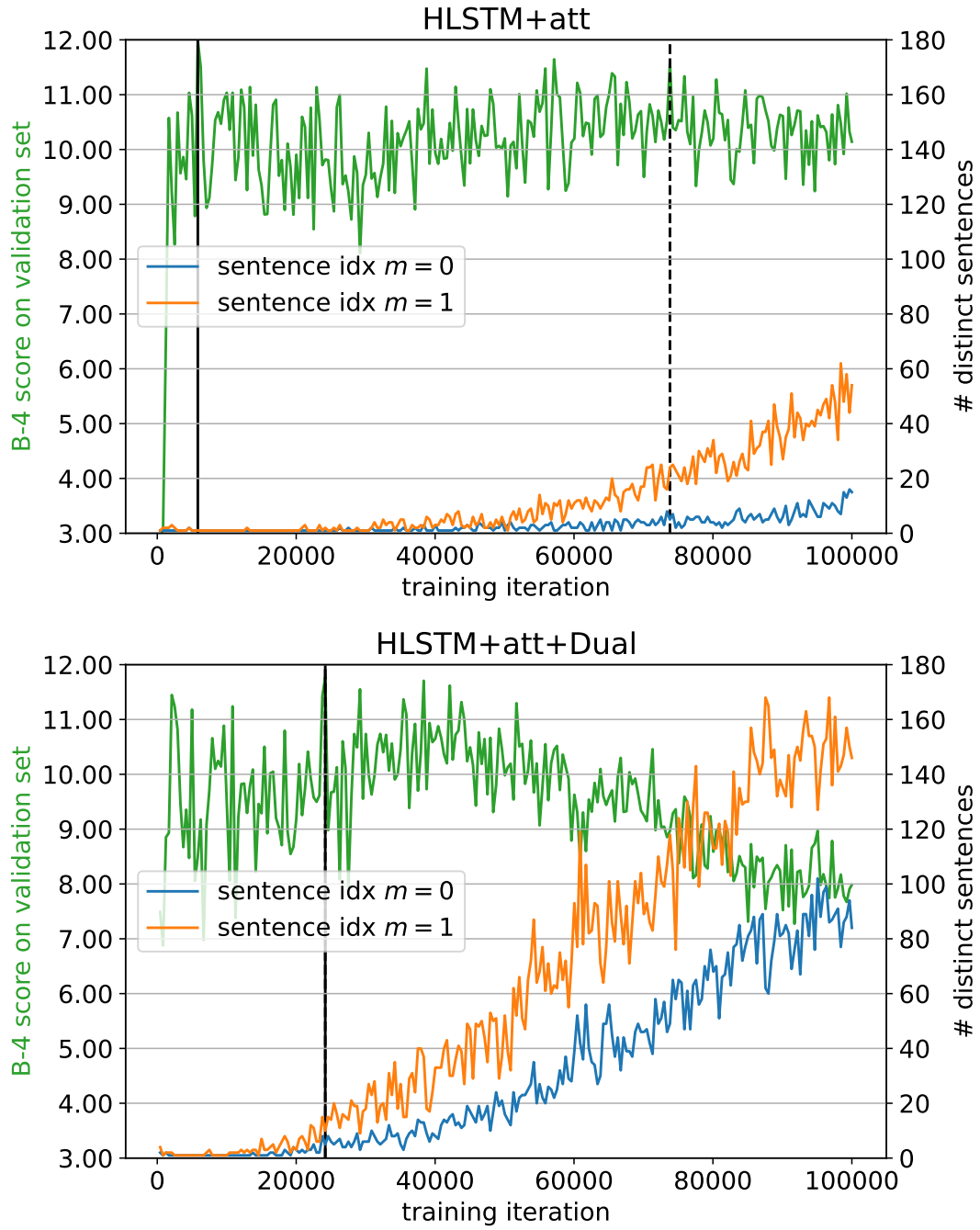


Figure 6.5: The number of distinct sentences of sentence $m = 0$ and $m = 1$ plotted against the BLEU-4 validation score (in green) over the course of training for *HLSTM+att* and *HLSTM+att+Dual*. The # of distinct sentences describes the number of different sentences generated by our model over the whole validation set for a specific sentence index m within each paragraph. We plot the # of distinct sentences for sentence indices $m = 0$ and $m = 1$. The solid line represents the training iteration with the maximum BLEU-4 score and the dashed line our selected model. In comparison, the ground-truth contains 1216 and 1540 distinct sentences for sentence indices $m = 0$ and $m = 1$, respectively.

per sentence index m within a generated paragraph for each validation point by performing the following steps:

1. We generate all paragraphs for the whole validation set.
2. We gather all sentences for every sentence position (i.e., index m) within the generated paragraphs.
3. Finally, we count the number of different sentences generated by our model for every index m.

In the following, we denote this count with *number of distinct sentences* for a sentence index m within all paragraphs over the validation set.

We notice that high BLEU-4 scores do not necessarily imply a high variability in generated sentences. Most notably, the highest scores can sometimes be observed when there are only 1 or 2 distinct sentences per sentence index resulting in very few different paragraphs.

In Figure 6.5, we show the number of distinct sentences for sentence indices $m = 0$ and $m = 1$ compared to the BLEU-4 score over the course of training. We see that the score of model *HLSTM+att* stays in the same limited range over the course of training. For example, it has a higher score even though it generates the very same paragraph for every sample in the validation set at training iteration 5838 (visualized by the vertical black line) than at training iteration 73,809 (visualized by dashed line). For the model *HLSTM+att+Dual*, we see that the score drops as more distinct sentences are generated. For this model, we also see that there is much more variability of sentences from the beginning on and also far more distinct sentences are generated in contrast to only using a single word LSTM.

Table 6.3: Absolute number of distinct sentences in the ground-truth validation set (GT) alongside the absolute number of generated sentences on the validation split of the dataset. We list the absolute counts per sentence index $m \in [0, 9]$ within the generated paragraph.

Model \ m	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
GT	1216	1540	1586	1549	1378	1086	725	477	278	171
CNN-RNN [139]	12	19	17	23	19	8	0	0	0	0
HLSTM+att	5	24	24	33	25	31	23	14	9	3
HLSTM	4	13	12	18	25	22	15	14	11	4
HLSTM+Dual	8	28	36	45	32	17	2	0	0	0
HLSTM+att+Dual	5	10	7	8	8	4	1	0	0	0

If we look at the number of distinct sentences generated per sentence index m in our chosen models compared to the ground-truth in Table 6.3, we still see a huge gap. Note

that paragraphs with only a single distinct sentence for any sentence index do not have additional benefit, since they are not dependent on the input image. For example, as we can see on the top graph in Figure 6.5, the *HLSTM+att* model at iteration 5838 (visualized by the vertical black line) yields the maximum BLEU-4 score. However, it does only produce a single sentence for all generated paragraphs for the first and second sentence indices. Considering that many sentences within the ground-truth only differ slightly but have a synonymous meaning, we find that results which do not possibly have the maximum score but a higher variability in generated paragraphs describe the input images in a better way. Thus, we chose to introduce the first stopping criterion in Section 6.6.1: We want our selected models to generate at least 4 distinct sentences for sentence index $m = 0$.

6.6.3 Dual Word LSTM with Abnormal Sentence Predictor

We present the scores on the held-out test set in Table 6.2 (in brackets). Over all evaluation metrics, our *HLSTM+att+Dual* model has the most improvement on CIDEr [138], which is designed for evaluating image descriptions, uses human consensus and considers the TF-IDF for weighting n-grams. This implies that our *HLSTM+att+Dual* model can catch more distinct n-grams in the reference paragraph. In addition, our *HLSTM+att+Dual* model is consistently better than other baselines in multi-gram BLEU, METEOR and ROUGE-L, indicating that the relevance is not sacrificed while distinctiveness is increased. In addition, we also compare our models with the dual word LSTMs from Section 6.5.2 against the vanilla HLSTM model inspired by Jing et al. [70]. As we already mentioned in Section 6.6.2, the number of distinct sentences per each sentence index starts to grow more rapidly during training when using two word LSTMs, which can be seen in the bottom part of Figure 6.5 when comparing it to the *HLSTM+att* model on top. We can also see that generating more distinct sentences does not account for better scores. However, when looking at the validation and test scores in Table 6.2, the dual word LSTM models often have higher scores than the single word LSTM models.

Table 6.4: Final results on the held-out test-set of the dataset for images that show abnormalities and normal cases. The first score describes the score over the images with abnormalities. The score in brackets is only calculated over images with normal cases.

Model	B-1	B-2	B-3	B-4	CIDEr	METEOR	ROUGE-L
HLSTM+att	30.9 (44.4)	19.0 (30.1)	12.9 (21.8)	9.1 (15.8)	25.9 (42.6)	12.8 (22.2)	25.0 (38.6)
HLSTM	32.3 (43.5)	19.4 (29.7)	12.8 (21.3)	8.8 (15.3)	24.6 (37.1)	13.2 (21.7)	25.8 (38.2)
HLSTM+Dual	**32.8** (41.2)	**20.6** (28.1)	**14.0** (20.0)	**9.8** (13.9)	**30.1** (31.8)	13.2 (19.5)	26.1 (36.0)
HLSTM+att+Dual	31.8 (**46.9**)	19.8 (**33.5**)	13.5 (**24.9**)	9.7 (**18.3**)	28.4 (**49.5**)	**13.5** (**22.8**)	**26.9** (**39.9**)

In Table 6.4, we report scores on the held-out test set for abnormal as well as normal (in brackets) images. The best-performing model for both normal and abnormal images is one of our dual models. The results also indicate that the performance is best on normal images and so effort should be given to further improve performance on abnormal images.

6.7 Summary

In this chapter, we presented a hierarchical LSTM architecture expanded by a dual word LSTM. Paired with an abnormality prediction module, we introduced dual word LSTMs, which are responsible for generating sentences that describe abnormal cases and normal cases, respectively. In order to train the abnormal sentence prediction module, we manually annotated each sentence in the IU chest X-ray dataset with a label indicating whether it describes an abnormality.

We then examined the correlation between the BLEU-n metrics and the number of distinct sentences generated by our model and observed that common evaluation metrics such as BLEU-4 do not necessarily imply a good evaluation criteria for multi-sentence medical reports. Furthermore, by introducing a dual word LSTM, we were able to increase the number of distinct sentences faster when selecting a corresponding stopping criterion. In contrast, a single word LSTM has the tendency to generate more probable sentences for every single image over and over again.

Part III

Language Models with Self-Attention for Image Description Generation and Video-to-Text Translation

Overview

In Part II of this thesis we discussed and presented image description models that generate natural language descriptions in a recurrent way, i.e., they include RNNs in order to generate a sentence word by word. Even though many works claim that LSTMs are able to alleviate or even prevent the problems associated with long-term dependencies, we have seen in Table 6.2 of Chapter 6 that an LSTM network with two hierarchy levels performs better than the baseline CNN-RNN model for descriptions that are longer than one sentence.

In addition, we also saw that attention mechanisms (Chapters 5 and 6) can help an LSTM to generate sentences with higher scores. For example, Anderson et al. [4] have shown that an image captioning model utilizing an attention mechanism can lay focus on different parts of the input image while generating different parts of a sentence. These successful applications of attention have ultimately led to a new architecture in NLP that primarily uses self-attention mechanisms in combination with fully-connected layers for extracting features from text and even for generating text. As it has happened before, this Transformer [137] architecture has been quickly adopted to other tasks than machine translation such as image captioning, object detection, video captioning, and many more with significant improvements. Furthermore, as Transformers have no more recurrent connections, another problem of RNNs was basically removed. Through unrolling recurrent cells, RNNs can become very deep networks, thus, vanishing and exploding gradients are a common problem the longer a sequence gets. In Transformers, however, each token of a sequence is fed to the same fully-connected layers before being passed to the multi-head attention. This means that tokens are not processed sequentially but in parallel. Therefore, Transformer networks do not get deeper as the sequence length grows, which means that vanishing and exploding gradients arise rarely. However, the matrix multiplication in the multi-head attention grows quadratically with the sequence length, which can be problematic for long sequences.

Because Transformers have set a new benchmark in many tasks and they combine multiple advantages in contrast to RNN networks, we discuss self-attentive models for generating descriptions in this part of the thesis. First, we introduce the task of video-to-text (VTT), which in comparison to the task of image captioning accepts video clips as inputs. For video data, the Transformer architecture is an even more natural fit, as videos are sequences of images and Transformers operate on sequence-based data on the input side as well as on the output side. Second, we focus on the models introduced in Part II and revisit them with a Transformer-based model and compare them to results and experiments from Chapters 4 and 5.

7 Fractional Positional Encoding for Transformers in Video-to-Text Translation for Synchronizing Audio-Visual Frames

In Part II, we discussed models that utilize RNN networks in order to generate textual descriptions for input images word by word. In this chapter, we will utilize the Transformer architecture and use video clips as inputs instead of images. This comes with a downside as it is naturally harder to describe what is happening in a video clip than in a single image. Islam et al. [69] argue that video captioning is more difficult than image captioning due to the following reasons: (1), in contrast to a still image, not all objects or actions of a video may be important for description generation. (2), we need to capture the motion and trajectory of related objects and make sense of the causality of events and respective objects. In image captioning, this is not needed as we cannot track these changes over time. (3), the events contained in a video can have various lengths and potentially overlap with other events. (4), a video captioning system needs to consider spatio-temporal features unlike an image captioning system that only operates on visual features. However, as videos are sequences of images, we can utilize the appealing properties of a Transformer, i.e., the encoder is designed to work on inputs that consist of sequential inputs such as video frames.

This section is based on following CVPR submission:

> **Synchronized Audio-Visual Frames with Fractional Positional Encoding for Transformers in Video-to-Text Translation** [54], Philipp Harzig, Moritz Einfalt and Rainer Lienhart, submitted to: IEEE International Conference on Image Processing, ICIP, 2022.

Additionally, we present the results of our participation in the TRECVID workshops. These works have been published as notebook papers:

> **Transforming Videos to Text (VTT Task) Team: MMCUniAugsburg** [55], Philipp Harzig, Moritz Einfalt, Katja Ludwig and Rainer Lienhart, TRECVID Workshop, 2020, virtual.

> **Extended Self-Critical Pipeline for Transforming Videos to Text (VTT Task 2021) – Team: MMCUniAugsburg**, to be published [56], Philipp Harzig,

Moritz Einfalt, Katja Ludwig and Rainer Lienhart, TRECVID Workshop, 2021, virtual.

In this chapter, we focus on the VTT task, which is actually quite similar to image captioning (see Chapters 4 and 6). In the following, we develop a model that is easy to implement and yet generates high-quality captions. We start with a Transformer modified to cope with video inputs as a baseline and investigate several improvements by adopting various techniques from the domain of image captioning. We focus only on promising extensions that do not rely on training multiple models or fusing them into an ensemble in order to design a model for transcribing videos to text that is easy to reproduce. Ultimately, we present a way to easily align video and audio features independent of their respective sampling rates. We align the features by extending the *positional encoding* to support fractional positions.

In this chapter, we discuss the following ideas and novel approaches applied to VTT:

- We develop a simple Transformer model for generating descriptions for short video clips. We reuse and adopt promising approaches from image captioning and human action classification for video clips.
- We present a combination of learning rate schedules that increases performance and shortens convergence time for VTT.
- Finally, we introduce *Fractional Positional Encoding* (FPE), an extension to the traditional positional encoding, which allows to synchronize video and audio frames independent on their respective sampling rate. By using FPE, we improve our CIDEr score by 37.13 points in comparison to the baseline. Furthermore, we achieve state-of-the-art scores on the MSVD and MSR-VTT datasets.

7.1 Related Work

Image Captioning with Recurrent Neural Networks Generating captions automatically from images is a task that has been widely studied. Most image captioning models are inspired by the machine translation encoder-decoder architecture and come with a vision CNN encoder and a language generating RNN [139, 73, 32]. Shortly after these initial works on image captioning, visual attention mechanisms have shown to benefit image description generation [152, 4] (also see Chapters 5 and 6).

Video-to-Text VTT is the natural continuation of image captioning. Instead of generating short descriptions for still images, VTT tries to infer descriptions from short video clips. Many works make use of 2D and/or 3D features in the encoder and generate the descriptions with an LSTM decoder. For the VTT task, 3D means that we operate on spatio-temporal input data. Pan et al. [104] use an encoder that utilizes 3D and 2D CNN features while the decoder is LSTM based. Gao et al. [42] implement an attention-based LSTM model with semantic consistency. First, they encode the visual frame-level

features with a single layer LSTM and use the outputs as inputs to the attention-based LSTM decoder. Concurrently, they optimize a *cross-view model* to enforce a consistency between the visual features and the sentence features generated by the attention-based LSTM decoder. Furthermore, Gan et al. [40] present an image captioning approach that utilizes tag information in the LSTM decoder. Tags are just the K most common words in the training set and trained with a multi-label classification task. More specifically, the hidden state of the LSTM decoder is then computed with an additional dependency on the predicted tag probabilities vector. Finally, they extend their approach to video captioning by concatenating average-pooled features from both a 2D and a 3D CNN. Another work by Gan et al. [39] is called *StyleNet*. It introduces a factored LSTM decoder module, that splits each of the parameter matrices $\mathbf{W}^{ix}, \mathbf{W}^{fx}, \mathbf{W}^{cx}$, and $\mathbf{W}^{ox}$ from Equations 2.4, 2.5, 2.6, 2.7, and 2.9 into three matrices. One of these three matrices is style-specific and is trained for a specific style factor. In particular, they train their model to generate the following three styles of sentences: factual captions, romantic sentences and humorous sentences. Finally, they extend their approach to video captioning by extracting and average-pooling features with the I3D network (see Sections 2.1.3 and 7.4.2).

Similar to image captioning works, VTT works have adopted traditional attention mechanisms: Long et al. [94] introduce an LSTM decoder with *multi-faceted attention*. In particular, they utilize semantic tags, motion features from a pretrained C3D [136], and temporal features from a pretrained ResNet-152 model in their attention mechanism. Wang et al. [142] propose a video captioning system with an attention-based LSTM decoder. Additionally, they introduce a reconstruction task, that aims to reconstruct visual feature vectors from the decoder LSTM's hidden states. Liu et al. [92] present *SibNet*. Its encoder is composed of two convolutional branches, i.e., a content branch and a semantic branch, which encode video content information and video semantic information, respectively. The content branch is implemented as an autoencoder and the semantic brand learns a visual-semantic joint embedding between the captions and videos. Both the content features and semantic features are utilized in an attention mechanism in the LSTM decoder. The *multimodal memory model (M3)* [143] builds a visual and textual shared memory that guides a visual attention mechanism. In particular, the model uses an external memory to store and retrieve textual and visual concepts. Pei et al. [109] also implement a video captioning model that leverages memory-attended information in a recurrent decoder. Chen et al. [18] implement a model with a two-layer LSTM that attends to motion information. They generate a rough spatial attention map by forwarding optical flow images through a CNN. Then a newly-introduced *gated attention recurrent unit (GARU)* creates refined attention maps which are concatenated and used in the two-layer decoder LSTM. Zhang et al. [157] also make use of optical flow information and extract motion features with the I3D [15] network in addition to content features extracted by a DCNN. Further works [141, 66, 159] also make use of an attention mechanism in combination with a recurrent decoder for description generation.

In order to improve the generation of descriptions for videos even further, some other works utilize object-level features for their architectures. For example, Aafaq et al. [1] encode spatio-temporal dynamics of a video by hierarchically applying the short-time

Fourier transform to the output activations of both a 2D-CNN's and a 3D-CNN's extraction layer. In addition, they utilize the features of an object detector network [118]. Zhang et al. [159] introduce a graph based object encoder that can learn an *object relational graph (ORG)*. This ORG is able to learn spatial and temporal relationships between objects. The object-level features are extracted with a Faster-RCNN [119] pretrained on MSCOCO [91] and the descriptions are generated with an LSTM. Zhang et al. [158] also present a *retrieve-copy-generate network* for video captioning. Their architecture combines encoder-decoder methods with a video-to-text retriever. A *copy-mechanism generator* based on an LSTM cell then generates the words under the guidance of visual features and sentences retrieved from the training set by the video-to-text retriever.

Transformer Networks One big leap for machine translation was the introduction of the Transformer architecture by Vaswani et al. [137]. By replacing recurrence with self-attention modules, they better utilized long-term dependencies and improved the state-of-the-art at a fraction of the training cost. Similar to the recurrent machine translation models, the Transformer architecture was quickly adopted in the task of image captioning [83, 24, 60, 154] (see Section 8.1). Another noteworthy work is the *X-linear attention network (X-LAN)* for image captioning [106]. X-LANs include an upgraded self-attention block that utilizes bilinear pooling but is technically different from the Transformer's scaled dot-product attention. Still, we classify them under the broad term of self-attention models. In their work, the authors also include the X-linear attention block into a Transformer called *X-Transformer*. The performance of the X-Transformer is slightly worse than X-LAN. However, during self-critical sequence training (SCST) [120] the X-Transformer is better than the X-LAN optimized with SCST.

Transformers for VTT and Datasets As Transformers operate on sequences of features, it is easy to modify this architecture to describe short video clips. In this work, we mainly focus on the VATEX Captioning dataset [90], which has been widely used in the VTT task. Furthermore, we validate our models on the MSR-VTT [151] and MSVD [17] datasets. Other video description datasets depicting everyday activities have been presented [7, 103]. We cover the VTT specific datasets in more detail in Section 7.1.

Lin et al. [90] implement a model that uses seven different kinds of features including I3D [15] features (see also Sections 2.1.3 and 7.4.2) and audio features. Then they use an X-LAN architecture to generate a video description. Zhu et al. [163] combine object-level features from a Faster-RCNN with motion features from three different feature extractors, including I3D. They feed these combined features in an ensemble consisting of 17 Transformer [137] and 15 X-LAN models. Both aforementioned works optimize their best models for the CIDEr score with the SCST [120] reinforcement learning approach. Finally, Guo et al. [49] utilize object-level features extracted from Faster-RCNN [119] in a Transformer encoder-decoder model.

Table 7.1: Different datasets and their respective number of video clips. Captions are available for every video. However, not every video was available for download on YouTube. The numbers of available videos from YouTube are also listed in the table.

Dataset	# Videos (clips)	# Captions	# Videos avail.	# Captions usable
VATEX [147]	41,269	349,910	38,109	323,950
MSR-VTT [151]	10,000	200,000	7773	155,640
MSVD [17]	2089	85,550	1970	80,838
AC-GIF [103]	163,183	164,378	163,183	164,378
TRECVID-VTT [8]	7485	28,183	5971	22,547

7.2 Datasets

We use the VATEX [147], MSR-VTT [151], MSVD [17], AC-GIF [103], and TRECVID-VTT [8] datasets for our video-to-text models. We depict details about these datasets in Table 7.1.

VATEX We mainly use the VATEX Dataset [147] for our experiments. The main reason for collecting this dataset was to create a new dataset with high-quality, diverse captions. Moreover, the authors introduce a bilingual dataset with both English and Chinese captions. The VATEX dataset is split into 4 sets, i.e., the training set, the validation set, the public test set, and the private test set. The VATEX dataset comes with 10 English and 10 Chinese captions per video clip. Most video clips have a length of 10 s.

The VATEX split contains 25,991, 3000, 6000 and 6278 video clips for the train, validation, public test split and private test split, respectively. We only use the English captions for our model. Because some videos were not available from YouTube at time of download we only could utilize 24,144, 2782, 5469 and 5714 video clips, respectively.

MSR-VTT Additionally, we train our final models on the MSR-VTT [151] dataset. This dataset is older than the VATEX dataset and its main motivation was to introduce a large-scale video description dataset that matches the scale of image description datasets such as MSCOCO [91]. MSR-VTT consists of 10,000 YouTube videos of which 7773 were available at the time of download. Each video clip is annotated with 20 sentences on average. We split MSR-VTT in train, validation and test with splits containing 90 %, 5 % and 5 % of the dataset.

MSVD Furthermore, we also validate our models on the MSVD [17] dataset. MSVD is the oldest video description dataset and concurrently the smallest of the datasets used in this chapter. Furthermore, it was one of the first datasets that collected linguistic descriptions for videos with a crowd-based data collection strategy. For MSVD, we follow the common practice and split the 1970 available video clips into three partitions of 1200, 100 and 670 for training, validation and test, respectively.

Auto-Captions on GIF The *Auto-Captions on GIF* [103] (AC-GIF) dataset was designed for pre-training VTT models. Because existing video-sentence datasets are mostly task-specific, i.e., they are mainly focused on specific domains such as cooking [26], the authors of the AC-GIF dataset tried to create a more generic dataset. They created their dataset by collecting GIFs and their respective alt-text HTML attributes from the web. As these alt-text attributes are mostly not accurate descriptions the dataset is predominantly thought for pre-training models and not thought of as a VTT dataset. The AC-GIF dataset contains 163,183 videos and 164,378 sentences. The total number of words is 1,619,648 with a vocabulary of 31,662 words.

TRECVID-VTT We use the official TRECVID-VTT dataset [6] which contains videos from the TRECVID-VTT challenges from 2016-2019. We only use the Twitter Vine subset of videos. In total, this subset contains $6,475$ videos from which we use $5,971$ available videos with $22,547$ captions. In all our experiments we train on 90% and validate the model on 10% of the videos.

7.3 Baseline Model

We utilize a slightly modified Transformer [137] as our baseline model and depict this baseline model in Figure 7.1. The Transformer architecture is built around the idea of transforming sequences from one domain to another. The original Transformer is a machine translation model that operates on sequences of tokens (words). However, we work on a different input domain (i.e., video clips) instead of sentences. Thus, we modified the encoder of the original Transformer architecture by altering its inputs. For the baseline architecture, we feed the encoder with embedded images

$$\mathcal{F}_e^I = [\mathcal{F}_{e,0}^I, \ldots, \mathcal{F}_{e,N_I-1}^I] \tag{7.1}$$

for every video frame instead of embedded tokens. In other words, for every frame i in a video, we take its image features $\mathcal{F}_i^I \in \mathbb{R}^{1\times 2048}$ and forward them through a fully-connected layer to embed these into the Transformer's model dimension d_{model}

$$\mathcal{F}_{e,i}^I = \mathcal{F}_i^I \cdot \mathbf{W}^{e,I} + \mathbf{b}^{e,I}, \tag{7.2}$$

where $\mathbf{W}^{e,I} \in \mathbb{R}^{2048\times d_{\text{model}}}$ and $\mathbf{b}^{e,I} \in \mathbb{R}^{d_{\text{model}}}$ are the weights and biases for the image embedding, respectively.

After embedding the image features, we add the positional encoding on top of these embeddings in order to maintain information about absolute and relative ordering of the sequence. As videos are sequences of frames, we can adopt the same positional encoding that Vaswani et al. [137] utilize for sequences of tokens. Our baseline model has $\mathcal{N} = 8$ encoder layers and outputs continuous representations

$$\mathbf{z} = [z_0, \ldots, z_{N_I-1}] \tag{7.3}$$

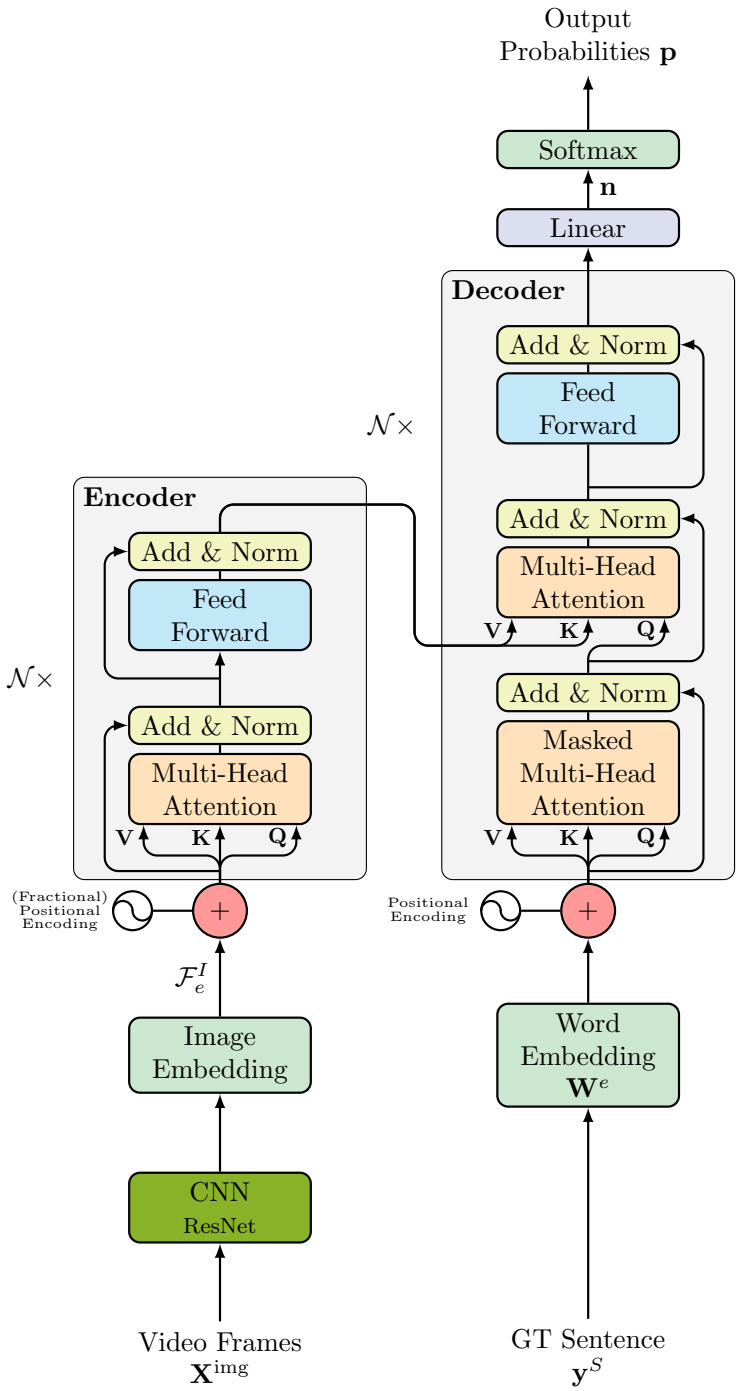


Figure 7.1: Our baseline model architecture is slightly modified from the original Transformer [137] to allow video frames as input to the encoder blocks. Original image redrawn from [137] and modified to match our baseline architecture.

of dimension $d_{\text{model}} = 512$. Our decoder also has $\mathcal{N} = 8$ layers and generates an output sequence

$$\mathbf{n} = [\mathbf{n}_1, \ldots, \mathbf{n}_{N-1}]. \tag{7.4}$$

We use a learned word embedding to convert the decoder's input tokens to vectors of dimension d_{model} and share the weight matrix with a learned linear projection layer to predict the probabilities of the next word [137, 115]. Given the embedded tokens and $\mathbf{z}$, the decoder generates its output one word $\mathbf{n}_t$ at a time. Similar to most encoder-decoder sequence models, the decoder uses the output of the previous steps as input to the current step in an auto-regressive way when generating text. Thus, we simply optimize the cross-entropy loss for every target sentence $\mathbf{y}^S$ with one-hot encoded words $\mathbf{y}_t^S$ during training

$$L(\mathbf{X}^{\text{img}}, \mathbf{y}^S) = -\sum_{t=0}^{N-2} \left(\log\left[\phi(\mathbf{n}_{t+1})\right] \cdot \mathbf{y}_{t+1}^S\right) \tag{7.5}$$

for every (video, sentence) pair. Note that $\cdot$ is the dot-product, and each video clip consists of video frames $\mathbf{X}^{\text{img}}$. Identical to all our models in Chapters 4, 5, and 6, we preprocess the ground-truth sentences according to Section 2.2.2. Particularly, we prepend the start-of-sequence token `<S>` and append the end-of-sequence token `</S>` to every ground-truth sentence $\mathbf{y}^S$ and optimize our model for every word except the start-of-sequence token. Identical to these models, our goal is to predict word $\mathbf{y}_{t+1}^S$ for the input $\mathbf{y}_t^S$, i.e., the Transformer's decoder generates $\mathbf{n}_{t+1}$ at iteration step t. Technically, we only iterate over the decoder during evaluation, i.e., when we want to generate a new caption for a given video. During training, we generate the outputs $\mathbf{n} = [\mathbf{n}_1, \ldots, \mathbf{n}_{N-1}]$ at once and optimize these according to Equation 7.5.

7.4 Video-to-Text Model

Our adapted Transformer baseline architecture is a model designed with natural language transduction in mind. Therefore, in order to gradually improve the baseline model, we employ techniques and methods from the related task of image captioning and adapt the architecture to use video clips as inputs. In the following, we motivate the changes made to the baseline architecture and preprocessing in order to improve the quality of the generated sentences.

7.4.1 Memory-Augmented Encoder

The weights learned for the self-attention layers only depend on pairwise similarities between the projected inputs, i.e., in our case the self-attention in the encoder only models pairwise relationships between single frames. Cornia et al. [24] argue that this property of the self-attention in Transformers leads to a limitation: A Transformer cannot learn and remember a-priori knowledge. In our case this would include prior knowledge about relationships between frames of videos which were already observed.

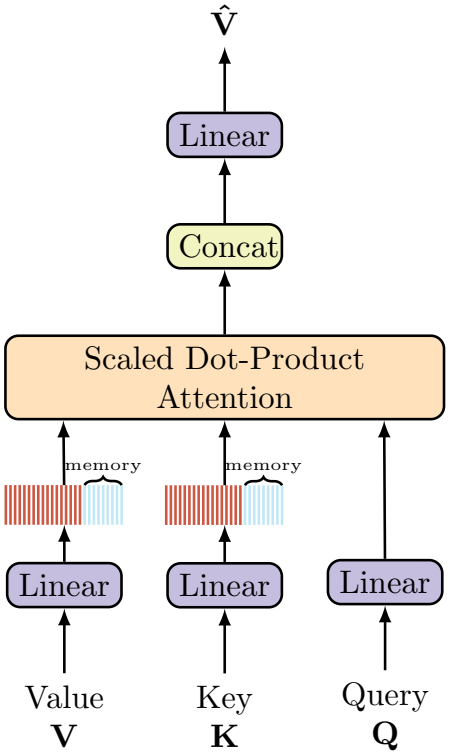


Figure 7.2: The memory-augmented attention from [24], which we use in our multi-head attention blocks. This image shows a single head of the multi-head attention. We append learned memory vectors to the linear projections of the key and value. These learned memory vectors are able to learn a-priori knowledge about the task at hand.

To mitigate this issue, we make use of the memory-augmented encoding [24], which encodes multi-level visual relationships with a-priori knowledge. In their original work, a Transformer works on inputs of image regions in an image captioning setting. The authors present a persistent, learnable memory vector which is concatenated to the key and value of the self-attention blocks of the Transformer's encoder (see Figure 7.2). These memory vectors allow to encode persistent a-priori knowledge about relationships between image regions: For example, if there is one region of a man and a second region showing a basket ball, it is difficult for a model to conclude concepts such as *player* or *game* without a-priori knowledge. In contrast to [24], we work with video sequences instead of still images with regions. Adapted to our architecture, we can encode prior knowledge about relationships between frames for each training video, which later can be transferred to unseen video samples. For example, we see in Section 7.5.2.1 that the same idea holds true for videos. In particular, our models seem to memorize the information that ice hockey is played on an ice rink. Our models that do not implement memory vectors do not generate sentences with this information.

7.4.2 Inflated 3D ConvNet

A quick and naive way to implement a simple VTT model is to use frame-by-frame features extracted by a CNN designed for image classification. However, some actions in a video clip can only be inferred by looking at temporal changes. A standard image-based CNN never gets to see what happens before or after a given frame and cannot understand these kinds of semantics, thus, is not able to reflect them in the image features. One way to extract features over multiple frames is using 3D convolutions that not only operate on the spatial dimensions but also convolve over the time dimension.

Therefore, we use the well-known Inflated 3D ConvNet (I3D; see Section 2.1.3, [15]) architecture in order to provide the encoder with better input features. In particular, we extract features from the videos with the RGB-I3D model, which was pretrained on the Kinetics Human Action Video dataset [74].

7.4.3 Subword and BERT Vocabulary

For the baseline model, we implement an ordinary dictionary that takes the $|\mathbf{V}|$ most frequent words into account, i.e., we order the words/tokens within the captions descending according to their frequency (e.g., *a* and *the* are the two most frequent words). However, an ordinary dictionary is limited by its size, e.g., 12,000 words. Rare words that are important for understanding some sentences are completely left out and replaced by an `<UNK>` token.

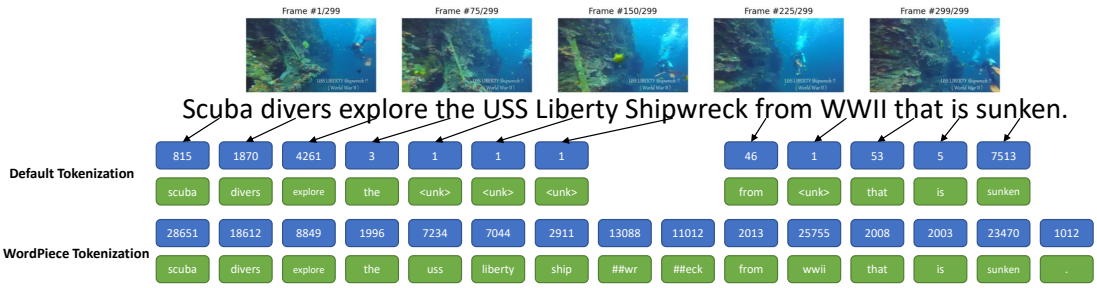


Figure 7.3: Example tokenization of a training sample from the VATEX dataset [147] (Video ID: tXzq4vvGabk) with both default tokenization and WordPiece tokenization. WordPiece tokenization splits up rare words into subwords.

A possible solution for this problem is the WordPiece tokenizer [150], which we employ in this work. The WordPiece tokenizer is a model optimized to maximize the language-model likelihood of the training data while minimizing the corpus size given a number of desired tokens. The goal is to represent rare words by splitting them up into word-pieces, which can later be recovered. For our vocabulary, we use the default BERT [29] tokens. In Figure 7.3, we depict a sample tokenization of one training sample. The default tokenization, for which we limit $|\mathbf{V}|$ to 12,000 words cannot tokenize rare words such as *USS*, *liberty*, *shipwreck*, or *WWII*. When using the WordPiece tokenizer with the BERT dictionary, we see that all words can be represented with tokens. Especially, the rare word *shipwreck* is split up into three subwords (*ship*, *##wr*, *##eck*). The leading

indicate a split-up word and we can reconstruct the whole word *shipwreck* from the three tokens.

7.4.4 Learning-Rate Scheduling

Similar to [137], we employ a learning rate schedule that linearly increases the learning rate for the first $N^{\text{warm-up}}$ training steps. After the warm-up phase, we decrease the learning rate $\eta(i)$ proportionally to the inverse square root of the current step i. We set $N^{\text{warm-up}} = 10000$ in our use case. In the following, we call this *schedule-default*:

$$\eta(i) = \frac{1}{\sqrt{d_{\text{model}}}} \cdot \min\left(\frac{1}{\sqrt{i}}, \frac{i}{(N^{\text{warm-up}})^{1.5}}\right). \tag{7.6}$$

For our models (see Section 7.5), it stands out that the validation score increases during the warm-up phase and continues to increase for two to three epochs during the slow decay of *schedule-default*. However, after that, the validation scores decrease continuously, i.e., our model starts to overfit after a short while. Since the validation scores are strongly dependent on the learning rate according to our observations, we want to prevent early overfitting by using a different learning rate schedule.

The SGDR (Stochastic Gradient Descent with Warm Restarts) [95] learning rate schedule is a promising approach, as it is applied successfully in other related works [93, 30, 47] and helps to improve scores while speeding up convergence.

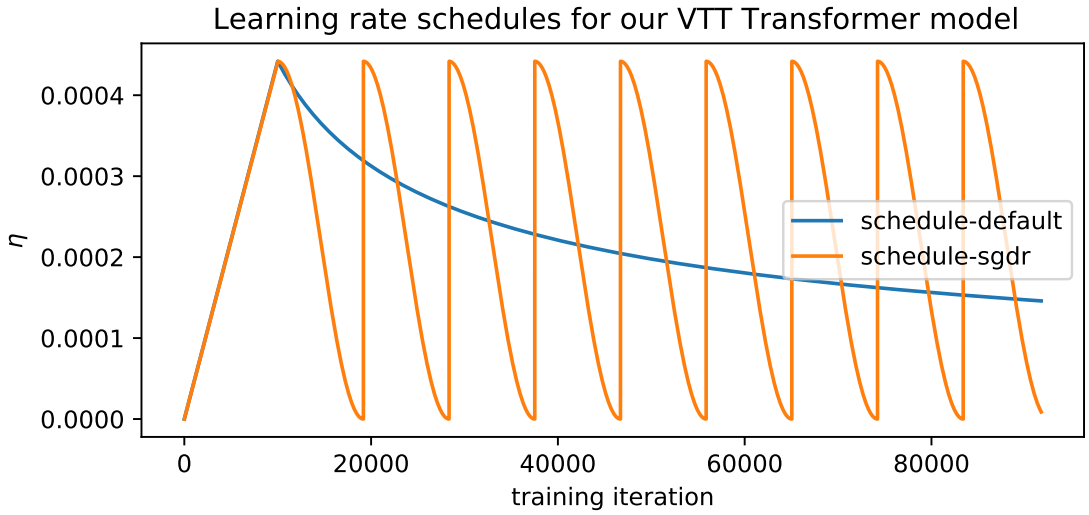


Figure 7.4: The blue line shows the default learning rate schedule for a Transformer with 10,000 warm-up steps. The orange line shows our learning rate schedule, a combination of SGDR [95] and the warm-up phase.

Initially, we found this technique to harm our final scores, i.e., the Transformer network did not seem to initialize correctly. However, when combining this approach with a warm-up phase, we did notice some improvements over *schedule-default*. Particularly, we find

that the learning-rate restarts harm the performance but the fast cosine decay helps our model to converge faster and with better scores (see Section 7.5.2). We depict *schedule-sgdr* alongside *schedule-default* in Figure 7.4. After the warm-up phase, we decay the learning rate for $T_0 = 5$ epochs. Other parameters according to [95] are $T_{mult} = 1.0$.

7.4.5 Naïve Fusion of Audio and Video Features

When generating descriptions from visual data of video clips, we can inherently only describe what we "see". However, some of the content reflected in the associated captions can only be derived when we also inject information about what we "hear". Since the videos of the VATEX dataset are videos from YouTube, we are able to extract raw audio from these video clips.

To turn these raw audio streams into usable information, we extract audio features with the VGGish [62] architecture that yields features of dimension $\mathbb{R}^{N^A \times 128}$. We forward these features through a fully-connected audio embedding layer to match $d_{\text{model}} = 512$. We show an updated version of our model in Figure 7.5.

Note that the number of audio frames N^A does not match the number of image frames or I3D frames N^I, respectively. Therefore, we cannot sum image and audio features and use these as new encoder input features. To mitigate this problem, we find an approach that utilizes properties of the Transformer architecture. Particularly, we speak of the property that all Transformer inputs have the same distance from each other, i.e., the MHA allows any input to attend to any other input. In contrast to LSTM architectures, this is a desirable property as long-term dependencies can be recognized and utilized. Inherently, this modeling does not take the order of the inputs into account. Thus, Vaswani et al. [137] introduced the positional encoding to their Transformer architecture in order to inject information about the absolute or relative position of the tokens in the sequence. Because the sequence length of image features or I3D features are varying, we cannot simply concatenate vision and audio features as the added positional encoding may signal the encoder that it receives a vision feature as input when in reality it is an audio feature.

As we still want to allow vision features to attend to audio features and vice versa, we assume a fixed starting position for all audio features which we set greater than the maximum number of vision input features $\hat{N}^I$ within the dataset. More specifically, we concatenate N^I vision and N^A audio features along the time dimension while adding the positional encodings for indices $[0, \ldots, N^I - 1, \hat{N}^I + 0, \ldots, \hat{N}^I + N^A - 1]$ onto them. We visualize the naïve approach in the top part of Figure 7.6, where $\hat{N}^I = 300$. Note that the loss function from Equation 7.5 is now also dependent on audio frames $\mathbf{X}^{\text{aud}}$:

$$L(\mathbf{X}^{\text{img}}, \mathbf{X}^{\text{aud}}, \mathbf{y}^S) = -\sum_{t=0}^{N-2} \left(\log \left[\phi(\mathbf{n}_{t+1}) \right] \cdot \mathbf{y}_{t+1}^S \right). \tag{7.7}$$

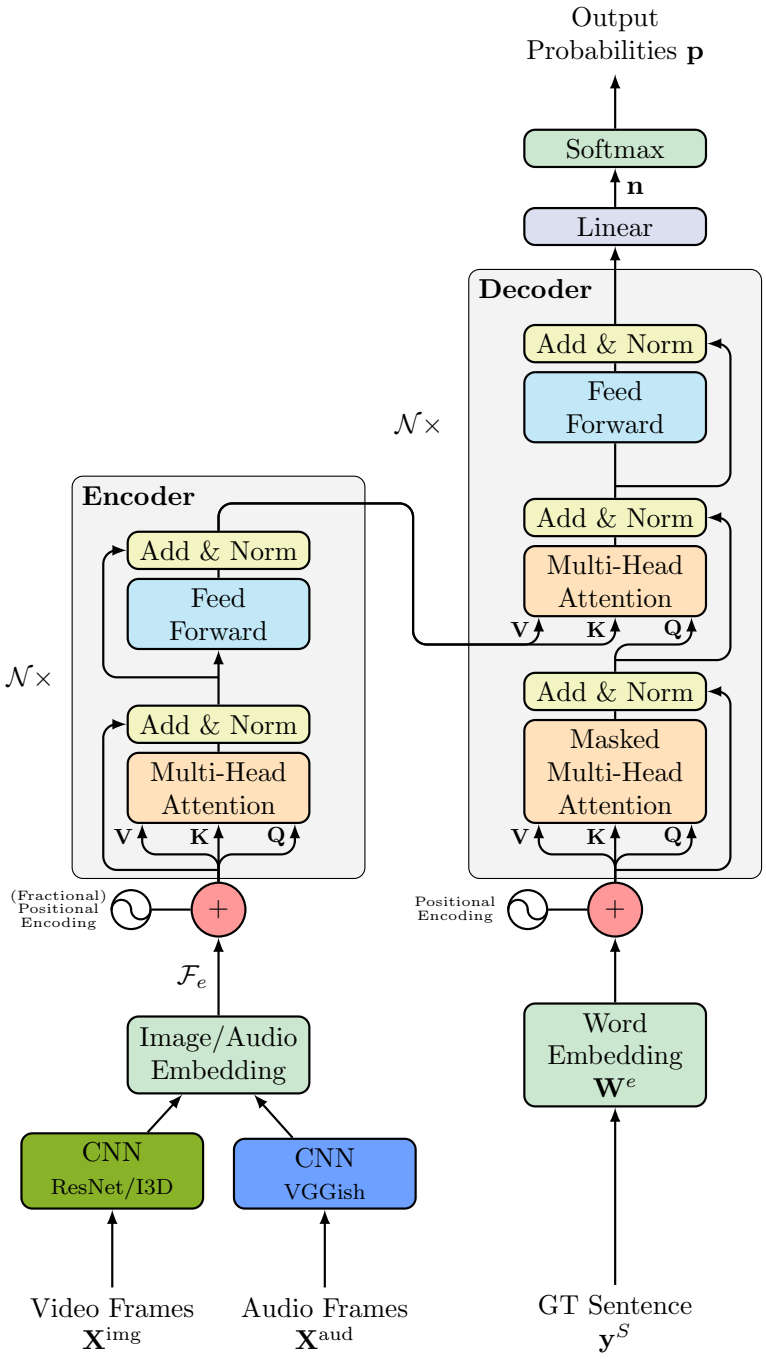


Figure 7.5: Our model is slightly modified from the baseline model in Figure 7.4. That is, the model can now utilize visual I3D features in addition to CNN features. Furthermore, our model makes use of audio features. Our embedding module can concurrently embed visual and audio features with two different methods: *naïve fusion* (see Section 7.4.5) and *fractional positional encoding* (see Section 7.4.6).

7.4.6 Fractional Positional Encoding

In addition to the naïve fusion of audio and video features, we present a novel way of aligning vision and audio features within a Transformer model. Our I3D frames and audio frames are not synchronized: for vision features, we extract single frames without resampling the video, and audio is resampled to 16 kHz.

Thus, an I3D frame at a given position represents a different timestamp for videos with different frame rates. If we resampled all videos to the same frame rate, we would still have no way of synchronizing the vision frames with the audio frames, as those sampling rates differ. In other words, the audio frame at a given position would not match the timestamp of the I3D frame at the same position.

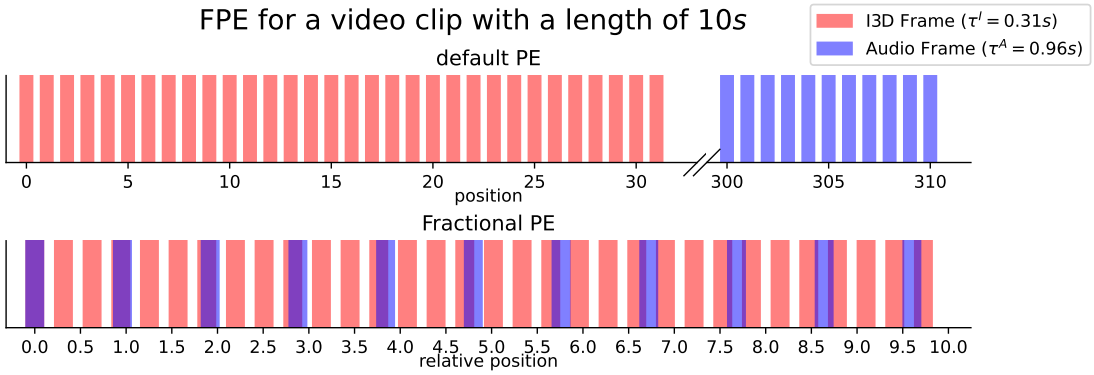


Figure 7.6: The default positional encoding for audio and video frames (on top) in comparison with the FPE (bottom) for an exemplary video. The video has 32 I3D frames and 11 audio frames. The lengths (τ) of audio and video frames differ.

In the original work [137], the positional encoding has no inherent meaning other than to define the relative position of a word. For our input data, however, vision and audio feature frames are aligned on the same time-axis and depend on their respective frame rate. Thus, we fix this problem by introducing the *Fractional Positional Encoding* (FPE) (see Figure 7.6).

FPE is an extension to the traditional positional encoding that allows positional encoding on a fractional level. In order to fully utilize the audio features, the Transformer needs to know which audio frame corresponds to which vision frame. To do so, we calculate two timestamp factors for every video within the dataset, i.e., an audio (τ^A) and a vision (τ^I) timestamp factor:

$$\tau^A = \frac{\tau}{N^A} \tag{7.8}$$

$$\tau^I = \frac{\tau}{N^I}, \tag{7.9}$$

where τ is the duration of the corresponding video clip. Both timestamp factors indicate the number of seconds each frame lasts.

During training, we then multiply the integer indices for each frame with the corresponding timestamp factor. Thus, we ensure that audio and video frames are properly aligned relative to their timestamp.

7.4.7 Self-Critical Sequence Training

In our baseline model, we optimize the objective of maximizing the likelihood of the next ground-truth word given previous ground-truth words and the encoder outputs. This approach is called *teacher forcing* [149] and has the serious drawback that the training phase is different from the inference phase (*exposure bias* [117]). The exposure bias problem describes that during inference, we can only sample the next word given previously sampled words. In contrast, we maximize the probability of the next word given ground-truth words during training. In addition, our models are trained with a cross-entropy loss and evaluated with non-differentiable metrics (e.g., CIDEr [138] and BLEU [108]). Preferably, we should train our model to optimize the metrics for the task and, thus, avoid the exposure bias issue. In the work *Self-Critical Sequence Training (SCST) for Image Captioning*, Rennie et al. [120] present a sequence model that is trained to mitigate these problems. SCST is a variation of the popular REINFORCE [148] algorithm that utilizes the outputs of the model's test-time inference algorithm.

First, we define terms and incorporate our model in a reinforcement learning setting with the identical terminology as in [120]: Our video description generation system can be seen as a single *agent* that interacts with an external *environment*. In our case, the environment consists of the audio-visual features and words. Our model consists of parameters θ, which define a policy π_θ. The policy results in an *action* which is the prediction of the next word. After each action, we update the network's internal state, i.e., the parameters of the network. After our model has generated the end-of-sequence token, the agent observes a *reward* r. In reinforcement learning, the agent is not told which action to take, rather the agent tries to find actions that maximize the observed reward. This reward can be an arbitrary function that evaluates the actions taken by the network. In our case, we choose the CIDEr and BLEU-4 metrics as the reward functions. For now, however, we consider only the CIDEr score as a reward function. As usual, we compute the CIDEr score by comparing a generated candidate sentence against a set of reference sentences. With these definitions, we can now change our training goal to minimize the negative expected reward identical to [120]:

$$L^{\mathrm{SCST}}(\mathbf{X}^{\mathrm{img}}, \mathbf{X}^{\mathrm{aud}}, \mathbf{y}^S; \theta) = -\mathbb{E}_{w^S \sim \pi_\theta}\left[r(w^S)\right], \tag{7.10}$$

where $w^S = [w_1^S, \ldots, w_{N-1}^S]$ is a sampled caption and w_{t+1}^S is a sampled word at iteration step t. Note that in case of greedy sampling $w_{t+1}^S = \arg\max(\mathbf{p}_{t+1}) = \arg\max(\phi(\mathbf{n}_{t+1}))$. $w^S \sim \pi_\theta$ stands for a caption w^S sampled with a policy π_θ. The negative expected reward is typically estimated with a single sample generated by π_θ:

$$L^{\mathrm{SCST}}(\mathbf{X}^{\mathrm{img}}, \mathbf{X}^{\mathrm{aud}}, \mathbf{y}^S; \theta) \approx -r(w^S), w^S \sim \pi_\theta. \tag{7.11}$$

If we now want to optimize our model to minimize the negative reward, we need to compute the gradient $\nabla_\theta L^{\mathrm{SCST}}(\mathbf{X}^{\mathrm{img}}, \mathbf{X}^{\mathrm{aud}}, \mathbf{y}^S; \theta)$. To do so, Rennie et al. [120] make

use of the REINFORCE algorithm [148], [130, Chapter 13] that states that the expected gradient of a non-differentiable reward function can be calculated as follows:

$$\nabla_\theta L^{\text{SCST}}(\mathbf{X}^{\text{img}}, \mathbf{X}^{\text{aud}}, \mathbf{y}^S; \theta) = -\mathbb{E}_{w^S \sim \pi_\theta}\left[r(w^S)\nabla_\theta \log \pi_\theta(w^S)\right] \tag{7.12}$$

Note that we write $\pi_\theta(w^S)$ for the probability that action w^S is taken at a given time (i.e., training iteration) when the environment is in some state with parameters θ [130, Chapter 13]. In our case, the state of the environment is given by the image features $\mathbf{X}^{\text{img}}$, the audio features $\mathbf{X}^{\text{aud}}$ and the start-of-sequence token. Similar to before, we can approximate the expected gradient by generating a single *monte-carlo* sample[1] w^S from π_θ for each training sample in a minibatch:

$$\nabla_\theta L^{\text{SCST}}(\mathbf{X}^{\text{img}}, \mathbf{X}^{\text{aud}}, \mathbf{y}^S; \theta) \approx -r(w^S)\nabla_\theta \log \pi_\theta(w^S). \tag{7.13}$$

Specifically, we generate our monte-carlo sample with categorical sampling. In contrast to greedy sampling, we do not select the most probable word at every iteration step of the sampling process but choose words according to their probability. For example, our model generates a categorical probability distribution[2] $\mathbf{p}_{t+1} = \phi(\mathbf{n}_{t+1})$ over $|\mathbf{V}|$ words at iteration step t. We then sample a word according to this probability distribution, whereas words with higher probability are more likely to be drawn but words with lower probability can still be drawn. For example, if word A has a probability of 0.2 and word B a probability of 0.1, we can expect on average to sample word A in two of ten draws, word B in one of ten draws, and a different word in seven of ten draws.

We could now optimize our model according to Equation 7.13. However, the gradients of the REINFORCE algorithm for monte-carlo samples typically suffer from high variance that lead to unstable learning updates (see [130, Chapter 13] and [120]). Therefore, the authors make use of the REINFORCE algorithm with a baseline, where they compute a reward relative to a baseline reward b. As long as the baseline reward does not depend on the action w^S, we can compute the gradient estimate with:

$$\nabla_\theta L^{\text{SCST}}(\mathbf{X}^{\text{img}}, \mathbf{X}^{\text{aud}}, \mathbf{y}^S; \theta) \approx -(r(w^S) - b)\nabla_\theta \log \pi_\theta(w^S). \tag{7.14}$$

We compute the baseline reward by first sampling a caption using the test-time greedy sampling with the current parameter set of our model. Then, we compute the CIDEr score of this caption which we denote by b. Note that - as required - the sampled caption is not dependent on the action w^S as we sample it with test-time greedy sampling. By subtracting the baseline reward, we can reduce the variance of the gradient estimate (see [130, Chapter 13] and [120]). Furthermore, we now use the model's test-time output during training which addresses the exposure bias issue.

[1] A monte-carlo sample is a single sample drawn in a monte-carlo simulation. A monte-carlo simulation tries to approximate a solution to a numerical problem by inspecting a large number of randomly generated samples.

[2] A categorical probability distribution describes the possible outcomes of a random variable that can fall into K possible categories. Each category has a defined probability and all probabilities sum up to 1.

In practice, we first greedily sample a caption for each video clip with our model in inference mode (greedy decoding). Second, we sample $\tilde{N}$ sentences w^S for the corresponding video clip in training mode using categorical sampling. Then, we calculate the CIDEr scores for the baseline caption and each sampled caption w^S, respectively. Subsequently, we adjust the reward of the sampled captions by subtracting the CIDEr score for the baseline caption. Thus, sampled captions with a higher CIDEr score than the baseline caption get a positive reward and vice versa. By optimizing for this objective, sampled captions with a higher CIDEr score will be increased in probability, while we try to make bad captions less likely. Note that we assign the same reward to every word of each sampled caption. Finally, we calculate the following loss for every training sample

$$L^{\mathrm{SCST}}(\mathbf{X}^{\mathrm{img}}, \mathbf{X}^{\mathrm{aud}}, \mathbf{y}^S; \theta) = -\frac{1}{\tilde{N}} \sum_{i=0}^{\tilde{N}-1} \left[(r(w_i^S) - b) \sum_{t=0}^{N-2} \log \phi(\tilde{\mathbf{n}}_{i,t+1}) \right], \tag{7.15}$$

where $\tilde{\mathbf{n}}_{i,t+1}$ is the output of our model with parameters θ with categorical sampling for the i-th caption at iteration step t. $\log \phi(\tilde{\mathbf{n}}_{i,t+1})$ is then the log probability for the respective sampled word. w_i^S is the i-th entire sampled caption. We see that the loss is calculated by weighting each word's log probability with the baseline adjusted reward. $r(\cdot)$ is the reward function as defined before.

For some our final models, we additionally optimize the BLEU-4 metric. Therefore, our reward function becomes

$$r(\cdot) = \lambda_{\mathrm{CIDEr}} \cdot r_{\mathrm{CIDEr}}(\cdot) + \lambda_{\mathrm{BLEU\text{-}4}} \cdot r_{\mathrm{BLEU\text{-}4}}(\cdot), \tag{7.16}$$

where $\lambda_{\cdot}$ is a weight for the corresponding metric.

In order to be able to sample new candidate sentences, we need to start from a fairly decent model to begin with. Therefore, we fine-tune the best model of ours with SCST. Integrating the SCST method in our Transformer architecture is not trivial. We need to back-propagate through the network for each generated word of each sampled caption. Therefore, SCST requires us to hold a copy of the network in GPU memory for each word. With $\tilde{N} = 5$ sampled sentences and a maximum sequence length of 30 for a sampled caption, we would have to keep 150 copies of the intermediate decoder in memory.

We limit GPU memory usage by only calling the encoder once, although this only approximates the case of separately calling the encoder for every sample. Because the dropout layers in the encoder behave differently for every execution during training, the outputs of the encoder would be different for every one of the 5 sampled captions. As we only call the encoder once, we feed the decoder with the same output for every single sampled caption. Furthermore, we have to limit the batch size per GPU to 4 in order to not exceed 40 GB of graphics memory of our NVIDIA A100 GPUs.

7.5 Experiments

In this Section, we first present the training configuration and important hyperparameters. Furthermore, we explain preprocessing techniques employed in order to properly

prepare the input data for our models. Then, we evaluate our models on a validation set to find a final model, which we evaluate on the held-out test set. Finally, we compare the model to the state-of-the-art on the VATEX, MSVD, and MSR-VTT datasets.

7.5.1 Training Configuration and Preprocessing

In the following, we present some of our implementation details, then we discuss our preprocessing strategy. In particular, we explain how we preprocessed audio-visual contents of the videos and how we prepared the ground-truth captions.

7.5.1.1 Implementation Details

Our model is implemented with TensorFlow 2 and we have published our code on GitHub[3]. As a baseline model, we implement the model from Section 7.3 with $d_{\text{model}} = 512$ and set the inner-layer dimensionality of the feed-forward network to $d_{\text{ff}} = 2048$. Our encoder and decoder each have $\mathcal{N} = 8$ layers with 8 parallel attention heads. We also adopt the the learning rate schedules from Section 7.4.4 with 10,000 warm-up steps. As optimizer, we use Adam [75] with $\beta_1 = 0.9, \beta_2 = 0.999$ and $\epsilon = 1 \cdot 10^{-8}$. We train for a maximum number of 50 epochs with a batch size of 128 and employ early stopping based on the validation CIDEr score. We choose an effective batch size of 128, i.e., we train with 2 NVIDIA A100 GPUs and a batch size of 64 each. The training takes about 15 hours for 50 epochs.

For fine-tuning our model with SCST [120], we repurposed our greedy sample algorithm to allow for sampling further captions with categorical sampling (see Section 7.4.7). We implemented the SCST training scheme like described in Algorithm 1. Basically, we greedily sample a caption in inference (test-time) mode and calculate its reward as the CIDEr score against all correct reference sentences $\mathbf{y}^S$. Then, we sample $\tilde{N} = 5$ more captions and calculate the reward by calculating their CIDEr scores and subtracting the baseline reward. We then calculate the loss by weighting each word's log probability with the final baseline adjusted reward and average the loss over the five captions. Finally, we calculate the gradients for this loss and update the network's parameters. Because of the huge memory demand of the SCST training, we lower the effective batch size to 16 (i.e. 4 GPUs with batch size 4) during the fine-tuning stage and use a constant learning rate of $\eta = 5 \cdot 10^{-6}$. When fine-tuning with SCST, the training time increases to 19 hours per epoch. The best validation scores are reached within 2-3 epochs of fine-tuning.

7.5.1.2 Preprocessing of Videos

Single Images. In order to process the videos in our model, we extract every frame of each video. We do not resample the videos to a fixed frame rate. We use ResNet-101 V2 [59] to compute features for the extracted frames by resizing the input images to 224×224 and using the average-pooled features with dimension $N^I \times 2048$, where N^I

[3] `https://github.com/fpe-vtt/ftt-vpe`

Algorithm 1 Self-critical sequence training

for each $v \in$ dataset **do**
  $\text{caption}_{\text{greedy}}, \mathbf{n} \leftarrow$ Greedy-Sample(v)
  $b \leftarrow$ CIDEr$(\text{caption}_{\text{greedy}}, \mathbf{y}^S)$
  for $i = 0$ to $\tilde{N} - 1$ **do**
    $\text{caption}_{\text{categorical}}, \tilde{\mathbf{n}} \leftarrow$ Categorical-Sample(v)
    $r \leftarrow$ CIDEr$(\text{caption}_{\text{categorical}}, \mathbf{y}^S)$
    $L_i^{\text{SCST}} \leftarrow -(r - b) \sum_{t=0}^{N-2} \log \phi(\tilde{\mathbf{n}}_{t+1})$
  end for
  $L^{\text{SCST}} \leftarrow \frac{1}{\tilde{N}} \sum_{i=0}^{\tilde{N}-1} L_i^{\text{SCST}}$
  Calculate $\nabla_\theta L^{\text{SCST}}$ and update parameters θ
end for

is the number of frames of the corresponding video clip. We embed the images with an image embedding layer that projects the features into the model's dimension $N^I \times d_{\text{model}}$.

I3D Features. We extract I3D features similar to frame-level features. Instead of forwarding frame images through the ResNet-101 V2 network, we extract video clip features with the RGB-I3D pretrained on the Kinetics Human Action Video dataset [74]. The I3D yields features of dimension $N^I \times 7 \times 7 \times 1024$, whereas in this case, N^I is the number of I3D frames, which is less than the original number of frames in the video due to the 3D-convolutions. Furthermore, we average-pool over the spatial dimensions (7×7) and use the same image embedding layer to embed the I3D features into the model dimension $N^I \times 512$.

Audio features. We take the audio of the video, resample it to 16 kHz and extract features with the VGGish [62] network. This network yields features of dimension $N^A \times 128$. Here, N^A is the number of audio features, which is different from N^I. If no audio stream for a video is existent, we create a dummy feature vector with all zeros and dimension 1×128.

7.5.1.3 Preprocessing of Tokens

For our models with the default vocabulary, we employ a simple text tokenizer that filters out special characters[4]. We limit the vocabulary to 12,000 tokens and replace less occurring words with the `<UNK>` token.

For our WordPiece models, we use the English BERT vocabulary together with the WordPiece tokenizer to generate the tokens. For some of our models, we load pretrained embedding weights[5] from the $\text{BERT}_{\text{SMALL}}$ model, which has $d_{\text{model}} = 512$ to match our architecture.

[4] !"#$%&()*+.,-/:;=?@[\]^_‘{|}~

[5] `https://tfhub.dev/google/small_bert/bert_uncased_L-8_H-512_A-8/1`

Table 7.2: Ablation study for our VTT Transformer models on the VATEX validation set. On the left, we list the names of the models with their respective configurations (|mv| = length of memory vector, ft We = fine-tuning of word embedding). On the right we list the validation scores (B-x = BLEU-x, M = METEOR, R = ROUGE-L, C = CIDEr).

Model	Features	|mv|	Vocabulary	ft We	FPE	lr Schedule	B-4	M	R	C
baseline	R101	0	Default	✓	—	Default	23.47	18.72	43.60	33.72
memvec	R101	64	Default	✓	—	Default	24.57	18.83	43.92	35.23
i3d-baseline	I3D	0	Default	✓	—	Default	27.92	20.85	46.33	49.13
i3d-memvec	I3D	64	Default	✓	—	Default	29.21	21.77	47.33	53.01
i3d-wp	I3D	64	WP	✓	—	Default	29.04	21.49	47.08	50.19
i3d-wp-audio	I3D+VGGish	64	WP	✓	—	Default	29.99	22.11	48.10	52.64
i3d-audio	I3D+VGGish	64	Default	✓	—	Default	30.40	22.04	48.08	51.72
i3d-bert	I3D	64	WP-BERT	—	—	Default	28.26	21.76	47.14	51.38
i3d-bert-audio	I3D+VGGish	64	WP-BERT	—	—	Default	30.35	22.06	47.90	53.16
i3d-bert-ft-audio	I3D+VGGish	64	WP-BERT	✓	—	Default	30.32	21.95	48.09	52.09
i3d-bert-audio-sgdr	I3D+VGGish	64	WP-BERT	—	—	sgdr	32.16	22.70	49.07	56.92
i3d-bert-audio-sgdr-FPE	I3D+VGGish	64	WP-BERT	—	✓	sgdr	32.43	23.81	49.60	61.80
SCST-Cider	I3D+VGGish	64	WP-BERT	—	—	$5 \cdot 10^{-6}$	28.73	23.08	48.20	68.87
SCST-Cider-B4	I3D+VGGish	64	WP-BERT	—	—	$5 \cdot 10^{-6}$	33.92	23.65	49.91	68.62
SCST-Cider-B4-FPE	I3D+VGGish	64	WP-BERT	—	✓	$5 \cdot 10^{-6}$	36.30	24.52	51.91	70.85

7.5.2 Results

In the following, we discuss the results of the extensions presented in Section 7.4. In particular, we describe how we gradually turn a vanilla Transformer architecture into an end-to-end trainable network that generates textual descriptions for short video clips. Several tricks and design decisions borrowed from the field of image captioning turn a model which was designed for machine translation into a video-to-text architecture, which can be easily adapted for multiple application scenarios. In Table 7.2, we depict results on the validation set of the VATEX dataset. Furthermore, in Figures 7.8, 7.9, and 7.10 we show generated captions for three example videos from the VATEX validation set for every model. Both when looking at the scores and the generated descriptions, we see that our *baseline* model scores worst across all metrics. The baseline model uses frame-level ResNet-101 features embedded with an image embedding layer in the encoder.

7.5.2.1 Memory-Augmented Encoder

Adding a memory vector to the key and value of the multi-head self-attention allows the encoder network to learn a-priori knowledge about relationships on an inter-frame level. For example, when we look at sentences generated for the third video clip in Figure 7.10, we see an ice hockey player doing some shots on a goal. Comparing the captions generated by the **memvec* models to the captions of the baseline models, we see the models have memorized that ice hockey often is played within an *ice rink*. In addition, for the model *memvec* with a memory vector (|mv|) of size 64, we see a slight boost in all scores except ROUGE-L, e.g., the CIDEr score is improved by about 1.5 points.

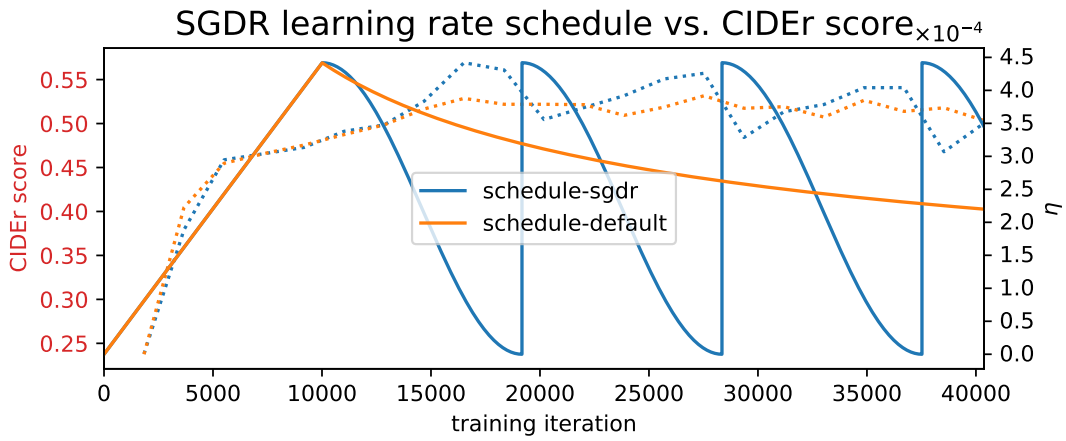


Figure 7.7: The course of learning rate plotted against the CIDEr validation score of models *i3d-bert-audio* and *i3d-bert-audio-sgdr*. We plotted the learning rates with solid lines and the corresponding validation scores with a dotted line style.

7.5.2.2 Image Features and I3D Features

One of the two extensions gaining the most in terms of CIDEr score is replacing frame-level image features with features from the RGB-I3D network. Looking at the model *i3d-baseline* that uses no memory-augmented encoder, we see an increase of 15.41 points and 4.45 points in CIDEr and BLEU-4, respectively. The memory-augmented encoder benefits from the I3D features in the same way, i.e., *i3d-memvec* gains 17.78 points and 4.64 points in CIDEr and BLEU-4, respectively. Thus, we train all other future models with I3D features and the memory-augmented encoder.

7.5.2.3 Naïve Fusion of Audio and Video Features

Most videos not only contain visual data but also audio data. Thus, it is obvious that some contents of a textual description of said video clip can only be described accurately when also using audio data. As already described in Section 7.4.5, we concatenate vision and audio features, however, using a trick with the positional embeddings in order to allow the network to extract vision-audio dependencies. When comparing the model *i3d-audio* with *i3d-memvec* we can observe no gains in performance, i.e., the CIDEr score is slightly worse with -1.29 points while BLEU-4 improves the score by 1.19, However, we will see in the next paragraph that combining audio features with the BERT dictionary will yield improvements.

7.5.2.4 Dictionaries and Tokenization

We already stated that by using WordPiece tokenization, we are able to generate rare words that do not occur in a vocabulary with a fixed size. However, this does not necessarily reflect on the scores as model *i3d-wp* shows. In comparison to *i3d-memvec*

we loose 2.82 points and 0.17 points in CIDEr and BLEU-4, respectively. Similarly, when initializing the word embedding with BERT embeddings (*i3d-bert*), we lose 1.63 points and 0.95 points for CIDEr and BLEU-4, respectively. However, in combination with the concatenated audio features, WordPiece tokenization gives us better results. For WordPiece tokenization with no initialization of the word embedding, we loose 0.37 points (CIDEr) and gain 0.78 points (B-4). When initializing the word embeddings (*i3d-bert-audio*) we get slightly higher scores. When comparing *i3d-bert-audio* with *i3d-bert*, we also see the benefit of audio features, which could not be seen beforehand. Note that we keep the BERT embeddings frozen during training, because fine-tuning them hurts performance (see model *i3d-bert-ft-audio*).

7.5.2.5 Learning Rate Scheduling

By replacing the default Transformer learning rate schedule with our modified version of SGDR, *i3d-bert-audio-sgdr* improves the performance by 3.76 points and 1.81 points in CIDEr and BLEU-4, respectively. As we have already discussed in Section 7.4.4, the fast decay of the SGDR schedule helps to boost our validation scores as we depict in Figure 7.7. After the warm-up phase of 10,000 steps, the validation accuracy makes another climb until it hits its maximum CIDEr score of 56.92 at the end of the first decay. However, we see that restarting the learning rate leads to a drop in performance. Our model can recover somewhat but never reaches the maximum score again and its performance declines slowly.

7.5.2.6 FPE

In contrast to the naïve fusion of audio and video features, FPE (*i3d-bert-audio-sgdr-FPE*) boosts performance across all metrics significantly. Most notably, synchronizing audio and video features by their relative position shows the hugest benefit on the CIDEr metric, where we gain 4.88 points. Even during self-critical fine-tuning (see Section 7.5.2.7), FPE (*SCST-Cider-B4-FPE*) achieves improvements across all metrics. FPE is also an effective way to allow for training on multiple datasets at once that have different respective frame rates. Furthermore, it allows for transfer learning without needing to adapt the new dataset to the frame rate that was trained on originally. Finally, FPE adds the benefit of easily aligning features from audio data without the need of implementing any additional synchronization logic. Thus, we conclude that FPE is an easy and effective way to synchronize audio and video features in Transformers.

7.5.2.7 SCST

We initialize the self-critical sequence training with the best models *i3d-bert-audio-sgdr* and *i3d-bert-audio-sgdr-FPE*. As reward function, we calculate the CIDEr score of the baseline caption and the sampled sentences. We see that directly optimizing the CIDEr metrics leads to big gains in the CIDEr metric, i.e., 68.87 points vs. 56.92 points. The difference of 11.95 points is the second biggest improvement besides replacing image features with I3D features. However, as we only optimize for the CIDEr metric in model

SCST-Cider, we lose 3.43 points on the BLEU-4 metric. We also lose some performance across all other metrics except METEOR. However, when directly optimizing for CIDEr and BLEU-4 (see model *SCST-Cider-B4*; we set $\lambda_{\text{CIDEr}} = \lambda_{\text{BLEU-4}} = 1.0$), we see that the CIDEr score is nearly identical while all BLEU-n scores get a significant boost. When combining SCST with FPE, our model produces the best results across all experiments and we improve by another 2.23 and 2.38 in CIDEr and BLEU-4, respectively.

7.5.3 Comparison with State-of-the-Art

We were not able to download all video files for the VATEX dataset from YouTube (see Table 7.1), thus we could not train, validate and test on the whole dataset. For the private test split of the VATEX dataset, we could download 5714 of 6278 videos, thus, missing features for 564 videos. However, the authors provide pre-extracted I3D features. But after closer inspection, we notice that these features do not match our I3D features. Additionally, we do not have audio features for those missing videos. Submitting generated descriptions to the evaluation server requires descriptions for every single of the 6,278 videos. Therefore, we used the VATEX authors' I3D features with no audio features for submitting results. In Table 7.3, we depict the results of our model *SCST-Cider-B4-FPE* trained in the same manner on both train and validation splits.

Our model does not perform as well as the models from the VATEX video captioning challenge (Zhu et al. [163] and Lin et al. [90]), who use ensembles of multiple models. However, across all peer-reviewed works on video captioning, we achieve similar performance on the reported metrics.

We suspect that the poorer performance of our model is due to three reasons:

1. We cannot utilize all features for the test video, i.e., 564 of the videos are missing, because they are unavailable on YouTube.

2. Our method does not use X-linear attention [106], which [163, 90] do use for their models.

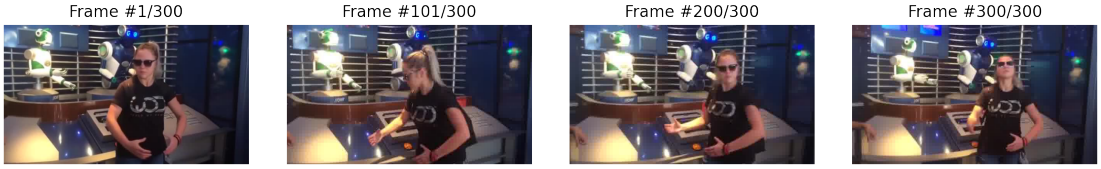
3. Also, [163, 90] train an ensemble of up to 19 networks to boost their performance. As we develop an application-centric, easy to reproduce model, we do not train an ensemble of models.

In addition, we also train our model on the MSVD and MSR-VTT datasets to prove the effectiveness of our method. On the MSVD dataset, our scores are below SemSynAN [111] but otherwise better than all other methods listed in Table 7.3. For MSR-VTT, however, our final model outperforms SemSynAN by 10.21 points in CIDEr and the model performs similarly to it for the other metrics.

Table 7.3: Comparison on VATEX, MSVD, and MSR-VTT datasets against state-of-the-art methods. For VATEX, we tested our model on the private test set with the evaluation server. For MSVD and MSR-VTT we use the test splits discussed in Section 7.1. I, M, O and A denote image, motion, object and audio features.

		Features				MSVD				MSR–VTT				VATEX			
Model	**Year**	**I**	**M**	**O**	**A**	**B-4**	**M**	**R**	**C**	**B-4**	**M**	**R**	**C**	**B-4**	**M**	**R**	**C**
M3	CVPR 2018 [143]	✓	✓	—	—	52.8	33.3	—	—	38.1	26.6	—	—	—	—	—	—
RecNet	ICCV 2018 [142]	✓	—	—	—	52.3	34.1	69.8	80.3	39.1	26.6	59.3	42.7	—	—	—	—
PickNet	ECCV 2018 [21]	✓	—	—	—	52.3	33.3	69.6	76.5	41.3	27.7	59.8	44.1	—	—	—	—
MARN	CVPR 2019 [109]	✓	✓	—	—	48.6	35.1	71.9	92.2	40.4	28.1	60.7	47.1	—	—	—	—
SibNet	ACM'MM 2018 [92]	✓	—	—	—	54.2	34.8	71.7	88.2	40.9	27.5	60.2	47.5	—	—	—	—
OA-BTG	CVPR 2019 [157]	✓	—	✓	—	56.9	36.2	—	90.6	41.4	28.2	—	46.9	—	—	—	—
GRU-EVE	CVPR 2019 [1]	✓	✓	✓	—	47.9	35	71.5	78.1	38.3	28.4	60.7	48.1	—	—	—	—
MGSA	AAAI 2019 [18]	✓	✓	—	✓	53.4	35	—	86.7	42.4	27.6	—	47.5	—	—	—	—
POS+CG	CVPR 2019 [141]	✓	✓	—	—	52.5	34.1	71.3	88.7	42	28.2	61.6	48.7	—	—	—	—
POS+VCT	ICCV 2019 [66]	✓	✓	—	—	52.8	36.1	71.8	87.8	42.3	29.7	62.8	49.1	—	—	—	—
ORG-TRL	CVPR 2020 [159]	✓	✓	✓	—	54.3	36.4	73.9	95.2	43.6	28.8	62.1	50.9	32.1	22.2	48.9	49.7
LSTM-TSA$_{IV}$	CVPR 2017 [105]					52.8	33.5	—	—	—	—	—	—	—	—	—	—
aLSTMs	IEEE ToM 2017 [42]	✓	✓	—	—	50.8	33.3	—	—	38	26.1		43.2	—	—	—	—
RCG	CVPR 2021 [158]	✓	✓	—	—	—	—	—	—	42.8	29.3	61.7	52.9	33.9	23.7	50.2	57.5
NSA	CVPR 2020 [49]	—	✓	✓	—	—	—	—	—	—	—	—	—	31.4	22.7	49	57.1
SemSynAN	CVPR 2021 [111]	✓	✓	—	—	**64.4**	**41.9**	**79.5**	**111.5**	**46.4**	**30.4**	**64.7**	51.9	—	—	—	—
VATEX	CVPR 2019 [147]	—	✓	—	—	—	—	—	—	—	—	—	—	28.7	21.9	47.2	45.6
SCST-Cider-B4-FPE[6]		—	✓	—	✓	51.22	34.73	72.69	103.2	45.91	30.25	64.12	**62.11**	33.28	22.74	49.56	54.63
Non peer-reviewed papers:																	
MV+HR	arXiv 2019 [163]	✓	✓	✓	—	—	—	—	—	—	—	—	—	**40.7**	**25.8**	**53.7**	**81.4**
MM-Feat	arXiv 2020 [90]	✓	✓	✓	✓	—	—	—	—	—	—	—	—	39.2	26.5	52.7	76
NITS-VC	arXiv 2020 [126]	—	✓	—	—	—	—	—	—	—	—	—	—	22	18	43	27

[4]We were only able to extract I3D and audio features for 5,714/6,278 video clips as the videos were no longer available on YouTube. We could use I3D features made available by the dataset's authors. These features, however, were different from our I3D features.



baseline: a man is standing in a restaurant and talking about it
memvec: a man is sitting at a table playing a drum set
i3d-baseline: a woman is standing in front of a faucet and she is holding a bottle
i3d-memvec: a young boy is playing with a machine and a woman is talking
i3d-wp: a young man is standing at a ball game and talking to the camera .
i3d-wp-audio: a woman is standing in front of a store and she is talking to a man .
i3d-audio: a young boy is standing at a counter and he is standing in a chair
i3d-bert: a man is standing in front of a car and he is pumping gas in the air .
i3d-bert-audio: a young girl is at a bar and she is playing the game
i3d-bert-audio-sgdr: a young girl is trying to get her balance on the machine .
i3d-bert-audio-sgdr-FPE: a young girl is playing a toy game while a man watches her .
SCST-Cider-B4: a man and a young man is playing with a football player .
SCST-Cider-B4-FPE: a young girl is using a machine to peel a game of meat .
GT: a woman moves like a robot in front of a robot exhibit.

Figure 7.8: Generated descriptions for the first of three example videos from the validation split. We see four frames of the video together with the frame number on the left and the generated caption for each model on the right. Video from the VATEX dataset [147].

7.5.4 Application on Unseen Datasets

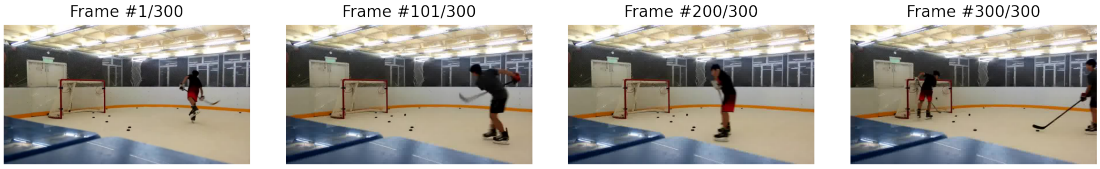
One possible application of our model is to use it as a generic video clip descriptor. Thus, we use our model *SCST-Cider-B4* to generate descriptions on other unseen datasets. In particular, we use a validation split (i.e., 100% of the data) of the MSR-VTT dataset and a validation split (i.e., 100% of the data) of the TRECVID-2020 VTT dataset [7]. In Table 7.4, we depict the scores on those two datasets. Looking at the scores for TRECVID-2020 dataset, we see that the model performs reasonably well, i.e., according to the TRECVID-2020 VTT challenge [7], our model would ranked the third-highest in CIDEr score even though it has never seen a single sample from the whole TRECVID dataset. For the MSR-VTT dataset, the scores are similar regarding CIDEr. For the other metrics, we see that the scores are slightly higher. However, when looking at the original work for the MSR-VTT dataset [151], the scores are worse than their presented

Table 7.4: Results of our best models on two different datasets (validation splits). The model has never seen data from any of those datasets.

Unseen Dataset	B-4	Met.	ROUGE-L	CIDEr
MSR-VTT [151]	21.5	24.6	49.4	19.8
TRECVID-2020 [7]	8.7	12.7	31.0	21.4

baseline: a man is flipping a pancake in a pan and catches it
memvec: a woman is sitting at a desk and talking about it
i3d-baseline: a man flips a pancake in the air and catches it
i3d-memvec: a woman is flipping a pancake in a pan and then she flips it
i3d-wp: a man is flipping a pancake in a frying pan .
i3d-wp-audio: a man is flipping a pancake in the air and catches it .
i3d-audio: a young girl is flipping a pancake and flipping it
i3d-bert: a man flips a pancake in a kitchen and flips it .
i3d-bert-audio: a man is flipping a pancake in a pan .
i3d-bert-audio-sgdr: a man is flipping a pancake in a pan .
i3d-bert-audio-sgdr-FPE: a man is flipping a pancake in a pan and then catches it .
SCST-Cider-B4: a man and a man flips a pancake and then in a frying pan .
SCST-Cider-B4-FPE: a young man is flipping a pancake in a frying pan .
GT: a young man flips pancakes out of a frying pan to his friend who catches them in his frying pan.

Figure 7.9: Generated descriptions for the second of three example videos from the validation split. We see four frames of the video together with the frame number on the left and the generated caption for each model on the right. Video from the VATEX dataset [147].



baseline: a group of people are playing a game of curling
memvec: a man is playing a game of curling in a rink
i3d-baseline: a hockey player is skating backwards and then turns to a stop
i3d-memvec: a group of people are practicing skating on an ice rink
i3d-wp: a person is skating on an ice rink and practicing ice skating .
i3d-wp-audio: a group of people are playing hockey in an arena .
i3d-audio: a hockey game is being played on an ice rink
i3d-bert: a group of people are skating around in a hockey rink .
i3d-bert-audio: a group of people are playing hockey in a gym .
i3d-bert-audio-sgdr: a group of people are playing hockey in a rink .
i3d-bert-audio-sgdr-FPE: a group of people are playing a game of soccer in a indoor rink .
SCST-Cider-B4: a man and a person is playing a hockey goal on an ice rink .
SCST-Cider-B4-FPE: a group of people are playing a game of hockey on a rink .
GT: two boys practice shooting hockey pucks into the net on an ice rink.

Figure 7.10: Generated descriptions for the third of three example videos from the validation split. We see four frames of the video together with the frame number on the left and the generated caption for each model on the right. Video from the VATEX dataset [147].

baseline architectures. The lowest scoring baseline architecture in [151] scores 35.4 in BLEU-4 comparing to our score of 21.5. However, our METEOR score is closer with 24.6 vs. 26.3. CIDEr scores are not reported in [151].

Naturally, if we compare the scores to models trained on the respective datasets the scores are worse. If we look at Tables 7.3 and 7.4, we see that when applying a well-performing model to the unseen MSR-VTT dataset, scores decrease by 24.41 and 42.31 for the BLEU-4 and CIDEr metric, respectively. However, note that Table 7.3 is evaluated on the test set and results from Table 7.4 are from the whole validation dataset. Also, if we look at Table 7.7, we see that the TRECVID scores decrease by 10.1 in CIDEr when not trained on the TRECVID-VTT dataset. Surprisingly, the BLEU-4 score is better but we suspect that the TRECVID organizers use a different script to evaluate their BLEU scores, as these are consistently worse than the scores returned by the MSCOCO evaluation script (see Section 7.6 for more details).

7.6 Experiments and Results on the TRECVID-VTT Dataset

In this part of the experiments, we discuss results on a different dataset, i.e., the TRECVID-VTT dataset. The model is partly identical to the model discussed in this chapter (i.e., Section 7.4). Results have been published in the following TRECVID workshop papers:

> **Transforming Videos to Text (VTT Task) Team: MMCUniAugsburg** [55], Philipp Harzig, Moritz Einfalt, Katja Ludwig and Rainer Lienhart, TRECVID Workshop, 2020, virtual

> **Extended Self-Critical Pipeline for Transforming Videos to Text (VTT Task 2021) – Team: MMCUniAugsburg**, to be published [56], Philipp Harzig, Moritz Einfalt, Katja Ludwig and Rainer Lienhart, TRECVID Workshop, 2021, virtual

7.6.1 Model Configuration

We implemented four models for the TRECVID-VTT task. In Table 7.5, we present the number of training samples for the data splits used for training the base and fine-tuned models. The first two models are pretrained on a merged dataset (data split #1 and #2) and then fine-tuned on the TRECVID-VTT only dataset (data split #3). Models 3 and 4 are both pretrained and fine-tuned on the same merged dataset (VATEX + 90 % TRECVID (extended); data split #4).

Our models use $\mathcal{N} = 8$ encoder and $\mathcal{N} = 8$ decoder blocks. We use 8 attention heads and a model dimension of $d_{\text{model}} = 512$. For the position-wise feed-forward networks, we set $d_{\text{ff}} = 2048$ as the inner-layer dimensionality. We use a memory-vector size of 64.

TRECVID-1 For our primary model, we first train a base model on the full MSR-VTT dataset and 90 % of the TRECVID-VTT dataset. We select the model by employing an early-stopping strategy on the CIDEr score of the remaining 10 % of the TRECVID-VTT (i.e., validation split) dataset. Furthermore, we train with a vocabulary of 12,000 of complete words. For fine-tuning, we use the base model and train it on the 90 % split of the TRECVID-VTT dataset (data split #3). We also use early-stopping to select our fine-tuned model *TRECVID-1-ft*.

TRECVID-2 Our second model is trained similarly, except we use AC-GIF (full), MSR-VTT (full), and TRECVID-VTT (90 %) for training the base model. For the second model, we use a subword text encoder with 20,283 subwords. It does not use complete words for the vocabulary but tries to build words from subwords, i.e., it splits words into subwords if a word is not in the initial dictionary (see Section 7.4.3). For fine-tuning the second model, we also use the TRECVID-VTT (90 %) split (data split #3).

TRECVID-3 We train the third model on the full VATEX dataset and 90 % of the TRECVID dataset (data split #4). Similar to the previous models, we use an early-stopping strategy on the CIDEr score to select a final model. In contrast to models 1 and 2, we train on I3D features instead of ResNet features. Furthermore, we add audio features for the VATEX part of our training set (the TRECVID dataset does not come with audio) and train the model with the modified learning rate schedule (see Section 7.4.7). Also, we change the vocabulary to be the default BERT subtoken vocabulary with 30,522 subword tokens. In contrast to models 1 and 2, we initialize the fine-tuned model (*TRECVID-3-ft*) with the base model and train it on the same dataset (data split #4), but enable self-critical sequence learning [120] with a constant learning rate $\eta = 5 \cdot 10^{-6}$.

TRECVID-4 Our fourth model has the same configuration as model 3, except we implemented X-linear attention blocks [106] within our Transformer. We fine-tune this model in the same way as model *TRECVID-3*.

7.6.2 Training Setting

We train our models in a multi GPU setting, i.e., we train the model on 5 NVIDIA Tesla V100 GPUs simultaneously. We use a batch size of 16 per GPU, resulting in an effective batch size of 80. We use the Adam [75] optimizer with $\beta_1 = 0.9, \beta_2 = 0.98$ and $\epsilon = 10^{-9}$. Similar to [137], we use a variable learning rate η over the course of the training. That is, we use a linearly increasing learning rate in a warm-up phase and decrease it afterwards proportionally to the inverse square root of the current training iteration. We employ the SCST strategy presented in Section 7.5.2.7 only for models *TRECVID-3* and *TRECVID-4*. In contrast to the original Transformer architecture, we use 10,000 for the number of warm-up steps.

Table 7.5: Data sources used for training our TRECVID-VTT base models. We also depict the total number of training and validation samples used. Extended means that we were able to use more videos for the second TRECVID challenge [56] in contrast to the first challenge [55].

Data split #	Data sources	# train samples	# val samples
1	MSR-VTT + 90 % TRECVID-VTT	175,902	2285
2	MSR-VTT + AC-GIF + 90 % TRECVID-VTT	303,380	2285
3	90 % TRECVID-VTT	20,262	2285
4	VATEX + 90 % TRECVID-VTT (extended)	273,314	3602

7.6.3 Results

First, we evaluate our models on a small validation part of the TRECVID-VTT dataset (the remaining 10 %) and present the results in Table 7.6. We already discussed above that we employed an early stopping strategy on the CIDEr score to select the best-performing models. Second, we discuss the performance on the held-out test set of these submitted models.

7.6.3.1 Ablation Study on the Validation Set

For the base model of our first model (*TRECVID-1*), we observed the best validation performance on TRECVID-VTT after 25 epochs with a CIDEr score of 17.55. We use this model to fine-tune only on the TRECVID-VTT dataset and call it *TRECVID-1-ft*. In doing so, we slightly improved the scores as can be seen in Table 7.6. We selected the final models for the TRECVID dataset based on the CIDEr scores and depict generated captions from model *TRECVID-1-ft* on the validation set in Figure 7.11.

For our second model, we chose the same approach but trained the base model on more data sources, namely MSR-VTT, AC-GIF and 90 % of TRECVID-VTT. The best scores were also observed after 25 epochs and are in the same range as our primary model. However, when fine-tuning the second model on MSR-VTT only, the scores decrease except the CIDEr score as can be seen in Table 7.6.

The third base model (*TRECVID-3*) reached its best validation scores after training for 43 epochs with a CIDEr score of 24.94. As described above, we fine-tuned this model with self-critical sequence learning (*TRECVID-3-ft*). In doing so, we significantly improved the scores as can be seen in Table 7.6. Furthermore, we clearly see that our observations from Section 7.5 also hold true for the TRECVID dataset: The combination of I3D features, audio features, the updated learning rate schedule, and self-critical sequence learning significantly boosts scores. In fact, we improve the CIDEr score by 13.14 (i.e., by 75 %) in comparison to *TRECVID-1-ft*.

For the fourth model (*TRECVID-4*), we chose the same extensions as for model 3 but trained the base model with a Transformer that employs X-linear attention blocks [106]. The best scores were observed after 15 epochs and the BLEU-4 and METEOR scores are in the same range as model *TRECVID-3*. However, the scores are not as good as *TRECVID-3*. Similarly, when fine-tuning the third model with self-critical

sequence learning, the scores did improve in contrast to the base model. Still, our model *TRECVID-3-ft* performs significantly better than *TRECVID-3*.

Table 7.6: Models and their respective validation scores. We highlight our final models in **bold** (first column). These models were submitted in order to be evaluated on the held-out test set. We validated all of our models after every epoch on 10 % of the TRECVID-VTT dataset to select a model with an early stopping strategy (column epoch stands for the training iteration we selected).

Model	epochs	ft	Features	|mv|	Vocabulary	lr Schedule	B-4	C	M
TRECVID-1	25	—	CNN	64	Default	Default	7.60	17.55	11.61
TRECVID-1-ft	8	✓	CNN	64	Default	Default	7.13	17.61	11.69
TRECVID-2	25	—	CNN	64	Default	Default	7.52	13.84	11.96
TRECVID-2-ft	1	✓	CNN	64	Default	Default	6.10	15.07	10.96
TRECVID-3	43	—	I3D	64	WP-BERT	sgdr	10.06	24.94	13.50
TRECVID-3-ft	3	✓	I3D	64	WP-BERT	$5 \cdot 10^{-6}$	**14.22**	**30.75**	**15.99**
TRECVID-4	15	—	I3D	64	WP-BERT	sgdr	10.88	22.63	13.65
TRECVID-4-ft	0.33	✓	I3D	64	WP-BERT	$5 \cdot 10^{-6}$	11.54	24.37	14.18

7.6.3.2 Performance on the Test Set

We submitted captions generated on the provided test videos (1700) to the TRECVID workshop [8, 7, 6] organizers, who evaluated these captions on the held-out test dataset.

Compared to our validation set scores, the evaluation on the test set yields worse results as can be been in Table 7.7. Especially, the BLEU score is much lower on the test data than on the evaluation data. We suspect that the TRECVID workshop organizers use a different script for the BLEU metric. However, the METEOR score is better on the test set. Our best model *TRECVID-3-ft* improves the CIDEr score by 17.50 (i.e., by 125 %) in comparison to *TRECVID-1-ft*.

We depict four videos and their generated captions in Figure 7.11. We see that for the first three videos our generated captions match the video content quite well. The first video does indeed look like a man talking to people in a classroom. Only if we look closer, we see that this is not a classroom, but rather some presentation in front of adults. In the second video, our model detects a young man singing and fails to recognize that there are two men, one of which is singing and the other is playing the piano. In the third video, our model detects a group of people who seem to be dancing. But it places them on a beach rather than in a pedestrian zone. In the fourth video, our model wrongly assumes that there is snow in a parking lot. However, it recognizes a man moving on the street in the daytime, but not the bicycle.

In Figure 7.12 we depict five more videos, but with captions generated by our better models *TRECVID-3-ft* and *TRECID-4-ft*. We see that for the first three videos our generated captions from the model *TRECVID-3-ft* match the video content quite well: The first video description is correct. Only if we look closer, we see that one person is giving the other person a massage instead of smiling at the camera. In the second video, our model detects correctly that we see a football field and a group of people who are

indeed playing football. In the third video, our model detects a young woman who looks into a camera. However, it fails to detect that the woman is cheering in the background with some apples laying in front of her. For the fourth video, the model correctly detects a man. But the man is not reading a book, rather he is showing an ad in front of his notebook. In the fifth video, the model detects that there is a basketball game going on. However, it shows the audience rather than basketball players sitting on a bench. But in the first frame, we see a basketball player, hence, the model may take this as a hint for generating the sentence.

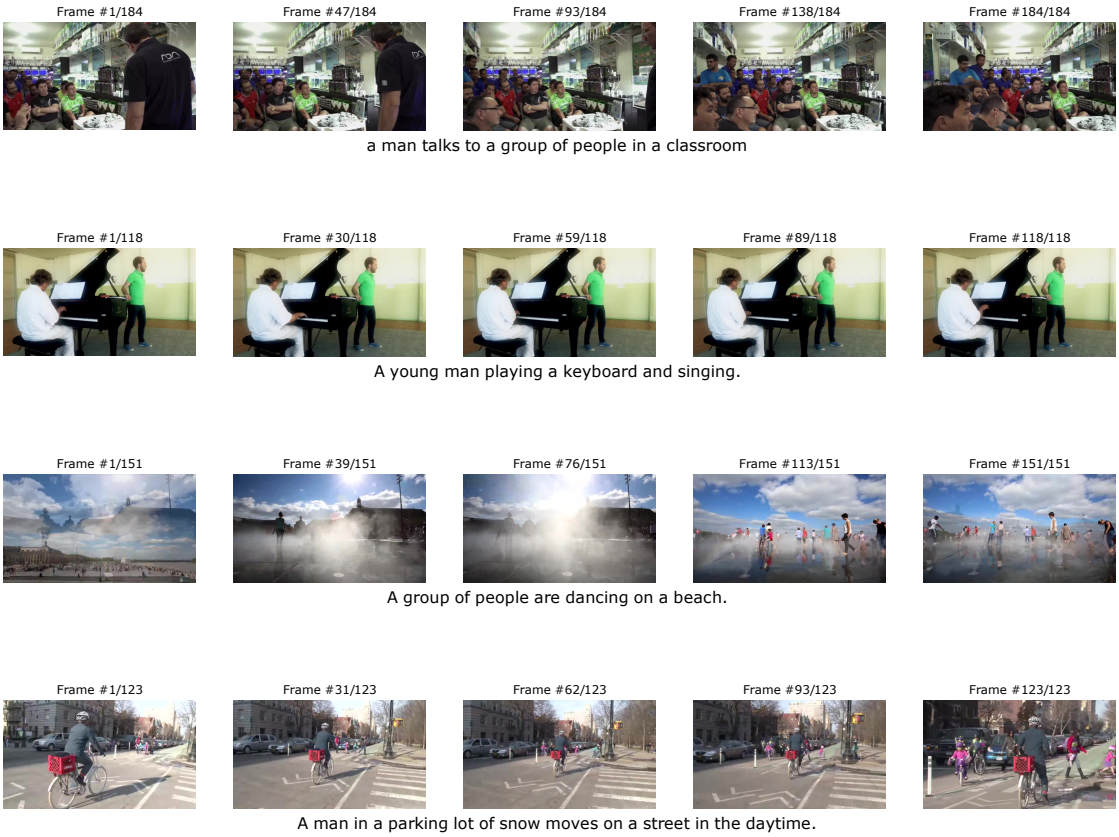


Figure 7.11: Four videos from the TRECVID-VTT [6] validation split and the corresponding captions generated by our model *TRECVID-1-ft*.

7.7 Summary

In this chapter, we presented a Transformer-based video-to-text architecture aimed at generating descriptions for short videos. Utilizing promising approaches from the related field of image captioning, we were able to gradually improve a vanilla Transformer designed for machine translation into an architecture that generates appropriate and matching captions for video clips. By combining motion features, audio features, a

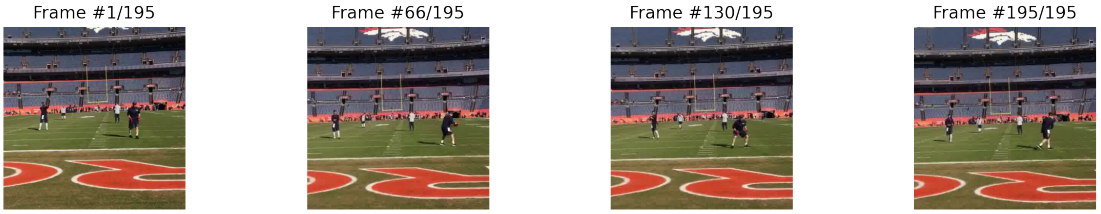
Table 7.7: Chosen models on the TRECVID-VTT dataset and their respective performance on the unseen test dataset.

Model	BLEU	CIDEr	METEOR
TRECVID-1-ft	1.80	14.00	20.20
TRECVID-2-ft	1.10	13.60	20.40
TRECVID-3-ft	**2.21**	**31.50**	**29.22**
TRECVID-4-ft	1.52	24.70	25.98

custom learning rate schedule, and a pretrained vocabulary, we establish a solid video captioning model. Furthermore, we introduce the novel fractional positional encoding to properly synchronize video and audio features with different sampling rates, which significantly improves results across all metrics. In combination with self-critical sequence training, we were able to considerably boost the performance of a baseline model by an absolute of 37.13 points or 210% in the CIDEr metric. Also, our final model configuration performs well on the MSR-VTT and MSVD datasets and even reaches near state-of-the-art performance without the need of an ensemble of up to 32 single models as in [163].

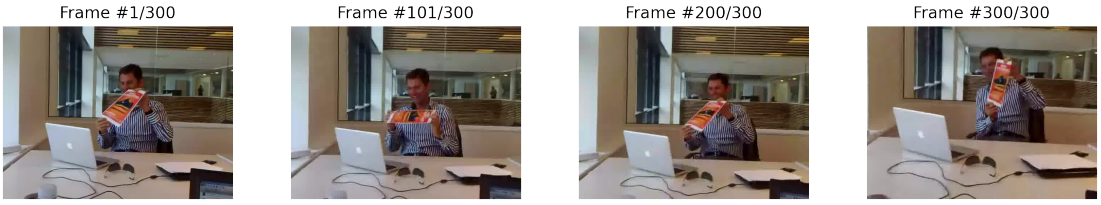
Finally, we present our findings on an entirely different dataset, namely the TRECVID-VTT dataset. In conclusion, we find that we can easily transfer our findings to other datasets. The relative improvements did also reflect on an ablation study for the TRECVID-VTT dataset and our models improved in the same way on the held-out test set.

TRECVID-3-ft young asian boy is holding another boy in front of him and smiles at the camera .
TRECVID-4-ft a young asian woman is crying in front of a camera in a room

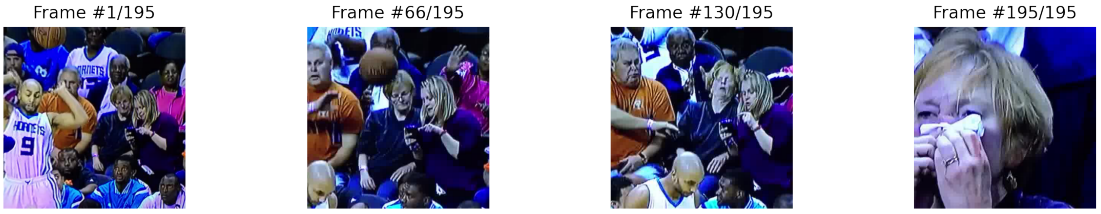


TRECVID-3-ft a group of people are playing football on a field .
TRECVID-4-ft a football player is running on a football field and kicks a field goal . and

TRECVID-3-ft a young asian woman talking to the camera .
TRECVID-4-ft a young asian woman is talking to the camera in a room



TRECVID-3-ft a man is sitting at a table and reading a book in a room .
TRECVID-4-ft a young man is using a knife to open a box . and



TRECVID-3-ft a group of basketball players are sitting on a bench at a game .
TRECVID-4-ft a group of soccer players are sitting in a basketball court and in front of a man

Figure 7.12: Five videos from the TRECVID-VTT validation dataset [6] and the corresponding captions generated by our models *TRECVID-3-ft* and *TRECVID-4-ft*.

8 Image Description Generation with Transformer Networks

Recurrent neural networks are a common architecture to model natural language generation tasks. Especially long short-term memory (LSTM) networks in combination with DCNNs are used to generate descriptions of images [139, 73, 71] (see also Chapters 4, 5, and 6). The architectures have matured over the years and introduced self-attention for LSTM layers [152]. These methods have also become more and more popular for machine translation tasks, whose encoder-decoder architecture originally inspired the Show and Tell model of Vinyals et al. [139]. Recently, Vaswani et al. [137] introduced a novel network architecture that is solely based on attention mechanisms and gets rid of convolutions altogether. Given the massive improvements in the task of sequence transduction and machine translation, it is natural to adapt this technique to image captioning [24].

In the following, we will revisit the models and problems from Chapters 4 and 5. The idea is to improve performance and make the models easier to train by porting them to the Transformer architecture [137]. Our main goal is to adapt these models to a more recent architecture and show that our previous ideas and concepts are also applicable and suitable for Transformers.

We will introduce modified architectures that include a Transformer style encoder-decoder. Afterwards, we try to calculate a baseline score and then implement the same ideas and changes from Chapters 4 and 5. Finally, in quantitative experiments, we compare the scores against the baseline Transformer models and their respective architectures from Part II. We find that the scores improve in contrast to the older reference models but also discover during the experimental evaluation that not every aspect of the results behaves as desired. For example, the SCA of our image captioning models decrease a little bit. Furthermore, the VQA-based Transformer does not generate as many new answers as its LSTM-based counterpart.

8.1 Related Work

In the following, we review related works about Transformer architectures for image captioning, visual question answering and also present a few works on non-autoregressive models for sequence generation as we make use of this technique for our Transformer-based VQA model.

Transformers for Image Captioning In the previous chapter, we already reviewed the work by Cornia et al. [24]. They presented an image captioning Transformer that makes

use of memory-augmented attention that allows to learn a-priori knowledge about relationships between image regions. Li et al. [83] introduce an *entangled transformer* for image captioning. Particularly, they introduce *entangled attention* that allows the Transformer to exploit semantic and visual information simultaneously. This semantic information is gathered by extracting semantic attributes (e.g., sidewalk street, dog, bike) with the same backbone DCNN but trained as a classification task on the 1000 most common words of the dataset [35]. Furthermore, He et al. [60] propose the *image transformer* that contains specifically designed encoder and decoder blocks. The encoder operates on three hierarchies of image regions: for every query region in the image there are its neighbour regions and child regions. They combine an LSTM layer with a decoder layer. The LSTM layer receives its inputs from the encoder and serves as a common memory module for the decoder layer. With their modifications they achieve state-of-the-art scores using the CIDEr metric. Finally, Yu et al. [154] also present a Transformer-based image captioning architecture. Their encoder makes use of region features extracted by a Faster-RCNN [119] model. Furthermore, they align regions (i.e., same coordinates and size) from different object detector models into an aligned multi-view before passing those to the encoder. Another model of theirs does not align the regions and rather just uses detections from different object detectors as encoder inputs. In the decoder, they embed the input tokens and pass them through a one-layer LSTM network before feeding these to a standard Transformer decoder. In combination with self-critical sequence learning, they take the lead on the MSCOCO captioning leaderboard at time of writing of their paper. We refer the reader to the related work in Section 7.1 for an overview of more Transformer-based architectures for image captioning and video-to-text.

Transformers in VQA Tan et al. [134] present the LXMERT (*Learning Cross-Modality Encoder Representations from Transformers*) framework that includes a large-scale Transformer equipped with three encoders. They pre-train their model with a large number of image-sentence pairs with multiple pre-training tasks that help to learn cross-modal relationships. These pre-training tasks include masked language modeling, masked object prediction, cross-modality matching and visual question answering. Then they fine-tune the pretrained model on VQA datasets and achieve state-of-the-art results. Wang et al. introduce the *Vision-Language pretrained Model* (VLMo) [144], which is the current leader in the VQA-v2 challenge. As the name suggests, their model is also pretrained before being fine-tuned on other tasks. They train multiple encoders with their new *Mixture-of-Modality-Experts* (MoME) Transformer that can encode multiple modalities in a single Transformer block. The MoME Transformer consists of three experts: The vision expert for image encoding, a language expert for text encoding and a vision-language expert for image-text fusion. They also pre-train VLMo on different pretraining tasks and fine-tune on the VQA task. Most other Tranformer-based works which achieved recent state-of-the art scores, are also trained on multiple pre-training tasks [84, 86, 20, 96].

Non-Autoregressive Methods for Sequence Generation In contrast to the Transformer-based VQA models from above, our VQA model is different in three ways: First, following the motivation of Chapter 5, we do not implement a classification approach for answering visual questions. Second, we do not employ pre-training tasks and fine-tune on the VQA task subsequently. Third, in contrast to all other models in this thesis, we generate answers in a non-autoregressive manner. Autoregressive models sample a sequence word by word whereas non-autoregressive models generate a sequence at once.

There have been some recent works for sequence generation with non-autoregressive models. For example, Gu et al. [46] present a NMT model that avoids the autoregressive way of generating text. By employing a few tricks such as self-critical sequence training and employing an autoregressive Transformer network as a teacher, they reach good performance scores. In particular, their non-autoregressive model performs 2.0 points worse in BLEU-4 than the autoregressive model on the *WMT14 En-De* machine translation corpus. Furthermore, Zhou et al. [162] present a *semi-autoregressive model for image captioning* (SATIC) that keeps an autoregressive property of the decoder while it predicts word chunks of the caption in a non-autoregressive fashion in parallel. Still, similar to the NMT task this model performs a little worse than its autoregressive counterpart. Gao et al. [41] present a non-autoregressive image captioning model where they mask different parts of the input sequences during training and remove these masks during inference. The performance of their masked non-autoregressive model surpasses the scores of the baseline non-autoregressive model by a lot but is still worse than comparable autoregressive models. There are other works that utilize non-autoregressive Transformers for the image captioning task: FNIC [37], MIR [81], and CMAL [48]. However, they all perform worse (see [162]) than autoregressive Transformer image captioning models.

In summary, most language generation models use auto-regression: For example the GPT models [14] and our VTT model (see Chapter 7). In contrast, most non-autoregressive models still perform worse than their autoregressive counterparts (see above). Particularly, these non-autoregressive models suffer from problems such as word repetition or the omission of words. However, as there is no iterative sampling involved and the sequences are generated at once, these models can decode sequences much faster than autoregressive models as the decoder does not need to be computed for every iteration step.

8.2 Transformer for Image Captioning with Multi-Task Training

Note that this section builds on Chapters 4 and 7. Therefore, we make use of definitions which we introduced in these chapters and refer the reader to these at multiple occasions. As opposed to our VQA model in the next section, we employ an autoregressive decoding strategy for the image captioning model in this section.

In particular, we look once more at the task presented in Chapter 4. More specifically, our goal is to generate captions for images that depict interactions between humans and branded products. Following the theme of this chapter, we replace the encoder-

decoder Show and Tell architecture (see Section 4.2 and [139]) with a Transformer-based encoder-decoder architecture. By doing so, we hope to develop a model that is easier to train and ultimately yields better scores than our LSTM-based model. Furthermore, we add our extensions from Chapter 4 to the Transformer-based model. Namely, we utilize the classification-aware loss (see Section 4.5.1) and image ratings module (see Section 4.5.2). In summary, our goal is to show that we can transfer our proposed methods from Chapter 4 to a state-of-the-art architecture based on the Transformer. Finally, we evaluate the new architecture and examine whether our conclusions from Chapter 4 are also valid for the Transformer-based model.

8.2.1 Transformer-Based Image Captioning Model

In Figure 8.1, we depict our Transformer-based image captioning model. The encoder and decoder of our model are the same as in Chapter 7 which we show in the center of the figure. Furthermore, we add the image ratings module and the classification-aware loss to the model. We show the image ratings module on the left side. The classification-aware loss is implemented identical to Section 4.5.1. It only operates on classwords, i.e., words of the vocabulary that represent brands. We depict the masking of words required for the classification-aware loss on the top center.

Encoder Similar to the baseline model of Chapter 7 (see Section 7.3), we embed our input with a DCNN. However, there are some differences: First, we only have a single image instead of a video since we want to create captions for single images. Second, we extract the image's features with an Inception-v3 DCNN [132] instead of a ResNet [59]. We use the Inception-v3 features as we want to compare the results of our Transformer-based models to the models from Chapter 4. Third, as the encoder of a Transformer operates on sequences, we utilize the full feature map $\mathcal{F}^I \in \mathbb{R}^{8\times8\times2048}$ with spatial information. We reshape the feature map into a sequence of length $K = 8 \cdot 8$. Therefore, our features are now of dimension $\mathbb{R}^{K\times2048}$ and we embed these features with an image embedding layer into the Transformer's model with dimension d_{model}:

$$\mathcal{F}_e^I = \mathcal{F}^I \cdot \mathbf{W}^{e,I} + \mathbf{b}^{e,I}, \tag{8.1}$$

where $\mathbf{W}^{e,I} \in \mathbb{R}^{2048\times d_{\text{model}}}$ and $\mathbf{b}^{e,I} \in \mathbb{R}^{d_{\text{model}}}$ are the weights and biases for the image embedding, respectively. Analogous to Chapter 7.3 this now yields a sequence of embedded image features

$$\mathcal{F}_e^I = [\mathcal{F}_{e,0}^I, \ldots, \mathcal{F}_{e,K-1}^I] \tag{8.2}$$

for every spatial location within the feature map. Note that this is in contrast to the average-pooled features used in Chapter 4 which gives the Transformer an advantage. Therefore, we also conduct experiments with an average-pooled feature map in Section 8.2.3. We add the default positional encoding (see Section 2.3.4) on top of the embedded image features.

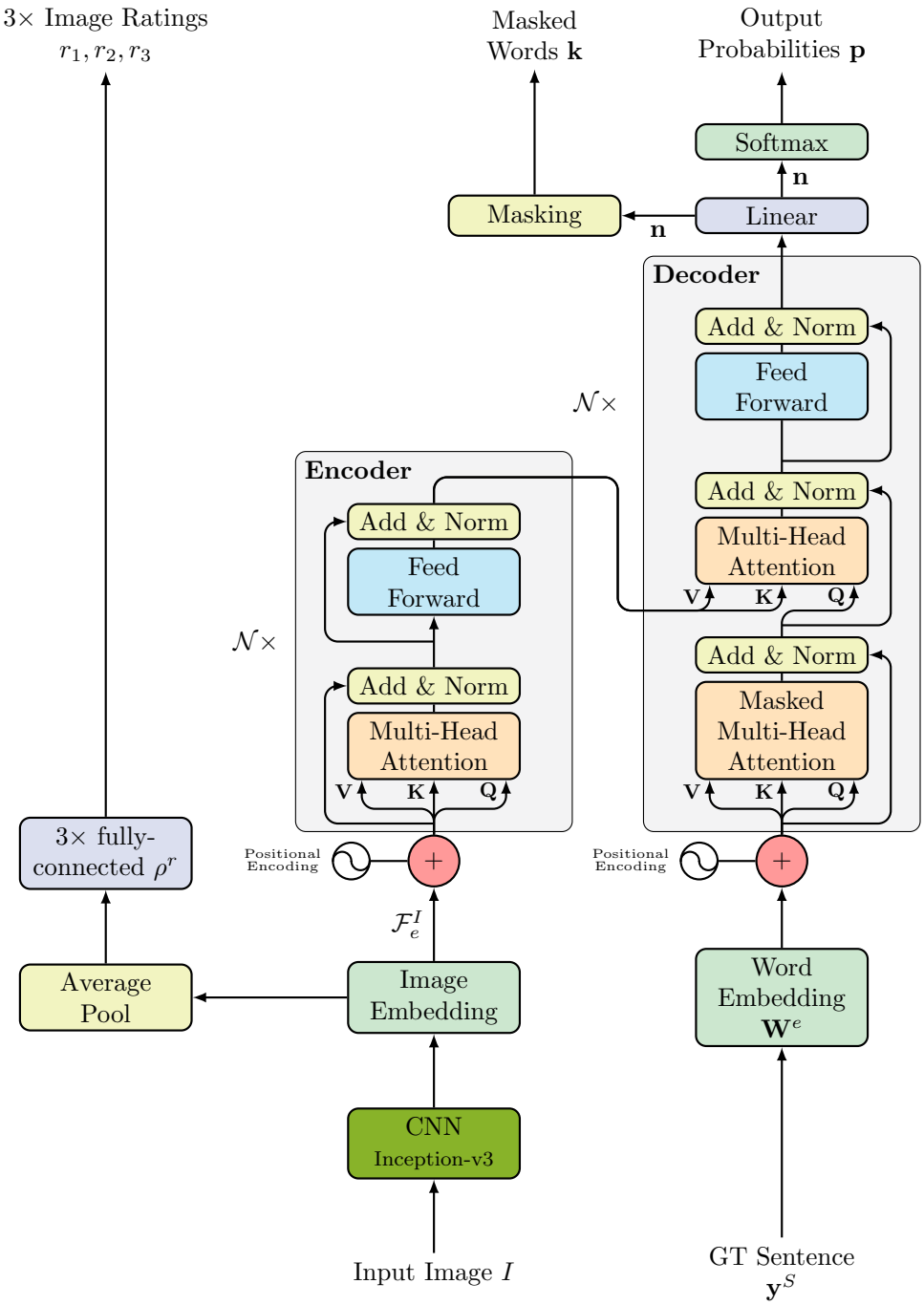


Figure 8.1: Our Transformer-based image captioning architecture for describing images that depict human-product interactions. We use the Transformer from Vaswani et al. [137] (see also Chapter 7) and add our extensions from Chapter 4. This includes the masking of output words for the classification-aware loss and the image ratings prediction.

Decoder Our decoder is implemented identical to Chapter 7 and depicted on the right side in Figure 8.1. Our ground-truth sentences $\mathbf{y}^S$ are preprocessed as outlined in Section 2.2.2. In particular, we prepend a start-of-sequence token and append an end-of-sequence token to every sentence and encode every word of a sentence with a one-hot vector. Note that we do not employ subword tokenization (see Section 7.4.3) and use the same vocabulary from Chapter 4 for representing our ground-truth sentences. We make use of a learned word embedding that embeds the input tokens into vectors of dimension d_{model}. Then, we add the default positional encoding onto the embedded vectors before feeding these to the decoder. We share the weight matrix of the word embedding with the linear projection layer (see linear layer on top of decoder) that outputs unnormalized scores for the next word. This linear layer projects the decoder's outputs back into the dimension of our vocabulary. Identical to the other chapters, we denote these outputs as $\mathbf{n}$. Then, we can optimize the cross-entropy loss for every input image I with ground-truth sentence $\mathbf{y}^S$:

$$L(I, \mathbf{y}^S) = -\sum_{t=0}^{N-2} \left(\log \left[\phi(\mathbf{n}_{t+1}) \right] \cdot \mathbf{y}_{t+1}^S \right), \tag{8.3}$$

where ϕ is the softmax activation function and $\cdot$ is the dot-product. Again, we optimize this loss for every word except the start-of-sequence token, i.e., indices $[1, \ldots, N-1]$.

Classification-Aware Loss As already mentioned, we add the classification-aware loss which we introduced in Chapter 4 to the Transformer-based model. The outputs $\mathbf{n}$ of the linear projection layer after the decoder represent scores over the vocabulary $\mathbf{V}$ for every output of the decoder. Identical to Section 4.5.1, we perform an element-wise multiplication of these outputs with a constant binary mask vector $\mathbf{m}$:

$$\mathbf{k}_{t+1} = \mathbf{n}_{t+1} \circ \mathbf{m}, \tag{8.4}$$

The mask vector has a one at every index of the vocabulary where the corresponding word is a classword (see Equation 4.5). We depict this operation by the yellow operation *Masking* on the top center of Figure 8.1 which yields masked words $\mathbf{k}$. Each of these masked words $\mathbf{k}_{t+1}$ represents the scores for iteration step t over the whole vocabulary but is set to zero for all words other than a classword. Because a classword should only occur once in a generated sentence, we sum over all iteration steps and apply the softmax function (ϕ):

$$\widehat{\widetilde{\mathbf{k}}} = \phi \left(\sum_{t=0}^{N-2} \mathbf{k}_{t+1} \right). \tag{8.5}$$

Then we can calculate the classification-aware loss L^{cls} just as in Equation 4.9:

$$L^{\text{cls}}(I, \mathbf{y}^{\text{cls}}) = -\log \left[\widehat{\widetilde{\mathbf{k}}} \right] \cdot \mathbf{y}^{\text{cls}}, \tag{8.6}$$

where $\cdot$ is the dot-product and $\mathbf{y}^{\text{cls}}$ is the one-hot encoded ground-truth vector for the classword included within the image I.

Image Ratings Furthermore, we extend our model with the image ratings module from Chapter 4.5.1. Again, we only use the soft-targets version (see Section 4.5.2.3) of the image ratings prediction because we chose this particular approach for our final LSTM-based model. We depict the image ratings module on the left side in Figure 8.1. In contrast to our LSTM-based model, we make use of spatial features in the Transformer-based model. For the image ratings module, however, we only need a flattened feature vector. Therefore, we first average-pool the embedded image features. Then, for every one of the three image ratings $r \in r_1, r_2, r_3$, we forward the average-pooled embedded features through a separate fully-connected layer ρ^r. As we use the soft-targets approach, the fully-connected layers have 5 output neurons and our ground-truth $\mathbf{y}^r \in \mathbb{R}^5$ is again encoded as a probability distribution. Then, we optimize the same loss as in Equation 4.14:

$$L^r(I, \mathbf{y}^r) = -\sum_{i=0}^{4} \left(\mathbf{y}_i^r \cdot \log\left[\sigma(\rho_i^r)\right] + (1 - \mathbf{y}_i^r) \cdot \log\left[1 - \sigma(\rho_i^r)\right]\right). \tag{8.7}$$

Identical to Section 4.5.2.3, we want to learn the probability of each individual rating. Thus, $\mathbf{y}_i^r$ and ρ_i^r are the i-th element of the ground-truth distribution and i-th neuron of the fully-connected layer's output, respectively.

Optimizing the Transformer-Based Model For our final models with classification-aware loss and image ratings module, we optimize all losses simultaneously:

$$\begin{aligned} L^{\text{total}} =& L(I, \mathbf{y}^S) + L^{\text{cls}}(I, \mathbf{y}^{\text{cls}}) + \\ & L^{r_1}(I, \mathbf{y}^{r_1}) + L^{r_2}(I, \mathbf{y}^{r_1}) + L^{r_3}(I, \mathbf{y}^{r_1}). \end{aligned} \tag{8.8}$$

This is the same loss as in Equation 4.15. Depending on the model configuration, we only optimize particular addends of the sum: For example, when the classification-aware loss and the ratings losses are disabled, we only optimize $L(I, \mathbf{y}^S)$.

8.2.2 Dataset and Training Configuration

Dataset Splits Like in Section 4.4, we use the *LogosSimple* and *LogosExtended* dataset splits. First, the *LogosSimple* dataset that only consists of images, descriptions and image ratings from the GfK-Dataset. Second, we make use of the *LogosExtended* dataset that consists of the training split of MSCOCO [91] and *LogosSimple* multiplied 8 times to compensate for the lower amount of training samples (see Table 4.1).

Training Configuration Our Transformer-based image captioning models consist of $\mathcal{N} = 6$ encoder and decoder blocks. We use 8 attention heads and a model dimension of $d_{\text{model}} = 512$. For the position-wise feed-forward networks, we set $d_{\text{ff}} = 2048$ as the inner-layer dimensionality. We use the Adam [75] optimizer with the default learning rate schedule from [137] (see also *schedule-default* in Section 7.4.4). Furthermore, we set the optimizer parameters to $\beta_1 = 0.9, \beta_2 = 0.999$ and $\epsilon = 1 \cdot 10^{-8}$. Similar to Chapter 7,

we set the number of warm-up steps to 10,000. We train all our models with a single NVIDIA A100 GPU and a batch size of 128. We select our models with an early-stopping strategy based on the CIDEr score and generate our captions with greedy sampling as opposed to beam search in Chapter 4. However, we utilize beam search decoding for our final Transformer-based model to be comparable to our older models. We implemented our LSTM-based model from Chapter 4 in TensorFlow 1 and our Transformer-based model in TensorFlow 2. Our weights from the Inception-v3 DCNN, which we fine-tuned on our 26 brand classes, are not compatible with the newer TensorFlow version. A conversion was also not possible as the networks are implemented in a different manner in both versions of the framework. Thus, in contrast to the final models of Chapter 4, we initialize the Inception-v3 DCNN only with weights pretrained on the 1000 ImageNet classes [122].

8.2.3 Experiments

In this section, we compare our Transformer-based model to our final results from Chapter 4. In particular, we selected the model *base* and the model *soft-targets+CA+ft* as reference models. Note that *base* is initialized with an Inception-v3 pretrained on the 1000 ImageNet classes. In contrast, the Inception-v3 of *soft-targets+CA+ft* was initially fine-tuned on our 26 brand classes.

base is our previous baseline, thus, we can compare it to the new baseline *baseT* because both include a Incpetion-v3 DCNN trained on the 1000 ImageNet classes. We denote all Transformer-based models with a "*T*" in the name. Furthermore, we chose *soft-targets+CA+ft* as a second reference model as this was our finally selected model from Chapter 4.

8.2.3.1 BLEU-4, METEOR and CIDEr Scores

We list the BLEU-4, METEOR and CIDEr scores of our models in Table 8.1. As comparison to the model *base* of Chapter 4, we train the model *baseT*. We initialize this model with an Inception-v3 that was pretrained on the 1000 ImageNet classes [122]. For a fair comparison, we only use the average-pooled feature map $\tilde{\mathcal{F}}^I$ in this experiment because our models from Chapter 4 did not employ spatial features. We see that all scores decrease a little bit. The CIDEr score is down by 2.25 points and BLEU-4 drops by 0.65 points. However, note that we use the full encoder even though the input sequence only has a length of 1 in this case. This means, that $\mathcal{N} = 6$ encoder blocks process a sequence of length 1. For another comparison to the LSTM-based model *base*, we removed the Transformer's encoder blocks altogether in model *baseT-Enc*. This model is closest to the LSTM-based model as we only extract average-pooled features with the Inception-v3, embed those with an image embedding layer, and pass the embedded features to $\mathcal{N} = 6$ Transformer decoder blocks. We see that this model performs similar to *base*. The CIDEr score is a little higher, while the BLEU-4 and METEOR scores are slightly less. We conclude that the Transformer's encoder is unnecessary when working on sequences of length 1. This comes as no surprise because the self-attention is effectively not present for

Table 8.1: Results for our Transformer-based models. The first column states the model number and the second column the model name, columns 3–5 state the dataset used (DS), whether we trained with the classification-aware loss (CA) and the kind of image ratings loss (IR). Columns 6–8 show the BLEU-4 (B-4), METEOR (Met) and CIDEr (Cid) scores. The last two columns show the overall accuraccy (OA) and mean accuracy (MA). *-BS* denotes that the model was evaluated with beam search decoding.

#	Method (Initialization)	DS	CA	IR	B-4	Met	Cid	OA	MA
—	**base (IC-v3)**	LS	—	LIN	54.80	34.10	177.10	75.66	58.72
1	**baseT (IC-v3)**	LS	—	—	54.15	33.22	174.84	69.15	49.84
2	**baseT-Enc (IC-v3)**	LS	—	—	54.28	33.98	177.99	71.60	59.11
3	**baseT+Spatial (IC-v3)**	LS	—	—	56.96	35.46	194.82	79.62	64.66
4	**baseT+Spatial+ft (IC-v3)**	LS	—	—	62.74	37.61	221.27	86.42	72.65
5	**baseT+Spatial+CA (IC-v3)**	LS	✓	ST	56.08	35.64	192.10	79.43	64.26
6	**baseT+Spatial+CA+ft (#4)**	LS	✓	ST	60.97	38.29	214.91	86.13	72.29
7	**fuseT+Spatial+CA (IC-v3)**	LE	✓	ST	57.19	35.88	195.25	78.81	62.69
8	**fuseT+Spatial+CA+ft (#7)**	LE	✓	ST	61.01	37.82	214.96	84.96	74.46
9	**soft-targetsT+Spatial+CA+ft (#8)**	LS	✓	ST	63.52	38.45	225.05	87.74	75.73
9-BS	**soft-targetsT+Spatial+CA+ft (#8)**	LS	✓	ST	**65.40**	**38.80**	**226.80**	89.25	**80.73**
—	**soft-targets+CA+ft (#10)**	LS	✓	ST	61.30	37.56	207.25	**91.11**	80.45

sequences of length 1. Particularly, the calculation of attention scores with the softmax function in Equation 2.14 returns a single attention weight of 1. Therefore, each encoder block only executes a projection of the values with layer normalization, followed by a feed-forward network with layer normalization both of which include a skip connection.

As a Transformer's encoder normally operates on a sequence of inputs, we embed all spatial locations of the output feature map of the Inception-v3 DCNN and reshape the spatial dimension into a one-dimensional sequence dimension in the models denoted by *+Spatial*. We see that the model *baseT+Spatial* improves scores significantly in comparison to *base*. CIDEr improves by 17.72 points and BLEU-4 by 2.16. When enabling fine-tuning of the Inception-v3 DCNN, scores improve even more. Note that models 1–3 are comparable to model *base* from Chapter 4 even though we did not train the image ratings module. This is because the image ratings module can only influence the caption generation through the Inception-v3 feature extractor network which we kept frozen for those models. The model *baseT+Spatial+ft* increases CIDEr and BLEU-4 scores by another 26.45 and 5.78 points, respectively. We see that *baseT+Spatial+ft* already surpasses all three common sentence scores of our final LSTM-based model *soft-targets+CA+ft*. Note that this model does not make use of the classification-aware loss or the multi-task learning objective introduced with the image ratings module.

When enabling the classification-aware loss in models 5 and 6, we notice that the scores decrease in comparison to models 3 and 4. This behavior is identical to the models in Chapter 4. However, the scores only decrease a little bit in comparison to the LSTM-based models. For instance, the CIDEr score of *baseT+Spatial+CA* decreases by

2.72 when compared against model #3. In Chapter 4, the model *base+CA* looses 21.17 points in CIDEr when compared against *base*.

In similar fashion to Chapter 4, we also train models on the *LogosExtended* (LE) data split which includes the MSCOCO train split in addition to the GfK-Captions dataset. First, we train model *fuseT+Spatial+CA* which yields slightly better results than the model trained on *LogosSimple* only. Furthermore, during fine-tuning (model *fuseT+Spatial+CA+ft*), we see that the scores develop into the same region as the corresponding model (model *baseT+Spatial+CA+ft*) trained on the *LogosSimple* data split.

Finally, just as in Chapter 4, we initialize our final model *soft-targetsT+Spatial+CA* with the weights from model *fuseT+Spatial+CA+ft* and fine-tune on the *LogoSimple* dataset only. During fine-tuning, we also allow changes to the Inception-v3 feature extractor network. This model scores even higher BLEU-4, METEOR and CIDEr scores for the *LogosSimple* dataset. In particular when compared to the LSTM-based model *base*, it improves the BLEU-4, METEOR, and CIDEr scores by 8.72, 4.35, and 47.95, respectively. Furthermore, it also boosts the scores of our final LSTM-based model *soft-targets+CA+ft* by 2.22, 0.89, and 17.8 points for BLEU-4, METEOR, and CIDEr, respectively. Until now we generated the captions for all models with greedy sampling. In Chapter 4, however, we employed the beam search strategy for decoding. Therefore, we additionally generated captions for our best model #9 by employing beam search decoding. When looking at these scores (denoted by model #9-BS), we see that the different sampling strategy yields the maximum BLEU-4, METEOR and CIDEr scores which we could observe for the *LogosSimple* dataset. We show predicted captions and the image ratings for the four images from Chapter 4 in Figure 8.2. More specifically, we show one of the generated captions of our final model #9-BS alongside the final LSTM-based model *soft-targets+CA+ft*.

8.2.3.2 Sentence Classification Accuracy

We also depict the overall accuracy (OA) and mean accuracy (MA) (see Section 4.6.1) in Table 8.1. These two accuracies evaluate whether the brand name of the product depicted in an input image was identified correctly. If we take a closer look to these accuracies, we see that our Transformer-based models consistently perform worse than our model from Chapter 4 with one exception. For example, model #4 only has an overall accuracy and mean accuracy of 86.42 % and 72.65 %, respectively. Compared to the model *soft-targets+CA+ft* from Chapter 4 it performs worse by an absolute of 4.69 % and 7.8 %. However, when employing beam search with a beam size of $b = 3$ just as in Chapter 4, the OA and MA increase: We additionally generated captions with beam search decoding for model #9. As more captions per sample are allowed to "match" the correct brand name (see Section 4.6.1), the score inherently increases. The MA is even slightly better than for model *soft-targets+CA+ft*. However, the OA is worse by nearly an absolute of 2 %. One factor for inferior performance could be attributed to the feature extractor DCNN: As we mentioned before, we were not able to convert the Inception-v3 parameters from TensorFlow 1 to TensorFlow 2. As outlined

in Section 4.6.2.1, we pretrained the Inception-v3 to classify the 26 brand classes of the GfK-Captions dataset. The final models of Chapter 4 were all initialized with the pretrained Inception-v3 DCNN. By employing the network that was pretrained on the brand classes, the OA and MA of the model *base* in comparison to model *base+L* (see Table 4.3) improved from 75.66 % to 90.94 % and from 58.72 % to 81.34 %, respectively.

Table 8.2: Results for our Transformer-based models. The first column states the model number and the second column the model name, columns 3–5 depict the ratings accuracies for ratings r_1, r_2, r_3.

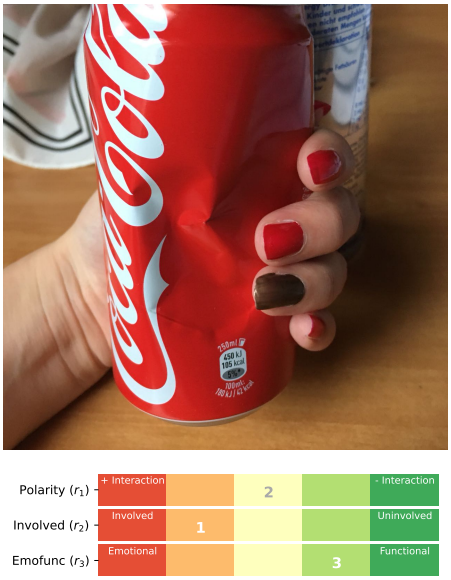
#	Method	$\textbf{accuracy}^{r_1}$	$\textbf{accuracy}^{r_2}$	$\textbf{accuracy}^{r_3}$
5	**baseT+Spatial+CA**	69.61	68.20	61.13
6	**baseT+Spatial+CA+ft**	73.50	69.61	68.90
7	**fuseT+Spatial+CA**	67.14	66.08	60.78
8	**fuseT+Spatial+CA+ft**	72.44	68.55	70.32
9	**soft-targetsT+Spatial+CA+ft**	**76.33**	68.55	**73.14**
—	**soft-targets+CA+ft**	74.60	**71.01**	70.65

8.2.3.3 Image Ratings

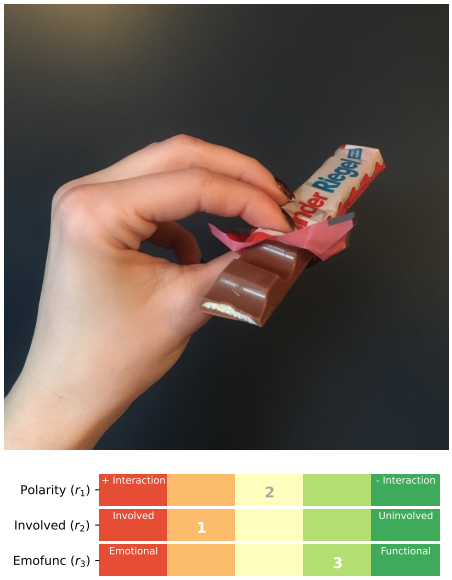
In Table 8.2, we show the image ratings accuracies of our Transformer-based models. The image ratings module is implemented in the same way as in Chapter 4. In particular, we append a single fully-connected layer ρ^r for every image rating $r \in \{r_1, r_2, r_3\}$ to the average-pooled features of the feature extractor DCNN. The only difference to our models from Chapter 4 is that we train with the Adam optimizer. Models #5 and #7 perform worse than our reference model *soft-targets+CA+ft* because we did not unfreeze the parameters of the Inception-v3 DCNN. If we look at the models where we allow fine-tuning of the feature extractor network, we see that the accuracies improve. Specifically, our model *soft-targetsT+Spatial+CA+ft* performs considerably better for the polarity (r_1) rating and the emofunc (r_3) rating. Only the involved (r_2) rating is worse by an absolute of 2.46 %. When we compare the accuracies to the LSTM-based models trained with the soft-targets approach (see Table 4.5), we see that the accuracies for ratings r_1 and r_2 of the Transformer-based models are within the same range. Only the accuracy for the emofunc rating (r_3) is better by over 2 %.

8.2.3.4 SPO Accuracy Metrics

In Table 8.3, we list the results of our Transformer-based models regarding the subject-predicate-object (SPO) accuracies (see Section 4.5.3). We see that all SPO accuracies of our models with greedy sampling (models #1-9) perform worse than our reference model *soft-targets+CA+ft*. For example, model #9 has a SPO_7 accuracy of 65.47 %, i.e., all generated sentences match the subject, predicate, and object simultaneously in 65.47 % of all cases. This is worse by an absolute of 4.53 % in comparison to the reference model.



(a) LSTM: *"a female hand holds a can of co-cacola above a tiled floor."*
Model #9: *"a hand is holding a can of co-cacola."*



(b) LSTM: *"a hand is holding a bar of kinder-riegel."*
Model #9: *"a hand is holding a kinder-riegel bar."*

(c) LSTM: *"a hand is holding a can of heinz."*
Model #9: *"a hand is holding a can of heinz."*

(d) LSTM: *"a woman is holding a nutella jar in front of her face."*
Model #9: *"a woman is holding a nutella jar in front of the person 's face."*

Figure 8.2: The same images (see Figure 4.6) and the predictions by the Transformer-based model #9 compared to our final model from Chapter 4.

However, there is a discrepancy on how we calculate the SPO metrics in comparison to Section 4.6.3.4: We sample $b = 3$ captions with beam search in Chapter 4 and employ a greedy sampling strategy for our Transformer-based models. Then to calculate the SPO score, we choose the caption that matches most of the three sentence clauses defined by us, i.e., subject, predicate, and object. For example, if one of the three sentences generated by beam search matches two sentences clauses and another one three sentence clauses, we select the later one. Therefore, the SPO accuracies of the LSTM-based model *soft-targets+CA+ft* are naturally higher than those of model #9 that utilize greedy sampling. However, when we generate the captions for model #9 with beam search decoding (denoted by #9-BS), the SPO accuracies improve significantly. Now all SPO accuracies are higher than those of the LSTM-based model. By generating 3 captions for each sample with beam search decoding, the accuracy of SPO_7 rises by an absolute of 6.42 %.

Table 8.3: Results for our Transformer-based models. The first column states the model number and the second column the model name, the following eight columns represent the SPO accuracies SPO_0–SPO_7. *-BS* denotes that the model was evaluated with beam search decoding.

#	Method	SPO_0	SPO_1	SPO_2	SPO_3	SPO_4	SPO_5	SPO_6	SPO_7
1	**baseT**	**0.00**	71.32	78.21	59.15	79.62	59.91	64.91	51.32
2	**baseT-Enc**	**0.00**	75.00	80.85	63.11	77.64	61.60	65.00	53.40
3	**baseT+Spatial**	**0.00**	78.21	82.74	65.94	80.66	65.47	68.49	56.04
4	**baseT+Spatial+ft**	**0.00**	79.53	85.00	69.72	84.53	70.00	73.77	62.17
5	**baseT+Spatial+CA**	**0.00**	77.36	81.13	65.00	80.85	65.00	68.11	56.32
6	**baseT+Spatial+CA+ft**	**0.00**	83.11	87.08	73.77	84.53	72.92	75.19	65.57
7	**fuseT+Spatial+CA**	**0.00**	76.98	83.96	67.36	78.58	65.66	68.87	58.30
8	**fuseT+Spatial+CA+ft**	**0.00**	81.42	86.13	72.64	81.60	70.09	72.64	63.68
9	**soft-targetsT+Spatial+CA+ft**	**0.00**	83.30	87.36	74.62	83.11	71.32	75.19	65.47
9-BS	**soft-targetsT+Spatial+CA+ft**	**0.00**	**86.89**	**90.75**	**80.57**	**86.60**	**77.08**	**79.72**	**71.89**
—	**soft-targets+CA+ft**	**0.00**	85.94	88.87	78.49	85.66	75.75	77.74	70.00

8.3 Generating Answers with a Transformer for Visual Questions

This section builds on Chapters 5 and 7. Similar to Section 8.2, we make use of definitions which we introduced in these chapters and refer the reader to these at multiple occasions.

In this section, we investigate the VQA task in light of the Transformer architecture. Similar to Chapter 5, we want to generate answers instead of implementing a classification-based approach. As we already outlined in Section 8.1, most related works implement their VQA models as a classification task. That is, they select an answer out of a predefined set of answers. This extends to works that use the Transformer architecture as backbone where they append a fully-connected layer to the decoder. This fully-connected layer is then trained to maximize a classification objective.

Like motivated in Section 5.1, we want to develop a model that is able to generate answers instead of only selecting an answer from a predefined set of answers. More specifically, we want our model to also generate less common answers (LCA) and entirely new answers which is not possible with a classification-based model. In contrast to our other Transformer architectures of this work, we try to generate the answer at once in a non-autoregressive fashion instead of generating the answer word by word. The non-autoregressive decoding is presenting itself as an alternative to autoregressive encoding in this case because we have a complete question available during inference. In particular, when generating an answer for a given image and question, we can pass the complete question to the decoder's input and then generate an answer at once. In contrast, if we generate a sentence word by word with autoregressive decoding, we need the previously predicted words for generating a new word. By employing this architecture, the decoder then also replaces the multi-modal fusion operation from Section 5.5.4.

Finally, we evaluate the new architecture and check if the VQA accuracies are better than those of Chapter 5. Furthermore we investigate if the Transformer-based model is also able to generate LCA and new anwers.

8.3.1 Transformer-Based VQA Model

We depict our Transformer-based VQA model in Figure 8.3. Again, we make use of the encoder and decoder from Chapter 7 and Section 8.2. This model is very similar to the model from Section 8.2, which we depict in Figure 8.1. First, the model is simpler as we do not have extensions like an image ratings module or the classification-aware loss. In fact, the encoder is built the same way as both models utilize spatial feature maps from the Inception-v3 DCNN in a sequential manner. The decoder, however, is built different from the VTT Transformer model presented in Chapter 7 and the image captioning transformer from Section 8.2. Additionally, the idea of how we create answer sentences is different from a traditional text generation model that samples sentences word by word. Instead the decoder is given a question and predicts the answer sentence at once. Still, we generate answers instead of using the classification-based approach which other VQA works utilize (see Section 5.2).

Encoder As we already outlined, the encoder is essentially the same as in the image captioning Transformer of Section 8.2. However, there is one small difference: In some of our models we utilize the memory augmented encoder (see Section 7.4.1). We feed an input image I through an Inception-v3 DCNN that extracts spatial features $\mathcal{F}^I \in \mathbb{R}^{8\times8\times2048}$. Then, we reshape the image's features into a sequence of length $K = 8 \cdot 8$ and embed these image features with an image embedding layer into the Transformer's model dimension $d_{\text{model}} = 512$ (see Equation 8.1). Note that we made use of the same spatial features in Chapter 5. Afterwards, we add the positional encoding (see Section 2.10) on the embedded features and feed the resulting sequence of vectors to our encoder blocks.

Decoder We already mentioned that the decoder of our Transformer-based VQA models is built differently than the models from Section 8.2 and Chapter 7. Although the

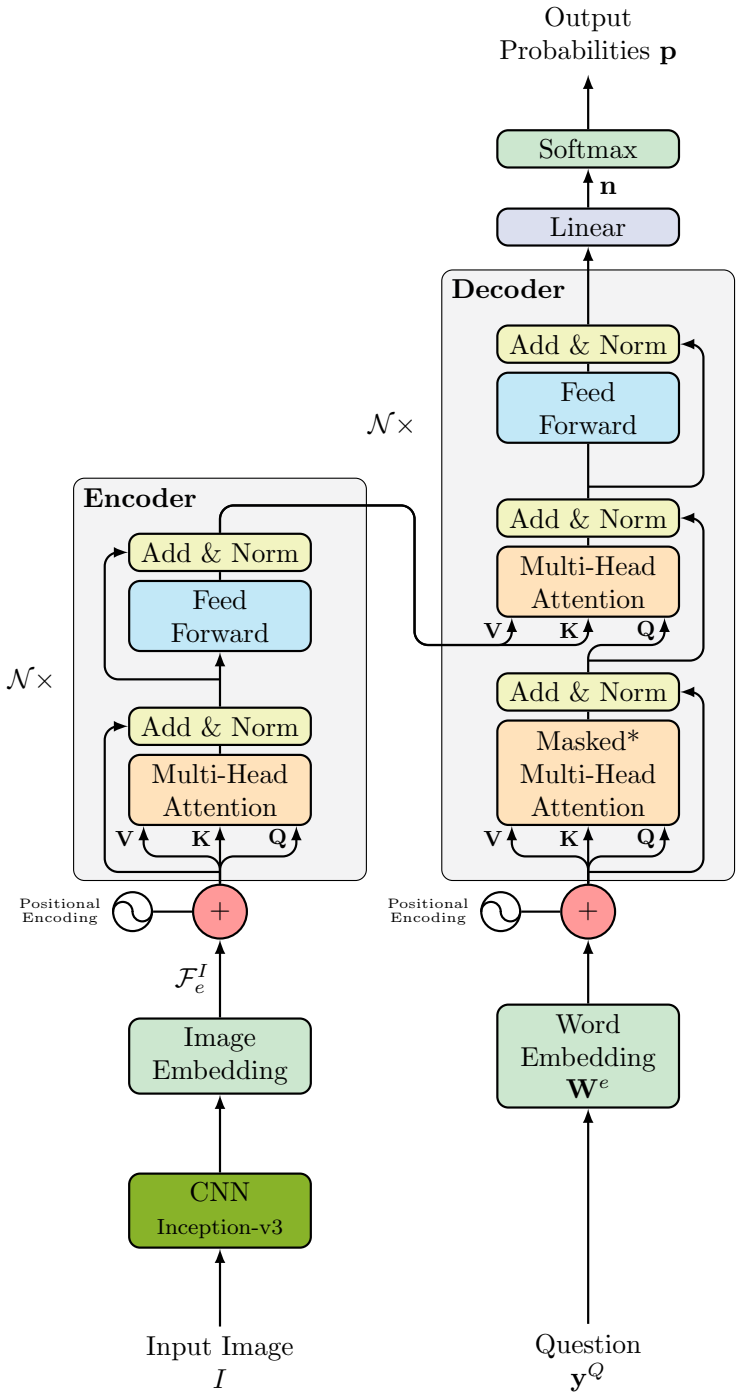


Figure 8.3: Our Transformer-based architecture that allows to generate answers at once for visual questions. We also use the Transformer from Vaswani et al. [137] (see also Chapter 7) and use it to generate answers for visual questions. We feed the decoder with embedded questions on the bottom right and directly generate the answer with the decoder. *The masked multi-head attention is different from other Transformer models (see Chapter 7 and Section 8.2.1). Specifically, we only mask padded input tokens instead of previous word indices (for more detail see Section 8.3.1).

basic structure is the same, we utilize this structure differently. Instead of feeding a ground-truth sentence as input and trying to predict the same shifted ground-truth sentence as output, we make use of different inputs and outputs: Our inputs to the decoder are embedded questions and the outputs are the answers. One crucial part of the visual question answering task is the multi-modal fusion of features (see Section 5.5.4). That is, the combination of the visual features from the input image and the features from the question asked. In our model, we combine the question features and image features in the decoder.

If we look at the decoder structure of the image captioning model in Figure 8.1 more closely, we see that the caption generation is dependent on the encoder outputs and the input to the decoder. In all previous language generation models in this work, we generate the captions in an autoregressive manner. That means that given the encoder outputs and some input words, we generate the next word with a decoding strategy like greedy sampling. Even though most non-autoregressive architectures are not superior to autoregressive models when generating text (see Section 8.1), we implement a non-autoregressive decoder. We made this architectural decision, because of multiple reasons. First, the input question is completely available during inference time. Second, adding a second decoder just for autoregressively generating answers or fusing the image and question in a different way would make our Transformer-based model more complex and difficult to train. Thus, a Transformer's decoder that can attend to the encoder's outputs given some inputs is an natural choice for fusing image and question features. In particular, the decoder then attends to the image features from the encoder given an input question. Furthermore, to keep our model consistent and comparable to the models from Chapter 5, we want to generate an answer instead of selecting the most-probable out of a predefined set of answers. Therefore, we modify the decoder of our Transformer-based VQA model to accept questions as inputs and train the objective of directly predicting answers. In contrast to the task of image captioning, we do not need to mask out input words in the decoder. The generated answer word of iteration step t is allowed to see all words of the input question as opposed to an image captioning decoder which is only allowed to see the words with iteration steps $< t$. Vaswani et al. [137] introduced this limitation so that predictions for position t can depend only on the known outputs at positions less than t.

We use a shared weight matrix for both embedding the questions and the linear projection that maps decoder outputs to the answer words. We prepend a start-of-sequence token and append an end-of-sequence token to both our questions and answers because we use the identical preprocessed dataset from Chapter 5. Furthermore, we pad all of our input questions to the length of the longest possible answer within the dataset. We need to do this because the inputs to the decoder can have different lengths than the output answers. For example, when a question only consists of five words and the corresponding ground-truth answer has a length of eight words, we could not generate answer words for the last three words of the answer. Thus, we would not be able to calculate a loss for these words. By padding the question to the length of the maximum answer length of the dataset, we make sure that the decoder always generates a sufficient number of outputs. Note that technically we still make use of the masked multi-head

attention to mask out the padded tokens of the input question. As the padded tokens are not part of the question, we do not attend over those in the decoder. We also add positional encodings to the embedded input question.

As with all our word generation models, we optimize the softmax cross-entropy loss for every image I with question $\mathbf{y}^Q$ and answer $\mathbf{y}^S$:

$$L(I, \mathbf{y}^Q, \mathbf{y}^S) = -\sum_{t=0}^{N-2} \left(\log\left[\phi(\mathbf{n}_{t+1})\right] \cdot \mathbf{y}_{t+1}^S\right), \tag{8.9}$$

where N is the length of the one-hot encoded ground-truth answer $\mathbf{y}^S$ and $\cdot$ is the dot-product. Note that the ground-truth answer includes the start-of-sequence and end-of-sequence token and we optimize the loss for word indices $[1, \ldots, N-1]$. Similar to other models in this thesis, we do not optimize for index 0 which is the start-of-sequence token. Still, we want to optimize for index $N-1$ so our final model is able to predict the end-of-sequence token and we are able to cut off the unwanted predictions.

8.3.2 Dataset and Training Configuration

Dataset We use the same preprocessed data splits from Section 5.3. Therefore, we make use of the same vocabulary and do not employ subword tokenization. Our questions and answers are also already preprocessed and include a start-of-sequence and end-of-sequence token. We train our models on the rebalanced train split which includes the original training split and 90 % of the original validation split of the VQA-v2 dataset. We validate our models on the remaining 10 % of the validation split.

Training Configuration We train our models on a single NVIDIA A100 GPU with a batch size of 1024. We empirically found that this batch size yields better results than lower batch sizes and is also faster to train. When fine-tuning the architecture, we unfreeze the Inception-v3 feature extractor network and need to lower the batch size to 128 due to memory constraints. We select our models with early stopping strategy based on the CIDEr validation score. Our Transformer uses a differing number $\mathcal{N}$ of encoder-decoder blocks which we discuss in the experiments section. We set $d_{\text{model}} = 512$ and use an inner-layer dimensionality of $d_{\text{ff}} = 2048$. As with all our other Transformer models, we use the Adam optimizer [75] with the default learning rate schedule from [137] and 10,000 warm-up steps. We also experimented with the SGDR learning rate schedule from [95] (see also Section 7.4.4). We initialize the Inception-v3 feature extractor networks with weights which were pretrained on the 1000 ImageNet classes (the same weights as in Section 8.2).

8.3.3 Experiments

Similar to Section 8.2, we compare the Transformer-based VQA models against the LSTM-based VQA models from Chapter 5. We also selected two LSTM-based reference models to which we compare the Transformer-based models. We depict the performance

of our models on the validation split in Table 8.4. Model numbers with the * suffix are LSTM-based models from Chapter 5. The other models (i.e., models #1–7) are our Transformer-based models.

8.3.3.1 VQA Accuracies

We choose the model #8 from Table 5.1 as LSTM-baseline model as it is our baseline model from Chapter 5 that utilizes soft-attention. Because the Transformer also utilizes attention over the spatial dimensions of the image's feature map, we selected this model as baseline to compare the Transformer-based VQA models to. We see that the baseline VQA Transformer (model #1) improves the overall VQA accuracy by 1.82 % in comparison to the LSTM-based model #8*.

We used $\mathcal{N} = 6$ encoder and decoder blocks for the baseline experiment because Vaswani et al. [137] proposed it in their original work and this parameter worked well for our image captioning pipeline from Section 8.2. However, as our dataset is really big and our models trained for a long time, we experimented with the number of encoder and decoder blocks and found that lowering the number of $\mathcal{N}$ improves the overall VQA accuracy. We see that the overall VQA accuracy of model #2 with $\mathcal{N} = 5$ rises by nearly 2 % in comparison to the baseline model with $\mathcal{N} = 6$. If we further reduce the number of encoder and decoder blocks to $\mathcal{N} = 4$, the performance starts to degrade (see model #3). Thus, we utilize $\mathcal{N} = 5$ encoder and decoder blocks for our final models.

We also experiment with the memory augmented encoder from Section 7.4.1. Models #1–3 already made use of the memory augmented encoder with a memory vector size of 64. We conduct two comparisons for models with no memory augmented encoding. In particular, we see that model #4 performs slightly worse than model #1 with a loss of 0.05 % in overall accuracy. However, model #5 with $\mathcal{N} = 5$ has a higher accuracy than model #2 (0.41 %). Because the memory augmented encoder does not harm performance and slightly improves it in case of model #5, we enable the memory augmented encoder with a memory vector size of 64.

For model #6, we tried to utilize the SGDR learning rate schedule from Section 7.4.4. However, as we see in the scores, this learning rate schedule does not improve scores as it does for our video-to-text Transformer. In fact, the overall accuracy decreases by 0.34 % compared to model #2. As SGDR does have the desired effect, we do not pursue it for our final model.

Finally, with model #7, we take the configuration of our best-performing Transformer-based model and additionally fine-tune the parameters of the Inception-v3 feature extractor network. In Chapter 5 this yielded an improvement of 2.91 % in the overall accuracy. With the Tranformer-based model (model #7), we also see an improvement of 2.1 % in overall accuracy. This is not as much as in the LSTM-based model, but still substantial when compared to the differences between different model configurations (compare models #1–6). Our final Transformer-based model improves performance by nearly 3 % in contrast to our final LSTM-based model.

These are scores on our validation split of the VQA-v2 dataset. We could not calculate the result of our final model on the test set as this is not public and the test server for

Table 8.4: Studies on our validation split of the VQA-v2 dataset. We analyze different combinations of hyperparameters in our model. The first column states the model #. Model numbers with the suffix * are our LSTM-based models from Table 5.1. *FT* stands for fine-tuning of all parameters (including the *Inception-v3* image feature extraction DCNN) and *lr-schedule* which learning rate schedule we used. |mv| states if we used the memory augmented encoder (see Section 7.4.1) and the size of the memory vector. The columns for validation performance show the accuracies on our validation split (10 % of the VQA-v2 validation split).

	Model Configuration				Validation Performance			
#	FT	lr Schedule	|mv|	$\mathcal{N}$	All	Y/N	Num	Other
8*	—	$\eta = 2.0$	0	—	53.54	73.33	34.81	43.34
1	—	Default	64	6	55.36	72.54	35.77	46.06
2	—	Default	64	5	57.33	75.63	38.33	46.89
3	—	Default	64	4	56.05	75.56	37.35	45.69
4	—	Default	0	6	55.41	73.59	36.49	45.53
5	—	Default	0	5	56.92	75.44	34.02	48.34
6	—	sgdr	64	5	56.99	74.67	35.99	47.70
7	✓	Default	64	5	**59.43**	**76.34**	**41.93**	**49.48**
11*	✓	$\eta = 2.0$	—	—	56.62	75.45	39.33	46.79

the specific dataset version used in Section 5.6.4 was not available. Note that current state-of-the-art methods and methods employing a Transformer such as LXMERT [134], ROSITA [25], and VLMo [144] perform substantially better than our model. However, these methods all use the classification-based approach and are therefore only comparable to our model in a limited degree.

8.3.3.2 Less Common Answers

A key contribution of Chapter 5 was the analysis on less common answers (LCA), most common answers (MCA) and new answers. We showed that our models are able to generate answers that are not contained in the 3000 most common answers and even new ones. As our Transformer-based models also generate answers instead of performing a classification task, we also investigate how these models perform for LCA, MCA, and new answers similar to Section 5.6.3. In Table 8.5, we examine the distribution of the generated answers among LCA, MCA and new answers, respectively. For example, 15.81 % of all generated answers by model #3 were less common answers, i.e., answers that are not part of the 3000 MCA within the training data split. We also list the VQA accuracy of those LCA. Thus for model #3, 16.99 % of the generated LCA were correct. Note that these are answers that cannot be given by classification-based models that

only consider the top-3000 answers. Furthermore, we list the proportion of MCA and new answers. New answers cannot be generated at all by classification-based models.

We see that the Transformer-based models have a higher LCA accuracy than the LSTM-based models. However, models #3 and #7 generate fewer LCA and new answers than models #8* and #11*. As a consequence they generate a lot more MCA than the LSTM-based models. As one of our goals is to generate new answers, this behaviour of the Transformer-based models is undesired. We suspect that this is an effect of the non-autoregressive generation of answers. We do not sample answers word by word as we did in Chapter 5, which allows for a better exploration of the room of possible answer sentences.

Table 8.5: Fractions of unique answers generated by our Transformer-based models. LCA are less common answers left out altogether by classification models. MCA are most common answers, which are the only answers generateable by classification models and new answers are newly generated sentences by our models not contained in the train split. The LCA accuracy describes the percentage of correctly generated answers out of the LCA set, i.e., these are correct answers given by our model, that classification models are not able to produce. Model numbers with the suffix ***** are our LSTM-based models from Table 5.1

#	LCA	LCA accuracy	MCA	new answers
8*	24.52 %	9.91 %	64.22 %	11.26 %
11*	19.49 %	8.10 %	69.74 %	10.77 %
3	15.81 %	16.99 %	81.83 %	2.36 %
7	13.61 %	13.42 %	81.53 %	4.86 %

8.4 Summary

In this chapter, we adapted the LSTM-based architectures from Chapter 4 and 5 to a Transformer-based architecture. Essentially, we swapped the LSTM decoder of these models with a Transformer decoder. Furthermore, instead of only encoding image features with a single image embedding layer, we utilized the encoder of a Transformer.

First, we adapted the image captioning model from Chapter 4. Instead of only using an encoder-decoder Transformer, we also transferred the classification-aware loss function and the multi-task training objective to the Transformer-based architecture. A baseline Transformer without the extensions was already able to surpass our baseline LSTM models from Chapter 4 in terms of common sentence evaluation metrics (BLEU-4, METEOR, and CIDEr). The same was true for models that did fine-tune the parameters of the Inception-v3 image feature extractor network. Finally, when utilizing the classification-aware loss and the concurrent prediction of image ratings during fine-tuning, we were able to reach the highest scores. In total, we did improve the CIDEr score of our best performing LSTM-based model from 207.25 to 226.80. We were also

able to match the performance of our image ratings module. However, one of the accuracies that measure whether we correctly identified brand names within the generated caption decreased (OA). But this problem could be addressed by utilizing the same pretraining of the Inception-v3 network from Chapter 4.

Second, we also utilized the Transformer architecture in a VQA model. Specifically, in similar fashion to Chapter 5, we created a VQA model that generates answers instead of selecting answers out of a predefined set of answers. In contrast to other Transformer models in this work, we presented a non-autoregressive model that generates the answer for the visual question at once instead of generating the answer word by word. In particular, we fed the decoder the question as inputs and predicted the answer in one step. Our final Transformer-based model did improve the VQA overall accuracy by nearly an absolute of 3 %. However, in contrast to our final LSTM-based model from Chapter 5, the model did create a higher proportion of most common answers. Thus as a consequence, the Transformer-based models do not create as many new answers as the LSTM-based models. Even though the VQA accuracies of these models are higher, this is an undesired behavior as one of our goals was to create previously unseen answers. We suspect that this is one effect of the non-autoregressive model. Investigating this downside may be one interesting idea for future research.

9 Conclusion and Outlook

9.1 Summary

In this work, we have explored different approaches for automatic generation of textual descriptions of visual data.

First, we shortly visited the area of language generation by using templates. We detected different findings within a video of a colonoscopy and determined their spatial locations. We then fed this information to a templating engine that creates a written report of said video by inserting the detections and their locations into a template.

Additionally, we investigated the task of image captioning in the special case of describing images that contain person-product interactions. One specific requirement was that generated captions should contain the correct brand name of the product, which we tackled with the classification-aware loss function. We then improved caption quality even more by optimizing a multi-task training objective that concurrently predicts three different image ratings. These ratings express the person-product interaction with three distinct properties. With two new metrics and traditional metrics, we performed a detailed analysis of our proposed image captioning model.

Furthermore, we looked at the task of visual question answering. Given its definition, this task does originally not include a language generating component but is designed to give one answer out of a predefined set of possible answers. As this limits the space of possible answers, we explored whether it is feasible to generate new natural language answers just as in the task of image captioning. We found that generating answers from scratch can have the benefit of producing new - previously unseen - and richer answers. Also, our model was able to generate less common answers which were not included in the predefined fixed set of answers.

In the last chapter of Part II, we discussed the generation of medical reports for chest X-ray images. The challenges for this task were twofold: First, doctors' reports are longer than just a single sentence. A standard LSTM often is not adequate to generate a paragraph of sentences as long-term dependencies of words spanning multiple sentences cannot be learned sufficiently. Thus, we implemented a hierarchical LSTM that allows to generate multiple sentences with two hierarchy levels of LSTM cells. In particular, one LSTM cell generates context vectors that represent a single sentence while the LSTM cell of the second hierarchy level then generates a sentence word by word when preconditioned with this context vector. Second, data bias problems made it hard to generate suitable descriptions as abnormal cases are very rare in comparison to normal cases. Thus, probabilistic models such as language models tend to overgenerate reports for normal cases. We addressed this problem by extending the hierarchical LSTM with an abnormality prediction module that selects a different LSTM depending

on whether it is more likely for a sentence to describe an abnormality or not. Also, we examined the correlation between the number of distinct sentences generated over the whole validation set and the BLEU-4 metric. These experiments allowed us to conclude that good doctors' reports do not necessarily have a good BLEU-4 score. Particularly, models that generated the same sentences for multiple samples of the validation set scored higher than models that generated sentences with higher variability.

In the third part, we looked at the Transformer architecture. Given its huge improvements both in computational efficiency and performance for language models, we applied it to our problems. First, we investigated description generation models for video clips. We implemented a Transformer to deal with the increased complexity due to video data instead of still images. We then included promising approaches from image captioning models into our video-to-text model. Furthermore, we presented the novel fractional positional encoding for Transformers that allows to easily align and synchronize audio-visual frames for video-to-text translation. In a series of experiments, we showed the effectiveness of our approach and improved scores considerably in comparison to a vanilla Transformer video-to-text model. Second, we revisited the models from Part II by changing the architectures to employ a Transformer with self-attention mechanism. The Transformer-based counterparts of the LSTM-based models from the first part of this thesis were able to improve the performance on some metrics even more. However, there were some downsides: Despite its higher CIDEr and BLEU-4 scores, the Transformer-based image captioning model was not able to include the names of the branded products with the same accuracy as its LSTM-based counterpart. The visual question answering model also achieved a higher overall accuracy but did not generate as much new answers as the LSTM-based model.

9.2 Outlook

The automated generation of textual descriptions of visual data (i.e., images and videos) has made remarkable progress in recent years. The Transformer architecture has set the bar even higher than traditional image captioning pipelines but the task of generating good and matching captions is nowhere near a solved problem. Throughout this thesis, we investigated interesting approaches and techniques that allowed us to automatically transform visual data into natural language. Although the end of the road seems to have been reached for RNN-based description generation models, there is still much potential for further research of Transformer-based models. This starts with models that implement many of the most recent approaches and are trained on huge datasets like our video-to-text model. Particularly, there are different aspects of limits that current state-of-the-art models have to struggle with: First, when we look at sentences generated by the best of our models, they might seem to mostly match the visual input, but they are far from perfect: For instance, rare activities, the scenery, or the actual count of objects are sometimes described incorrectly. Second, generated descriptions could be more detailed and vivid. This aspect, however, can only be improved by utilizing more data and condensing this knowledge into a model. Third, a huge area for improvement

are metrics that evaluate generated sentences. As we have explored a few times in this thesis, common machine translation metrics are not always sufficient or a good choice when trying to assess whether a generated description is good or bad. Here, different and task-specific metrics can be helpful to automatically determine sentence quality. A human evaluation is still the best method to detect errors in the description. Even more pressing is the evaluation of generated medical reports which not only have to "sound" good but also have to be correct from a medical perspective. Here, the manual evaluation is even more difficult and cost-intensive as only domain experts have the knowledge and experience to rate if a report is correct.

We expect future research to go into the following different directions:

One promising direction is to employ unsupervised pretraining for Transformer networks. The BERT [29] model has shown that with a large-scale unsupervised pretraining on huge existing text corpora, Transformers can vastly improve performance for related tasks such as machine translation. Also, VideoBERT [128] employed self-supervised pretraining on joint distributions over linguistic and visual tokens and improved performance on the related tasks of action classification and video captioning.

Another interesting aspect of future research is the area of architectural design decisions. Traditional DCNNs have matured over the recent years and have been improved greatly by the introduction of extensions to the architecture. The comparatively young Transformer architecture could benefit from such architectural changes. Also, integrating new basic machine learning concepts and ideas from other areas of research could improve Transformers efficiency-wise as well as performance-wise.

As we already outlined, existing metrics are far from perfect. Especially with our model for visual question answering, the existing accuracy metric is meaningless as generated answers are only compared to a pool of fixed answers. After all, our model is capable to generate more diverse answers and *new* answers. In contrast, other approaches predict the most probable answer from a set of candidate answers. Developing new metrics in order to better evaluate and understand automatically generated sentences is vital as the only alternative - manually evaluating - is far more time-intensive and expensive.

List of Figures

List of Tables

Bibliography

[1] N. Aafaq, N. Akhtar, W. Liu, S. Z. Gilani, and A. Mian. Spatio-temporal dynamics and semantic attribute enriched visual encoding for video captioning. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019, pages 12487–12496. Computer Vision Foundation / IEEE, 2019.

[2] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[3] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. A. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. Tensorflow: A system for large-scale machine learning. In K. Keeton and T. Roscoe, editors, 12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016, Savannah, GA, USA, November 2-4, 2016, pages 265–283. USENIX Association, 2016.

[4] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018, pages 6077–6086. Computer Vision Foundation / IEEE Computer Society, 2018.

[5] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh. VQA: visual question answering. In 2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015, pages 2425–2433. IEEE Computer Society, 2015.

[6] G. Awad, A. A. Butt, K. Curtis, J. Fiscus, A. Godil, Y. Lee, A. Delgado, J. Zhang, E. Godard, B. Chocot, L. Diduch, J. Liu, Y. Graham, G. J. F. Jones, , and G. Quénot. Evaluating multiple video understanding and retrieval tasks at trecvid 2021. In Proceedings of TRECVID 2021. NIST, USA, 2021.

[7] G. Awad, A. A. Butt, K. Curtis, J. G. Fiscus, A. Godil, Y. Lee, A. Delgado, J. Zhang, E. Godard, B. Chocot, L. L. Diduch, J. Liu, A. F. Smeaton, Y. Graham, G. J. F. Jones, W. Kraaij, and G. Quénot. TRECVID 2020: A comprehensive campaign for evaluating video retrieval tasks across multiple application domains. In G. Awad, A. A. Butt, K. Curtis, J. G. Fiscus, A. Godil, Y. Lee, A. Delgado, J. Zhang, E. Godard, B. Chocot, L. L. Diduch, J. Liu, A. F. Smeaton, Y. Graham, G. J. F. Jones, W. Kraaij, and G. Quénot, editors, 2020 TREC Video Retrieval Evaluation, TRECVID 2020, Gaithersburg, MD, USA, December 8-11, 2020. National Institute of Standards and Technology (NIST), 2020.

[8] G. Awad, A. A. Butt, K. Curtis, Y. Lee, J. G. Fiscus, A. Godil, A. Delgado, J. Zhang, E. Godard, L. L. Diduch, A. F. Smeaton, Y. Graham, W. Kraaij, and G. Quénot. TRECVID 2019: An evaluation campaign to benchmark video activity detection, video captioning and matching, and video search & retrieval. In G. Awad, A. A. Butt, K. Curtis, Y. Lee, J. Fiscus, A. Godil, A. Delgado, J. Zhang, E. Godard, L. L. Diduch, A. F. Smeaton, Y. Graham, W. Kraaij, and G. Quénot, editors, 2019 TREC Video Retrieval Evaluation, TRECVID 2019, Gaithersburg, MD, USA, November 12-13, 2019. National Institute of Standards and Technology (NIST), 2019.

[9] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In Y. Bengio and Y. LeCun, editors, 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.

[10] S. Banerjee and A. Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization, pages 65–72, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

[11] K. Barnard and D. A. Forsyth. Learning the semantics of words and pictures. In Proceedings of the Eighth International Conference On Computer Vision (ICCV-01), Vancouver, British Columbia, Canada, July 7-14, 2001 - Volume 2, pages 408–415. IEEE Computer Society, 2001.

[12] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada, pages 1171–1179, 2015.

[13] J. Bernal, F. J. Sánchez, and F. Vilariño. Towards automatic polyp detection with a polyp appearance model. Pattern Recognit., 45(9):3166–3182, 2012.

[14] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger,

T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, 2020.

[15] J. Carreira and A. Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. In 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, pages 4724–4733. IEEE Computer Society, 2017.

[16] R. A. Caruana. Multitask connectionist learning. In Connectionist Models Summer School, pages 372–379, 1993.

[17] D. L. Chen and W. B. Dolan. Collecting highly parallel data for paraphrase evaluation. In D. Lin, Y. Matsumoto, and R. Mihalcea, editors, The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA, pages 190–200. The Association for Computer Linguistics, 2011.

[18] S. Chen and Y. Jiang. Motion guided spatial attention for video captioning. In The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019, pages 8191–8198. AAAI Press, 2019.

[19] X. Chen, H. Fang, T. Lin, R. Vedantam, S. Gupta, P. Dollár, and C. L. Zitnick. Microsoft COCO captions: Data collection and evaluation server. CoRR, abs/1504.00325, 2015.

[20] Y. Chen, L. Li, L. Yu, A. E. Kholy, F. Ahmed, Z. Gan, Y. Cheng, and J. Liu. UNITER: universal image-text representation learning. In A. Vedaldi, H. Bischof, T. Brox, and J. Frahm, editors, Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXX, volume 12375 of Lecture Notes in Computer Science, pages 104–120. Springer, 2020.

[21] Y. Chen, S. Wang, W. Zhang, and Q. Huang. Less is more: Picking informative frames for video captioning. In V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, editors, Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIII, volume 11217 of Lecture Notes in Computer Science, pages 367–384. Springer, 2018.

[22] N. Chinchor. MUC-4 evaluation metrics. In Fourth Message Uunderstanding Conference (MUC-4): Proceedings of a Conference Held in McLean, Virginia, June 16-18, 1992, 1992.

[23] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In A. Moschitti, B. Pang, and W. Daelemans, editors, Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL, pages 1724–1734. ACL, 2014.

[24] M. Cornia, M. Stefanini, L. Baraldi, and R. Cucchiara. Meshed-memory transformer for image captioning. In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020, pages 10575–10584. Computer Vision Foundation / IEEE, 2020.

[25] Y. Cui, Z. Yu, C. Wang, Z. Zhao, J. Zhang, M. Wang, and J. Yu. ROSITA: enhancing vision-and-language semantic alignments via cross- and intra-modal knowledge integration. In H. T. Shen, Y. Zhuang, J. R. Smith, Y. Yang, P. Cesar, F. Metze, and B. Prabhakaran, editors, MM '21: ACM Multimedia Conference, Virtual Event, China, October 20 - 24, 2021, pages 797–806. ACM, 2021.

[26] P. Das, C. Xu, R. F. Doell, and J. J. Corso. A thousand frames in just a few words: Lingual description of videos through latent topics and sparse object stitching. In 2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013, pages 2634–2641. IEEE Computer Society, 2013.

[27] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier. Language modeling with gated convolutional networks. In D. Precup and Y. W. Teh, editors, Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017, volume 70 of Proceedings of Machine Learning Research, pages 933–941. PMLR, 2017.

[28] D. Demner-Fushman, M. D. Kohli, M. B. Rosenman, S. E. Shooshan, L. Rodriguez, S. K. Antani, G. R. Thoma, and C. J. McDonald. Preparing a collection of radiology examinations for distribution and retrieval. J. Am. Medical Informatics Assoc., 23(2):304–310, 2016.

[29] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, and T. Solorio, editors, Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers), pages 4171–4186. Association for Computational Linguistics, 2019.

[30] M. Ding, Z. Yang, W. Hong, W. Zheng, C. Zhou, D. Yin, J. Lin, X. Zou, Z. Shao, H. Yang, and J. Tang. Cogview: Mastering text-to-image generation via transformers. CoRR, abs/2105.13290, 2021.

[31] G. C. T. Documentation. Advanced guide to inception v3 on cloud tpu. `https://cloud.google.com/tpu/docs/inception-v3-advanced/`, 2015. accessed at 11/22/2021.

[32] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, T. Darrell, and K. Saenko. Long-term recurrent convolutional networks for visual recognition and description. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015, pages 2625–2634. IEEE Computer Society, 2015.

[33] D. Dong, H. Wu, W. He, D. Yu, and H. Wang. Multi-task learning for multiple language translation. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers, pages 1723–1732. The Association for Computer Linguistics, 2015.

[34] D. Elliott and F. Keller. Image description using visual dependency representations. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL, pages 1292–1302. ACL, 2013.

[35] H. Fang, S. Gupta, F. N. Iandola, R. K. Srivastava, L. Deng, P. Dollár, J. Gao, X. He, M. Mitchell, J. C. Platt, C. L. Zitnick, and G. Zweig. From captions to visual concepts and back. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015, pages 1473–1482. IEEE Computer Society, 2015.

[36] A. Farhadi, S. M. M. Hejrati, M. A. Sadeghi, P. Young, C. Rashtchian, J. Hockenmaier, and D. A. Forsyth. Every picture tells a story: Generating sentences from images. In K. Daniilidis, P. Maragos, and N. Paragios, editors, Computer Vision - ECCV 2010, 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV, volume 6314 of Lecture Notes in Computer Science, pages 15–29. Springer, 2010.

[37] Z. Fei. Fast image caption generation with position alignment. CoRR, abs/1912.06365, 2019.

[38] A. Fukui, D. H. Park, D. Yang, A. Rohrbach, T. Darrell, and M. Rohrbach. Multimodal compact bilinear pooling for visual question answering and visual grounding. In J. Su, X. Carreras, and K. Duh, editors, Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin,

Texas, USA, November 1-4, 2016, pages 457–468. The Association for Computational Linguistics, 2016.

[39] C. Gan, Z. Gan, X. He, J. Gao, and L. Deng. Stylenet: Generating attractive visual captions with styles. In 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, pages 955–964. IEEE Computer Society, 2017.

[40] Z. Gan, C. Gan, X. He, Y. Pu, K. Tran, J. Gao, L. Carin, and L. Deng. Semantic compositional networks for visual captioning. In 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, pages 1141–1150. IEEE Computer Society, 2017.

[41] J. Gao, X. Meng, S. Wang, X. Li, S. Wang, S. Ma, and W. Gao. Masked non-autoregressive image captioning. CoRR, abs/1906.00717, 2019.

[42] L. Gao, Z. Guo, H. Zhang, X. Xu, and H. T. Shen. Video captioning with attention-based LSTM and semantic consistency. IEEE Trans. Multim., 19(9):2045–2055, 2017.

[43] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. Convolutional sequence to sequence learning. In D. Precup and Y. W. Teh, editors, Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017, volume 70 of Proceedings of Machine Learning Research, pages 1243–1252. PMLR, 2017.

[44] Y. Gong, L. Wang, M. Hodosh, J. Hockenmaier, and S. Lazebnik. Improving image-sentence embeddings using large weakly annotated photo collections. In D. J. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part IV, volume 8692 of Lecture Notes in Computer Science, pages 529–545. Springer, 2014.

[45] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh. Making the V in VQA matter: Elevating the role of image understanding in visual question answering. In 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, pages 6325–6334. IEEE Computer Society, 2017.

[46] J. Gu, J. Bradbury, C. Xiong, V. O. K. Li, and R. Socher. Non-autoregressive neural machine translation. In 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net, 2018.

[47] J. Gu, G. Wang, J. Cai, and T. Chen. An empirical study of language CNN for image captioning. In IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017, pages 1231–1240. IEEE Computer Society, 2017.

[48] L. Guo, J. Liu, X. Zhu, X. He, J. Jiang, and H. Lu. Non-autoregressive image captioning with counterfactuals-critical multi-agent learning. In C. Bessiere, editor, Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020, pages 767–773. ijcai.org, 2020.

[49] L. Guo, J. Liu, X. Zhu, P. Yao, S. Lu, and H. Lu. Normalized and geometry-aware self-attention network for image captioning. In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020, pages 10324–10333. Computer Vision Foundation / IEEE, 2020.

[50] P. Harzig, S. Brehm, R. Lienhart, C. Kaiser, and R. Schallner. Multimodal image captioning for marketing analysis. In IEEE 1st Conference on Multimedia Information Processing and Retrieval, MIPR 2018, Miami, FL, USA, April 10-12, 2018, pages 158–161. IEEE, 2018.

[51] P. Harzig, Y. Chen, F. Chen, and R. Lienhart. Addressing data bias problems for chest x-ray image report generation. In 30th British Machine Vision Conference 2019, BMVC 2019, Cardiff, UK, September 9-12, 2019, page 144. BMVA Press, 2019.

[52] P. Harzig, C. Eggert, and R. Lienhart. Visual question answering with a hybrid convolution recurrent model. In K. Aizawa, M. S. Lew, and S. Satoh, editors, Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval, ICMR 2018, Yokohama, Japan, June 11-14, 2018, pages 318–325. ACM, 2018.

[53] P. Harzig, M. Einfalt, and R. Lienhart. Automatic disease detection and report generation for gastrointestinal tract examination. In L. Amsaleg, B. Huet, M. A. Larson, G. Gravier, H. Hung, C. Ngo, and W. T. Ooi, editors, Proceedings of the 27th ACM International Conference on Multimedia, MM 2019, Nice, France, October 21-25, 2019, pages 2573–2577. ACM, 2019.

[54] P. Harzig, M. Einfalt, and R. Lienhart. Synchronized audio-visual frames with fractional positional encoding for transformers in video-to-text translation. CoRR, abs/2112.14088, 2021.

[55] P. Harzig, M. Einfalt, K. Ludwig, and R. Lienhart. Transforming videos to text (VTT task) team: MMCUniAugsburg. In G. Awad, A. A. Butt, K. Curtis, J. G. Fiscus, A. Godil, Y. Lee, A. Delgado, J. Zhang, E. Godard, B. Chocot, L. L. Diduch, J. Liu, A. F. Smeaton, Y. Graham, G. J. F. Jones, W. Kraaij, and G. Quénot, editors, 2020 TREC Video Retrieval Evaluation, TRECVID 2020, Gaithersburg, MD, USA, December 8-11, 2020. National Institute of Standards and Technology (NIST), 2020.

[56] P. Harzig, M. Einfalt, K. Ludwig, and R. Lienhart. Extended self-critical pipeline for transforming videos to text (TRECVID-VTT task 2021) – team: MMCUni-Augsburg. CoRR, abs/2112.14100, 2021.

[57] P. Harzig, D. Zecha, R. Lienhart, C. Kaiser, and R. Schallner. Image captioning with clause-focused metrics in a multi-modal setting for marketing. In 2nd IEEE Conference on Multimedia Information Processing and Retrieval, MIPR 2019, San Jose, CA, USA, March 28-30, 2019, pages 419–424. IEEE, 2019.

[58] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, pages 770–778. IEEE Computer Society, 2016.

[59] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV, volume 9908 of Lecture Notes in Computer Science, pages 630–645. Springer, 2016.

[60] S. He, W. Liao, H. R. Tavakoli, M. Y. Yang, B. Rosenhahn, and N. Pugeault. Image captioning through image transformer. In H. Ishikawa, C. Liu, T. Pajdla, and J. Shi, editors, Computer Vision - ACCV 2020 - 15th Asian Conference on Computer Vision, Kyoto, Japan, November 30 - December 4, 2020, Revised Selected Papers, Part IV, volume 12625 of Lecture Notes in Computer Science, pages 153–169. Springer, 2020.

[61] L. A. Hendricks, S. Venugopalan, M. Rohrbach, R. J. Mooney, K. Saenko, and T. Darrell. Deep compositional captioning: Describing novel object categories without paired training data. In 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, pages 1–10. IEEE Computer Society, 2016.

[62] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. W. Wilson. CNN architectures for large-scale audio classification. In 2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017, New Orleans, LA, USA, March 5-9, 2017, pages 131–135. IEEE, 2017.

[63] S. A. Hicks, M. Riegler, P. H. Smedsrud, T. B. Haugen, K. R. Randel, K. Pogorelov, H. K. Stensland, D. Dang-Nguyen, M. Lux, A. Petlund, T. de Lange, P. T. Schmidt, and P. Halvorsen. ACM multimedia biomedia 2019 grand challenge overview. In L. Amsaleg, B. Huet, M. A. Larson, G. Gravier, H. Hung, C. Ngo, and W. T. Ooi, editors, Proceedings of the 27th ACM International Conference on Multimedia, MM 2019, Nice, France, October 21-25, 2019, pages 2563–2567. ACM, 2019.

[64] S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural Comput., 9(8):1735–1780, 1997.

[65] M. Hodosh, P. Young, and J. Hockenmaier. Framing image description as a ranking task: Data, models and evaluation metrics. J. Artif. Intell. Res., 47:853–899, 2013.

[66] J. Hou, X. Wu, W. Zhao, J. Luo, and Y. Jia. Joint syntax representation learning and visual cue translation for video captioning. In 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019, pages 8917–8926. IEEE, 2019.

[67] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, pages 2261–2269. IEEE Computer Society, 2017.

[68] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In F. R. Bach and D. M. Blei, editors, Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015, volume 37 of JMLR Workshop and Conference Proceedings, pages 448–456. JMLR.org, 2015.

[69] S. Islam, A. Dash, A. Seum, A. H. Raj, T. Hossain, and F. M. Shah. Exploring video captioning techniques: A comprehensive survey on deep learning methods. SN Comput. Sci., 2(2):120, 2021.

[70] B. Jing, P. Xie, and E. P. Xing. On the automatic generation of medical imaging reports. In I. Gurevych and Y. Miyao, editors, Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers, pages 2577–2586. Association for Computational Linguistics, 2018.

[71] J. Johnson, A. Karpathy, and L. Fei-Fei. Densecap: Fully convolutional localization networks for dense captioning. In 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, pages 4565–4574. IEEE Computer Society, 2016.

[72] D. Jurafsky and J. H. Martin. Speech and Language Processing: An Introduction to Natural Language Processing,Computational Linguistics, and Speech Recognition. Stanford University, University of Colorado at Boulder, third edition draft edition, 2021.

[73] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015, pages 3128–3137. IEEE Computer Society, 2015.

[74] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman. The kinetics human action video dataset. CoRR, abs/1705.06950, 2017.

[75] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.

[76] R. Kiros, R. Salakhutdinov, and R. S. Zemel. Unifying visual-semantic embeddings with multimodal neural language models. CoRR, abs/1411.2539, 2014.

[77] J. Krause, J. Johnson, R. Krishna, and L. Fei-Fei. A hierarchical approach for generating descriptive image paragraphs. In 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, pages 3337–3345. IEEE Computer Society, 2017.

[78] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L. Li, D. A. Shamma, M. S. Bernstein, and L. Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. Int. J. Comput. Vis., 123(1):32–73, 2017.

[79] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States, pages 1106–1114, 2012.

[80] G. Kulkarni, V. Premraj, S. Dhar, S. Li, Y. Choi, A. C. Berg, and T. L. Berg. Baby talk: Understanding and generating simple image descriptions. In The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, Colorado Springs, CO, USA, 20-25 June 2011, pages 1601–1608. IEEE Computer Society, 2011.

[81] J. Lee, E. Mansimov, and K. Cho. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii, editors, Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018, pages 1173–1182. Association for Computational Linguistics, 2018.

[82] C. Y. Li, X. Liang, Z. Hu, and E. P. Xing. Knowledge-driven encode, retrieve, paraphrase for medical image report generation. In The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019, pages 6666–6673. AAAI Press, 2019.

[83] G. Li, L. Zhu, P. Liu, and Y. Yang. Entangled transformer for image captioning. In 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019,

Seoul, Korea (South), October 27 - November 2, 2019, pages 8927–8936. IEEE, 2019.

[84] L. H. Li, M. Yatskar, D. Yin, C. Hsieh, and K. Chang. Visualbert: A simple and performant baseline for vision and language. CoRR, abs/1908.03557, 2019.

[85] S. Li, G. Kulkarni, T. L. Berg, A. C. Berg, and Y. Choi. Composing simple image descriptions using web-scale n-grams. In S. Goldwater and C. D. Manning, editors, Proceedings of the Fifteenth Conference on Computational Natural Language Learning, CoNLL 2011, Portland, Oregon, USA, June 23-24, 2011, pages 220–228. ACL, 2011.

[86] X. Li, X. Yin, C. Li, P. Zhang, X. Hu, L. Zhang, L. Wang, H. Hu, L. Dong, F. Wei, Y. Choi, and J. Gao. Oscar: Object-semantics aligned pre-training for vision-language tasks. In A. Vedaldi, H. Bischof, T. Brox, and J. Frahm, editors, Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXX, volume 12375 of Lecture Notes in Computer Science, pages 121–137. Springer, 2020.

[87] Y. Li, X. Liang, Z. Hu, and E. P. Xing. Hybrid retrieval-generation reinforced agent for medical image report generation. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada, pages 1537–1547, 2018.

[88] Z. Li, C. Wang, M. Han, Y. Xue, W. Wei, L. Li, and L. Fei-Fei. Thoracic disease identification and localization with limited supervision. In 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018, pages 8290–8299. Computer Vision Foundation / IEEE Computer Society, 2018.

[89] C.-Y. Lin. ROUGE: A package for automatic evaluation of summaries. In Text Summarization Branches Out, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[90] K. Lin, Z. Gan, and L. Wang. Multi-modal feature fusion with feature attention for VATEX captioning challenge 2020. CoRR, abs/2006.03315, 2020.

[91] T. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. In D. J. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V, volume 8693 of Lecture Notes in Computer Science, pages 740–755. Springer, 2014.

[92] S. Liu, Z. Ren, and J. Yuan. Sibnet: Sibling convolutional encoder for video captioning. In S. Boll, K. M. Lee, J. Luo, W. Zhu, H. Byun, C. W. Chen, R. Lienhart, and T. Mei, editors, 2018 ACM Multimedia Conference on Multimedia Conference, MM 2018, Seoul, Republic of Korea, October 22-26, 2018, pages 1425–1434. ACM, 2018.

[93] X. Liu, H. Li, J. Shao, D. Chen, and X. Wang. Show, tell and discriminate: Image captioning by self-retrieval with partially labeled data. In V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, editors, Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XV, volume 11219 of Lecture Notes in Computer Science, pages 353–369. Springer, 2018.

[94] X. Long, C. Gan, and G. de Melo. Video captioning with multi-faceted attention. Trans. Assoc. Comput. Linguistics, 6:173–184, 2018.

[95] I. Loshchilov and F. Hutter. SGDR: stochastic gradient descent with warm restarts. In 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net, 2017.

[96] J. Lu, D. Batra, D. Parikh, and S. Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, pages 13–23, 2019.

[97] J. Lu, X. Lin, D. Batra, and D. Parikh. Deeper lstm and normalized cnn visual question answering model. `https://github.com/VT-vision-lab/VQA_LSTM_CNN`, 2015.

[98] M. Luong, Q. V. Le, I. Sutskever, O. Vinyals, and L. Kaiser. Multi-task sequence to sequence learning. In Y. Bengio and Y. LeCun, editors, 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings, 2016.

[99] T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. In L. Màrquez, C. Callison-Burch, J. Su, D. Pighin, and Y. Marton, editors, Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015, pages 1412–1421. The Association for Computational Linguistics, 2015.

[100] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In Y. Bengio and Y. LeCun, editors, 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings, 2013.

[101] T. Mikolov, W. Yih, and G. Zweig. Linguistic regularities in continuous space word representations. In L. Vanderwende, H. D. III, and K. Kirchhoff, editors, Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA, pages 746–751. The Association for Computational Linguistics, 2013.

[102] C. Olah. Understanding lstm networks. `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`, 2015. accessed at 11/13/2021.

[103] Y. Pan, Y. Li, J. Luo, J. Xu, T. Yao, and T. Mei. Auto-captions on GIF: A large-scale video-sentence dataset for vision-language pre-training. CoRR, abs/2007.02375, 2020.

[104] Y. Pan, T. Mei, T. Yao, H. Li, and Y. Rui. Jointly modeling embedding and translation to bridge video and language. In 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, pages 4594–4602. IEEE Computer Society, 2016.

[105] Y. Pan, T. Yao, H. Li, and T. Mei. Video captioning with transferred semantic attributes. In 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, pages 984–992. IEEE Computer Society, 2017.

[106] Y. Pan, T. Yao, Y. Li, and T. Mei. X-linear attention networks for image captioning. In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020, pages 10968–10977. Computer Vision Foundation / IEEE, 2020.

[107] K. Papineni, S. Roukos, T. Ward, J. Henderson, and F. Reeder. Corpus-based comprehensive and diagnostic mt evaluation: Initial arabic, chinese, french, and spanish results. In Proceedings of the Second International Conference on Human Language Technology Research, HLT '02, page 132–137, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[108] K. Papineni, S. Roukos, T. Ward, and W. Zhu. Bleu: a method for automatic evaluation of machine translation. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA, pages 311–318. ACL, 2002.

[109] W. Pei, J. Zhang, X. Wang, L. Ke, X. Shen, and Y. Tai. Memory-attended recurrent network for video captioning. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019, pages 8347–8356. Computer Vision Foundation / IEEE, 2019.

[110] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In A. Moschitti, B. Pang, and W. Daelemans, editors, Proceedings

of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL, pages 1532–1543. ACL, 2014.

[111] J. Perez-Martin, B. Bustos, and J. Pérez. Improving video captioning with temporal composition of a visual-syntactic embedding*. In IEEE Winter Conference on Applications of Computer Vision, WACV 2021, Waikoloa, HI, USA, January 3-8, 2021, pages 3038–3048. IEEE, 2021.

[112] K. Pogorelov, K. R. Randel, T. de Lange, S. L. Eskeland, C. Griwodz, D. Johansen, C. Spampinato, M. Taschwer, M. Lux, P. T. Schmidt, M. Riegler, and P. Halvorsen. Nerthus: A bowel preparation quality video dataset. In Proceedings of the 8th ACM on Multimedia Systems Conference, MMSys 2017, Taipei, Taiwan, June 20-23, 2017, pages 170–174. ACM, 2017.

[113] K. Pogorelov, K. R. Randel, C. Griwodz, S. L. Eskeland, T. de Lange, D. Johansen, C. Spampinato, D. Dang-Nguyen, M. Lux, P. T. Schmidt, M. Riegler, and P. Halvorsen. KVASIR: A multi-class image dataset for computer aided gastrointestinal disease detection. In Proceedings of the 8th ACM on Multimedia Systems Conference, MMSys 2017, Taipei, Taiwan, June 20-23, 2017, pages 164–169. ACM, 2017.

[114] M. F. Porter. An algorithm for suffix stripping. Program, 14(3):130–137, 1980.

[115] O. Press and L. Wolf. Using the output embedding to improve language models. In M. Lapata, P. Blunsom, and A. Koller, editors, Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers, pages 157–163. Association for Computational Linguistics, 2017.

[116] P. Rajpurkar, J. Irvin, K. Zhu, B. Yang, H. Mehta, T. Duan, D. Y. Ding, A. Bagul, C. Langlotz, K. S. Shpanskaya, M. P. Lungren, and A. Y. Ng. Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning. CoRR, abs/1711.05225, 2017.

[117] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba. Sequence level training with recurrent neural networks. In Y. Bengio and Y. LeCun, editors, 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings, 2016.

[118] J. Redmon and A. Farhadi. YOLO9000: better, faster, stronger. In 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, pages 6517–6525. IEEE Computer Society, 2017.

[119] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, Advances in Neural Information Processing

Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada, pages 91–99, 2015.

[120] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel. Self-critical sequence training for image captioning. In 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, pages 1179–1195. IEEE Computer Society, 2017.

[121] C. J. V. Rijsbergen. Information Retrieval. Butterworth-Heinemann, 2nd edition, 1979.

[122] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. Int. J. Comput. Vis., 115(3):211–252, 2015.

[123] H. Saggion, D. Radev, S. Teufel, and W. Lam. Meta-evaluation of summaries in a cross-lingual environment using content-based metrics. In COLING 2002: The 19th International Conference on Computational Linguistics, 2002.

[124] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018, pages 4510–4520. Computer Vision Foundation / IEEE Computer Society, 2018.

[125] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In Y. Bengio and Y. LeCun, editors, 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.

[126] A. Singh, T. D. Singh, and S. Bandyopadhyay. NITS-VC system for VATEX video captioning challenge 2020. CoRR, abs/2006.04058, 2020.

[127] R. Socher, A. Karpathy, Q. V. Le, C. D. Manning, and A. Y. Ng. Grounded compositional semantics for finding and describing images with sentences. Trans. Assoc. Comput. Linguistics, 2:207–218, 2014.

[128] C. Sun, A. Myers, C. Vondrick, K. Murphy, and C. Schmid. Videobert: A joint model for video and language representation learning. In 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019, pages 7463–7472. IEEE, 2019.

[129] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada, pages 3104–3112, 2014.

[130] R. S. Sutton and A. G. Barto. Reinforcement Learning: An Introduction. A Bradford Book, Cambridge, MA, USA, 2018.

[131] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015, pages 1–9. IEEE Computer Society, 2015.

[132] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, pages 2818–2826. IEEE Computer Society, 2016.

[133] N. Tajbakhsh, S. R. Gurudu, and J. Liang. Automated polyp detection in colonoscopy videos using shape and context information. IEEE Trans. Medical Imaging, 35(2):630–644, 2016.

[134] H. Tan and M. Bansal. LXMERT: learning cross-modality encoder representations from transformers. In K. Inui, J. Jiang, V. Ng, and X. Wan, editors, Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019, pages 5099–5110. Association for Computational Linguistics, 2019.

[135] D. Teney, P. Anderson, X. He, and A. van den Hengel. Tips and tricks for visual question answering: Learnings from the 2017 challenge. In 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018, pages 4223–4232. Computer Vision Foundation / IEEE Computer Society, 2018.

[136] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In 2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015, pages 4489–4497. IEEE Computer Society, 2015.

[137] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pages 5998–6008, 2017.

[138] R. Vedantam, C. L. Zitnick, and D. Parikh. Cider: Consensus-based image description evaluation. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015, pages 4566–4575. IEEE Computer Society, 2015.

[139] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015, pages 3156–3164. IEEE Computer Society, 2015.

[140] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: Lessons learned from the 2015 MSCOCO image captioning challenge. IEEE Trans. Pattern Anal. Mach. Intell., 39(4):652–663, 2017.

[141] B. Wang, L. Ma, W. Zhang, W. Jiang, J. Wang, and W. Liu. Controllable video captioning with POS sequence guidance based on gated fusion network. In 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019, pages 2641–2650. IEEE, 2019.

[142] B. Wang, L. Ma, W. Zhang, and W. Liu. Reconstruction network for video captioning. In 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018, pages 7622–7631. Computer Vision Foundation / IEEE Computer Society, 2018.

[143] J. Wang, W. Wang, Y. Huang, L. Wang, and T. Tan. M3: multimodal memory modelling for video captioning. In 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018, pages 7512–7520. Computer Vision Foundation / IEEE Computer Society, 2018.

[144] W. Wang, H. Bao, L. Dong, and F. Wei. Vlmo: Unified vision-language pre-training with mixture-of-modality-experts. CoRR, abs/2111.02358, 2021.

[145] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and R. M. Summers. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, pages 3462–3471. IEEE Computer Society, 2017.

[146] X. Wang, Y. Peng, L. Lu, Z. Lu, and R. M. Summers. Tienet: Text-image embedding network for common thorax disease classification and reporting in chest x-rays. In 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018, pages 9049–9058. Computer Vision Foundation / IEEE Computer Society, 2018.

[147] X. Wang, J. Wu, J. Chen, L. Li, Y. Wang, and W. Y. Wang. Vatex: A large-scale, high-quality multilingual dataset for video-and-language research. In 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019, pages 4580–4590. IEEE, 2019.

[148] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach. Learn., 8:229–256, 1992.

[149] R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. Neural Comput., 1(2):270–280, 1989.

[150] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. CoRR, abs/1609.08144, 2016.

[151] J. Xu, T. Mei, T. Yao, and Y. Rui. MSR-VTT: A large video description dataset for bridging video and language. In 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, pages 5288–5296. IEEE Computer Society, 2016.

[152] K. Xu, J. Ba, R. Kiros, K. Cho, A. C. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In F. R. Bach and D. M. Blei, editors, Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015, volume 37 of JMLR Workshop and Conference Proceedings, pages 2048–2057. JMLR.org, 2015.

[153] Y. Xue, T. Xu, L. R. Long, Z. Xue, S. K. Antani, G. R. Thoma, and X. Huang. Multimodal recurrent model with attention for automated radiology report generation. In A. F. Frangi, J. A. Schnabel, C. Davatzikos, C. Alberola-López, and G. Fichtinger, editors, Medical Image Computing and Computer Assisted Intervention - MICCAI 2018 - 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part I, volume 11070 of Lecture Notes in Computer Science, pages 457–466. Springer, 2018.

[154] J. Yu, J. Li, Z. Yu, and Q. Huang. Multimodal transformer with multi-view visual representation for image captioning. IEEE Trans. Circuits Syst. Video Technol., 30(12):4467–4480, 2020.

[155] Z. Yu, J. Yu, J. Fan, and D. Tao. Multi-modal factorized bilinear pooling with co-attention learning for visual question answering. In IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017, pages 1839–1848. IEEE Computer Society, 2017.

[156] Z. Yu, J. Yu, C. Xiang, J. Fan, and D. Tao. Beyond bilinear: Generalized multi-modal factorized high-order pooling for visual question answering. CoRR, abs/1708.03619, 2017.

[157] J. Zhang and Y. Peng. Object-aware aggregation with bidirectional temporal graph for video captioning. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019, pages 8327–8336. Computer Vision Foundation / IEEE, 2019.

[158] Z. Zhang, Z. Qi, C. Yuan, Y. Shan, B. Li, Y. Deng, and W. Hu. Open-book video captioning with retrieve-copy-generate network. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021, pages 9837–9846. Computer Vision Foundation / IEEE, 2021.

[159] Z. Zhang, Y. Shi, C. Yuan, B. Li, P. Wang, W. Hu, and Z. Zha. Object relational graph with teacher-recommended learning for video captioning. In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020, pages 13275–13285. Computer Vision Foundation / IEEE, 2020.

[160] B. Zhou, A. Khosla, À. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, pages 2921–2929. IEEE Computer Society, 2016.

[161] B. Zhou, Y. Tian, S. Sukhbaatar, A. Szlam, and R. Fergus. Simple baseline for visual question answering. CoRR, abs/1512.02167, 2015.

[162] Y. Zhou, Y. Zhang, Z. Hu, and M. Wang. Semi-autoregressive transformer for image captioning. In IEEE/CVF International Conference on Computer Vision Workshops, ICCVW 2021, Montreal, BC, Canada, October 11-17, 2021, pages 3132–3136. IEEE, 2021.

[163] X. Zhu, L. Guo, P. Yao, J. Liu, S. Lu, Z. Yu, W. Liu, and H. Lu. Multi-view features and hybrid reward strategies for vatex video captioning challenge 2019. CoRR, abs/1910.11102, 2019.