Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

FAKULTÄT
FÜR MATHEMATIK, INFORMATIK
UND NATURWISSENSCHAFTEN

Master thesis

COMMUNICATION WITH HUMAN MOTIVATION: How solving the Image Captioning-Retrieval problem creates conversations

December 5, 2018

Fabian Alexander Karl
Intelligent Adaptive Systems WiSe 2016/17
Student ID: 6886047
E-Mail: 6karl@informatik.uni-hamburg.de

Primary Supervisor: **Prof. Dr. Chris Biemann**
Department of Language Technology (LT)

Secondary Supervisor: **Dr. Mikko Lauri**
Department of Computer Vision (CV)

In cooperation with **novomind AG**

We can only see a short distance ahead,
but we can see plenty there that needs to be done.
– *Alan Turing*

Abstract

The motivation for artificial language generation is often explicitly modeled and not derived from human motivation. Language generation in humans is primarily motivated by information exchange. This work applies a similar objective to an artificial language generation system. We design and implement a Neural Network architecture that describes the content of images and retrieves them solely based on the generated descriptions. This way we simulate a simple conversation about images. The main objective of our model is information exchange in form of successful image retrieval. While doing this we impose the constraint that produced image descriptions should be as similar to human generated language as possible. In the end, we can show a strong increase in information exchange while losing some grammatical correctness in the generated descriptions.

Acknowledgments

This thesis marks the end of a 6-year long journey through different academical institutions. I am very grateful and happy for everyone who was part of this journey and helped me on my way.
A special thanks goes to my two advisors, Chris and Mikko. You two enabled this thesis and invested so much time in defining the topic, reading through drafts, discussing ideas and giving critical feedback. Thank you for all your help and support.
I am grateful for the collaboration with novomind AG. This thesis would not have been possible without the support of various people from inside the company.
I want to thank my parents. Your continuous support throughout my life enabled any of this, in the first place.
Thank you, Hannah, for your ideas, your belief in me and for the occasional motivational push, whenever I needed one.

Contents

Abbreviation

AI	Artificial Intelligence
AID	Automated Image Description
BLEU	BiLingual Evaluation Understudy
CBOW	Continuous Bag-of-Words
CCA	Canonical Correlation Analysis
CIDEr	Consensus-based Image Description Evaluation
CNN	Convolutional Neural Network
DCCA	Deep Canonical Correlation Analysis
FFNN	Feed Forward Neural Network
GAN	Generative Adversarial Network
GRU	Gated Recurrent Unit
IC	Image Captioning
ICR	Image Captioning-Retrieval
IP	Image Projection
IR	Image Retrieval
KCCA	Kernel Canonical Correlation Analysis
LSTM	Long-Short-Term Memory
METEOR	Metric for Evaluation of Translation with Explicit ORdering
ML	Machine Learning
MLE	Maximum Likelihood Estimation
MSCOCO	Microsoft Common Objects in Context
MSE	Mean-Squared Error

NLIS	Natural Language Image Search
NLP	Natural Language Processing
NLTK	Natural Language Tool Kit
NN	Neural Network
ReLU	Rectified Linear Unit
ResNet50	Residual Neural Network 50
RL	Reinforcement Learning
RNN	Recurrent Neural Network
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
Skip-gram	Continuous Skip-gram
VSE	Visual-Semantic Embedding
WE	Word Embedding
WF	Word Frequency
WGAN	Wasserstein GAN

List of Figures

List of Tables

1 What artificial conversational systems lack and how to fix it

Artificial replication of human behavior and reasoning is one of the main goals of Machine Learning (ML) and Artificial Intelligence (AI). It is no surprise that research into replicating human's unique ability to speak and to converse is popular among scientists. The close relation between thoughts and words deeply connects humans to language. Turing [119] famously defined the ability to hold a complex conversation as AI. While this point is heavily debated [105], it is safe to say that our language makes us human. Any artificial system capable of human language will be received human-like by us in a certain way.

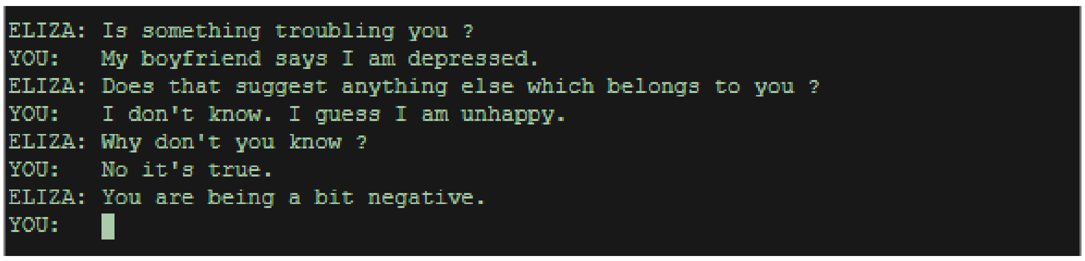
1.1 Why do conversational systems fail?

An impressive amount of research into conversational systems both in the scientific [63, 83] and the commercial sector [1, 66] could make one believe that the challenge of conversational systems is already close to being solved. Sadly, quite the contrary is the case.

To show the reason for this and to motivate this thesis, a short history of conversational systems is needed.

1.1.1 A short history about conversational systems

The very first conversational systems were rule-based pattern matching systems like Weizenbaum's Eliza in 1966 [127] or Simmons' question-answer system in 1970 [108]. The idea that human language and reasoning can be modeled, once enough hand-crafted



```
ELIZA: Is something troubling you ?
YOU:   My boyfriend says I am depressed.
ELIZA: Does that suggest anything else which belongs to you ?
YOU:   I don't know. I guess I am unhappy.
ELIZA: Why don't you know ?
YOU:   No it's true.
ELIZA: You are being a bit negative.
YOU:
```

Figure 1.1: A short conversation with Eliza [127]. Eliza is a mock Rogerian psychotherapist with a simple rule-based answering scheme, developed in 1966.

rules are created was strongly researched until the sudden rise of interest in Neural Networks (NNs). Figure 1.1 shows a short conversation with Eliza, one of the first chatbots.

Rule-based approaches often work great for expert systems, to understand simple commands or to simulate intelligent behavior. Due to unwanted, unavoidable interaction between too many created rules, these approaches can probably never achieve human language performance. Human language has such a complex nature that handmade rules alone cannot recreate it. Rule-based methods are still the dominant technique in commercial applications (Siri, Alexa, Cortana, etc.) since they are predictable and retraceable.

In 2015, the first sequence-to-sequence model was presented by Vinyals and Le [123]. They propose, to train a model to predict the most likely sentence based on a given input sentence. This approach was inspired by machine translation [114], has since been replicated multiple times [78, 106, 111, 131] and is now considered the state-of-the-art architecture for conversational agents.

Many studies base their research on top of the sequence-to-sequence architecture. He et al. [45], for example, try to increase the quality of the conversations by introducing a Dynamic Knowledge Graph Network. Many studies use Reinforcement Learning (RL) [29, 70] to increase the quality of conversations by defining their own measurements of how successful a conversation is. This is questionable since a hand-crafted metric for scoring the quality of a conversation is needed. Evaluating the quality of a produced sentence or a whole conversation, however, is still an open research problem, difficult to solve, due to the complex nature of language and conversations.

Mathur and Singh [83] offer a thorough survey about the history and the current state of conversational systems.

This quick overview of conversational systems is meant to show where this thesis is separating itself from the commonly used approaches for dialogue generation. In our opinion, the motivation for most of the existing approaches to communication is ill-formulated, because the motivation for language generation is explicitly modeled.

The motivation to create language, however, does not lie in the creation of language itself. Language generation should not be motivated by its own generation but by a truly important and useful goal-state. Mathur and Singh [83] capture this nicely when they state that especially sequence-to-sequence models *"can theoretically never solve the problem of* [language] *modelling"* since *"the objective function that is being optimized does not capture the actual objective achieved through human communication, which is typically longer term and based on exchange of information rather than next step prediction."*

The generation of artificial conversations is trying to mimic human behavior. In order to **mimic human behavior**, one should first closely examine the human **motivation for said behavior**.

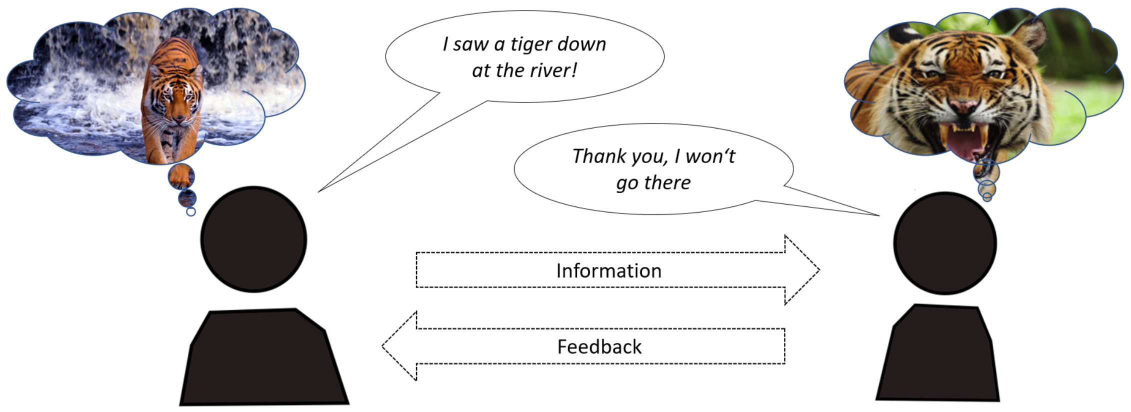


Figure 1.2: At an earlier stage in evolution, information exchange through language could directly increase the chances of survival. The feedback can be in form of language, actions or non-actions.

1.1.2 Why do humans communicate?

The primary, evolutionary motivation for human communication is information exchange [60]. It allows bigger and stronger social structures and has a strong cultural impact since it allows knowledge and information to be shared. It allows defining clear rules and structures, to give precise information about certain topics and to pass knowledge over generations [43].

Many animals use communication, but only the *homo sapiens* developed highly sophisticated and complex language systems, tailored to their environment [12], in order to optimize **information exchange** and ultimately increase survival chances.

1.1.3 How is communication defined in this thesis?

Since information exchange seems to be the main driver for the development of human communication and language, it is chosen as the primary motivation for communication. For this study, a successful communication is therefore defined as the following:

1. *One system creates a series of tokens with the intention of communicating an internal state/ knowledge/ thought.*

2. *A second, separated system, receives these tokens and is able to understand them, to recreate the internal state.*

3. *Both systems receive feedback on how well the second system could recreate the original state, representing the level of success of the communication.*

If both systems can do their designated task well enough, information from one system can be exchanged to another system and with that, a simple conversation emerges. A simplified, possible scenario of a conversation is shown in Figure 1.2.

1.2 Communication with human motivation

This thesis is proposing the idea and the implementation of a system that is capable of natural language communication, driven by human-like motivation factors. The two key requirements of this system are:

1. *Conversation between two artificial systems or agents is inspired by the human motivation of information exchange.*
2. *Communication takes place in human understandable language.*

The focus of this work lies in the idea that the main drive for communication is not simply to produce language but that language is the means to an end, similar to the human motivation of language and communication. It is a key aspect of this thesis that the objective to produce language is not simply the production of language but the exchange of information, similar to Steels [112].

1.2.1 How can human motivation be modeled?

To give artificial systems exactly the same motivation for language as humans are given by evolution, complex simulations would be needed that try to model the environment, the internal human state and the interactions between humans. This is computationally infeasible at the present date and ethically questionable. Giving an artificial system evolutionary motivation would include the drive to survive as long as possible and to replicate its genetic material as much as possible. A simpler and safer approach is needed.

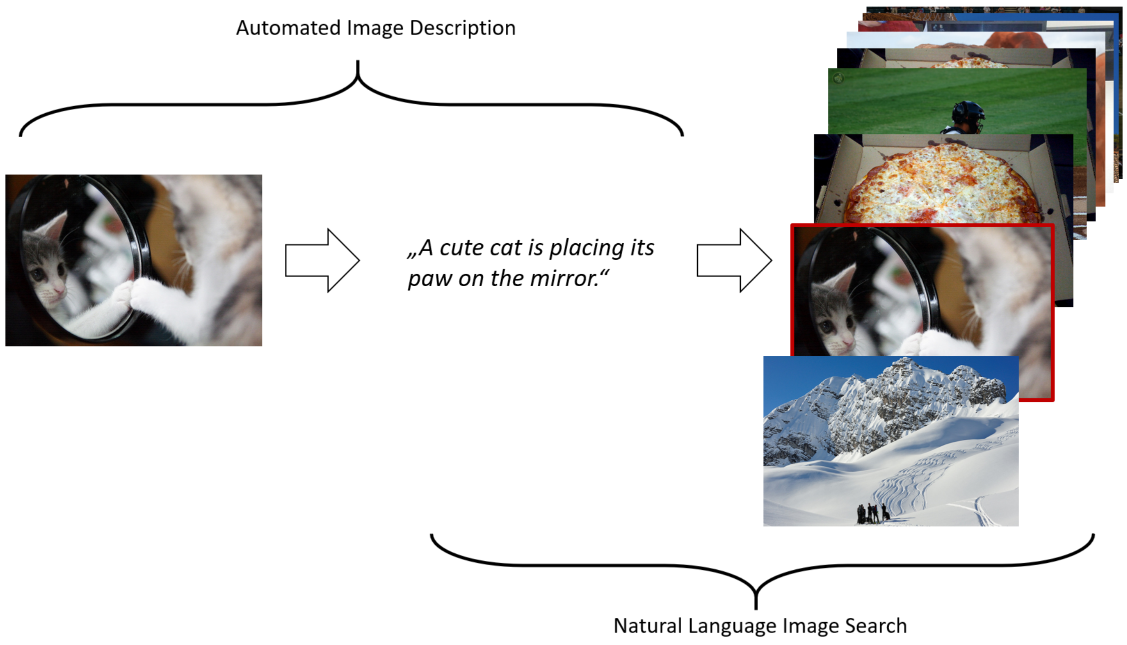
In order to create a situation, where two systems have to exchange information via language, the Image Captioning-Retrieval (ICR) problem is proposed. This problem is derived from the Image Retrieval (IR) problem that describes the task of finding the closest match to a query image in a large database of images. By adding the constraint that the system has to produce a natural language search query in order to find the closest matching image, the ICR problem is described. Figure 1.3 shows this in a simple and abstract way.

The problem of ICR can be split into Automated Image Description (AID) and Natural Language Image Search (NLIS). AID describes a given image with a sentence in natural language. NLIS uses descriptions or search queries as input and tries to retrieve the closest or if possible the same image out of a range of candidate images.

The next section will show, why the ICR problem is well suited to simulate simple conversations.

1.2.2 Can Image Captioning-Retrieval offer human-like motivation?

It is proposed that ICR can offer the needed motivation to exchange information in natural language. By creating a feedback loop from the retrieval performance of the NLIS,


Automated Image Description
„A cute cat is placing its paw on the mirror.“
Natural Language Image Search

[16]

Figure 1.3: The ICR problem: The image is first described and then retrieved among a number of candidate images.

both systems can be optimized, based on the quality and amount of information exchanged. If the AID system describes the image well enough and the NLIS system is able to retrieve the correct or a very similar image, information was successfully exchanged.

The success of the communication can now be measured mathematically, by ranking the candidate images based on the NLIS system's belief, similar to Hodosh et al. [50]. The success is proportional to the belief/probability for the correct image. By quantifying the quality of the communication and by using it as feedback, both systems can be trained to maximize it. The system should learn to generate a successful conversation, by generating and understanding an image description. The motivation for this task comes mainly from the objective of information exchanged.

Providing the motivation for information exchange through communication alone is not the complete solution, though. If the objective of the system is simply to optimize information exchange, the system will invent its own optimized language. The constraint, that information should not simply be exchanged in any language or code but in human language is a non-trivial challenge and another reason for ICR as the system of choice.

Thanks to extensive datasets with annotated images, AID allows grounding the internal representation of the system (images) to human understandable language or descriptions. The system can be trained to produce descriptions in the desired human language. Once the AID system is grounded, reusing it for different tasks, while the mapping from image pixels to words stays intact, should be possible.

1.2.3 Possible use-cases

The main application for a conversational system is the usage as a new interface for interacting with artificial systems. Right now, keyboard, computer-mouse, controller and touch-display are the most commonly used hardware to interact with one's devices. Adding a language interface to devices will open a whole new market for language controlled gadgets; similar to the revolution of touch-displays. Since voice-to-speech and speech-to-voice systems are getting close to human performance [4, 33, 42], language control basically introduces voice control.

There are many situations, where voice control will increase safety (car control), usability (device control) and simply convenience (smart homes, smart assistants, gaming).

The production of properly motivated conversation-like interaction loops between two systems could be the foundation of more sophisticated systems. It is our hope that once the principle framework for communication is better understood and modeled, more complex conversations (back-and-forth) can be built upon it. The image retrieval game could, for example, be expanded to a question-answer game about the images.

When two systems are able to communicate in human language with each other, they build the baseline for a system that can communicate with humans. The NLIS system has been trained to receive natural language input, to process this input in a meaningful way and to associate it with a fitting image. By changing the objective of detecting the correct image to a different action-space, various systems can be created.

A system capable of acting as an API between human and machine, via language, needs a deep understanding of natural language. To create language systems that are complex enough to do so, a thorough training process, with a concrete and meaningful goal state, is necessary. This work aims to lay the foundation for that.

If the proposed feedback loop allows successful communication, with human-like motivation, it would be a small step closer to real human-machine communication with natural language.

1.2.4 Additional applications

The created subsystems can be used independently for various tasks.

AID can be used to label articles from online shops for example. An application for visually impaired people can also be imagined, where the system describes images or videos taken by the person. In general, the idea of artificial commentators or narrators could be realized in the future. The automated captioning of taken photos is another possible application. AID allows enriching the textual content of an image in form of annotations or keywords. This can help with text-based image search.

NLIS can mainly be used to query images with natural language, which would allow image search without any form of annotations or keywords needed. Search algorithms based on image content can be very useful.

1.3 Content, contribution and research question

This chapter gives a broad overview of the following study, names the key contributions and the research questions.

1.3.1 Content

Our work is separated into three main chapters with the same internal structure. The three chapters respectively represent AID in Chapter 4, NLIS in Chapter 5 and ICR in Chapter 6. Each of these chapters contains a thorough literature research about the respective topic (Sections 4.1, 5.1, 6.1), followed by the in-depth explanation of the most promising approaches (Sections 4.2, 5.2, 6.2). After that, the experimental setup is described (Sections 4.3, 5.3, 6.3). The results, compared to related work (Sections 4.4, 5.4, 6.4), and a discussion about the chosen approach finish each chapter (Sections 4.5, 5.5, 6.5).

The three chapters are preceded by Chapter 2 about basic NN techniques and Chapter 3, where the used dataset is analyzed descriptively. Chapter 7 finally concludes the whole study.

1.3.2 Contribution

The main contributions of this thesis are:

- The simple and straightforward formulation for artificial communication with human-like motivation.
- A broad and thorough survey over AID, NLIS, and ICR, including the most recent ideas and approaches.
- The replicative implementation and evaluation of state-of-the-art AID and NLIS models.
- The implementation and evaluation of an ICR system that communicates, motivated by the objective of information exchange.

1.3.3 Research questions

From a scientific point of view, the main research question of this thesis is:

- Are the architecture and implementation proposed in this study capable of solving the ICR problem in a way that simple conversations driven by information exchange emerge?

The main research question includes three sub-questions:

- Can AID be solved well enough that it can capture the most important features of an image and transform them into natural language?
- Is it possible to train a NLIS system to find the same or similar images with only the generated description as input?
- Can both the AID and NLIS system be optimized simultaneously by the feedback of how successful the retrieval was?

2 Neural Network techniques

This chapter covers high-level Neural Network (NN) concepts, needed to understand the following chapters of this work. If the reader is familiar with these concepts this chapter can either be skipped for now and revisited when needed, or it can be used as a refreshment of knowledge.

2.1 Neural Networks

NNs are non-linear, universal function approximation systems. This means, theoretically, they can approximate any possible function. Since any process can be formalized as function, NNs have the potential to model any process and make predictions about what the outcome of this process might be under novel conditions or inputs. In practice, this is only possible once enough high-quality training data is provided and the system is trained on it sufficiently. Even in that case, the optimal solution of any problem is always theoretical and might not be found due to sub-optimal solutions. NNs have shown fascinating results both in science and in commercial products.

There is a large amount of literature on NNs in the form of books [41, 44, 102], papers [34, 103] and in countless very informative and intuitive blogs [37, 56, 84, 88]. For deeper understanding of the mathematical concepts of gradient descent, forward- and backpropagation, error functions and learning optimization, these sources are recommended.

NNs are functions, projecting an input x onto an output $\hat{y}$. Both input and output can have arbitrary numeric shapes and dimensions. The projection from x to $\hat{y}$ can be seen as a forward propagation of the input through the network. All input values are multiplied with weighted connections w between neurons, they are summed up and an activation function is used to output the state of the next neuron. The activation function is thus defining, how much of the current activation is propagated to the next node. One of the most frequently used activation function nowadays is the Rectified Linear Unit (ReLU) function. The ReLU function is the same as $\max[0, x]$, where x is the input to the function.

This process is repeated until the output layer of the network is reached and $\hat{y}$ is generated. Figure 2.1 shows how the activation is propagated forward for one step through a network.

Mathematically, a NN simply performs multiple matrix multiplications, starting with the input matrix X and the first weight matrix of the network $W^{in \to 1}$ (Eq. 2.1). The activation function f_{act} is applied (Eq. 2.2) and the current state S of the network is transformed

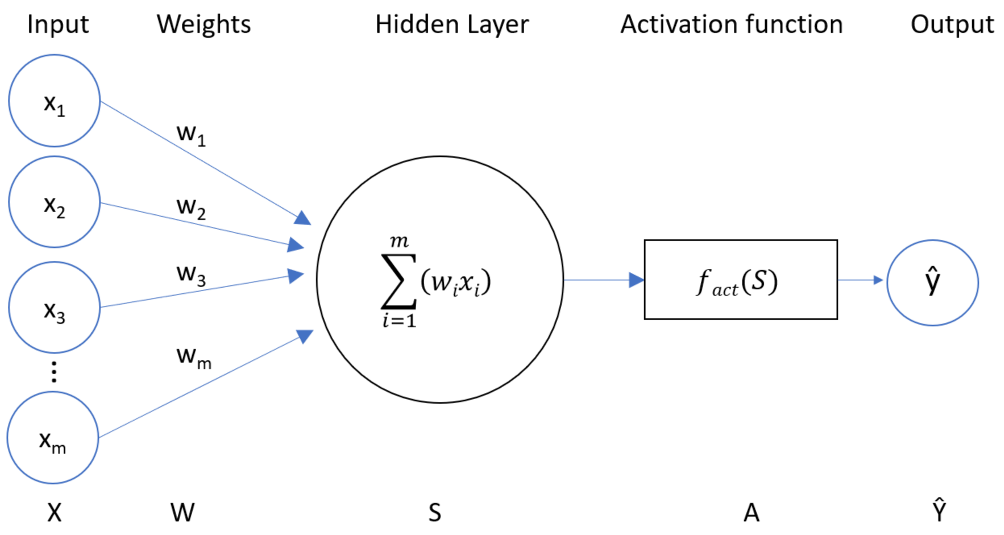


Figure 2.1: A visualization of the process leading from the input of a neuron to its output. X can either stand for the input of the network or for the activation of the previous layer of the network and the output can either be propagated to the following layer or could be the final output of the network.

into the activation state, A. A is multiplied with the next weight matrix of the network (Eq. 2.3). This process is repeated until the state S^{out} of the network is reached and the output $\hat{Y}$ is generated by applying the output function f_{out}. The output function f_{out} often differs from the activation function f_{act}. Classic activation functions are for example the Sigmoid function, returning a value between 0 and 1 or the Softmax function, yielding a normalized distribution over a number of output nodes.

$$S^1 = XW^{in\rightarrow 1} \tag{2.1}$$
$$A^1 = \mathrm{f}_{act}(S^1) \tag{2.2}$$
$$S^2 = A^1 W^{1\rightarrow 2} \tag{2.3}$$
$$A^2 = \mathrm{f}_{act}(S^2) \tag{2.4}$$
$$\ldots \tag{2.5}$$
$$S^{out} = A^{out-1} W^{out-1\rightarrow out} \tag{2.6}$$
$$\hat{Y} = \mathrm{f}_{out}\left(S^{out}\right) \tag{2.7}$$

Once $\hat{Y}$ is generated, it can be compared to the target Y and a prediction error can be calculated. In order to minimize this error, an error or loss function $\boldsymbol{L}$ is defined, mapping the current weights of the network $\boldsymbol{\theta}$ to the average error. Equation 2.8 calculates the average error L. L itself depends on the learning task, but it always calculates the error between a single target y^i and the outcome of the neural network f that is dependent on a single input x^i and its current weights $\boldsymbol{\theta}$.

$$\boldsymbol{\mathcal{L}}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} \mathrm{L}\big(y^i, f(\mathbf{x}^i, \boldsymbol{\theta})\big) \tag{2.8}$$

A common loss function L is the Mean-Squared Error (MSE), shown in Equation 2.9. The prediction y for a single sample is subtracted from the target $\hat{y}$ and squared. This is repeated for a batch of samples, to generate an averaged error.

$$\mathrm{L}(y, \hat{y}) = (y - \hat{y})^2 \tag{2.9}$$

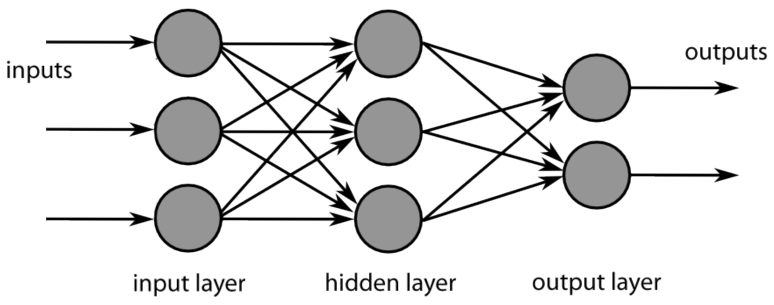
In order to decrease the error, the partial derivative of the loss function $\boldsymbol{\mathcal{L}}$, with respect to every single weight $\boldsymbol{\theta^i}$, connected to an output neuron, of the error function is calculated and the gradient of it is determined. Descending on the gradient in small steps, governed by a defined learning rate, each weight is adapted and slowly and steadily the error of the model is decreased. This process is called gradient descent.

This gradient can only be calculated for the weights connected to the output neurons. When calculating the gradient for a weight not directly connected to an output neuron, the previously calculated error of all the following units is summed and used instead of the real error. This process is called backpropagation since it needs to start at the end of the network and work its way back to the input layer.

By using these methods, the weights of a NN are optimized depending on the chosen loss function and the given data. Different types of NN architectures will be presented in the following sections.

2.2 Feed Forward Neural Networks

Feed Forward Neural Networks (FFNNs) are "classic" NNs. They are fully connected, meaning every neuron from layer l is connected to every neuron from layer $l+1$, and they go forward, meaning in the direction from input to output. A typical input for a FFNN is $x \in \mathbb{R}^f$, where f is a set of features, but they are not limited to this input shape. The output can have any chosen shape. Figure 2.2 shows how FFNNs are normally visualized.



[32]

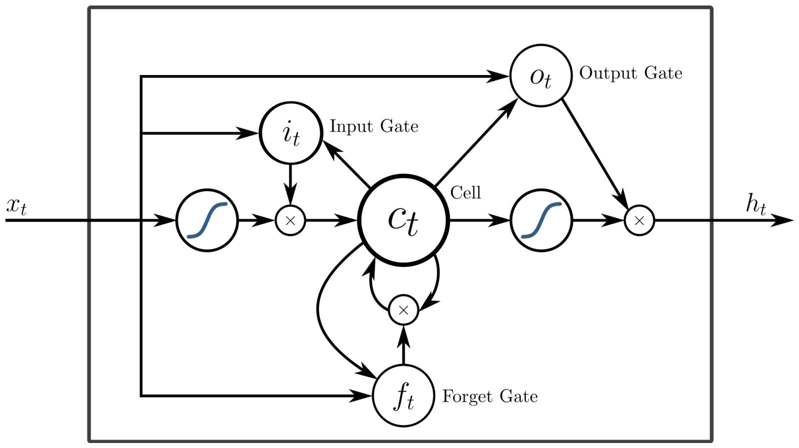
Figure 2.2: A simple FFNN with three input and two output nodes.

2.3 Recurrent Neural Networks

Recurrent Neural Networks (RNNs), as the name says, have recurrent connections over a sequence of input features. They normally receive an input sequence $x \in \mathbb{R}^{t \times f}$, where t is the number of time or sequence steps and f is the number of features that represent each step. The first time-step of x is propagated through the network just like in a FFNN. When the second step of the sequence is used as input, the output from the previous time-step is additionally used as input for the current time-step. This allows the network to have an internal or hidden state over the complete sequence of inputs.

Since regular RNNs suffer from the problem of *vanishing/exploding gradient* [48], gated units like Long-Short-Term Memory (LSTM) cells [49] or Gated Recurrent Units (GRUs) [18] are often preferred [20]. The problem of *vanishing/exploding gradient* describes the exponential decrease or increase of the gradients, when backpropagating them. It occurs when the network has too many layers. In order to calculate the gradients for early layers, multiple matrix multiplications have to be calculated. If a lot of the weights of these matrices are either smaller or bigger than 1, the final result will move exponentially towards either 0 or ∞, respectively.

Gate units have trainable gates that determine when to remember and when to forget information. Intuitively, the input gate controls the amount of information entering the cell, the forget gate controls the amount of information that stays in the cell and the output gate controls the amount of information exiting the cell (Figure 2.3). These gates are basically single neurons with the Sigmoid activation function, squashing the output between 0 and 1. This limitation between 0 and 1 hinders the gradient from *exploding*, which would happen, if the activation is larger than 1, and backpropagated too many times. This gate acts as a binary switch between forgetting and keeping the information. This way, the gradients either stay constant or they are set to zero.



[38]

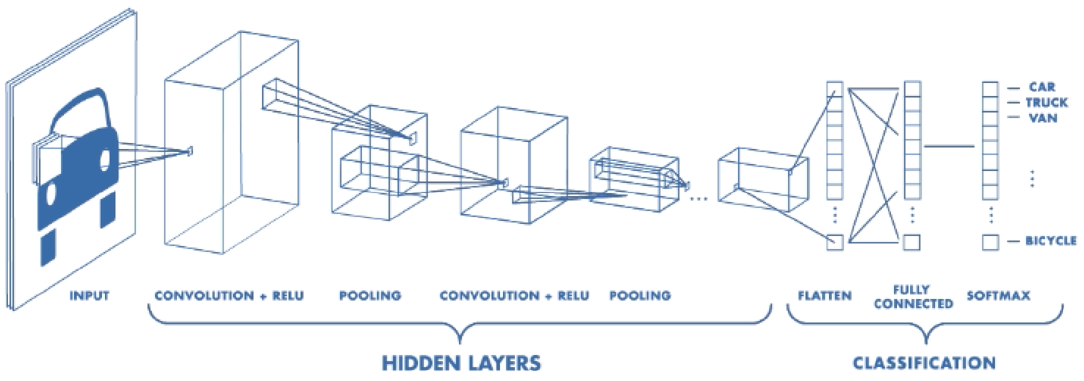
Figure 2.3: A peephole LSTM cell. x_t describes the input from all sources: input or previous layers and recurrent connections. h_t is the hidden state of the network that will be propagated to the next unit and to the next sequence step.

2.4 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are most famous for their performance in image processing, especially image-classification [24, 46, 65, 109, 116] and image-detection tasks [93]. A regular FFNN network connects every node from layer l to every node of its succeeding layer $l+1$. A CNN only connects a certain number of nodes to one of its succeeding nodes. This is similar to applying a filter or a convolution to an image; thus the name CNN.

This approach saves a lot of computational time, on the one hand, on the other hand, local context becomes more important in comparison to a FFNN, where the whole input is considered at once. Between the convolutional layers, a CNN normally has pooling layers, decreasing the size of the image representation. After the last convolutional layer, fully connected layers are often added, e.g. classification layer.

Figure 2.4 shows schematically, how a CNN for image classification can look like. The input image is projected to the output layer by various convolutional, pooling and fully connected layers.



[21]

Figure 2.4: An abstract view on a CNN for image classification.

2.5 Autoencoders

Autoencoders, first mentioned in 1985 by Rumelhart et al. [100], map a given input into a context space and back to the original input. By doing this, a compressed context vector with the most important information of the input vector is created.

When using a NN as autoencoder [97], the network normally has a bottleneck in form of a hidden layer, smaller than input or output layer. The activation of this hidden layer is used as context vector.

Almost all autoencoders, used today, fall under either one of these four categories: Denoising autoencoders [122], sparse autoencoders [35], contractive autoencoders [98] or variational autoencoders [97].

2.6 Generative Adversarial Networks

Generative Adversarial Networks (GANs) were proposed by Goodfellow et al. [36] in 2014 and have since seen an incredible amount of attention in the scientific community. The general idea is equally simple and brilliant.

If the objective is posed, to reproduce a certain sort of available data, it has historically been approached by applying Maximum Likelihood Estimation (MLE), to maximize the similarity between target and generated data. The problem with this approach is the lack of wrong samples. Training a system on what it is supposed to generate but never on what it is not supposed to generate will result in a weak system.

GANs solve this, by continuously using the output of a generator G, during its own development towards generating better and better samples, as negative samples. These negative samples, together with the original samples are used to train a discriminator D to discriminate between real samples and generated ones. G is trained to generate samples so that the D believes it to be a real sample.

Both systems are trained alternately and try to "beat" their counter player. This is often done in a min-max fashion, where the generator aims to maximize the error of the discriminator and the discriminator aims to minimize its own error.

During this process, G is unaware of the real data distribution and usually uses random noise as a starting point to learn to replicate the real data using only the feedback from D. If G receives meaningful input (e.g. image) that is needed for the generation, it is called a conditional GAN, since the generator creates samples, conditioned on a certain input [87].

A recent improvement of GANs is called Wasserstein GAN (WGAN) [5, 39] and introduces the usage of the Wasserstein distance, also called Earth Mover's distance [99], as loss function for GANs. This allows for a more stable training process and an interpretative loss function.

2.7 Word embedding models

When working with mathematical models, like NNs, words have to be transformed from their textual form to a numerical form. A simple way to do this is by converting every word into one-hot encoded vectors, where every word is represented by a vector $e \in (0,1)^V$, where V is the vocabulary size. Every embedding vector contains $V-1$ zeros and a single one at position i, representing the ith word of the vocabulary. The disadvantages of this commonly used technique are that the size of the created word-vectors can become very large and that different one-hot encoded words have no visible relation to each other.

In order to solve both shortcomings, different approaches have been proposed. In 1990 Deerwester et al. [23] proposed latent semantic analysis to create a numeric representation of documents and words by counting. More recently, Mikolov et al. [85] proposed dense word embeddings that can be regarded as an effective, continuous approximation of count-based distributional models. Levy et al. [69] compare the two approaches and report that both show similar overall performances over various tasks. Due to the simplicity, effectiveness, and availability of pretrained dense embedding models, they are often chosen as word embedding models nowadays.

When using dense word embeddings, words are not embedded as discrete one-hot vectors but as a continuous dense vector of m float values. Every vector is defined by $e \in \mathbb{R}^d$, where d is normally much smaller than V. For this technique, words are embedded by using a NN that works similar to an autoencoder. Words are embedded into one-hot vectors, before they are transformed into dense embeddings via either the Continuous Bag-of-Words (CBOW) or the Continuous Skip-gram (Skip-gram) approach. Skip-gram trains a model to project every word onto its n neighboring words. This is done for every word throughout huge corpora and the resulting hidden layer is taken as new dense word embedding. CBOW does the reverse. The neighboring words of every word are projected on the word itself.

The size of the hidden layer can be chosen to be much smaller than the vocabulary size. Often the word embeddings have 50, 100 or 300 dimensions. A second benefit of this method is the fact that words with similar neighborhoods have similar representations. Animal-words, for example, like *cat* or *dog* are likely represented by similar vectors, since their neighboring words will often be similar. This technique is commonly referred to as word2vec.

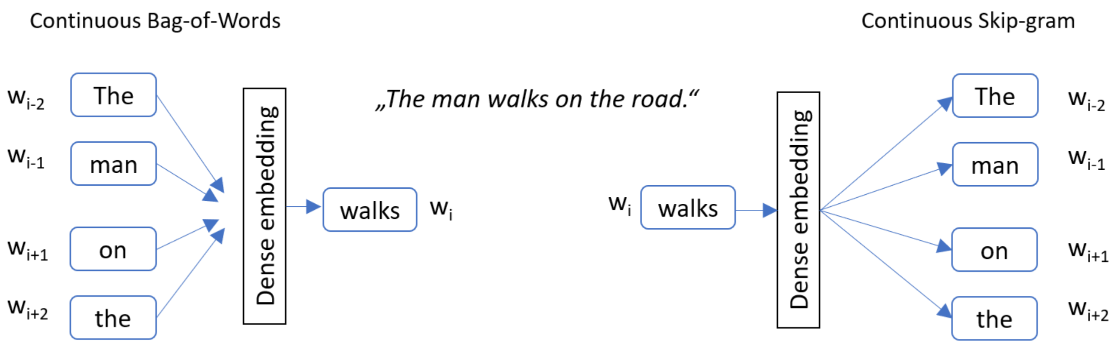


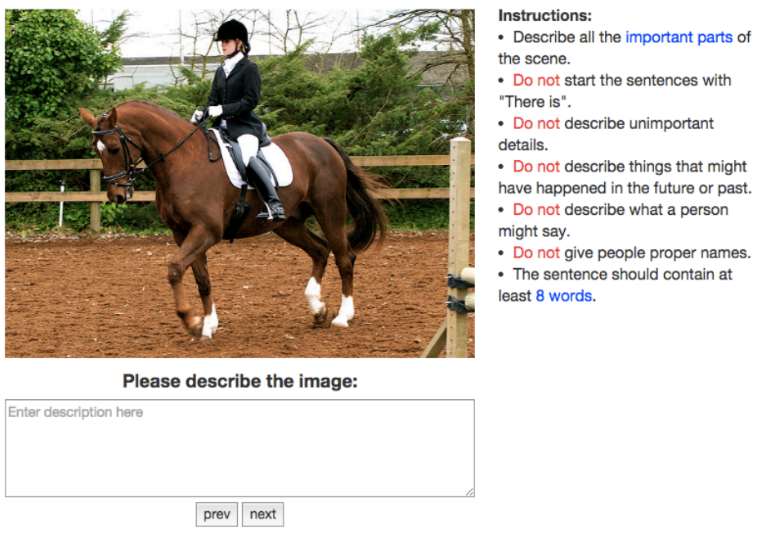
Figure 2.5: The CBOW model and the Skip-gram model. Words are mapped onto their neighborhood or the other way around. The resulting hidden layers are called dense word embeddings.

A disadvantage of word2vec is the fact that large text files (e.g. Common Crawl: 600B tokens) contain too many different words, many of them only occurring once or twice [136]. The resulting word2vec model becomes incredibly large and many words have an inaccurate vector representation. Word2vec models solve this by only using words, occurring more than t times in the given corpus. By doing this, though, many tokens are unknown to the model.

Fasttext [11, 86] tries to solve this issue by not assigning a word-vector to every single word but instead uses sub-information of the words to create the embeddings. This concretely means, using character n-grams to make up words. Character n-grams describe any occurrence of n consecutive characters in a text. By default, n-grams of sizes between 3 and 6 are used to build-up models. When using the model, n-grams can be recombined to build different words. While embeddings of known words can be drawn directly from the model, it also has the capability of computing representations for unknown words based on their character sequence. With this method, more words can be stored while the models stay relatively lean.

3 The MSCOCO dataset

The Microsoft Common Objects in Context (MSCOCO) [16] dataset was used to train and validate all three models: Automated Image Description (AID), Natural Language Image Search (NLIS) and Image Captioning-Retrieval (ICR). The dataset contains 123,287 images of various size and content. Most images display common, everyday scenes. Each image was annotated/described by five different individuals with the main objective to "Describe all the important parts of the scene." [16]. Figure 3.1 shows, how the interface for the human annotators looked like. As one can see from the instructions, the descriptions are focused on the most important parts of the image.



Instructions:
- Describe all the important parts of the scene.
- Do not start the sentences with "There is".
- Do not describe unimportant details.
- Do not describe things that might have happened in the future or past.
- Do not describe what a person might say.
- Do not give people proper names.
- The sentence should contain at least 8 words.

Please describe the image:

[16]

Figure 3.1: The user interface for human annotators.

The dataset has two different splits. The 2014 split contains 82,783 training images, 40,504 validation images. The 2017 training split contains 118,287 images and the validation split 5,000 images. In addition, there are 40,775 test images with hidden annotations. 5,000 of those test images were randomly selected and annotated with 40 different captions since it was shown that many evaluation measures can have a higher correlation with human judgment when given more, different reference sentences [120].

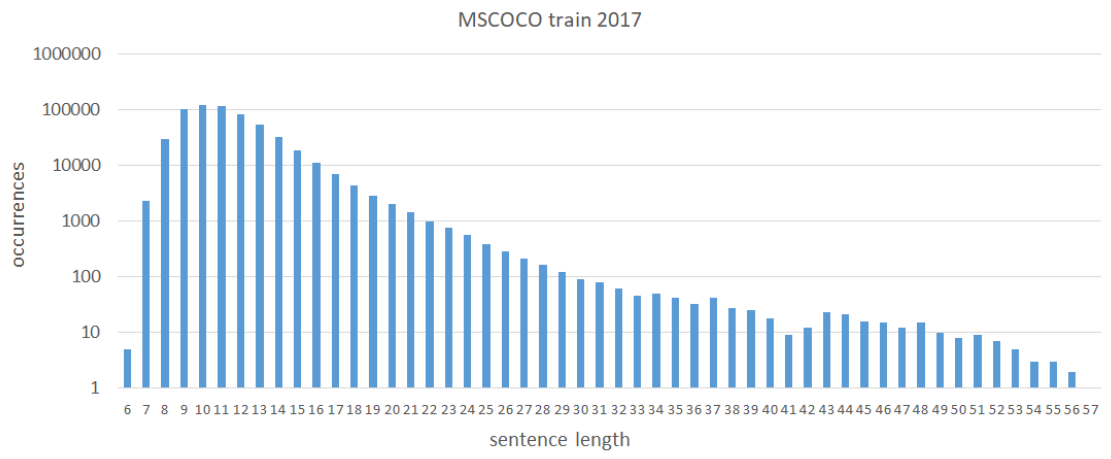


Figure 3.2: Length of sentences and number of occurrences in the MSCOCO 2017 training set.

Sentences, describing the images from the 2017 training set range from 5 to 57 words in length, including punctuation. Figure 3.2 shows that most of the 519,753 sentences have a length of 10 words. The number of sentences is exponentially decreasing with increasing sentence length.

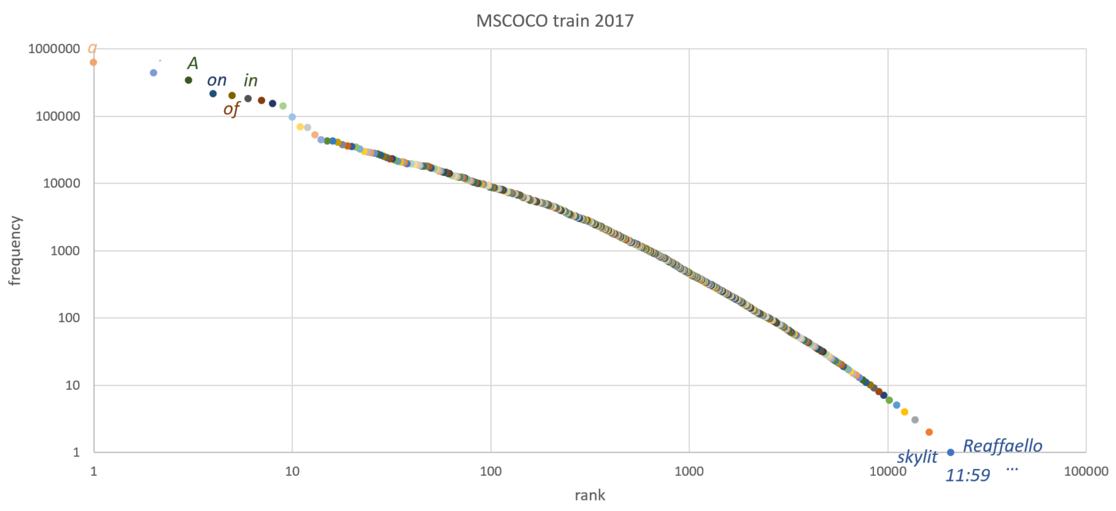


Figure 3.3: Frequency of each word is plotted against its rank in a log-log plot.

The MSCOCO 2017 training set contains 6,687,792 words. Figure 3.3 shows the frequency of every word in ranked order. In order to rank the words, they are first sorted decreasingly by frequency. The most frequent word gets the rank of 1. The rank for the next word is calculated by adding the number of words with the previous frequency to the rank of the previous word (Eq. 3.3). This is basically a min-sorting by frequency, where all words with the same frequency get the lowest rank.

$$\text{rank}(w_{f=1}) = 1 \tag{3.1}$$

$$\text{rank}(w_{f=2}) = \text{rank}(w_{f=1}) + \text{occurrences}(w_{f=1}) \tag{3.2}$$

$$\text{rank}(w_{f=n}) = \text{rank}(w_{f=n-1}) + \text{occurrences}(w_{f=n-1}) \tag{3.3}$$

The graph displays the typical distribution of word-occurrences in human language, following Zipf's law [136]. Human language contains a relatively small number of words that are used extremely frequent. These words are visible in the upper left corner of the graph. Some of them are marked with the word they represent. The most frequent word *"a"* occurred 633,904 times alone. On the other side of the spectrum, many words occur only once or twice. The extreme dimensions of this phenomenon are visualized in the log-log plot in Figure 3.3.

The marked dot with word frequency of 1 represents 16,837 different words out of a total of 37,678 unique words. This means 44.7% of all words occur only a single time. At the same time, these 16,837 words make up only a tiny fraction (0.025%) of the total word count of 6,687,792.

When masking every word with 4 or fewer occurrences with *unknown*, the number of different words decreases to 12,203 (32.38% of 37,678 different words) while losing the information from 39,764 words. This equals only 0.059% of the total 6,687,792 words. This technique is often applied to keep the size of the used vocabulary tractable.

4 Automated Image Description

This chapter covers the first of two systems needed to create a simple conversation about images: Automated Image Description (AID). Most of the related work presented in this chapter uses the terminology: Image Captioning (IC). This is normally done when the created descriptions are only short sentences and caption just the most important feature of the image. Image description is a broader term, including IC but also more detailed descriptions. Since the ultimate goal is to create rich and diverse natural language descriptions, the terminology AID has been chosen.

The following sections show how AID was and is tackled in related work (Section 4.1). The selected approach to AID is explained in more detail in Section 4.2. Section 4.3 covers the experimental setup, including the implementation details and the experiment design. In the end of this chapter, the results of the experiments are displayed, compared to similar studies (Section 4.4) and critically discussed (Section 4.5).

4.1 Related work

This section will focus mostly on how encoder-decoder architectures can be used to solve AID, how these models are optimized in related work and how they are traditionally evaluated.

Different approaches, like retrieval based methods [26, 82, 89, 90, 118, 132] have been proposed to solve the AID problem, but since one goal of this thesis is to generate novel descriptions, they are not mentioned further. For an overview of the history of AID, the available datasets and the popular evaluation metrics Bernardi et al. [10] is recommended.

4.1.1 Encoder-decoder architecture

Encoder-decoder models always follow the same general pattern (Figure 4.1):

1. *Computer vision techniques are used to encode an image into a context/image vector, representing the information of the image in a condensed way.*
2. *The image vector is decoded into a sequence of tokens that best describe the content of the image.*

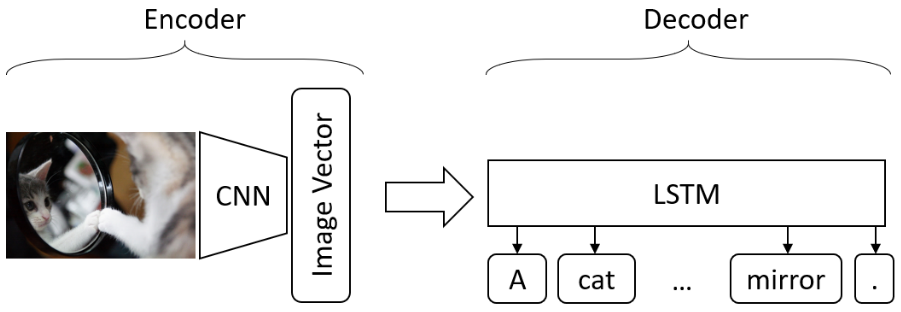


Figure 4.1: The encoder-decoder CNN-LSTM architecture from a high-level perspective.

Encoder

Encoding in this context describes the process of embedding an image into a vector, preferably in a lower-dimensional space. Optimally, the created vector contains a maximum amount of information about the image in form of a dense representation. The result is called context vector or image vector. Different approaches have been proposed to do this.

Farhadi et al. [31] use spatial relationships, Yang et al. [134] use corpus-based relationships and Kulkarni et al. [67] applied spatial and visual attributes to the image.

Nowadays, this step is often performed with a Convolutional Neural Network (CNN) as image encoder [17, 26, 27, 57, 61, 62, 124, 125]. All the mentioned studies use pretrained CNNs, normally trained on the ImageNet dataset [24] for image classification. Karpathy and Fei-Fei [57] use a Region Convolutional Neural Network to explicitly add spacial image information and Xu et al. [130] use an attention mechanism to focus more on salient regions in images.

Decoder

The decoding step describes the transformation of an image vector to a series of tokens or words. It can also be implemented in various ways.

Historically, template-based models [31, 134] were commonly used. Templates, designed by experts, are filled by selecting the most likely objects, actions, attributes, and prepositions based on, for example, a Hidden Markov Model. By their nature, templates strongly determine the structure of the generated sentences.

A more flexible approach is the use of language models. They are less constrained and can create novel text. A language model is a probabilistic model that, given n words, will return a probability for every word in a defined vocabulary of being the next word. This is traditionally done with word n-grams [67, 71] or with Neural Networks (NNs) [47]. A word n-gram is a sequence of any n words, occurring in a given text. Word n-gram models can be seen as discrete and NNs as continuous language models.

Chen and Zitnick [17], Donahue et al. [27], Karpathy and Fei-Fei [57], Kiros et al.

[61, 62], Mao et al. [81], Vinyals et al. [124, 125] and Devlin et al. [26] all use a CNN for encoding images and a Recurrent Neural Network (RNN) as language model. RNNs are often implemented in form of Long-Short-Term Memory (LSTM) cells or Gated Recurrent Units (GRUs), since they have better capability of remembering longer sequences.

The usage of NNs as encoder and decoder differs from previously used methods mainly in the fact that no hand-crafted rules, no templates, no corpus and no predefined categories like word types or such are needed. Instead, a large annotated training dataset is required to train the weights of the network.

4.1.2 Optimization strategies

Encoder-decoder architectures in form of CNN-RNN networks have claimed their place as state-of-the-art architecture when it comes to AID. The leader-boards of image captioning competitions like the MS COCO 2015 Image Captioning Task [16] clearly show this. While the architectures of the used models are similar throughout the literature, their optimization strategies vary greatly.

Maximum Likelihood Estimation

N-grams and neural models are both commonly optimized with Maximum Likelihood Estimation (MLE) [17, 26, 27, 57, 61, 62, 81, 124, 125]. This is normally done, by minimizing the cross-entropy between a generated word probability distribution and the *true* distribution. The *true* distribution refers to a one-hot vector of the *true* word.

When training a RNN, with MLE, to generate a sequence of word probabilities, two training strategies can be applied. The first option is to apply every generated word, to the already existing sentence, in order to create the new sentence, needed for the prediction of the next word (Figure 4.2b). This approach is very time-consuming, since the variety of words combined with the sentence length allows for an incredible amount of different sentences, with only a small partition being correct sentences of the language.

In order to overcome this problem, *teacher forcing* is commonly used. *Teacher forcing* describes the process, where input and target are the same sentences. Any wrong predictions in the training phase have no influence on the rest of the training episode (Figure 4.2a). This allows a fast training with good results but will often lead to the *exposure bias* [92]. This means, once the model is in the prediction phase, it can no longer rely on a target sentence as input since there is none. If the model creates unseen word sequences in this phase, it will react unpredictably. Since the model was only trained on correct sequences, it has never learned to recover from such a situation.

Bengio et al. [8] try to solve this by introducing generated sentences during training, but Huszar [53] shows that this will not converge to the true data distribution.

Reinforcement Learning

Bengio et al. [8] propose a different idea to work around the *exposure bias*. They gradually add a Reinforcement Learning (RL) [6, 115] reward, derived directly from the BiLingual Evaluation Understudy (BLEU) [91] score, which measures overlap between generated and *true* output (Section 4.1.3).

More studies use this technique to achieve higher evaluation measures. Liu et al. [76], Ranzato et al. [92], Rennie et al. [96] and Liu et al. [75] use the REINFORCE [128] algorithm to maximize scores like BLEU [91] or Consensus-based Image Description Evaluation (CIDEr) [120] (Section 4.1.3). Zhang et al. [135] do the same, but they use the Actor-Critic framework [7, 129].

Generative Adversarial Networks

A different approach to avoid the *exposure bias* is the usage of Generative Adversarial Networks (GANs) [22, 68, 72]. Instead of comparing sentences directly to the *ground truth*, a discriminator is trained to judge how well image-sentence pairs fit together. The generator is only trained to fool the discriminator. This way, the system has a more implicit objective, of creating human-like captions instead of the direct objective to copy the *ground truth* sentences. It should be noted here that GAN models are normally pretrained with MLE, to decrease training time.

4.1.3 Evaluation

The question of how to score produced sentences or even dialogues is still an open research topic. There are two different approaches, that are commonly seen in the related work.

Word *n*-gram evaluation measures

The most widely used evaluation metrics are *n*-gram evaluation measures. One measures the similarity between two sentences or paragraphs by comparing their respective occurrences of word *n*-grams.

The **BLEU** [91] score represents a modified precision of *n*-grams. The BLEU score is calculated by counting all the word *n*-gram matches between the candidate and a *ground true* sentence. Equation 4.1 shows how to calculate the BLEU score.

$$\text{BLEU} = \frac{m_{max}}{n_c} \tag{4.1}$$

Let m_{max} be the number of matches between the candidate and a *ground true* sentence, with the limit at the **max**imum total count of the *n*-gram in the *ground truth*. Let n_c be the total number of *n*-grams in the candidate sentence.

The **Recall-Oriented Understudy for Gisting Evaluation (ROUGE)** [73] score is very similar to the BLEU score. It represents either the recall or the precision between candidate and *ground truth* (Equation 4.2).

$$\begin{aligned} \text{ROUGE}_{\text{precision}} &= \frac{m}{n_c} \\ \text{ROUGE}_{\text{recall}} &= \frac{m}{n_{gt}} \end{aligned} \tag{4.2}$$

Let m be the number of matches and n either the total number of n-grams in the candidate or the *ground truth* sentence.

The **Metric for Evaluation of Translation with Explicit ORdering (METEOR)** [25] score is the harmonic mean between the precision and the recall, where precision is weighted 9 times more important than recall. METEOR also adds a penalty for exact copies of the *ground truth* sentence.

CIDEr [120] is a weighted statistic over n-grams designed for AID. CIDEr places more importance on n-grams that frequently occur in the *true* sentence describing an image and reduces the weight of n-grams that occur across all *true* descriptions with a high frequency. It provides a measure of word salience by discounting words with high overall frequencies and increasing the weight of special words, only occurring in a few *true* sentences. Thus, it rewards rich and diverse sentences over generic ones. It was shown that CIDEr has a higher correlation with human judgment, as compared to conventional evaluation measures like BLEU or ROUGE [120].

Word n-gram scores are fast and easy to calculate and offer comparable metrics for various Natural Language Processing (NLP) tasks. It should be noted here that these scores do not automatically represent the quality of the generated sentence. Comparing the generated sentences to multiple *ground truth* sentences increases the validity of evaluation measures being a good representation for the quality of captions but does not make them perfect measurements. They further only measure the similarity of words, not the similarity of meaning.

The interpretations of n-gram evaluation measures results should be done with care. A BLEU score of 1, for example, can only be reached by exactly copying the ground truth sentence. Quoting Papineni et al. [91]: *"The BLEU metric ranges from 0 to 1. Few translations will attain a score of 1 unless they are identical to a reference translation. For this reason, even a human translator will not necessarily score 1.* [...] *on a test corpus of about 500 sentences (40 general news stories), a human translator scored 0.3468 against four references and scored 0.2571 against two references."* CIDEr on the other hand can reach values beyond one. Chen et al. [16] offer additional information on n-gram evaluation measures and their agreement with human judgment.

Ranking

Hodosh et al. [50] offer a different evaluation technique for AID. They propose to evaluate a system based on its capability to rank image-sentence pairs in a retrieval task. This means the performance is measured based on how well the system can retrieve the correct caption for an image or the correct image for a caption. Hodosh et al. [50] argue that this is a more accurate way to determine whether a system has learned the correct mappings from image to textual information. They also report a higher correlation between retrieval scores and human evaluations compared to traditional evaluation measures. Next to n-gram scores it is the most frequently used evaluation metric for AID systems. There is some argument about whether ranking methods caption the quality of image descriptions better than traditional evaluation measures do.

Chen and Zitnick [17], Donahue et al. [27], Kiros et al. [61, 62], Mao et al. [81], Vinyals et al. [124] and Karpathy and Fei-Fei [57] all test their model on its image and sentence retrieval performance. For caption retrieval this is intuitively done, by generating the probabilities for a given image and ranking n *ground truth* captions based on their similarity to it. For image retrieval, the probabilities for n images are generated by the AID model and ranked based on their similarity to every single *ground truth* query sentence.

4.2 CNN-LSTM network

In order to build an AID baseline model, the CNN-LSTM encoder-decoder architecture with MLE was selected and implemented. This section explains in detail how the model works.

The AID model receives an image $x^{im} \in \mathbb{R}^{h \times w \times c}$, where h, w, c are the height, width and color dimension, and a sequence of words and will output a probability distribution for the next word. This step will be repeated until the end of sentence symbol (*end-symbol*) is generated or a fixed maximum sentence length is reached.

The input image is resized and fed through a CNN with the parameters θ_ϕ that extract the most important image features and project the image onto a vector $\phi(x^{im}, \theta_\phi) \in \mathbb{R}^k$, where k is the output dimension of the CNN. The respective description sentence is embedded in a dense word embedding, resulting in the second model input $x^{se} \in \mathbb{R}^{t \times d}$, where t is the number of words in a sentence and d is the dimensionality of every dense word embedding.

x^{se} is fed word by word into a LSTM layer. $\phi(x^{im}, \theta_\phi)$ can either be infused once at the beginning of a training sequence or it can be added at every generation step. The latter is more common since it allows the image information to be present at every generation step. The LSTM block is followed by a block of fully connected layers and a final softmax layer, squeezing the model output into t probability distributions with $\mathrm{P}(y_t | x^{se}_{1 \to t-1}, \phi(x^{im}, \theta_\phi))$, where y_t is the probability over the vocabulary V at timestep t, x^{se} is the information from the previous words and $\phi(x^{im}, \theta_\phi)$ is the image vector.

At training time, x^{se} and the target $y \in \mathbb{R}^{t \times d}$, with the same shape as x^{se}, are representations of the same *ground truth* sentence. This is called teacher forcing. It creates a quick and stable learning process but can lead to the *exposure bias* [92], described in Section 4.1.2. x^{se} is shifted one time-step into the future by adding a *start-symbol* at its beginning. This way, word y_t equals x^{se}_{t+1} and the model is trained to predict the next word of the sentence x^{se}. An *end-symbol* is appended to the y, to make input and output have the same length. Additionally, by adding the *end-symbol*, the network is trained to end every sentence with this token.

At every step t, the unfinished sentence $x^{se}_{1 \to t-1}$ is combined with the image-vector $\phi(x^{im}, \theta_\phi)$, fed into the LSTM block and projected to produce $\mathrm{P}(y_t)$. This is done until the end of the sequence or until a defined, maximum sentence length is reached. Figure 4.2a) shows a training run from an abstract perspective.

When testing the model, only $\phi(x^{im}, \theta_\phi)$ is given to the system. The model starts, like in the training phase, with $\hat{x}^{se}$, containing only the *start-symbol*, as first input and generates $\mathrm{P}(\hat{y}_t)$. Depending on the selection mode (e.g. greedy pick), one word y_t from $\mathrm{P}(\hat{y}_t)$ is selected. The difference to the training phase is that every produced word $\hat{y}_t$ is directly added to the previous input $\hat{x}^{se}_{1 \to t-1}$. The previous sentence input $\hat{x}^{se}$ is constantly updated and serves as new input. Figure 4.2b) shows how the prediction process looks like. This way the model is tested and used.

The classic prediction paradigm is to greedily pick the word with the highest probability at each generation step. In order to further increase evaluation measures, different selection strategies can be applied. Beam-search is classically used for this purpose, where the N most probable words are followed instead of just the most probable. Donahue et al. [27] empirically show that a beam width N of 3-5 shows optimal results and generally boosts results between 1-5 percent points.

A different prediction method is to sample multiple sentences randomly by the probabilities given by the softmax, and selecting the sentence with the highest log-likelihood. When doing this, it further increases the performance to decrease the temperature parameter τ of the softmax function. By decreasing τ, the softmax is transformed further towards a one-hot distribution and by increasing τ the distribution is gradually becoming more uniform. Donahue et al. [27] empirically show that using 100 random samples and $\tau = 0.75$ yields even higher results than using the more popular beam-search approach. However, each approach significantly increases the prediction time per image.

4.3 Experimental setup

This section gives detailed information about how the AID model was implemented and the different experiments that were executed. Table 4.2 offers a comprehensive overview of different hyper-parameter settings and the resulting evaluation measures. The hyper-parameters that were adapted are marked with their respective column names in the

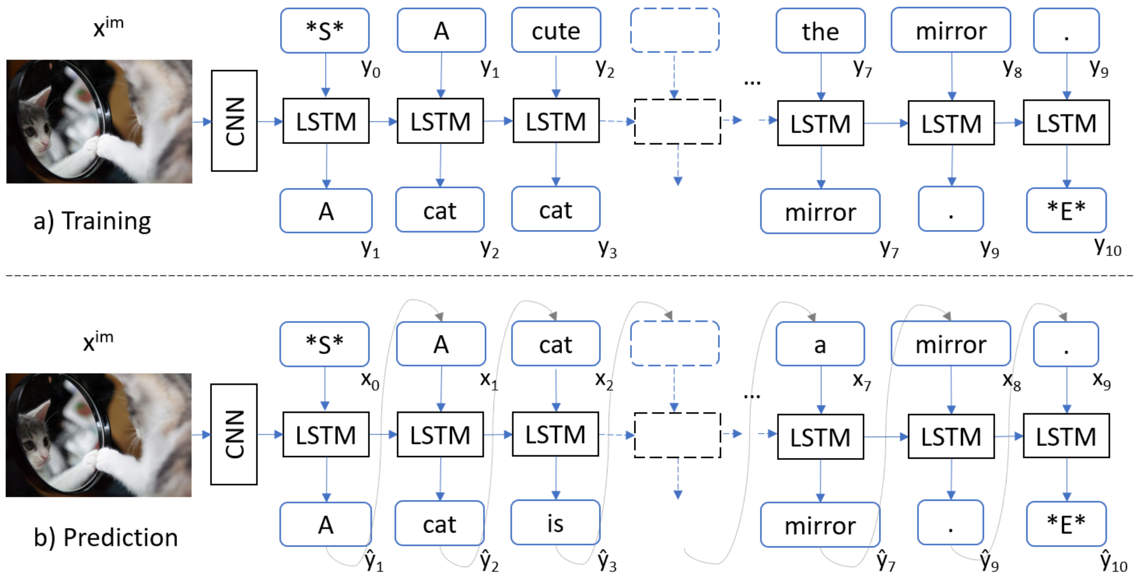


Figure 4.2: a) In training, the input and the target are the same sentence, simply shifted by one timestep. b) In the prediction phase, for not annotated images or during testing, the generated word is appended to the unfinished input sentence and used for the next generation step.

following sections.

Data preprocessing

Words were transformed to contain only non-capital letters and every word occurring less than 5 times was masked as *unknown*. This decreases the vocabulary size in the MSCOCO 2014 data split to 8,857 words. The vocabulary size for the 2017 split for this setting is 10,331. Accepting every word with a minimum frequency of 10 yields 7,456 different words. The vocabulary sizes were determined based on the training datasets only since this would be the available data in a realistic use-case. Both of these steps are necessary to reduce the size of every single sample and to allow training in reasonable time. Every sentence was padded or truncated at the end to have a fixed sentence length of either 16 or 20 words, including either *start-* or *end-symbol*. All training samples in each batch have to have the same length to process them. Since most of the sentences have a length of 10 words, 16 and 20 was chosen as maximum length, to keep the size of the input and output relatively small, while containing a maximum amount of information. Each word was encoded into a one-hot feature vector.

All images were resized to have $224 \times 224 \times 3$ pixels (Section 4.2) and fed through a pretrained CNN to retrieve their respective image vector. The AID model receives only the preprocessed image vectors. This greatly decreases the runtime of the algorithm. Calculating image vectors on the fly is only necessary when the complete CNN is being fine-tuned. This is very time-consuming since the same image has to be processed by the network again every epoch.

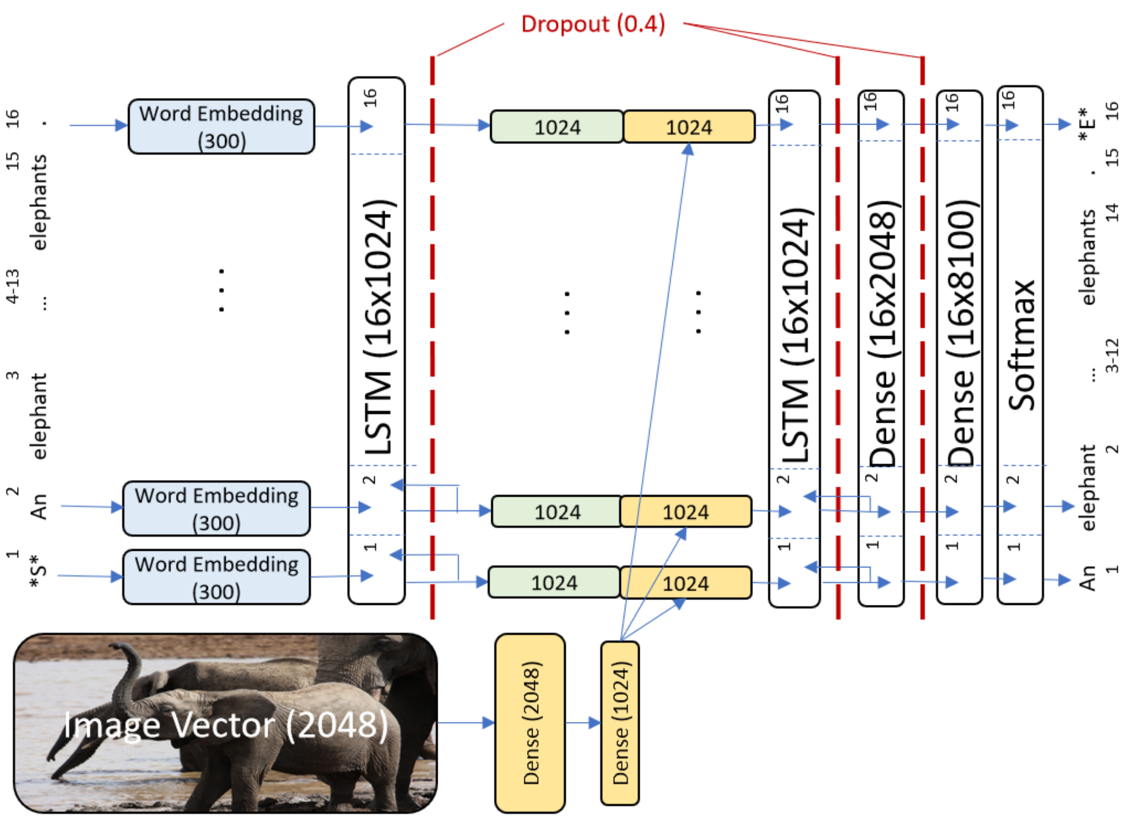


Figure 4.3: AID architecture of our final model.

Implementation

One-hot representations of words are first mapped to pretrained dense vectors from a dense fasttext [86] model, pretrained on Wikipedia dumps.

Following the implementation by Donahue et al. [27] and the hyper-parameter evaluation by Soh [110], a two-layered stacked LSTM architecture with multiple dropout layers was selected to process the embedded sentence word by word. As recommended by Donahue et al. [27], the image information was infused at every generation step, not into the top-level LSTM but after the first (L=1) or second (L=2) LSTM layer.

Soh [110] reports, that dropout layers with a dropout rate (D) between 0.2 and 0.3 optimally hinder the model from overfitting, while leaving it with enough information to successfully perform its task.

Residual Neural Network 50 (ResNet50) [46] was selected as CNN model. It showed the best empirical results when compared to three other commonly used CNN architectures. The pretrained weights are taken from the Keras website[1].

Further, experiments to optimize the architecture of the AID model were conducted. The final architecture is shown in Figure 4.3 and the evaluation measures of different model designs is shown in Table 4.2.

[1]`https://keras.io/applications/`

Table 4.1: Comparing different CNN architectures. ResNet50 shows the best results regarding all chosen evaluation measures. VGG16 shows almost the same results, but the model is nearly five times larger than ResNet50. Our best results are in **bold**. *P=Parameters, D=Depth, T5=Top5-Accuracy, B1-B4=BLEU1-4, M=METEOR, C=CIDEr, R_L=ROUGE_L*

CNN	P	D	T5	B1	B2	B3	B4	C	M	R_L
Inception [116]	$2.38 * 10^7$	159	0.944	0.16	0	0	0	0	0.04	0.2
Xception [19]	$2.29 * 10^7$	126	0.945	0.5	0.28	0.16	0.1	0.2	0.14	0.37
VGG16 [109]	$1.38 * 10^8$	23	0.901	0.63	0.44	0.3	0.2	0.61	0.2	0.46
ResNet50 [46]	$2.56 * 10^7$	**168**	**0.929**	**0.65**	**0.46**	**0.32**	**0.22**	**0.68**	**0.21**	**0.48**

Experiment

All experiments were run either 6 or 13 epochs. As reported by Soh [110], AID models normally learn the most during the first 5 training epochs and can hardly increase their performance with regard to the evaluation measures afterward (Figure 4.4). *N*-gram measures, however, do not directly indicate the quality of the generated sentences and Soh [110] shows examples, where the quality of the descriptions increases further even if this is not measurable quantitatively by the used evaluation measures. For this reason, all experiments were first conducted with 6 training epochs and once the best model was determined, it was trained for 13 epochs. Adam [59] was used as optimizer with the default learning rate of 0.001.

After every training episode, the whole validation dataset was used to predict a caption with greedy picking for every image and all used evaluation measures were calculated with the evaluation script provided by Microsoft Common Objects in Context (MSCOCO).

When experimenting with different parameter settings, the MSCOCO 2014 data-split was chosen, since most of the related work uses the 2014 split, and thus allowed a direct comparison of the performance of the model. Before uploading the results to the MSCOCO test server, the model was trained on the 2017 split, since it contains more training data, while still keeping 5,000 images for validation.

4.4 Results

The first empirical results are reported on the performance of different pretrained CNN models. Even though ResNet50 [46] did not perform best on either the top-1 or top-5 accuracy image classification task, it produced the best results compared to VGG16 [109], Xception [19] and Inception [116]. ResNet50 also only contains $2.56 * 10^7$ parameters, compared to the second best-performing VGG16 with $1.38 * 10^8$ parameters. Based on these results, ResNet50 was selected for all further experiments.

ResNet50 was combined with different decoder architectures to reach similar results as observed in the literature. Finetuning the model on the validation dataset of MSCOCO

Table 4.2: Showing the best performance for the MSCOCO validation 2014 dataset compared to the results from Donahue et al. [27]. Our best results are in **bold**. *Co=CNN model, V=VGG16, R=ResNet50, F=fine-tuned CNN, L=LSTM layers before merge with image vector, D=Dropout rate, TD=Time distributed dense vector after first LSTM, W=minimum word frequency, B1-B4=BLEU1-4, M=METEOR, C=CIDEr, R_L=ROUGE_L,*

Model	Co	F	L	D	TD	W	B1	B2	B3	B4	C	M	R_L
Our Model	R	N	2	0.25	-	8	0.65	0.47	0.32	0.23	0.68	0.21	0.48
Our Model	R	N	2	0.25	-	5	0.66	0.47	0.33	0.23	0.68	0.21	0.48
Our Model	R	N	1	0.25	512	5	0.66	0.48	0.34	0.23	0.71	0.22	0.49
Our Model	R	N	1	0.4	1024	5	**0.67**	**0.49**	**0.35**	**0.24**	**0.75**	**0.22**	**0.49**
LRCN [27]	V	N	1	-	-	-	0.67	0.49	0.35	0.25	0.77	0.23	0.50
LRCN [27]	V	Y	1	-	-	-	0.70	0.52	0.37	0.27	0.84	0.24	0.51

Table 4.3: C40 results from different studies, taking part in the 2015 COCO Caption Challenge Competition. Results are acquired through uploading the captions for the MSCOCO 2014 test dataset to the evaluation server. They are sorted by CIDEr score. Our best results are in **bold**. *B1-B4=BLEU1-4, M=METEOR, C=CIDEr, R_L=ROUGE_L*

Model	B1	B2	B3	B4	C	M	R_L
VSA [57]	0.828	0.701	0.566	0.446	0.692	0.28	0.603
UVS [62]	0.848	0.747	0.633	0.517	0.752	0.294	0.635
Our Model	**0.835**	**0.718**	**0.591**	**0.472**	**0.753**	**0.294**	**0.623**
m-RNN [81]	0.890	0.801	0.690	0.578	0.896	0.320	0.668
Human [74]	0.880	0.744	0.603	0.471	0.910	0.335	0.626
LRCN [27]	0.895	0.804	0.695	0.585	0.934	0.335	0.678

2014 resulted in the model shown in Figure 4.3 and the evaluation measures displayed in Table 4.2 for the validation dataset.

The results of Donahue et al. [27], without finetuning the CNN, could almost be replicated. Only small differences can be observed in the CIDEr score, which is 0.02 smaller than the reported value by Donahue et al. [27]. METEOR and ROUGE are 0.01 behind the state-of-the-art paper and the rest of the scores have the same value. According to Donahue et al. [27], finetuning the complete CNN can increase scores between 0.01 and 0.07 depending on the metric.

Using the same model, training it on the 2017 split and uploading the generated descriptions to the official evaluation server results in the evaluation measures reported in Table 4.3. The results are reported together with results from similar studies. They are all derived through the C40 dataset provided by the 2015 COCO Caption Challenge Competition. C40 refers to a dataset with 5,000 images annotated with 40 captions per image [16].

Figure 4.4 shows the performance of the model reported in Table 4.2, row 3, and in Table 4.3 during its training phase. Most of the scores only increase slightly (<0.025) after the first training epoch. CIDEr increases the most with a difference of 0.099 from first to

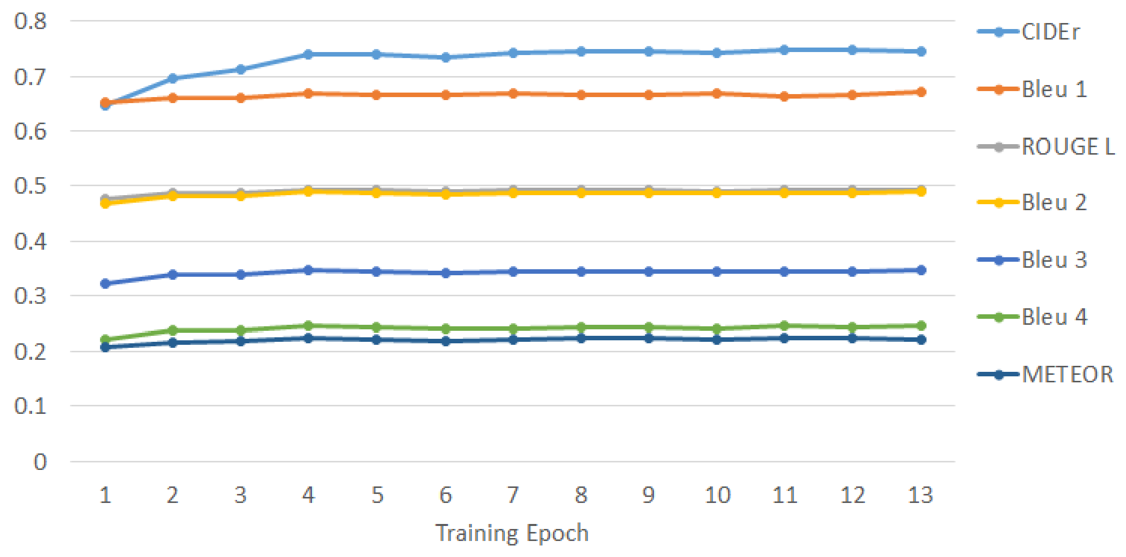
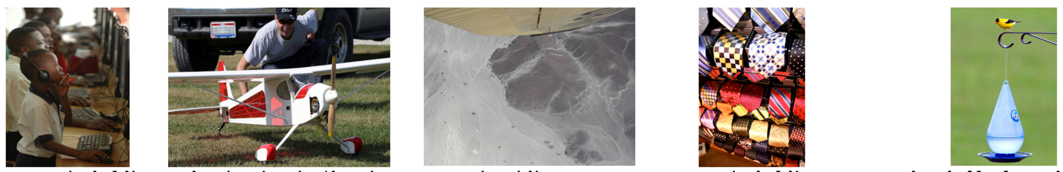


Figure 4.4: Evaluation measures calculated after every training epoch.



a woman is holding a box of food .

a dog jumping in the air catching a frisbee .

a man is riding a skateboard on a ramp .

a man is holding a teddy bear in a room .

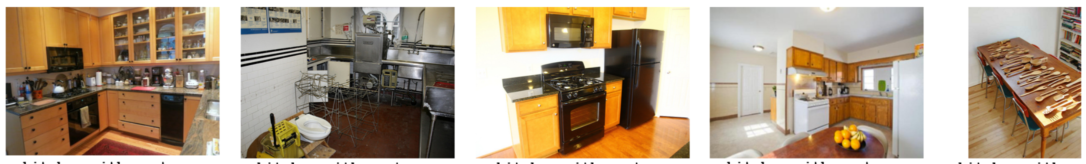
a baseball player holding a bat in his hand .

Figure 4.5: Badly described images

13th epoch.

Figure 4.5, 4.6 and 4.7 show example images with descriptions. Figure 4.5 shows images with wrong or very bad descriptions. The descriptions are grammatically correct, and they might fit a different image, but they are not well suited for the image they are supposed to describe.

Figure 4.6 shows different images with very similar, template-like descriptions. The images all show some sort of kitchen or dining room and the respective descriptions all sound like they could be generated with a template. In general, they are very similar and all start with the same five words. This is also observable for similar image categories like bathroom, street, skating or surfing images.



a kitchen with a stove , sink , and a stove .

a kitchen with a stove , stove , stove , and a stove .

a kitchen with a stove , stove , stove , and a stove .

a kitchen with a stove , sink , and refrigerator .

a kitchen with a stove , a stove , a stove , a microwave , a

Figure 4.6: Very generically, template-like described images.

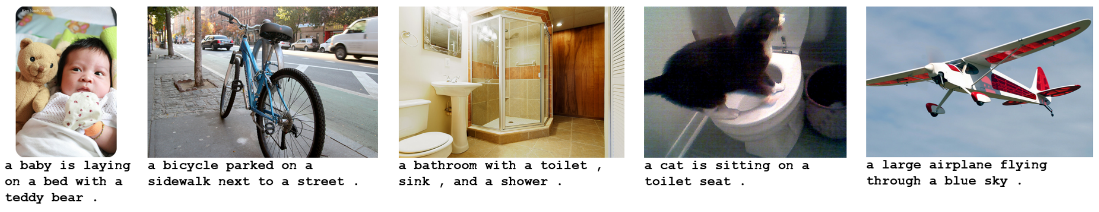

a baby is laying on a bed with a teddy bear .
a bicycle parked on a sidewalk next to a street .
a bathroom with a toilet , sink , and a shower .
a cat is sitting on a toilet seat .
a large airplane flying through a blue sky .

Figure 4.7: Well described images.

Finally, Figure 4.7 shows images described well by the algorithm. These images also show high evaluation measures, especially high CIDEr scores. The content of the images is accurately described and the grammatical and syntactical structure of the sentences is correct.

4.5 Discussion

This chapter describes, how an encoder-decoder model, trained with MLE can yield a baseline model for AID. The model was not created with the intention to maximize evaluation measures but to be used as pretrained model, in a larger Image Captioning-Retrieval (ICR) network, presented in Chapter 6.

Quantitative Analysis

In Table 4.2, the validation results from our model are compared to a previously published paper [27] that uses a similar architecture and the same loss. Looking at the experimental settings that are most similar to our settings, the differences in results is very small (~0.01).

Our results can support the claim of Donahue et al. [27] that the infusion of the image information between two stacked LSTM layers yields optimal results. We can only partly confirm the observation of Soh [110] on the dropout rate between layers. We could observe an even better result with a dropout rate of 0.4 compared to his proposed rate between 0.2-0.3. Additionally, we could observe better results, when including a temporal distributed dense layer between the two LSTM layers that was concatenated with the image information at every timestep.

Our results further strengthen the evidence for encoder-decoder architectures, trained with MLE to generate descriptions with strong baseline evaluation measures.

A correlation between evaluation measures, especially CIDEr [120] and human judgment is empirically shown by Chen et al. [16]. Table 4.3 shows the evaluation measures, humans achieved, when asked to perform the image description task. The fact, that Donahue et al. [27] achieve better results than the human baseline shows that the relationship between human judgment and the chosen evaluation measures is not perfect. *N*-gram based evaluation measures capture mostly, how similar the sentences are in wording and not in meaning, compared to *true* sentences.

A second explanation for the higher results of an artificial system compared to the human baseline could be the definition of the task. Machines, trained to maximize the similarity between the generated sentences and an underlying *true* sentence, while humans, if not explicitly told otherwise, simply describe images intuitively, without trying to achieve a high resemblance to an already existing human description. For a human annotator, his description is the *true* description, he will not try to create a generic sentence, that has a high chance of matching *n*-grams with other human descriptions.

When comparing our results to the human baseline in Table 4.3, a strong trend in the same direction is visible. From a quantitative perspective, this can be seen as a clear indicator that our model is capable of capturing the most essential image features and to transform them into sentence representations, similar to a human annotator. The comparison to different related work further shows, that our approach, even if it was only intended to be a baseline, beats other approaches or yields similar results.

Figure 4.4 shows that the evaluation measures only really improve in the first few training epochs. From the different scores, CIDEr shows the biggest improvement after the first training epoch. This training curve is typical for MLE training with *true* annotations as training input. The system quickly learns repeating patterns, like the start of sentences with "a". The robustness of the model is questionable since it has only seen correct sentences and uses greedy pick in the prediction phase. This way, little chance is given to novel sentences, which is beneficial for the system, because it would likely not be able to recover from unseen sequences of words. A qualitative analysis in the next section sheds more light on the performance of our model.

Qualitative Analysis

Expecting a random selection of generated captions allows an interesting insight into how the AID model works and where it fails. The samples in Figure 4.5 are completely wrong descriptions for the images at hand. However, the descriptions themselves are grammatically correct and could describe a different image very well. This means, the model, in a way, categorizes images and then adapts the sentence only slightly based on details in the image itself. In other words, once the system predicts the first two or three words, the rest of the sentence is more or less determined. These results are very similar to other studies, also training their model with MLE. Chapter 6 explains this behavior in more detail, and show, how it could be averted.

Figure 4.6 shows a different flaw in some of the generated sentences. All the described kitchens have very similar descriptions. It feels like they all follow a fixed template that only allows certain words to change. Similar findings have also been reported before when training with MLE as only loss. The model finds a general description that has a relatively high correlation to all similar images. It finds common sentence structures that satisfy a maximum amount of images to a maximum degree. The result yields high correlation between output and target, but the sentences sound stiff and template like.

Apart from these examples, most of the generated descriptions fit the image content very well. The sentences still sound generic or stiff sometimes, but they have a correct grammatical structure and describe the content in a meaningful and correct way.

Future work

There are different proposals to increase the quality of the generated descriptions. Some approaches focus more on maximizing the evaluation measures while other focus on naturalness and how human-like the sentences are.

Using attention mechanism or regional image information can increase performance [57, 130]. Donahue et al. [27] show that finetuning the complete CNN increases evaluation measures. Another way to improve results is to use a different sampling method. Instead of greedy picking, beam-search or multiple-random-sampling can be used and can increase evaluation measures by up to 0.05. This increases prediction time by a large factor, however, making it useless for real-time applications. Using these methods will maximize the performance achievable with MLE.

In order to further increase the quality of the generated sentences, a different loss function has to be chosen. MLE is normally still used to pretrain the network. After a certain epoch, the objective is switched to a more implicit one. Directly optimizing a chosen evaluation measure with RL has shown to generate less generic captions. Another approach is to train the model to describe an image and then retrieve it again. Since this approach is very similar to ICR, it will be covered in more depth in Chapter 6. These approaches can produce less generic descriptions with improved performance.

Conclusion

In conclusion, a strong baseline AID system has been created. The system can successfully describe given images with truly meaningful sentences. It was not our intention to create a new state-of-the-art performance with this model but to create a baseline model with strong word-meaning grounding that can be used as pretrained model in further tasks.

In Chapter 6 the trained system takes its place in a larger ICR network, in order to simulate simple conversations about images. We are confident that the pretrained model is capable of this task.

5 Natural Language Image Search

The transformation of an image into a series of words completes the first part of a successful communication. The next step is to reverse this process. A second system receives the created string and ranks all available images based on their similarity to the description. The objective of the system is to rank the described image as high as possible.

Section 5.1 shows how similar studies try to solve the problem of Natural Language Image Search (NLIS), sometimes also called Sentence-Based Image Search. Section 5.2 explains how this task can be approached with Visual-Semantic Embedding (VSE) and a triplet ranking loss. Section 5.3 describes the experimental setup, including data preprocessing, implementation details and experiment design. In Section 5.4 the results are displayed and in and Section 5.5 these results are evaluated and discussed critically.

5.1 Related work

One of the earliest ideas on how to approach NLIS was to use meta-data like annotations, keywords or descriptions of the candidate images. If those are not available, the images have to be annotated by hand or in an automated fashion (Section 4). A query sentence is compared to the descriptions of all possible images and an *n*-gram similarity measurement (Section 4.1.3) is used to calculate the closest match. This is similar to text-based image retrieval and has a long history of research [15, 117].

Most of the online search engines used this method for their image search service until image processing techniques became reliable enough [55]. In 2011, Schroff et al. [104] categorized images, returned by Google Images on simple search words and report the worst class (shark) precision of only 32%. This might have been due to keyword-search focused algorithms with little or no image processing.

Since the actual image content offers the primary source of information and because new Machine Learning (ML) techniques like Convolutional Neural Networks (CNNs) work very well on continuous state spaces like images, the retreat to only using keywords or annotations is no longer necessary today. Instead, the usage of hybrid models [55, 79] or pure image content based approaches is preferred.

5.1.1 Ranking by AID

Section 4.1.2 shows that image ranking is often used to evaluate Automated Image Description (AID) Systems [50]. This is a valid approach to NLIS, but it has a big disadvantage. In order to rank candidate images based on a query sentence, the log-likelihood for every image has to be calculated. For one image this is done by feeding the query sentence and the image into the AID system (Section 4.2) and storing the log-likelihoods for every *true* word. In order to rank a number of candidate images, this process has to be repeated for every image. This whole image search has to be repeated for every new query sentence. There is no possibility of preprocessing the images since the AID system always needs the query sentence and the image as input. For a growing database of candidate images, this is not a feasible method for NLIS.

5.1.2 Visual-semantic embedding

A common approach to image ranking is the mapping of sentences and images into a shared VSE space, where they can easily be compared by a distance measurement like the Euclidean Distance [9, 13] or the Cosine Similarity [113] (Figure 5.1.2).

Using this method allows the preprocessing of a large image database into vectors in the VSE space. For the prediction, the query is simply projected into the same space and the distance to all candidate image vectors can be calculated.

Canonical Correlation Analysis

Canonical Correlation Analysis (CCA) is used to calculate a linear projection of two vectors into a common space that maximizes their correlation [51]. Since many problems are not solvable through linear CCA projections alone, Kernel Canonical Correlation Analysis (KCCA) was proposed by Akaho [2] and reused for NLIS [50]. Recently, Deep Canonical Correlation Analysis (DCCA) was proposed to overcome the need for a fixed kernel in KCCA. In DCCA, Neural Networks (NNs) are used to project images and sentences into a shared space. The weights of the networks are then trained to maximize the correlation between all pairs, called the CCA loss [3, 14].

Eisenschtat and Wolf [28] propose a different loss. Instead of maximizing the correlation directly, they minimize the L2 loss, to simplify the model. They map an image vector directly onto its respective sentence vector and vice versa with symmetrical networks. Then, they take the hidden state of the middle layer between input and output of each network and try to minimize their Euclidean distance.

Autoencoder

A similar way to realize VSE is to formalize it as an autoencoder [17], where image vectors are used as input and target, so that $x^{im} = y^{im}$. The model is trained to perform AID and reverse the processes. This results in a system that transforms image vectors into sentences and a second system that transforms sentences back to the original input vector. NLIS can be performed by only using the second part of the network that will project a given sentence as close as possible to its respective image vector. This can be seen as a

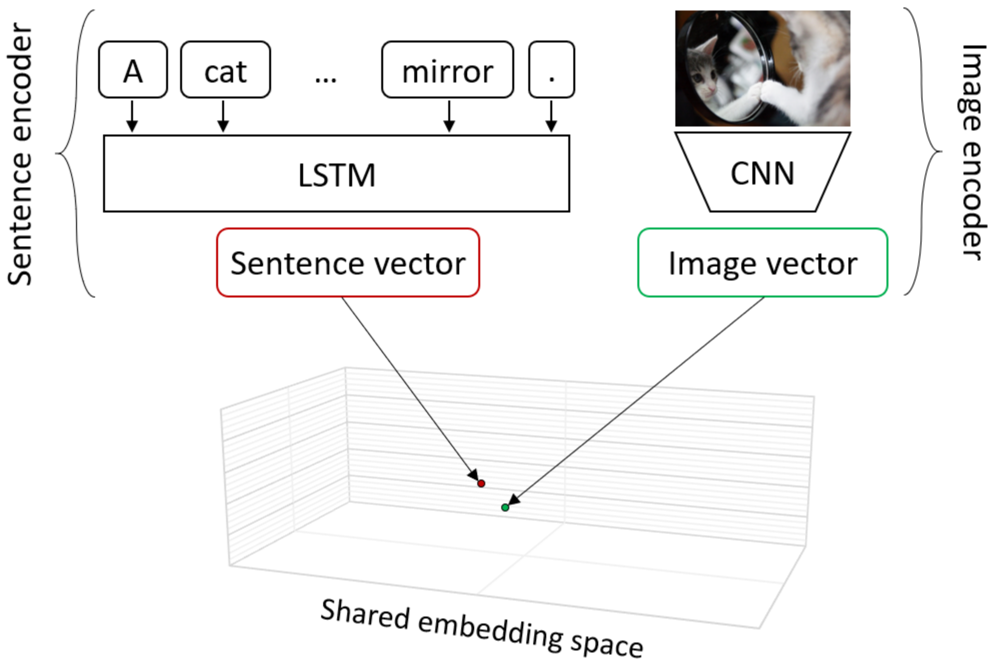


Figure 5.1: Visual-semantic embedding shown in three dimensions. The actual embedding space has a higher number of dimensions than shown here.

regression problem, with the objective to minimize the distance between the generated image vector $\hat{y}$ and the *true* image vector x^{im}. A problem with this approach is the fact that the model is only trained to produce similar image vectors but not to produce distinct image vectors with regard to the other, incorrect candidate images.

Triplet ranking loss

The most commonly used way to train a model to distinctly differ between correct and incorrect candidates is the usage of the triplet ranking loss [30, 57, 58, 77, 94, 126]. One triplet contains a query element (e.g. image caption), a correct candidate (e.g. correct image) and an incorrect candidate (any other image). The objective of the model is to decrease the similarity for any wrong query-candidate pair below a set margin compared to the similarity of the *true* query-candidate pair.

Karpathy and Fei-Fei [57], Karpathy et al. [58], Ren et al. [94] and Wang et al. [126] all use attention mechanisms to determine more or less important sub-regions in the images and to directly connect parts of the image to certain words.

Faghri et al. [30] propose an improvement of the triplet ranking loss, by only selecting the hardest negative sample, meaning the one wrong query-candidate pair that scored the highest similarity, as loss for a mini-batch. Liu et al. [77] go one step further and mine unannotated similar images from the web and use them as hard negative samples to make the model more sensitive to details in the images.

Wang et al. [126] also experimented with a similarity network that trained with a logistic loss to generate numbers close to 1 for well-fitting pairs and outputs close to -1 for bad fits. They show, that this approach *"fails miserably for image-sentence retrieval"* [126].

5.1.3 Evaluation

The performance of ranking models is traditionally reported by the recall of the model for different ranks. r@1, r@5, and r@10 refer to how many queries resulted in the correct candidate being ranked within the top 1, 5 or 10 elements. The median rank is also reported commonly. These metrics are highly dependent on the number of candidate images at prediction time. For the Microsoft Common Objects in Context (MSCOCO) dataset, 1,000 and 5,000 candidate images are normally chosen as candidate size.

Every image in the MSCOCO dataset is annotated with 5 different sentences. Using 1,000 images for prediction results in 5,000 different image-sentence pairs. Different approaches can be used in order to resolve this mismatch between the number of images and descriptions. Karpathy et al. [58] only use one of the five sentences for every image. This reduces the testing dataset to 1,000 image-sentence pairs. Similar to this approach, one can calculate the ranks for all five sentences, separately, and take the average over the single runs. The result should be similar to selecting only one sentence. This method will be called average ranking.

The more common approach in similar studies is to query one image against 5,000 images and to select the highest rank of the 5 *true* captions. When retrieving images, the average ranks from 5,000 sentence queries are used. Faghri et al. [30] repeat this process 5 times of 1000 images at a time, and calculate the average over all runs, thus covering the whole test dataset of 5,000 images. If not specified, the reported ranks have been created with this method.

5.2 VSE network with triplet ranking loss

An image $x^{im} \in \mathbb{R}^{h \times w \times c}$, where h, w, c are the height, width and color dimension, is transformed into a feature representation $\phi(x^{im}, \theta_\phi) \in \mathbb{R}^{d_\phi}$. This can be done by extracting the last fully connected layer of a pretrained CNN (e.g. Residual Neural Network 50 (ResNet50) [46]) before the categorization layer. Let ϕ be the image encoding model, with model parameters θ_ϕ.

Corresponding, a sentence $x^{se} \in \mathbb{R}^{t \times d}$, where t is the number of words in a sentence and d is the dimensionality of every dense word embedding, is transformed into $\psi(x^{se}, \theta_\psi) \in \mathbb{R}^{d_\psi}$. An approach for this is to embed the sentence and to feed it through a Recurrent Neural Network (RNN) (normally Long-Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU)). Let ψ be the sentence encoder model with model parameters θ_ψ.

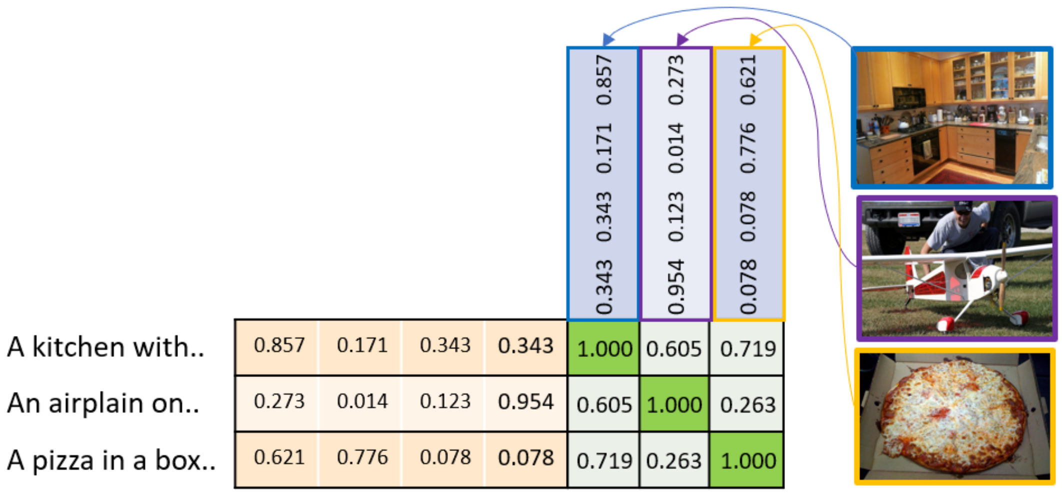


Figure 5.2: By matrix multiplication of the embedded visual-semantic vectors, the similarity between a batch of samples can be calculated in a single mini-batch. The diagonal of the resulting similarity matrix describes the similarities between the correct image-sentence pairs, while all other fields describe similarities between wrong pairs. In this perfect case, the corresponding image and sentence vectors are the identical transposes of one another. Multiplying two L2-normalized vectors results in the Cosine Similarity [113].

$$\mathrm{f}_{im}(x^{im}, W_{im}, \theta_\phi) = \left\| W_{im}^T \phi(x^{im}, \theta_\phi) \right\|_2 \tag{5.1}$$

$$\mathrm{f}_{se}(x^{se}, W_{se}, \theta_\psi) = \left\| W_{se}^T \psi(x^{se}, \theta_\psi) \right\|_2 \tag{5.2}$$

Both feature representations are then mapped into the shared embedding space, with size e, by linear projection with weight matrices $W_{im} \in \mathbb{R}^{d_\phi \times e}$ and $W_{se} \in \mathbb{R}^{d_\psi \times e}$. The resulting projections are normalized with the L2-Norm to lie on the unit hypersphere.

$$\mathrm{s}(im, se) = \mathrm{f}_{im}(x^{im}, W_{im}, \theta_\phi) \cdot \mathrm{f}_{se}(x^{se}, W_{se}, \theta_\psi) \tag{5.3}$$

The similarity between an image-sentence pair is defined as the inner product between the two normalized vectors, resulting in the Cosine Similarity [113].

$$\mathcal{L}(\theta, B_{im}, B_{se}) = \frac{1}{N} \sum_{n=1}^{N} \mathrm{L}(im = B_{im_n}, se = B_{se_i}, B_{im} \setminus im, B_{se} \setminus se) \tag{5.4}$$

For a batch of images, $B_{im} = \{x^{im}\}_{n=1}^{N}$, and corresponding sentences, $B_{se} = \{x^{se}\}_{n=1}^{N}$, the batch loss is calculated by comparing every image against every sentence and vice versa. At every loop, one image-sentence pair is seen as *true* pair, marked as (im, se).

The similarity of this pair is compared to the similarities between the image and all other sentences or the sentence and all other images respectively. A batch of sentences, without the correct sentence, is denoted as $B_{se} \setminus se$ and a batch of images without the correct image as $B_{im} \setminus im$.

In practice, this can efficiently be done by matrix multiplication between all image-vectors and all sentence-vectors, as shown in Figure 5.1.3. The diagonal of this similarity matrix can be subtracted row-wise and column-wise to create the losses between all possible pairs.

The optimized parameters are defined by $\theta = \theta_\psi, W_{im}, W_{se}$, when both image and sentence projection are trained, or by $\theta = \theta_\psi, W_{se}$, when only the sentence projection is trained.

$$\mathrm{L}_{\mathrm{SH}}(im, se, \hat{im}, \hat{se}) = \sum_{\hat{se}} [\alpha - \mathrm{s}(im, se) + \mathrm{s}(im, \hat{se})]_+ + \sum_{\hat{im}} [\alpha - \mathrm{s}(im, se) + \mathrm{s}(\hat{im}, se)]_+ \tag{5.5}$$

L_{SH} is defined as the sum of hinges and describes the classic triplet ranking loss. Let α be the margin that the similarity of all wrong image-sentence pairs should be smaller than the similarity of the correct image-sentence pair. $\mathrm{s}(im, se)$ describes the similarity of the correct image-sentence pair while $\mathrm{s}(im, \hat{se})$ describes the similarity between an incorrect image-sentence pair. $[x]_+ = \max(0, x)$, in order to stop the loss from becoming negative. The second term is symmetrical to the first term. In the first term, an image is fixed and the similarity with different candidate sentences are calculated and returned. In the second term, a sentence is fixed and all other images are looped over to calculate the similarities.

$$\mathrm{L}_{\mathrm{MH}}(im, se, \hat{im}, \hat{se}) = \max_{\hat{se}} [\alpha - \mathrm{s}(im, se) + \mathrm{s}(im, \hat{se})]_+ + \max_{\hat{im}} [\alpha - \mathrm{s}(im, se) + \mathrm{s}(\hat{im}, se)]_+ \tag{5.6}$$

L_{MH} is defined as max of hinges and refers to selecting the negative sample with the highest loss in every mini-batch. The only difference between L_{MH} and L_{SH} is the selection of the biggest error, $\max_{\hat{se}}[\alpha - \mathrm{s}(im, se) + \mathrm{s}(im, \hat{se})]_+$, instead of the summation of errors, $\sum_{\hat{se}}[\alpha - \mathrm{s}(im, se) + \mathrm{s}(im, \hat{se})]_+$.

5.3 Experimental setup

This section explains in detail how the NLIS experiments were executed and evaluated with the above-described VSE network and triplet ranking loss.

Data preprocessing

Similar to Chapter 4, all input sentences are either padded or truncated to contain 16 tokens, in order to guarantee same sequence lengths. No *start- or end-symbol* is required for this task. Sentences are tokenized with the function provided by the Natural Language Tool Kit (NLTK)[1]. In order to reduce the vocabulary size, words are lowercased, and words with a lower frequency than 5, 10 or 16, depending on the experiment setting, are replaced with an *unknown word* token. All words are encoded in one-hot vectors before feeding them into the VSE network.

All images are transformed into image vectors by resizing them to $224 \times 224 \times 3$ pixels (Section 5.2), feeding them through the pretrained ResNet50 [46] and average-pooling the last layer before the final classification layer. Images can either be preprocessed and the image vectors are stored on disk, or they can be calculated on the fly. The first method drastically decreases run time for multiple experimental setting.

Experiments with augmented images and the use of random image crops were conducted based on the setup proposed by Faghri et al. [30]. Creating various versions of every image and feeding them through the CNN increases the run time of the algorithms immensely.

Since first experiments conducted with image augmentation and random cropping showed no increase in performance, this approach was no longer followed, due to computational constraints.

Implementation

The first layer of the sentence encoder branch embeds every word into a dense word embedding. Pretrained embedding weights are taken from a fasttext [86] model, pretrained on Wikipedia dumps. The fasttext model contains 1 million words and their respective embeddings. The same model is also used to reduce the size of unknown words when encoding them as one-hot vectors. If a word would be marked as unknown, due to its low frequency in the corpus, but the fasttext model contains this word, the 10 most similar words, calculated by the fasttext model, are considered to replace it. If any of these words is in the vocabulary of the model, it is chosen instead.

After the embedding layer, every 300-dimensional word representation is processed by 1,024 LSTM cells. The final output, after all 16 words were fed through the LSTM, is projected by a final dense layer onto a 1,024-dimensional space.

Image vectors, returned by the CNN, are also projected into the same space by a single dense layer.

The dot products between all normalized sentence embeddings and all normalized transposed image embeddings are calculated per batch, resulting in a similarity matrix with size $n \times n$, where $n = batch\,size$. Figure 5.1.3 exemplifies this for a batch size of

[1] `https://www.nltk.org/`

three. The loss is calculated using either L_{SH} or L_{MH} and returned.

Experiments

MSCOCO 2017 split was used for all experiments. All models were trained on 118,287 images with 5 annotations each and evaluated on a held-out set of 5,000 or 1,000 images. The evaluation with 1,000 images results in 5,000 image-sentence pairs since every image is annotated with 5 different sentences.

Reported results were observed after 20 training epochs. Training was started with 5 epochs of L_{SH} to generate a stable loss in the beginning. L_{MH} was used after that for the following 15 training epochs. When starting directly with L_{MH}, the loss would often converge against $2 * \alpha$, which can be interpreted as all images and sentences being mapped at exactly the same position. For the first 10 episodes a learning rate of 0.0002 was chosen, before decreasing it to 0.00002. Hyperparameter settings and implementation are strongly inspired by Faghri et al. [30].

5.4 Results

Similar to the last chapter, qualitative and quantitative results are reported in this section and discussed in the next section.

Figure 5.4 shows a few example search queries from the validation set and the best ranked image by model 7 from Table 5.1. The images with the red markings are the *true* images, belonging to the query. One can see that the correct image is often within the best 5 ranks and two times on the first rank. If the image is not within the best 5 ranks, the image content still fits the description.

The results for different experimental settings are reported qualitatively in Table 5.1. The parameter, required Word Frequency (WF), and whether the Word Embedding (WE) layer and the Image Projection (IP) layer are trainable, were selected as independent variables. WF describes how often a word has to occur in all training sentences to not be masked with *unknown*. WB shows if the word embedding layer was finetuned during the training process. IP signals whether the image projection layer W_{im} is trained. A number in the IP column signals how many episodes W_{im} was trained before its weights were frozen. All results are reported for 1000 images and 1000 respective descriptions. This does not influence the image retrieval ranks strongly, but it decreases the sentence retrieval ranks compared to similar literature. Instead of querying for 1 out of 5 correct sentences within 5,000 candidates, we always query 1 sentence out of 1000 candidates. This is not the same way as most of the related studies evaluate their models, but it allows a direct comparison to our Image Captioning-Retrieval (ICR) model in the next chapter. Since our ICR model only generates 1 and not 5 descriptions, the results will be directly comparable with the AVG rankings of our NLIS network.

In order to determine the best model, the sums over all six retrieval ranks were compared. Following this metric, higher WFs, resulting in a smaller vocabulary sizes, showed better results than the baseline WF of 4, inspired by Donahue et al. [27]. EB had little influence on the results. The best results are observed when the IP layer was trained for 4 episodes before freezing its weights.

Table 5.2 shows the results from this work in context with similar studies. The results from Kiros et al. [62] are taken from Faghri et al. [30], because Kiros et al. [62] do not report results for the MSCOCO dataset. Our model could not replicate the state-of-the-art results reported by Faghri et al. [30]. The results are similar in performance to Karpathy and Fei-Fei [57] and Kiros et al. [62].

Table 5.1: Image ranking results for 1,000 validation images from MSCOCO 2017 validation split. For every image, one out of five captions was selected for the ranking evaluation. Results for different experimental settings are given, sorted by the sum of ranks. Our best results are in **bold**. *#=Row number, WF=Word frequency threshold, EB=Embedding layer trainable, IP=Image projection layer trainable*

				Sentence Retrieval			Image Retrieval			
#	WF	EB	IP	r@1	r@5	r@10	r@1	r@5	r@10	SUM
1	5	T	T	31.32	65.90	79.92	31.28	65.52	79.22	353.16
2	5	F	F	34.74	67.58	79.60	31.38	63.94	77.28	354.52
3	5	F	T	31.50	67.10	80.52	31.46	66.74	80.02	357.34
4	5	T	4	31.74	67.38	80.94	31.78	66.32	80.18	358.34
5	5	F	4	31.74	67.84	80.38	31.18	67.14	80.78	359.06
6	16	F	4	**32.48**	67.44	**81.96**	31.70	66.80	**81.08**	361.46
7	10	T	4	32.36	**68.54**	81.70	**33.02**	**67.20**	80.56	**363.38**

Table 5.2: Image ranking results for 1,000 images and 5,000 respective descriptions from MSCOCO 2017 split for different NLIS studies. Results are sorted by r@1 sentence retrieval performance. Our best results are in **bold**.

	Sentence Retrieval				Image Retrieval			
Origin	r@1	r@5	r@10	Med r	r@1	r@5	r@10	Med r
VSA [57]	38.4	69.9	80.5		27.4	60.2	74.8	3.0
Our Model	**39.9**	**69.8**	**80.1**	**2.0**	**32.0**	**66.3**	**80.8**	**3.0**
UVS [62]	43.4	75.7	85.8	2.0	33.0	67.2	80.6	3.0
Order [121]	46.7		88.9	2.0	37.9		85.9	2.0
2WayNet [28]	55.8	75.2			39.7	63.3		
sm-LSTM [52]	53.2	83.1	91.5	1.0	40.7	75.8	87.4	2.0
EmbNet [126]	54.0	84.0	91.2		43.3	76.8	87.6	
VSE++ [30]	64.6	90.0	95.7	1.0	52.0	84.3	92.0	1.0

5.5 Discussion

Similar to Chapter 4, results are discussed first from a quantitative and then from a qualitative perspective, in this section.

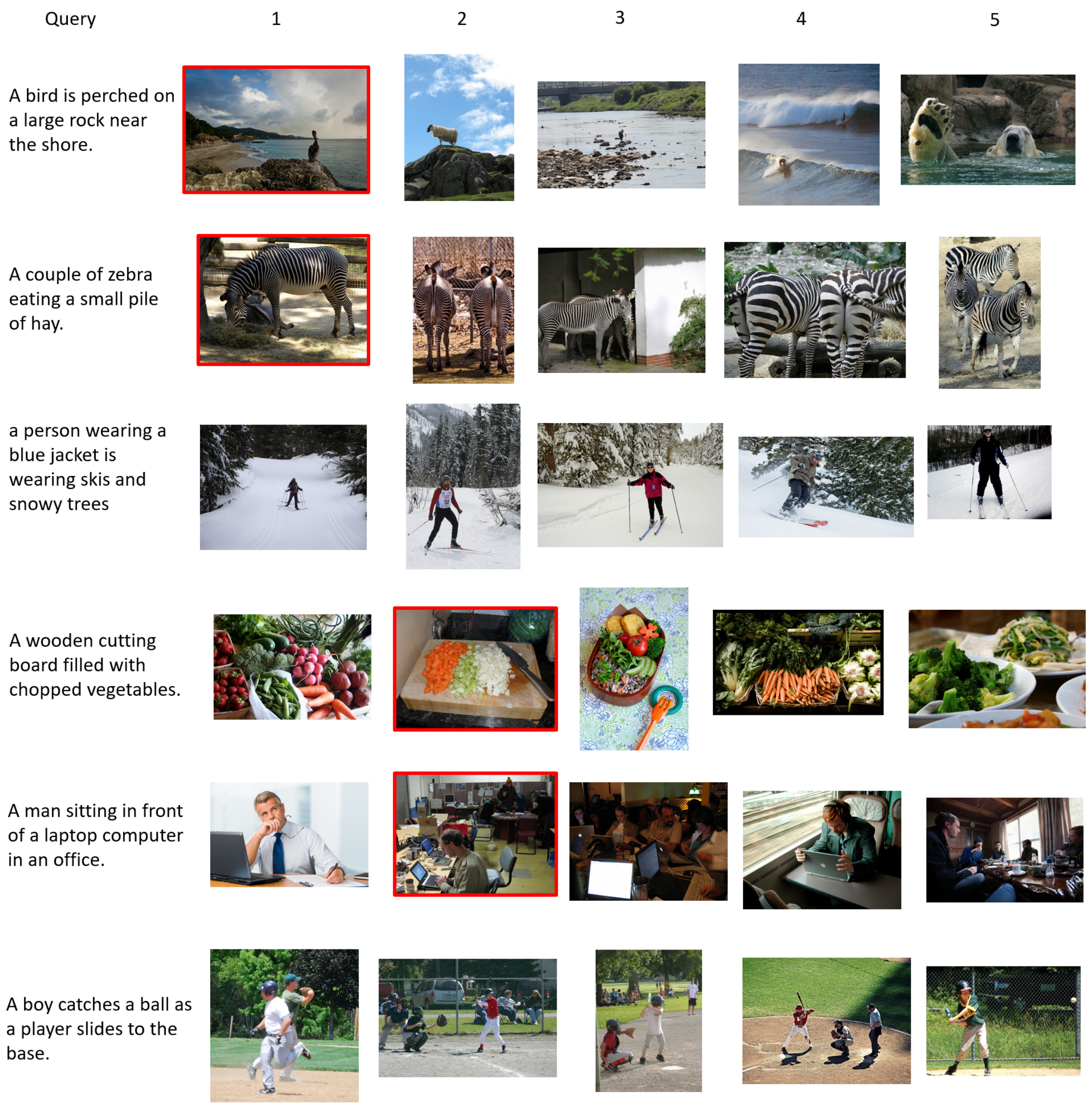

Query
1
2
3
4
5
A bird is perched on a large rock near the shore.
A couple of zebra eating a small pile of hay.
a person wearing a blue jacket is wearing skis and snowy trees
A wooden cutting board filled with chopped vegetables.
A man sitting in front of a laptop computer in an office.
A boy catches a ball as a player slides to the base.

Figure 5.3: Randomly selected queries from the validation dataset and the five most similar images predicted by the image ranking system. The images marked red are the *true* images.

Table 5.3: Direct comparison between our implementation with the implementation from Faghri et al. [30]. Both models use the VGG19 as image encoder (CNN) and use a single crop of the image (1C). The results from Faghri et al. [30] are between 3-5% better than ours.

			Sentence Retrieval				Image Retrieval			
Model	Img	CNN	r@1	r@5	r@10	Med r	r@1	r@5	r@10	Med r
VSE++ [30]	1C	VGG19	43.6	74.8	84.6	2	33.7	68.8	81	3
Our VSE++	1C	VGG19	35.3	69.1	80.1	3	28.7	63.2	78.4	3

Quantitative results

Experiments with different hyperparameters show an optimal ranking performance with a vocabulary size of 7456. Most experiments were executed with a WF of 5, following Donahue et al. [27], resulting in a vocabulary size of 10,331. A higher minimum WF of 10 occurrences yields a vocabulary of 7,456 words and increases the r@1 image retrieval performance of the model by almost 2%. Decreasing the vocabulary size further to 5950 (WF=16) shows slightly worse results than a WF of 10. This shows that, given the complexity of the problem and the amount of data, a lower word frequency threshold does not automatically increase the performance. A smaller vocabulary decreases the complexity of the model and allows a better processing of the content that is not marked as unknown. Additionally, creating a sufficient mapping from words to image sections relies on enough occurrences of both. Only receiving a word 5-9 times might be too little to allow the network to create a correct mapping.

Table 5.1 further shows that training the IP layer for 4 episodes in the beginning and then freezing it, yields optimal results. This is likely due to the fact that training the IP layer for too long leads to overfitting, training it not at all, leads to underfitting. Once the image projections are frozen, after training them for a few episodes, the model can only adapt the position of the sentence representations. This way the model has fewer degrees of freedom and has to generalize better over the training data.

Training the EB layer did not affect the outcome of the model greatly, as rows 4 and 5 show. The pretrained embeddings are already trained on huge amounts of data. It seems like finetuning does not really improve the quality of the embeddings any further.

Table 5.2 places our results in context with similar studies. Sadly, the state-of-the-art results form Faghri et al. [30] could not be replicated, even though, we followed their implementation. We could build a strong baseline model, with similar results to Kiros et al. [62] and Karpathy and Fei-Fei [57].

Kiros et al. [62] use VSE with a triplet loss to embed images and sentences in a shared space. The improvements from Faghri et al. [30] compared to Kiros et al. [62] are *"a novel loss, the use of augmented data, and fine-tuning"*[30] of their CNN network. They additionally use ResNet152 as image encoder.

Replicating their improved loss function and augmenting the dataset in a similar way, our results are far from the performance reported by Faghri et al. [30]. In Table 5.3 we tried to exactly replicate the experimental settings. Both models use the same pretrained CNN and do not finetune it. Faghri et al. [30] only use the training data from the 2014 split, while we use the larger training-dataset from the MSCOCO 2017 split. Both models use the same image preprocessing, where only one crop is taken from the image. This comparison of baselines clearly shows that Faghri et al. [30] achieve higher results with very similar experimental settings.

We are not sure, why this lack of performance is observed. The trained model is performing relatively well and even beats the performance of Karpathy and Fei-Fei [57], but the impressive results from Faghri et al. [30] could not be replicated. This is either due to a flaw in our implementation or due to an unattended detail in their implementation.

Our NLIS network, nevertheless, produces an acceptable ranking performance. When querying images, more the 80% of the correct images are within the top-10 ranked images. Around 32-33%, depending on the random selection of the validation images, are ranked at the top-1 position. This performance should be sufficient for the ICR network, where the NLIS system will function as performance measurement for the AID network. The ranking performance is qualitatively evaluated in the next section.

Qualitative results

Almost all images in Figure 5.4, whether they are the *true* images or not, could be described by the query sentence. Considering the image in the first row, the query does not fit images 2, 4 or 5 well. However, the backgrounds of the images are all very similar and all fit the description *"large rock near the shore"* in one way or another.

This could show, that the image encoder was not explicitly trained to detect salient objects. The CNN was trained on the ImageNet [24] dataset to classify images. This means it has not explicitly learned to pay special attention to what we might consider relevant in an image. If our definition of relevance and the objective to correctly classify the image overlap, the system will implicitly learn a form of attention towards certain shapes or colors. In the case, where the background of images with the same category is more informative than the actual entity, the system will focus on the background instead. This could have happened in the first query.

Query 4 has a similar effect. The word *"vegetables"* is dominating the image search and is much more important than *"cutting board"*, the word that would actually help the system to find the correct image.

Both of these examples should remind us, that supervised learning does not equal thinking. Humans quickly assume some sort of general knowledge from an artificial system. This general knowledge, e.g. detecting salient objects or the special meaning of the word *"board"*, when preceded by the word *"cutting"* instead of *"management"* or *"snow"*, cannot be assumed, however.

Query 2 shows, that the word *"zebra"* is very easy to find in images and is probably highly salient both in images and in sentences. There probably is a very high correlation between *"zebra"* and black-and-white stripes in a horse shape.

Queries 3, 5 and 6 show the ambiguity in the dataset and in language in general. Language is an abstraction of our visual input and internal representation. It is never a 1-to-1 mapping but always a many-to-many mapping. In this case, many images are well described by a single image description. This makes it hard to impossible for an artificial system or for a human to get a perfect ranking score.

On the other hand, these queries show, how well the system generalizes and how well the mapping from sentence to image and vice versa is learned. Small details in both the descriptions and the images are often not processed perfectly, but in general, the images fit the queries very well.

Future work

Faghri et al. [30] report that through the usage of hard negatives, image augmentation and finetuning of the image encoder, state-of-the-art ranking results can be achieved. Many studies show, that the selection of the image encoder will determine the performance of the resulting NLIS network. Assuming the state-of-the-art performance of Faghri et al. [30] is replicated, additional image attention might further increase the accuracy of the network. Huang et al. [52] use additional attention mechanisms and can improve their performance. Combining these attention mechanisms with hard negative sampling could increase the performance even more.

Liu et al. [77] use unlabeled but similar images as additional hard negative samples and report an increase in their model performance. Adding this semi-supervised training approach could further increase the ranking performance of a network.

Conclusion

We created a NLIS model that successfully maps similar image-description pairs together in a latent, shared space. The goal performance set by Faghri et al. [30] could not be reached. Instead, our performance is similar to other strong baseline studies. A qualitative analysis additionally shows, the network learned a meaningful correlation between descriptions and images that should allow the further usage of the NLIS model in our ICR network in Chapter 6.

6 Image Captioning-Retrieval

Chapters 4 and 5 show how Automated Image Description (AID) and Natural Language Image Search (NLIS) can be approached on their own. In order to satisfy our research question, they are combined in this chapter. In the Image Captioning-Retrieval (ICR) task, an image is described in natural language, the generated description is compared to a number of candidate images and the original or most similar image is retrieved. This does not only satisfy the requirements for a simple form of communication, but it also allows for an optimization of the AID system by the ranking performance of the NLIS system. In human terms, this would relate to someone reformulating their sentence, until a second person understands the meaning of the sentence.

6.1 Related work

The task of ICR has been approached before. The focus, however, always lies on the generation of sentences resembling human speech and not the information exchange in general. Related research is presented in the following sections.

6.1.1 Autoencoder

Section 5.1.2 mentions the usage of an autoencoder for bi-directional image description and retrieval. It was shown by Salakhutdinov and Hinton [101], how an autoencoder can be used to embed documents into bit-codes. Krizhevsky and Hinton [64] and Yan et al. [133] do the same with images and show that using 32, 64 or 128 bit layers in the hidden state allow a fast and accurate Image Retrieval (IR). Extending this embedding to a one-hot embedding of the size of the vocabulary would allow to embed images into sentences and back into images. In order to create natural language embeddings, the system would need additional feedback from a labeled dataset when generating the hidden representation, the description.

In the previously mentioned work by Chen and Zitnick [17] exactly this is done. Image vectors are inputted and the model is trained to produce image descriptions based on the ground truth sentences with Maximum Likelihood Estimation (MLE). Then the original image vector is reproduced based on this hidden representation. They generate the image vector not only after the sentence is completed but after every generated word and then feed the partially complete image vector back into the language generation layer. By doing this, they are trying to add a visual memory to the sentence generation steps. The already mentioned flaw of this approach is the fact that only correct image-sentence pairs are utilized for the recreation of the image vector. Thus, a lot of training data, namely the wrong image-sentence pairs, are neglected.

6.1.2 Triplet Loss

The in Section 5.2 described triplet-ranking-loss cannot only be used to optimize an image ranking system but also a complete ICR system. Instead of using only human annotations, the descriptions generated by an AID network can be used instead.

The produced probability distributions of single words, generated by the AID system, have to be transformed into a discrete distribution, representing single words. One could feed the NLIS system directly with probability vectors, but that would make the NLIS system unusable for further tasks since it is trained on distributions and not on discrete words. It also defeats the purpose of generating language and not a high dimensional representation of an image instead.

When the softmax, representing a single word probability distribution, is transformed into a discrete hardmax (argmax), the complete ICR network is no longer differentiable. This means, the ranking results from the NLIS network cannot be used to optimize the AID network. Since this is a necessary property of the ICR network, a work-around is needed.

Dai et al. [22], Liang et al. [72], Ren et al. [95] and Liu et al. [77] use Reinforcement Learning (RL) and treat the rank of the correct image as a negative reward. They all use this approach, in order to optimize evaluation measures in an adversarial framework. The NLIS network functions as discriminator network, trying to differentiate between generated and *true* samples. When using RL, there is no need for a continuous, differentiable function, so argmax can be used to greedily pick words. The only thing RL needs, is a reward after each episode. On the downside, RL needs much more training time, and if there is an alternative way, to solve a problem with gradient based loss optimization, it normally outperforms RL.

A different work-around is to transform each word probability distribution into a distribution as close as possible to a discrete distribution while still being differentiable.

Gumbel [40], Jang et al. [54] and Maddison et al. [80] show how to use the gumble softmax trick to transform probability distributions into continuous distributions, closely resembling one-hot vectors. Shetty et al. [107] apply this trick to train their AID model in an adversarial fashion. With the gumble softmax trick, it is possible, to calculate a loss, based on the output of the discriminator and directly backpropagate this loss, to optimize both the AID and the NLIS network.

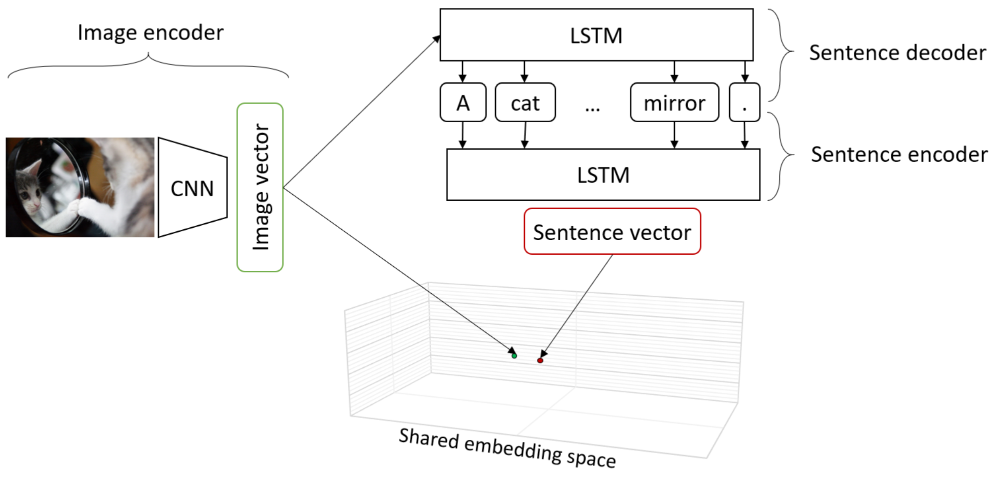


Figure 6.1: The ICR system from a high-level perspective.

6.2 Image Captioning-Retrieval network

The AID system from Chapter 4 is combined with the NLIS system described in Chapter 5 to result in our ICR network. The network can be trained from scratch, or the subsystems can be pretrained as described in their respective chapters. Figure 6.1.2 shows the architecture of our ICR model.

In order to overcome the problem of discrete word representations, the gumble softmax trick [54] is used to transform probability distributions into pseudo-one-hot-representations.

The original Gumbel-Max trick [40] is a simple and efficient way to draw samples from a categorical distribution with class probabilities π. $g \in (0, 1)$ is called the gumble distribution and is calculated from u, drawn from a Uniform distribution between 0 and 1.

$$u \sim \text{Uniform}(0, 1) \tag{6.1}$$

$$g = -\log(-\log(u)) \tag{6.2}$$

$$z = \text{one hot}\left(\underset{i}{\text{argmax}}[g_i + \log(\pi)]\right) \tag{6.3}$$

Since argmax is non-differentiable, the continuous softmax function is used as an approximation. τ is the temperature of the softmax. The smaller τ is, the closer the distribution is to a one-hot encoding. y_i is the k-dimensional resulting word distribution.

$$y_i = \frac{\exp((\log(\pi_i) + g_i)/\tau)}{\sum_{j=1}^{k} \exp((\log(\pi_i) + g_i)/\tau)} \qquad \text{for } i, ..., k \tag{6.4}$$

Figure 6.2 shows, how a distribution is transformed into a softmax, an argmax and a

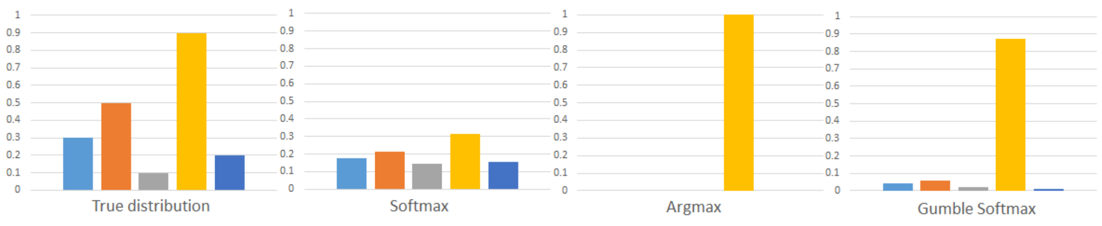


Figure 6.2: The same distribution in different transformations.

gumble softmax distribution.

A second challenge is the sampling of novel sentences. The system needs a complete input sentence x^{se}, to be able to determine the probability for every sub-sentence $x^{se}_{t_1:t_i}$. This means it is not possible to sample word by word on the fly and pass the final sentences to the NLIS system. Consequently, a sentence first has to be predicted for a certain image, with the prediction technique used in Section 4.2. Then this sentence can be used as input sentence x^{se} together with its corresponding image vector, $\phi(x^{im}, \theta_\phi)$. The AID model then basically produces the same sentences again but without interrupting the sampling process of words. When the resulting sentence vector $\hat{y}$ is transformed with the gumble softmax activation function, one word is randomly chosen based on the probabilities and is transformed into a high probability close to one. This means, the word with the highest probability will not be picked every time, as it would be the case when picking greedily. All other words will receive a very low probability, close to 0.

Let $\gamma(\hat{y})$ be the gumble softmax output. Together the original image vector $\phi(x^{im}, \theta_\phi)$ and $\gamma(\hat{y})$ are fed into the NLIS network to output a similarity matrix, containing similarities between every image and every sentence. From this similarity matrix, either the sum of hinges loss or the max of hinges loss (Section 5.2) can be calculated and used to train the whole network or only specific layers of it. The matrix also yields the rank for every image and sentence.

6.3 Experimental setup

The same preprocessing steps from previous chapters were executed. All input sentences have a sentence length of 16 and the WF is set to 9 since this yielded optimal results in Chapter 5.

Microsoft Common Objects in Context (MSCOCO) 2017 split was used for training and validation. In one training epoch, 20,000 images are randomly selected from the training dataset and annotated by the current AID network. As explained in Section 6.2, descriptions for images first have to be created in a separated prediction phase by the AID model. In a second step, these sentences are used as input sentences for the ICR system. Only this way, the AID system can create novel sentences and pass them to the NLIS system without interrupting the computation.

Our AID network has only been pretrained on correct sentences. Starting training with only self-generated sentences right from the start leads to an instant decrease in performance since the model has no time to adjust to the new and different input. To counter this issue, novel self-generated descriptions are slowly added to the ground truth sentences. This means, the ICR system randomly selects one description from a list of descriptions, containing both ground truth sentences and generated sentences. At the beginning of training, all ground truth sentences are added to their respective description list. At every epoch, the newly generated descriptions are added to their respective list. When the length of the list is greater than a defined value (INF=5, 10, 15), a random sentence is dropped from the list. This way, novel sentences are slowly infused into the list of *true* sentences.

Depending on the experiment setting, the embedding layers (EM), the image projection layers (IP) or the complete NLIS network (IR) were frozen and excluded from training.

After the training phase, all 5,000 validation images are annotated by the AID system and 1,000 of the resulting annotations, together with their respective images, were processed by the NLIS model to return the ranking performance of the model. Evaluation measures for all 5,000 created descriptions are calculated and stored.

Besides the previously used MLE loss for the AID system that can only be calculated when *true* descriptions for the images are available, a second, different loss was defined. This loss, called Sentence-Entropy loss (ENT) by us, is defined as the entropy over the sum of all word probabilities. This loss is supposed to reduce the repetitions of words in a sentence. If the probability for the same word is high in many words, the sum over these probabilities will have a very low entropy. If all words have different probabilities, the sum is more spread out and has a higher entropy. This loss is not minimized but maximized in order to increase the entropy over all words, and thus the information content of sentences.

For many experiments, true image-sentence pairs (TP) were added to the output from the AID network. If this is the case, 64 image-sentence pairs from the AID system and 64 true image-sentence pairs directly from the dataset are concatenated and then forwarded to the NLIS network in one mini-batch. If this is not the case, 128 image-sentence pairs were produced by the AID system and passed to the NLIS system in one mini-batch. Both methods result in a 128×128 similarity matrix for one mini-batch.

The model was trained for 50-60 epochs with Adam as optimizer. The learning rate is set to 0.0002 for the first 20 epochs and then decreased to 0.00002 for the rest of the training process.

6.4 Results

The results from various experimental settings are presented in this chapter. The experiments can be separated into two categories, one where the image ranking is the only or

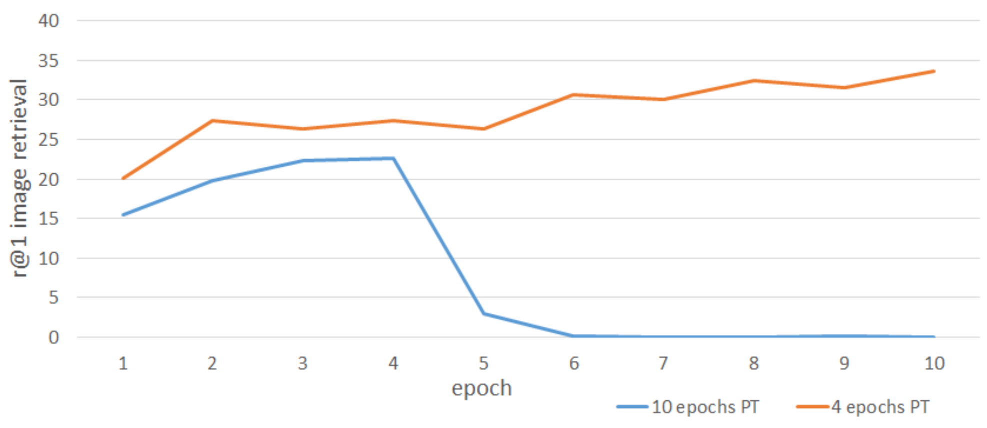


Figure 6.3: Same experimental design with different pretrained AID models. The one model is pretrained for 4 episodes, the other one for 10 episodes. The second model, pretrained for 10 epochs, performs very badly.

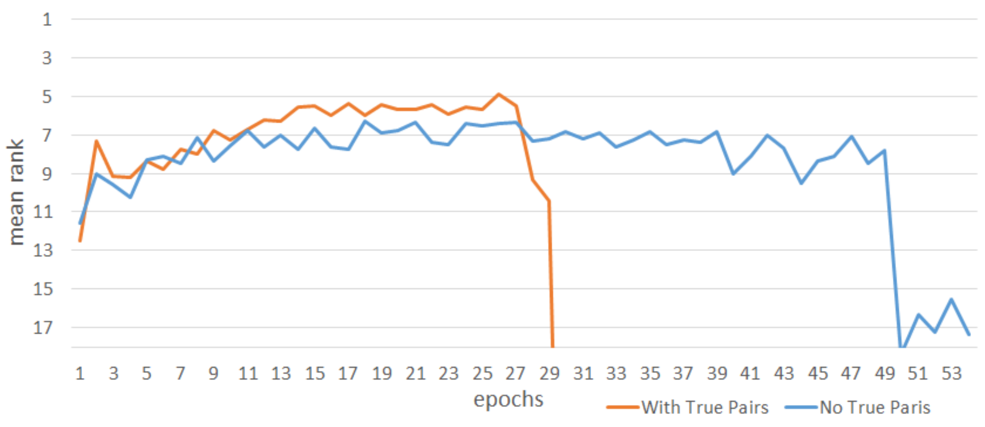


Figure 6.4: Mean rank (low is good) is plotted per epoch. When all layers are trained to increase ranking, performance plummets at some point.

the most dominant objective, and one where an increase of the n-gram based evaluation measures of the generated sentences is considered the main goal.

Optimizing image ranks

We experimented with two different pretrained AID networks. One was trained for 10 epochs and one for 4 epochs. Figure 6.4 shows, that the model pretrained for 10 epochs performs very badly and results in a r@1 ranking performances close to 0 after a few episodes. The model pretrained for 4 epochs performed much better and was selected for all further experiments.

The first set of experiments was designed to solely optimize information exchange from the AID system to the NLIS network. This means all layers were trainable and the ranking loss from the NLIS system was used to optimize the complete ICR network.

Figure 6.4 shows that this method resulted in an increase in ranking performance but

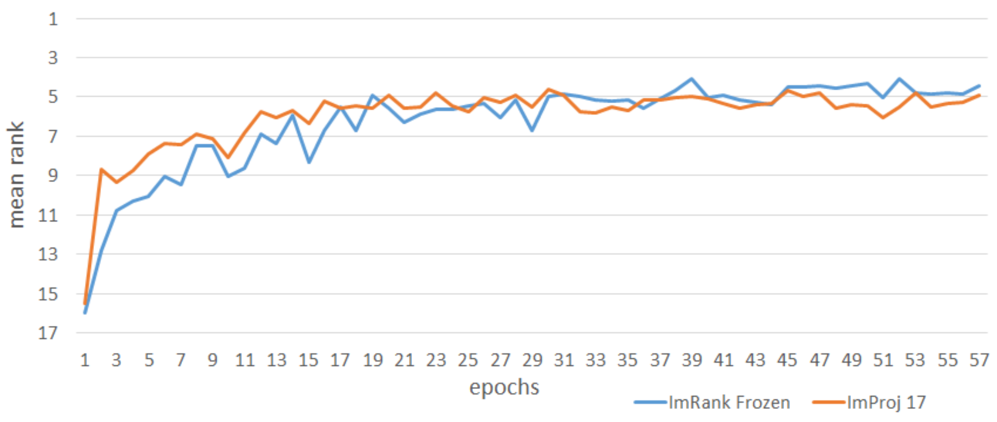


Figure 6.5: Best training runs. Blue line represents row 5 and orange line represents row 3 from Table 6.1

Table 6.1: Ranking retrieval results for different experimental settings. *TP=True Pairs, IR=Image Ranking network trainable, IP=Image Projection layer trainable, INF=Infusion list size, ENT=Entropy loss used, ImSum=Sum over all image scores, C=CIDEr*

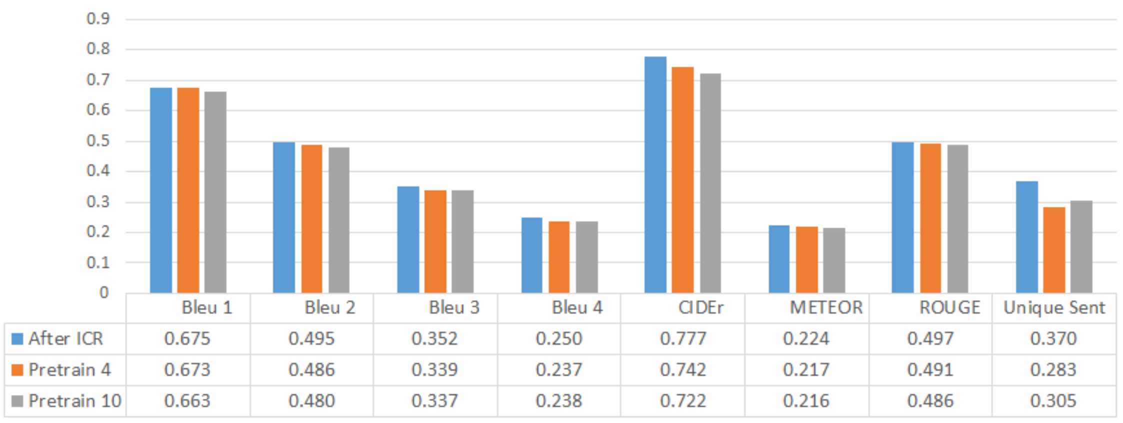
					Sentence Retrieval			Image Retrieval				
TP	IR	IP	INF	ENT	r@1	r@5	r@10	r@1	r@5	r@10	ImSum	C
F	T	F	10	T	34.7	72.1	86.9	33.2	69.7	83.7	186.6	0.061
F	F	F	10	T	40.8	77.8	89.9	38.8	76.3	88.9	204.0	0.101
T	T	17	10	T	44.4	79.9	90.5	40.8	79.1	89.7	209.6	0.049
T	F	T	10	T	47.6	84.2	93.6	43.1	81.8	92.5	217.4	0.094
T	F	T	10	F	44.8	82.5	93.1	43.9	82.9	90.9	217.7	0.097
T	F	T	15	T	46.2	86.4	93.6	46.0	83.0	93.0	222.0	0.083
Pretraining Baseline					32.4	68.5	81.7	33.0	67.2	80.6	180.8	0.742

an unstable training process. After a certain epoch, the performance drops drastically. The self-generated image descriptions also change completely at this point.

Freezing the image projection layers from both networks after a certain training epoch stabilizes the training process. Completely freezing the NLIS network yields the best ranking results (Figure 6.4). Table 6.1 shows different experiment settings and the resulting ranking scores as well as the Consensus-based Image Description Evaluation (CIDEr) score for the evaluation sentences.

The table shows that the usage of true image pairs (TP) always increases the ranking performance of the network. The best experimental setting could improve image r@1 results by 10,9% compared to our baseline from training the NLIS with the *true* descriptions. CIDEr scores for all experiments decrease from 0.72 to around 0.01.

Figure 6.5 shows a selection of images and different annotations. The first annotation, PT, is the annotation generated by the AID system, after pretraining. GT shows three of the five *ground truth* captions, only the AID system was trained on. The last sentence (ICR) is the generated description from our best performing model, from Table 6.1.



	Bleu 1	Bleu 2	Bleu 3	Bleu 4	CIDEr	METEOR	ROUGE	Unique Sent
■ After ICR	0.675	0.495	0.352	0.250	0.777	0.224	0.497	0.370
■ Pretrain 4	0.673	0.486	0.339	0.237	0.742	0.217	0.491	0.283
■ Pretrain 10	0.663	0.480	0.337	0.238	0.722	0.216	0.486	0.305

Figure 6.6: Using MLE and ranking loss together in order to improve *n*-gram evaluation measures and diversity of generated descriptions.

Optimizing *n*-gram evaluation measures

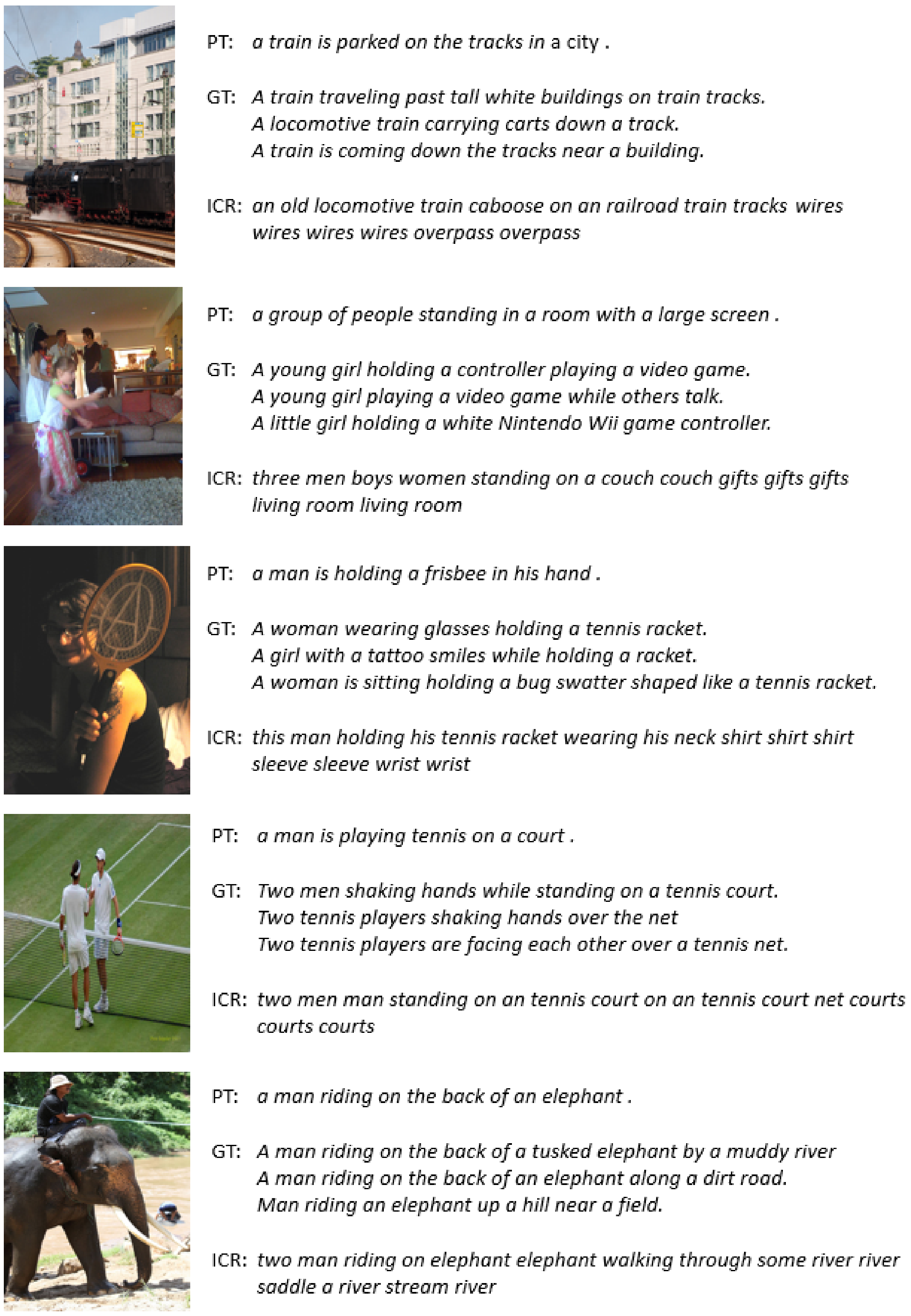
The models, optimized only with the loss from our NLIS network, show a clear improvement in ranking scores but also a strong decrease in sentence evaluation measures. In order to improve image ranking as well as sentence evaluation measures, a set of experiments was designed. The results shown in Figure 6.4 are taken from the most successful model in this regard. In order to achieve slightly better evaluation measures, the NLIS network was frozen from the beginning. There were no new image descriptions infused, but only the *true* descriptions were used. The model was optimized based on MLE loss and ranking loss combined.

The resulting model achieves slightly higher languages scores (CIDEr +0.035) and generates more unique sentences. The percentage of generated unique sentences increased by more than 7 points. The number of unique sentences is calculated by dividing the total number of generated sentences by the number of uniquely generated sentences. This can be a strong indicator of how diverse the generated sentences are. Models trained only with MLE often suffer from a high number of repeating sentences [77, 107].

The image retrieval performance of this model only reaches a r@1 of 5.0 for 1,000 candidate images. The evaluation measures are calculated with 4,992 images from the validation dataset of the MSCOCO 2017 split.

6.5 Discussion

The results are analyzed and discussed in this section. In the end, possible future work and a conclusion are presented.



PT: *a train is parked on the tracks in a city .*

GT: *A train traveling past tall white buildings on train tracks.*
A locomotive train carrying carts down a track.
A train is coming down the tracks near a building.

ICR: *an old locomotive train caboose on an railroad train tracks wires wires wires wires overpass overpass*

PT: *a group of people standing in a room with a large screen .*

GT: *A young girl holding a controller playing a video game.*
A young girl playing a video game while others talk.
A little girl holding a white Nintendo Wii game controller.

ICR: *three men boys women standing on a couch couch gifts gifts gifts living room living room*

PT: *a man is holding a frisbee in his hand .*

GT: *A woman wearing glasses holding a tennis racket.*
A girl with a tattoo smiles while holding a racket.
A woman is sitting holding a bug swatter shaped like a tennis racket.

ICR: *this man holding his tennis racket wearing his neck shirt shirt shirt sleeve sleeve wrist wrist*

PT: *a man is playing tennis on a court .*

GT: *Two men shaking hands while standing on a tennis court.*
Two tennis players shaking hands over the net
Two tennis players are facing each other over a tennis net.

ICR: *two men man standing on an tennis court on an tennis court net courts courts courts*

PT: *a man riding on the back of an elephant .*

GT: *A man riding on the back of a tusked elephant by a muddy river*
A man riding on the back of an elephant along a dirt road.
Man riding an elephant up a hill near a field.

ICR: *two man riding on elephant elephant walking through some river river saddle a river stream river*

Figure 6.7: Next to every image, the description generated by the pretrained AID model (PT), three of the ground truth descriptions (GT) and the descriptions, generated after training the AID model together with the NLIS model (ICR).

Optimizing image ranks

Figure 6.5 shows our main findings. The sentences created after the pretraining are grammatically correct and describe the image content more or less accurately. Generated descriptions show less grammatical structure after the AID system was trained to maximize the ranking performance, but the content of the sentence describes the image in much more detail and correctness. The increase in information can not only be observed in the ranking scores, shown in Table 6.1, but also in the generated image descriptions in Figure 6.5.

The generated sentences after ICR training often contain repeating words, and they do not contain the *end-symbol* anymore. Both of these effects are likely due to the pretraining of the system. During the pretraining phase, only correct sentences were used as input for the model. In the ICR training phase, new sentences are generated and used for training. Additionally, since the gumble softmax trick is a statistical sampling method, the word with the highest probability is not always picked, as it has been before with greedy picking. This means the system encounters new situations that it has to deal with. Since it was trained with teacher-forcing, it has developed little robustness against these novel situations. The repetition of the same word seems to be an indicator of this. When plotting the average number of different words per sentence (Figure 6.5), one can see that during training the repetition of words becomes less frequent. The average occurrences is calculated by dividing the number of words by the number of different words. A sentence with every word occurring only a single time would thus get the occurrence score of 1. Figure 6.5 shows the word occurrences for the two models from row 4 and 5 in Table 6.1. While in both runs, word repetitions are decreasing during training, the Entropy-loss, described earlier (orange graph), can significantly decrease the number of repetitions. This does not greatly influence the overall ranking scores (Table 6.1). Repetitions are unusual for human speech, but the encoding of meaning into repetitions by an artificial system is quite likely. E.g. the more dominant an object is, the more often it is repeated in the sentence.

Even though, the grammatical correctness of the sentences decreases with the ICR training process, the grounding of used words stays intact. This means, the network still describes parts of the image with the words, learned in the pretraining phase. This is highly relevant for a system trying to mimic human language. It means that the system does not forget the correlations from image parts to words, even when trained on a different task. This is not only the case when the NLIS is frozen. When this part of the network is trainable, however, the training process is less stable. In the two experiment runs, plotted in Figure 6.4, this is clearly visible. In the episode, where the ranking performance drops, the system suddenly creates completely different words and sentences for images. This indicates it is no longer grounded to the human meaning of words.

When training with a different pretrained AID model, that was trained for 10 and not 4 episodes, grounding was also lost quickly (Figure 6.5). It seems that an AID system

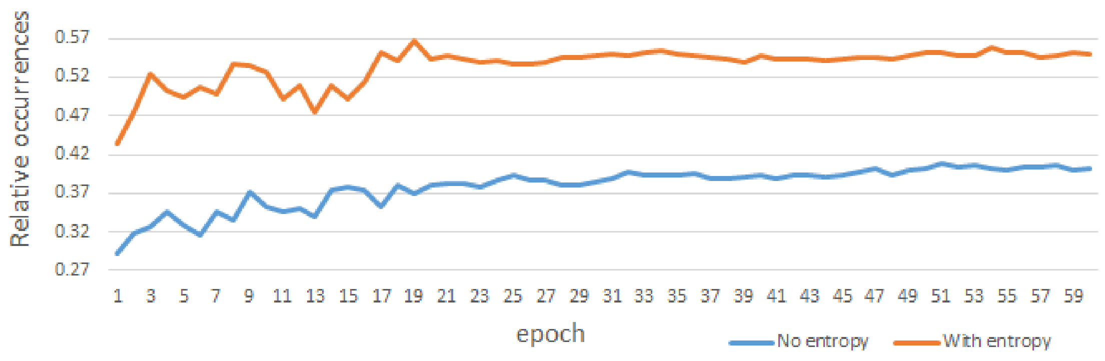


Figure 6.8: Average occurrences of different words in sentences. A word occurrence of 1 means no word appears more than once in a sentence. The smaller the number the more repetitions are in a sentence.

trained for around 4 episodes presents an optimal starting point for our ICR network. These results are counter-intuitive since it was assumed that more pretraining would strengthen the grounding and improve further training. A possible explanation for this phenomenon could be that the model trained for 10 episodes was trained for too long on correct sequences and could not transit to self-generated sentences anymore.

Regarding the first image in Figure 6.5, one can see, that the description after the ICR training includes *"an old locomotive"* instead of only *"a train"*. The description also contains *"wires overpass"*, describing the electrical wires over the train, even though this information was not present in any of the 5 human annotated sentences. This shows that the model is no longer explicitly trained on the *true* sentences but has a much more implicit objective. In order to optimize the ranking performance, additional, highly distinct image information was transformed into words. The fifth image in Figure 6.5 shows similar increases in content and detail description. The information that the elephant is *"walking through some river"* is crucial to distinctly rank this image over other elephant images.

In the fourth picture, the new description is less general. The pretrained system is producing a generic sentence, more or less fitting to any tennis scene. The description, generated after the ICR training is more accurate in its context. The same is true for image number 2 and 3. In general the image content is described in more detail and in more accuracy. The sentences are less grammatical than before, however.

These findings are satisfying and show that our objective trains our system to transfer information rather than to copy sentences. The fact that the created sentences are still grounded to human meaning and still show some human-like structure show that our language system, once pretrained, keeps its relations between objects and words intact. Our main goal of increasing the amount of exchanged information is clearly reached.

Optimizing *n*-gram evaluation measures

Similar studies [22, 77, 107] normally define their objective as the optimization of description evaluation measures and the increase of diversity and naturalness of their generated sentences. Figure 6.4 shows the changes in evaluation measures and in the number of unique, generated sentences of our best approach with this objective. Even though evaluation measures increase only slightly, there is a 7% increase in the number of generated unique sentences. This implies that the additional ranking loss, combined with MLE loss can improve both evaluation measures and diversity among the descriptions. The ranking performance suffers heavily in this experimental setting. This is due to the necessary freezing of the ranking weights and the MLE loss. The network can neither adjust the generated sentences well enough nor can it adjust the ranking parameters. All other experimental settings resulted in a strong decrease of all *n*-gram evaluation measures.

Future work

Our model can further be trained with unlabeled data. Since the loss is derived directly from the ranking performance, descriptions are not necessary for training after the transition phase form *true* to self-generated descriptions. A training with more novel and different images will likely increase the ranking performance and thus the sentence quality further. Since plain, unlabeled images are available in large magnitudes on the web, the amount of training data is immense. It would be very interesting to observe the result after training our model with a large unlabeled dataset.

In order to force the system to produce better grammar, the generated sentence could be fed into an *n*-gram or Long-Short-Term Memory (LSTM) language model. This model would be trained on a large corpus and would penalize unknown *n*-gram or sentence structures.

Similar, this could be done in an adversarial fashion. The generated and ill-formulated sentences would be used as wrong samples and human sentences as correct ones. The discriminator is trained to distinguish wrong sentences, whereas the generator, our AID system in this case, is trained to fool the discriminator. This way, the AID system has to replicate the general form and structure of human sentences in order to fool the discriminator. This should highly increase the grammatical correctness of generated descriptions.

A different approach to counter the problem of bad grammar and multiple word repetitions in sentences can be found in RL. Liu et al. [77] also optimize their AID network with a NLIS system. Next to this, they optimize the CIDEr score of the generated sentences with RL. This way, the model is not as constrained as to when training with MLE, but language typical structures, like ending sentences with ".", stay intact. RL also allows to simply add the result of a grammar checker or language checker of the generated sentences to the reward. This way, grammatical errors are punished and avoided. RL offers simple methods to constrain the outcome of the model in a favorable way.

Research in these directions is highly promising and should be pursued further.

Conclusion

We clearly show how training an AID network with a more implicit objective like the ranking results from our NLIS network can improve the amount of information captured in the generated sentences. The newly generated sentences are not grammatically perfect but showed a strong trend towards normal human-generated sentences. More importantly, after training our ICR model, generated descriptions capture more distinct details of images and describe more aspects of the images. The ranking performance was increased by a large margin. The language grounding of different words and concepts stays intact during training. Pretraining the AID model for a few episodes on the *true* descriptions is enough to build up a strong grounding between entities on images and words.

When strictly constraining the model to create valid, grammatically correct sentences, we could observe a slight improvement of sentence quality and diversity, when combining MLE loss and image ranking loss.

7 Final conclusion

In our endeavor to simulate simple conversations about images, we constructed and pre-trained *speaker* and *listener* as two single systems at first. The *speaker* was implemented in form of an Automated Image Description (AID) network, trained with Maximum Likelihood Estimation (MLE). The *listener* was realized in form of a Natural Language Image Search (NLIS) model, trained with a triplet-ranking-loss.

Combining both systems and training them with the objective of optimizing information exchange in the form of a successful Image Retrieval (IR) was realized in Chapter 6. A clear increase in ranking performance can be observed when only training the AID network with the ranking loss provided by our NLIS network. The improved information exchange is not only visible in the ranking performance but also in the generated descriptions themselves. Even though they lose some of their grammatical correctness and language-typical structure, their information content increases. This is clearly visible in the form of more accurate, more detailed and more distinct descriptions.

The created subsystems show strong baseline performances in their respective fields and can further be used for various tasks. The research questions stated in Section 1.3.3 will now be answered.

- Can AID be solved well enough that it can capture the most important features of an image and transform them into natural language?

We were able to create an AID system with similar performance to related work that, like our approach, use MLE as primary loss. The generated descriptions are grammatically correct and capture the most important features of the image accurately in most cases. A grounding between words/phrases and their representations in images could successfully be created.

- Is it possible to train a NLIS system to find the same or similar images with only the generated description as input?

Even though our model does not achieve state-of-the-art performance, it is nevertheless capable of correctly correlating image and sentence contents. We created a well functioning and scalable model to rank images based on query sentences and vice versa. Especially qualitative results show that even if the correct image is not ranked at the top-1 position, most of the highest ranked images satisfy the search query. There is room for improvement when it comes to detailed information in images and descriptions, but for our purpose the NLIS model performs sufficiently.

- Can both the AID and NLIS system be optimized simultaneously by the feedback of how successful the retrieval was?

With our system architecture, it was not possible to optimize both ranking performance and sentence quality at the same time. We could either observe a strong increase in ranking performance or a slight increase in sentence quality. In both cases, the other metric suffered during the training process.

- Are the architecture and implementation proposed in this study capable of solving the Image Captioning-Retrieval (ICR) problem in a way that simple conversations driven by information exchange emerge?

This work was designed to simulate simple conversations driven by information exchange. This was successfully implemented and resulted in an increase of detail and information in the created image descriptions. Our AID system adapted its generation process in such a way that the NLIS network could rank the candidate images with a higher accuracy. The amount of information in the description and thus the conversation could be increased.

While more information was embedded in the descriptions, the generated words were still grounded in the human meaning of the words. The correlation between words and certain entities that was learned in the pretraining phase stayed intact. Generated sentences are still human-readable even though they loose grammatical correctness.

This loss of grammatical structure was expected but still underestimated. We believe that an additional *sentence quality* loss combined with the experimental settings to increase the ranking scores will yield both an increase in ranking performance and a high grammatical quality.

Overall, our proposed architecture and implementation were capable of creating simple conversations about images in human-readable syntax motivated primarily by the objective of information exchange.

This work has strengthened our belief that language generation and comprehension are not processes learnable by the explicit goal of one or the other. Language offers a mapping from a high dimensional to discrete space. It offers the exchange of complex information in an equally complex but agreed-upon system. An explicit modeling of language is likely not possible. More effort should be placed in implicitly modeling, with objectives like information exchange in order to solve tasks, requiring content, that can only be transferred by language. The proposed language game in this work builds one of the most basic games there is: describing and finding an image.

More sophisticated games, like solving riddles, answering questions, walking through a maze or executing commands can all be implemented based on language instructions. These games all have to be designed in a way that succeeding is a direct implication of information exchange. If this approach is used, while language grounding and correct grammar are enforced and guaranteed for, we will have a real chance of solving the problem of language generation and comprehension in a more complete and satisfying way.

Bibliography

[1] Abdul-Kader, S. A. and Woods, J. (2015). Survey on chatbot design techniques in speech conversation systems. *International Journal of Advanced Computer Science and Applications*, 6(7):72–80.

[2] Akaho, S. (2001). A Kernel Method For Canonical Correlation Analysis. In *International Meeting of the Psychometric Society*, Osaka, JP.

[3] Andrew, G., Arora, R., Bilmes, J., and Livescu, K. (2013). Deep canonical correlation analysis. In *International Conference on Machine Learning*, volume 28, pages 1247–1255, Atlanta, US.

[4] Arik, S. Ö., Chrzanowski, M., Coates, A., Diamos, G., Gibiansky, A., Kang, Y., Li, X., Miller, J., Raiman, J., Sengupta, S., and Shoeybi, M. (2017). Deep voice: Real-time neural text-to-speech. *Computing Research Repository*, abs/1702.07825.

[5] Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein Generative Adversarial Networks. In *International Conference on Machine Learning*, volume 70, pages 214–223, Sydny, AU.

[6] Arulkumaran, K., Deisenroth, M. P., Brundage, M., and Bharath, A. A. (2017). Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38.

[7] Barto, A. G., Sutton, R. S., and Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *Transactions on systems, man, and cybernetics*, 5:834–846.

[8] Bengio, S., Vinyals, O., Jaitly, N., and Shazeer, N. (2015). Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. In *Conference on Neural Information Processing Systems*, volume 28, pages 1171–1179, Montréal, CA.

[9] Bentley, J. L. (1975). Multidimensional Binary Search Trees Used for Associative Searching. *Communications of the Association for Computing Machinery*, 18(9):509–517.

[10] Bernardi, R., Cakici, R., Elliott, D., Erdem, A., Erdem, E., Ikizler-Cinbis, N., Keller, F., Muscat, A., and Plank, B. (2016). Automatic Description Generation from Images: A Survey of Models, Datasets, and Evaluation Measures. *Journal of Artificial Intelligence Research*, 55(1):409–442.

[11] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

[12] Boroditsky, L. (2017). How language shapes the way we think. https://www.ted.com.

[13] Burba, F., Ferraty, F., and Vieu, P. (2009). k-Nearest Neighbour method in functional nonparametric regression. *Journal of Nonparametric Statistics*, 21(4):453–469.

[14] Chandar, S., Khapra, M. M., Larochelle, H., and Ravindran, B. (2016). Correlational neural networks. *Neural Computing*, 28(2):257–285.

[15] Chang, S. K. and Hsu, A. (1992). Image information systems: where do we go from here? *Transactions on Knowledge and Data Engineering*, 4(5):431–442.

[16] Chen, X., Fang, H., Lin, T., Vedantam, R., Gupta, S., Dollár, P., and Zitnick, C. L. (2015). Microsoft COCO Captions: Data Collection and Evaluation Server. *Computing Research Repository*, abs/1504.00325.

[17] Chen, X. and Zitnick, C. L. (2015). Mind's eye: A recurrent visual representation for image caption generation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2422–2431, Boston, US.

[18] Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *Computing Research Repository*, abs/1406.1078.

[19] Chollet, F. (2017). Xception: Deep Learning with Depthwise Separable Convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1800–1807, Honolulu, US.

[20] Chung, J., Gülçehre, Ç., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *Computing Research Repository*, abs/1412.3555.

[21] Cornelisse, D. (2018). An intuitive guide to convolutional neural networks. https://medium.freecodecamp.org/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050.

[22] Dai, B., Fidler, S., Urtasun, R., and Lin, D. (2017). Towards Diverse and Natural Image Descriptions via a Conditional GAN. In *International Conference on Computer Vision*, pages 2989–2998, Venice, IT.

[23] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the Association for Information Science and Technology*, 41(6):391–407.

[24] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, Miami Beach, US.

[25] Denkowski, M. and Lavie, A. (2014). Meteor universal: Language specific translation evaluation for any target language. In *Ninth Workshop on Statistical Machine Translation*, pages 376–380, Baltimore, US.

[26] Devlin, J., Cheng, H., Fang, H., Gupta, S., Deng, L., He, X., Zweig, G., and Mitchell, M. (2015). Language Models for Image Captioning: The Quirks and What Works. In *Conference on Empirical Methods in Natural Language Processing*, pages 100–105, Lisbon, PT.

[27] Donahue, J., Hendricks, L. A., Rohrbach, M., Venugopalan, S., Guadarrama, S., Saenko, K., and Darrell, T. (2017). Long-Term Recurrent Convolutional Networks for Visual Recognition and Description. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):677–691.

[28] Eisenschtat, A. and Wolf, L. (2017). Linking Image and Text with 2-Way Nets. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1855–1865, Honolulu, US.

[29] English, M. and Heeman, P. A. (2005). Learning Mixed Initiative Dialog Strategies By Using Reinforcement Learning On Both Conversants. In *Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 1011–1018, Vancouver, CA.

[30] Faghri, F., Fleet, D. J., Kiros, R., and Fidler, S. (2018). VSE++: Improved Visual-Semantic Embeddings. In *Proceeding of the British Machine Vision Conference*, Newcastle upon Tyne, UK.

[31] Farhadi, A., Hejrati, M., Sadeghi, M. A., Young, P., Rashtchian, C., Hockenmaier, J., and Forsyth, D. (2010). Every Picture Tells a Story: Generating Sentences from Images. In *European Conference on Computer Vision*, pages 15–29, Crete, GR.

[32] Feng, W. (2017). Learning apache spark with python. http://web.utk.edu/ wfeng1/spark/fnn.html.

[33] Gibiansky, A., Arik, S. Ö., Diamos, G. F., Miller, J., Peng, K., Ping, W., Raiman, J., and Zhou, Y. (2017). Deep voice 2: Multi-speaker neural text-to-speech. In *International Conference on Neural Information Processing Systems*, pages 2962–2970, Long Beach, US.

[34] Goh, A. T. (1995). Back-propagation neural networks for modeling complex systems. *Artificial Intelligence in Engineering*, 9(3):143–151.

[35] Goodfellow, I., Lee, H., Le, Q. V., Saxe, A., and Ng, A. Y. (2009). Measuring Invariances in Deep Networks. *Advances in Neural Information Processing Systems*, 22:646–654.

[36] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 27, pages 2672–2680, Montréal, CA.

[37] Gorman, B. (2017). Introduction to neural networks. http://blog.kaggle.com/2017/11/27/introduction-to-neural-networks.

[38] Graves, A., Mohamed, A., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, Vancouver, CA.

[39] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved Training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*, volume 30, pages 5767–5777, Long Beach, US.

[40] Gumbel, E. J. (1954). *Statistical theory of extreme values and some practical applications; a series of lectures*. U.S. Government Printing Office, Washington, US.

[41] Gurney, K. (1997). *An introduction to neural networks*. Taylor & Francis, Inc., Bristol, UK.

[42] Hannun, A. Y., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., and Ng, A. Y. (2014). Deep Speech: Scaling up end-to-end speech recognition. *Computing Research Repository*, abs/1412.5567.

[43] Harari, Y. N. (2015). *Sapiens: a brief history of humankind*. Harper, New York City, US.

[44] Haykin, S. (1998). *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, Upper Saddle River, US.

[45] He, H., Balakrishnan, A., Eric, M., and Liang, P. (2017). Learning Symmetric Collaborative Dialogue Agents with Dynamic Knowledge Graph Embeddings. *Computing Research Repository*, abs/1704.07130.

[46] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, Las Vegas Valley, US.

[47] He, X. and Deng, L. (2017). Deep Learning for Image-to-Text Generation: A Technical Overview. *IEEE Signal Processing Magazine*, 34(6):109–116.

[48] Hochreiter, S. (1998). The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(2):107–116.

[49] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

[50] Hodosh, M., Young, P., and Hockenmaier, J. (2013). Framing Image Description as a Ranking Task: Data, Models and Evaluation Metrics. *Journal of Artificial Intelligence Research*, 47:853–899.

[51] Hotelling, H. (1936). Relations between two sets of variates. *Biometrika*, 28(3/4):321–377.

[52] Huang, Y., Wang, W., and Wang, L. (2017). Instance-Aware Image and Sentence Matching with Selective Multimodal LSTM. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7254–7262, Honolulu, US.

[53] Huszar, F. (2015). How (not) to Train your Generative Model: Scheduled Sampling, Likelihood, Adversary? *Computing Research Repository*, abs/1511.05101.

[54] Jang, E., Gu, S., and Poole, B. (2016). Categorical Reparameterization by Gumbel-Softmax. *Computing Research Repository*, abs/1611.01144.

[55] Jing, Y. and Baluja, S. (2008). VisualRank: Applying PageRank to Large-Scale Image Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30:1877–1890.

[56] Kang, N. (2017). Multi-Layer Neural Networks with Sigmoid Function— Deep Learning for Rookies. https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f.

[57] Karpathy, A. and Fei-Fei, L. (2017). Deep Visual-Semantic Alignments for Generating Image Descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):664–676.

[58] Karpathy, A., Joulin, A., and Fei-Fei, L. F. (2014). Deep Fragment Embeddings for Bidirectional Image Sentence Mapping. In *Advances in Neural Information Processing Systems*, pages 1889–1897, Montréal, CA.

[59] Kingma, D. and Ban, L. (2014). Adam: A Method for Stochastic Optimization. *Computing Research Repository*, abs/1412.6980.

[60] Kirby, S. (2007). The evolution of language. *Oxford Handbook of Evolutionary Psychology*, pages 669–681.

[61] Kiros, R., Salakhutdinov, R., and Zemel, R. (2014a). Multimodal Neural Language Models. In *International Conference on Machine Learning*, volume 32, pages 595–603, Beijing, CN.

[62] Kiros, R., Salakhutdinov, R., and Zemel, R. S. (2014b). Unifying Visual-Semantic Embeddings with Multimodal Neural Language Models. *Computing Research Repository*, abs/1411.2539.

[63] Klopfenstein, L. C., Delpriori, S., Malatini, S., and Bogliolo, A. (2017). The Rise of Bots: A Survey of Conversational Interfaces, Patterns, and Paradigms. In *Conference on Designing Interactive Systems*, pages 555–565, Edinburgh, UK.

[64] Krizhevsky, A. and Hinton, G. E. (2011). Using very deep autoencoders for content-based image retrieval. In *European Symposium on Artificial Neural Networks*, Bruges, BE.

[65] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In *International Conference on Neural Information Processing Systems*, pages 1097–1105, Lake Tahoe, US.

[66] Kuligowska, K. (2015). Commercial Chatbot: Performance Evaluation, Usability Metrics and Quality Standards of Embodied Conversational Agents. *Professionals Center for Business Research*, 2:1–16.

[67] Kulkarni, G., Premraj, V., Dhar, S., Li, S., Choi, Y., Berg, A. C., and Berg, T. L. (2011). Baby talk: Understanding and generating image descriptions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1601–1608, Washington, US.

[68] Kusner, M. and Hernández-Lobato, J. M. (2016). GANS for Sequences of Discrete Elements with the Gumbel-Softmax Distribution. In *Conference on Neural Information Processing Systems*, Barcelona, ES.

[69] Levy, O., Goldberg, Y., and Dagan, I. (2015). Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

[70] Li, J., Monroe, W., Ritter, A., Jurafsky, D., Galley, M., and Gao, J. (2016). Deep Reinforcement Learning for Dialogue Generation. In *Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Austin, US.

[71] Li, S., Kulkarni, G., Berg, T. L., Berg, A. C., and Choi, Y. (2011). Composing Simple Image Descriptions Using Web-scale N-grams. In *Conference on Computational Natural Language Learning*, pages 220–228, Portland, US.

[72] Liang, X., Hu, Z., Zhang, H., Gan, C., and Xing, E. P. (2017). Recurrent Topic-Transition GAN for Visual Paragraph Generation. In *International Conference on Computer Vision*, pages 3382–3391, Venice, IT.

[73] Lin, C.-Y. (2004). ROUGE: A Package for Automatic Evaluation of summaries. In *Association for Computational Linguistics*, pages 74–81, Spain, ES.

[74] Lin, T., Maire, M., Belongie, S. J., Bourdev, L. D., Girshick, R. B., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft COCO: Common Objects in Context. *Computing Research Repository*, abs/1405.0312.

[75] Liu, S., Zhu, Z., Ye, N., Guadarrama, S., and Murphy, K. (2016). Optimization of image description metrics using policy gradient methods. *Computing Research Repository*, abs/1612.00370.

[76] Liu, S., Zhu, Z., Ye, N., Guadarrama, S., and Murphy, K. (2017). Improved Image Captioning via Policy Gradient optimization of SPIDEr. In *International Conference on Computer Vision*, pages 873–881, Venice, IT.

[77] Liu, X., Li, H., Shao, J., Chen, D., and Wang, X. (2018). Show, tell and discriminate: Image captioning by self-retrieval with partially labeled data. *Computing Research Repository*, abs/1803.08314.

[78] Luan, Y., Ji, Y., and Ostendorf, M. (2016). LSTM based Conversation Models. *Computing Research Repository*, abs/1603.09457.

[79] Luo, B., Wang, X., and Tang, X. (2003). World wide web based image search engine using text and image content features. *Society for Optical Engineering*, 5018:123–131.

[80] Maddison, C. J., Mnih, A., and Teh, Y. W. (2016). The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. *Computing Research Repository*, abs/1611.00712.

[81] Mao, J., Xu, W., Yang, Y., Wang, J., and Yuille, A. L. (2014). Deep Captioning with Multimodal Recurrent Neural Networks (m-RNN). *Computing Research Repository*, abs/1412.6632.

[82] Mason, R. and Charniak, E. (2014). Nonparametric Method for Data-driven Image Captioning. *Transactions of the Association for Computational Linguistics*, 2:592–598.

[83] Mathur, V. and Singh, A. (2018). The rapidly changing landscape of conversational agents. *Computing Research Repository*, abs/1803.08419.

[84] Mazur, M. (2015). A step by step backpropagation example. https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/.

[85] Mikolov, T., Chen, K., Repositoryado, G. S. C. R., and Dean, J. (2013). Efficient estimation of word representations in vector space. *Computing Research Repository*, abs/1301.3781.

[86] Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., and Joulin, A. (2018). Advances in Pre-Training Distributed Word Representations. *Language Resources and Evaluation*.

[87] Mirza, M. and Osindero, S. (2014). Conditional Generative Adversarial Nets. *Computing Research Repository*, abs/1411.1784.

[88] Olah, C. (2018). Colah's blog. http://colah.github.io/.

[89] Oliva, A. and Torralba, A. (2001). Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope. *International Journal of Computer Vision*, 42(3):145–175.

[90] Ordonez, V., Kulkarni, G., and Berg, T. L. (2011). Im2Text: Describing Images Using 1 Million Captioned Photographs. In *Advances in Neural Information Processing Systems*, pages 1143–1151, Granada, ES.

[91] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: A Method for Automatic Evaluation of Machine Translation. In *Association for Computational Linguistics*, pages 311–318, Philadelphia, US.

[92] Ranzato, M., Chopra, S., Auli, M., and Zaremba, W. (2016). Sequence level training with recurrent neural networks. In *International Conference on Learning Representations*, San Juan, PR.

[93] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *International Conference on Neural Information Processing Systems*, volume 1, pages 91–99, Montreal, CA.

[94] Ren, Z., Jin, H., Lin, Z., Fang, C., and Yuille, A. (2017a). Multiple instance visual-semantic embedding. In *Proceeding of the British Machine Vision Conference*, London, UK.

[95] Ren, Z., Wang, X., Zhang, N., Lv, X., and Li, L.-J. (2017b). Deep Reinforcement Learning-Based Image Captioning with Embedding Reward. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1151–1159, Honolulu, US.

[96] Rennie, S. J., Marcheret, E., Mroueh, Y., Ross, J., and Goel, V. (2017). Self-Critical Sequence Training for Image Captioning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1179–1195, Honolulu, US.

[97] Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, volume 32, pages 1278–1286, Beijing, CN.

[98] Rifai, S., Vincent, P., Muller, X., Glorot, X., and Bengio, Y. (2011). Contractive auto-encoders: Explicit invariance during feature extraction. In *International Conference on Machine Learning*, pages 833–840, Bellevue, US.

[99] Rubner, Y., Tomasi, C., and Guibas, L. J. (2000). The Earth Mover's Distance as a Metric for Image Retrieval. *International Journal of Computer Vision*, 40(2):99–121.

[100] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. MIT Press, Cambridge, US.

[101] Salakhutdinov, R. and Hinton, G. (2009). Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969 – 978.

[102] Schalkoff, R. J. (1997). *Artificial neural networks*. McGraw-Hill Higher Education, New York City, US.

[103] Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, 61:85–117.

[104] Schroff, F., Criminisi, A., and Zisserman, A. (2011). Harvesting Image Databases from the Web. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(4):754–766.

[105] Searle, J. R. (1980). Minds, brains, and programs. *Behavioral and Brain Sciences*, 3(3):417–424.

[106] Shang, L., Lu, Z., and Li, H. (2015). Neural Responding Machine for Short-Text Conversation. In *International Joint Conference on Natural Language Processing*, volume 1, pages 1577–1586, Beijing, CN.

[107] Shetty, R., Rohrbach, M., Hendricks, L. A., Fritz, M., and Schiele, B. (2017). Speaking the Same Language: Matching Machine to Human Captions by Adversarial Training. In *International Conference on Computer Vision*, pages 4155–4164, Venice, IT.

[108] Simmons, R. F. (1970). Natural language question-answering systems: 1969. *Communications of the Association for Computing Machinery*, 13(1):15–30.

[109] Simonyan, K. and Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *Computing Research Repository*, abs/1409.1556.

[110] Soh, M. (2016). Learning CNN-LSTM Architectures for Image Caption Generation. https://cs224d.stanford.edu/reports/msoh.pdf.

[111] Sordoni, A., Galley, M., Auli, M., Brockett, C., Ji, Y., Mitchell, M., Nie, J.-Y., Gao, J., and Dolan, B. (2015). A Neural Network Approach to Context-Sensitive Generation of Conversational Responses. *Human Language Technologies*, pages 196–205.

[112] Steels, L. (2015). *The Talking Heads experiment: Origins of words and meanings*. Language Science Press, Berlin, DE.

[113] Subhashini, R. and Kumar, V. J. S. (2010). Evaluating the performance of similarity measures used in document clustering and information retrieval. In *International Conference on Integrated Intelligent Computing*, pages 27–31, Bangalore, IN.

[114] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, Montréal, CA.

[115] Sutton, R. S. and Barto, A. G. (1998). *Reinforcement learning: An introduction*. MIT Press, Cambridge, US.

[116] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, Boston, US.

[117] Tamura, H. and Yokoya, N. (1984). Image database systems: A survey. *Pattern Recognition*, 17(1):29 – 43.

[118] Torralba, A., Fergus, R., and Freeman, W. T. (2008). 80 Million Tiny Images: A Large Data Set for Nonparametric Object and Scene Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1958–1970.

[119] Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, 49:433–460.

[120] Vedantam, R., Zitnick, C. L., and Parikh, D. (2015). CIDEr: Consensus-based image description evaluation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4566–4575, Bosten, US.

[121] Vendrov, I., Kiros, R., Fidler, S., and Urtasun, R. (2015). Order-Embeddings of Images and Language. *Computing Research Repository*, abs/1511.06361.

[122] Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *International Conference on Machine Learning*, pages 1096–1103, Helsinki, FI.

[123] Vinyals, O. and Le, Q. V. (2015). A Neural Conversational Model. *Computing Research Repository*, abs/1506.05869.

[124] Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2014). Show and Tell: A Neural Image Caption Generator. *Computing Research Repository*, abs/1411.4555.

[125] Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2017). Show and Tell: Lessons Learned from the 2015 MSCOCO Image Captioning Challenge. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):652–663.

[126] Wang, L., Li, Y., and Lazebnik, S. (2017). Learning Two-Branch Neural Networks for Image-Text Matching Tasks. *Computing Research Repository*, abs/1704.03470.

[127] Weizenbaum, J. (1966). ELIZA - A Computer Program for the Study of Natural Language Communication Between Man and Machine. *Communications of the Association for Computing Machinery*, 9(1):36–45.

[128] Williams, R. J. (1992). Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8(3-4):229–256.

[129] Witten, I. H. (1977). An Adaptive Optimal Controller for Discrete-Time Markov Environments. *Information and Control*, 34:286–295.

[130] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, volume 37, pages 2048–2057, Lille, FR.

[131] Xu, Z., Liu, B., Wang, B., Sun, C., and Wang, X. (2017). Incorporating loose-structured knowledge into conversation modeling via recall-gate LSTM. In *International Joint Conference on Neural Networks*, pages 3506–3513, Anchorage, US.

[132] Yagcioglu, S., Erdem, E., Erdem, A., and Cakici, R. (2015). A Distributed Representation Based Query Expansion Approach for Image Captioning. In *International Joint Conference on Natural Language Processing*, volume 2, pages 106–111, Beijing, China.

[133] Yan, C., Xie, H., Yang, D., Yin, J., Zhang, Y., and Dai, Q. (2018). Supervised Hash Coding With Deep Neural Network for Environment Perception of Intelligent Vehicles. *Transactions on Intelligent Transportation Systems*, 19(1):284–295.

[134] Yang, Y., Teo, C. L., Daumé, III, H., and Aloimonos, Y. (2011). Corpus-guided Sentence Generation of Natural Images. In *Conference on Empirical Methods in Natural Language Processing*, pages 444–454, Edinburgh, UK.

[135] Zhang, L., Sung, F., Liu, F., Xiang, T., Gong, S., Yang, Y., and Hospedales, T. M. (2017). Actor-Critic Sequence Training for Image Captioning. *Computing Research Repository*, abs/1706.09601.

[136] Zipf, G. K. (1949). *Human Behavior and the Principle of Least Effort*. Addison-Wesley, Boston, US.

Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe angefertigt und mich anderer als der im beigefügten Verzeichnis angegebenen Hilfsmittel nicht bedient habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Hamburg, den __________ Unterschrift: _________________________