INDEPENDENT STUDY

STATE-OF-THE-ART AUTOMATED IMAGE DESCRIPTION SYSTEMS

Written by: Fabian Karl
Student ID: 6886047
University of Hamburg

Examiner: Prof. Dr. Chris Biemann
Department of Language Technology
University of Hamburg

Advisor: Prof. Dr. Simone Frintrop
Department of Computer Vision
University of Hamburg

July 18, 2018

Abstract

Automated Image Description (AID) is a highly interesting task connecting image processing and language generation. Recent models using ideas from Generative Adversarial Network (GAN) and Reinforcement Learning (RL) claim to be the new state-of-the-art approach, when it comes to creating descriptions more natural and human like than previous models. For a longer time Vector-to-Sequence (Vec2Seq) models, inspired by ideas from Sequence-to-Sequence (Seq2Seq), used most famously for machine translation, were considered state-of-the-art. Image descriptions are learned in an end-to-end fashion and report state-of-the-art performance for most word n-gram based scores like BLEU (Papineni, Roukos, Ward, & Zhu, 2002) or ROUGE (Lin, 2004). Both models, together with a brief history of AID, are explained and compared in this survey.

In order to test the performance of automated end-to-end image description systems with Neural Network (NN), a Vec2Seq model was implemented and validated. This model builds a baseline model for AID with NN and can also be used for pretraining of possible future RL-GAN models.

Contents

Abbreviation

A3C	Asynchronous Advantages Actor-Critic.
AID	Automated Image Description.
CNN	Convolutional Neural Network.
DQN	Deep Q-Learning.
GAN	Generative Adversarial Network.
LSTM	Long-Short-Term Memory.
NN	Neural Network.
RL	Reinforcement Learning.
RNN	Recurrent Neural Network.
Seq2Seq	Sequence-to-Sequence.
Vec2Seq	Vector-to-Sequence.

1 Introduction

Understanding and describing visual stimuli is one of the brain's best skills. In order to allow artificial systems to perform something similar, Automated Image Description (AID) (or Image Captioning) systems are built. This field of research between Image Processing and Language Generation is not yet completely solved and there have been interesting new approaches recently.

Images can be described in multiple ways, depending on the information that is most important to the viewer. In the same way, image descriptions can be analyzed from different view-points (Shatford, 1986; Jaimes & fu Chang, 2000). Image description in this paper refers to a description of the entities in the image, their attributes, actions and relations to each other. Further information like location, events outside of the image or any other assumed knowledge are outside of the scope of AID-Systems, presented in this work.

In simple terms: An image goes in and a description of what is happening on the image comes out.
Applications of AID can be found in creating article descriptions for merchants, automatically captioning the photographs from your last trip or automated scene description for visually impaired people. Besides these applications, a human-like understanding of visual scenes and an expression of those through human language is a corner stone of any further research into humanoid artificial communication and reasoning. This paper mainly focuses on recent methods and models that have tried to solved the task of AID quite successfully.

Section 3 gives a short overview of the scientific history until roughly 2015. Section 4 describes the first of two recent ideas of how to solve AID in an end-to-end fashion without hand-crafted rules or templates. Section 5 then explains the most recent advances to create descriptions even more natural and diverse, even more human like.
Based on the theoretical foundation of these sections, the encoder-decoder model was selected and replicated in order to create a baseline AID system (Section 6). A final conclusion is given in Section 7.

This work is intended as both a survey of the recent advances in AID as well as a replication study of one of the state-of-the-art methods for it.

(Chen & Zitnick, 2015)

Figure 1: Whenever telling someone about an event one observed, in a way, automated image description is done by our brains. This process is highly subjective since the brain describes only what it deems to be interesting.

2 Evaluation of language generation

Scientific progress can only happen, if a measurement of success is defined first. Without being able to observe how good the outcome on an experiment is, it is impossible to improve on it. The measurement or evaluation of how good a generated sentence is, is not trivial.

Most metrics measure the quality of sentences by comparing the occurrences of n-grams in the generated text and in human created text (*ground-truth*). An n-gram is a sequence of any n words than occur in a given text. The BLEU score (Papineni et al., 2002) calculates the precision (True positives / Predicted condition positives) of n-grams and the ROUGE score (Lin, 2004) calculates the recall of n-grams (True positives / Condition positives). METEOR (Denkowski & Lavie, 2014) is a combination of both the precision and the recall of n-grams. CIDEr (Vedantam, Zitnick, & Parikh, 2015) is a weighted statistic over n-grams.

Defining quality of generated text with n-grams is an easy and cheap way to make model-outcomes comparable and to create a score that one can strive to increase. What scores based on n-grams suffer the most, is originality in generation. Almost every concept can be phrased in different ways, using completely different sentences. Two semantically different sentences can have a very similar or the same meaning and both can have a high quality in their formulation from a human perspective. If a score only regards n-gram similarity, two sentences like described could have very different scores.

A possible solution to this problem might lie in training a system to discriminate between human created sentences and non-human created sentences. If the system has enough

training data it should be able to score sentences based on their composition and not simply based on *n*-grams. This approach brings the disadvantage that different discriminators, trained on different datasets are not comparable with each other. This is one of the most important criterions of a score, though, since it allows different models to be compared directly. Section 5 will describe how this way of scoring sentences can be applied to AID.

A recent new metric called SPICE (Anderson, Fernando, Johnson, & Gould, 2016) was recently proposed. Instead of matching between *n*-grams, it focuses on linguistic entities and their relations and attributes.

3 History of related work

Before discussing two state-of-the-art methods for AID in more detail, the general history and other related work is covered in this section. Bernardi et al. (2016) separate all studies they found on AID into two categories, where one has another two sub-categories. The first approach is called 'Description as generation from visual input'.

3.1 Description as generation from visual input

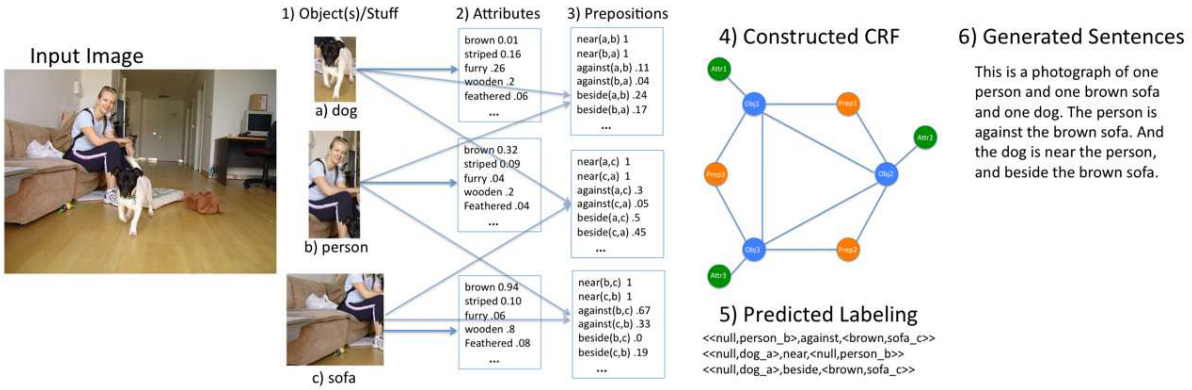
Studies in this category share the same general idea of first analyzing the visual content of an image and then generating text that represents this content. Bernardi et al. (2016) describe the general process like this:

1. *Computer vision techniques are used to classify the scene type, to detect the objects present in the scene, to predict their attributes and the relationships between them, and to recognize the actions taking place.*

2. *This is followed by a generation phase that turns the detector outputs into words or phrases. These are then combined to produce a natural language description of the image, using techniques from natural language generation (e.g., templates, n-grams, grammar rules).*

Both of these steps can be implemented in various ways. For the first phase, the image representation can be modeled differently. Farhadi et al. (2010) use spatial relationships, Yang, Teo, Daumé, and Aloimonos (2011) use corpus-based relationships and Kulkarni et al. (2011) applied spatial and visual attributes. There are many more approaches that all share the fact that they use some sort of hand crafted feature space to capture what is happening in a scene. Instead of these techniques, the usage of a end-to-end model like a Neural Network (NN) would also be possible here (Section 4.2).

The second step, the generation of description can either be based on templates or on some sort of language model. Yang et al. use templates for example that are filled by selecting the most likely objects, actions, attributes and prepositions based on a Hidden Markov Model. Using a language model is a less constrained way to create novel text. Kulkarni et al. and Li, Kulkarni, Berg, Berg, and Choi (2011) both used n-gram language models to generate image descriptions. A language model is used to predict the probabilities of the next word, given a history of n words. This can be done with n-grams but also with Recurrent Neural Network (RNN). RNNs can be seen as continuous language models, producing a probability



(Kulkarni et al., 2011)

Figure 2: An example for a generative model. Image processing steps create a representation of objects, relations and attributes. A generative model uses this representation to create text.

distribution over all possible words based on the history of produced words. Section 4 will give more information on RNNs as language models.

Figure 2 shows how these two steps can look like schematically.

3.2 Description as a Retrieval Task

Bernardi et al.'s (2016) second category of studies are called retrieval based methods. This category is split into two subcategories: Retrieval in visual space and retrieval in multimodal space. Works in the first subcategory try to retrieve a set of candidate images based on a chosen similarity measure. Then the descriptions of the candidates are either ranked and the best one is selected or they are modified and/or combined to create a new description for the image at hand.

The second category, retrieval in multimodal space, describes a similar approach, but the relation between image and text is mapped to a multimodal space. Then a query is used on the representation space to perform cross-modal (image-sentence) retrieval. Bernardi et al. (2016) describe encoder-decoder based models as retrieval-models in multimodal space. This is arguable, since there is no direct retrieval from a corpus of images or image-text pairs.

3.3 Alternative Split

X. He and Deng (2017) split early approaches of image description in template matching models (Farhadi et al., 2010; Li et al., 2011; Kulkarni et al., 2011) and retrieval based meth-

ods (Ordonez, Kulkarni, & Berg, 2011; Hodosh, Young, & Hockenmaier, 2013). They consider encoder-decoder models with deep NN as its own category of models.

The next chapter will show, how encoder-decoder models look like and how they work.

4 Encoder-decoder models

Encoder-decoder models consist of an encoder model and a decoder model. Similar to the general principle proposed in Section 3.1, the encoder is extracting features from an image and represents it in some numerical way and the decoder interprets this representation and generates text conditioned on it. Every produced token is directly compared to the ground truth sentence. This is called *teacher forcing*, since it forces the system to produce the same or very similar output to the ground truth.

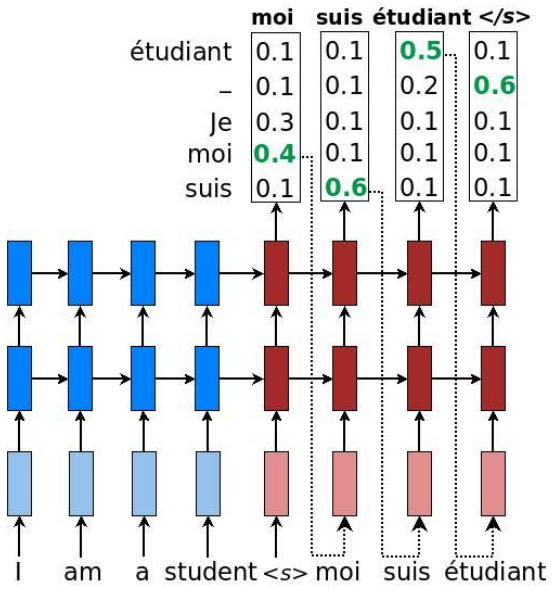
In difference to the previously described systems, this one is end-to-end trainable. This means, no conventional score like BLEU (Papineni et al., 2002) or ROUGE (Lin, 2004) are used to improve the performance of the model. They still are used to test the performance of the system, in oder to compare it to other models. In addition, no hand crafted features like templates or fixed word type (adjective, noun, verb, etc.) have to be added by human experts. The usage for this framework was first proposed for language-sequence-to-language-sequence (classic Sequence-to-Sequence (Seq2Seq)) pipelines, most famously used for sentence translation.

4.1 The idea behind Sequence-to-Sequence

Seq2Seq models have shown great performance in automated language translation systems (Bahdanau, Cho, & Bengio, 2014; Chung, Gulcehre, Cho, & Bengio, 2015; Sutskever, Vinyals, & Le, 2014). There is an intuitive step-by-step tutorial by Tensorflow (Luong, Brevdo, & Zhao, 2017) explaining how to build a Seq2Seq machine translation system.

The idea behind Seq2Seq is schematically shown on a machine translation example in Figure 3. The encoder is normally built by Long-Short-Term Memory (LSTM) cells followed by multiple dense layers. The LSTM, first proposed in 1997 by Hochreiter and Schmidhuber, is an extension of the RNN that performs much better on longer sequences. Chung, Gulcehre, Cho, and Bengio (2014) offer a comprehensive evaluation of the differences and advantages of gated RNNs like the LSTM. Dense layers are regular feed forward layers with no recurrent connections. The output of the encoder is often called a context vector. It incorporates the extracted information of the sentence by the network.

The decoder has basically the same structure as the encoder, but the output is not only stored after the full input-sequence is ran through the network completely, but instead every single timestep, the result from the network is saved, and itself fed back into the network as additional information for the next timestep. The output for every timestep is a softmax



(Luong et al., 2017)

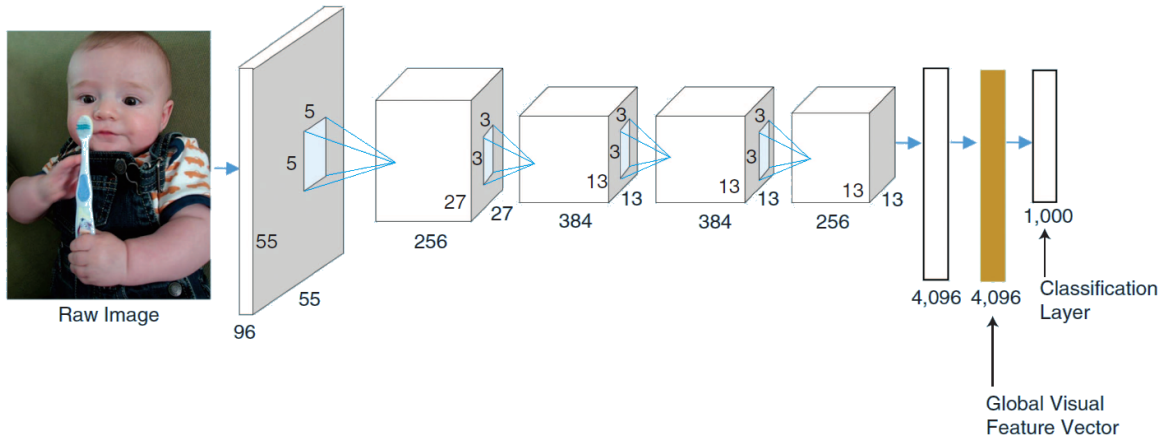
Figure 3: An example for a Seq2Seq machine translation system. The input *I am a student </s>* (END_SYMBOL) is transformed into a internal representation in form of a numeric vector. This vector is then used to produce one token after the other, always including the information of the previously produced words.

layer, representing a vocabulary of fixed size. This output can be used in every timestep to calculate an error between predicted output and *true* output. The *true* output could be the correctly translated sentence for example. If the network only predicted the *true* word with a probability of 0.4 an error of 0.6 can be derived. This is called a loss function and the errors can be accumulated and used to train/optimize the network.

Training a NN is almost always done by first calculating the gradients of every weight in the network for the chosen loss function. This gradient is then backpropergated through the whole network and all weights are adjusted (Schmidhuber, 2015; Haykin, 1994) in order to minimize the error.

The example in Figure 3 shows how an English sentence is fed into the encoder. The produced context vector together with a START_SYMBOL is then fed into the decoder to produce the first real word of the sentence. In the next step, the START_SYMBOL and the first word of the sentence together with the context vector is used to determine the probability distribution over the next word. This process is repeated until a fixed sentence length is reached or the END_SYMBOL is generated.

Next to machine translation, this approach has also been used to create conversational bots (Vinyals & Le, 2015).



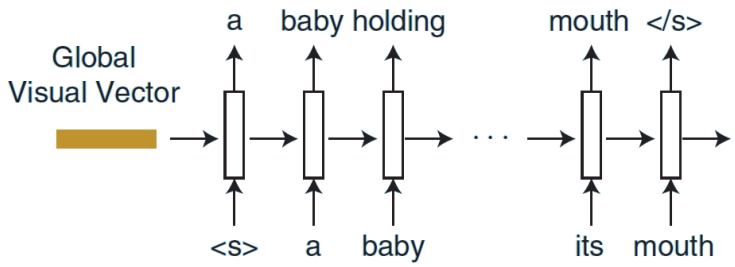
(X. He & Deng, 2017)

Figure 4: A simplified look on a deep CNN. The raw image pixels are first repeatedly filtered, then fully connected dense layers interpret the results of the filters and finally a classification softmax layer categorizes the image. If the classification layer is dropped, the same network can be used to create a visual feature vector that can be used in further tasks.

4.2 Image to text-sequence

Almost the same idea can be used for automated image description (Kiros, Salakhutdinov, & Zemel, 2014a, 2014b; Devlin et al., 2015; Donahue et al., 2017; Chen & Zitnick, 2015; Vinyals, Toshev, Bengio, & Erhan, 2014; Karpathy & Fei-Fei, 2017). The decoder stays exactly the same. The encoder is switched from a sequence analyzing LSTM to an image analyzing Convolutional Neural Network (CNN). Like LSTMs for sequences, CNNs have shown state-of-the-art performance for the last couple years when it comes to image processing. The idea has existed since 1989 when LeCun et al. first proposed it. Like NNs in general, it took quite a while until the idea of CNNs got really accepted in scientific circles. Nowadays, especially their successes in image classification (K. He, Zhang, Ren, & Sun, 2016; Krizhevsky, Sutskever, & Hinton, 2012; Simonyan & Zisserman, 2014) and object detection (Ren, He, Girshick, & Sun, 2015) has made them the go-to for many tasks involving images.

CNNs basically adapt the weights of different filters, by being trained on a supervised dataset (Deng et al., 2009). Image classification, where the network is trained to categorize between different animals and objects, is a famous task in computer vision. With existing labeled datasets, the network is trained to adapt its filters, to optimally analyze and categorize each given image. Comparing these filters between different studies and even different tasks shows that they often share common features (Zeiler & Fergus, 2014). Early filters are often



(X. He & Deng, 2017)

Figure 5: Taking the visual feature vector from Figure 4 and feeding it through a decoder as explained in Section 4.1 will yield the most probable selection of tokens. By having a *true* label/description for the image, every generation step will yield a loss that can be used to train the network.

automatically adapted in the training process to have a high activation for simple shapes, like vertical or horizontal lines. Filters in deeper layers react to more specific shapes, like the tire of a car or maybe the face of a dog.

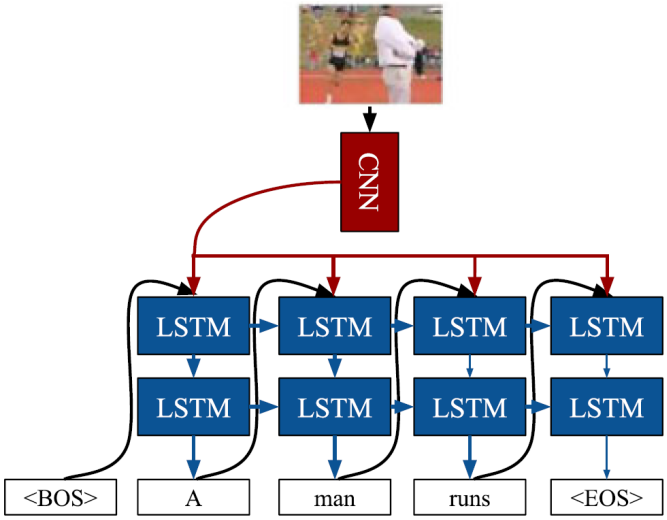
Figure 4 and Figure 5 show this process in a schematic way. The input image is fed into a pretrained CNN. Either the classification layer of the CNN was dropped beforehand, or simply the activation of one of the previous layers is taken as visual feature vector. From the CNN's point of view, this feature vector should represent the most important information of the given image. This vector, together with its respective ground truth description, is then fed into a generative decoder model word by word and creates a sequence of tokens.

Figure 6 shows a more detailed and specific architecture used by Donahue et al. (2017). They use two layers of stacked LSTM cells and use the image vector as input at every generation step.

These models will be called Vector-to-Sequence (Vec2Seq) models. The same way as the Seq2Seq models, the decoder compares the generated sentence to the *ground-truth* description, fed into the network. By doing so, an error for training the network can be calculated. To measure the quality of the output, an n-gram based score can be used for validation and testing purposes.

4.3 Conclusion

Seq2Seq and Vec2Seq models offer a solid approach to language generation in an end-to-end fashion. State-of-the-art performance, regarding BLEU score for example, can and have been



(Donahue et al., 2017)

Figure 6: A possible complete architecture for image description with NNs. The image is transformed into a visual vector and appended to the input data for every step of the sentence. Two stacked layers of LSTMs take the state (representing the previously produced tokens) and the image vector and generate a probability distribution over all possible tokens.

reached using this model. The only thing this model needs is a sufficient training corpus and it will implicitly learn an n-gram language model and produce sentences based on it. The framework is also quite simple and the principle ideas behind it are easy to grasp. So where is the 'but'?

Models from this family map an input to a sequence of tokens. This approach is very useful, if the generated sequence should stick as close as possible to the *ground-truth* i.e. the sentences produced by humans. This is exactly where language differs from other sequences. A correct syntactical and semantical sentence can have many different facets. To force a system to produce sentences with an underlying n-gram model or LSTM model will often yield sentences very similar to the ones trained on. Not only this, but the generated sentences often lack diversity and show little human-like creativity. They will have a high resemblance to each other and to the training corpus which by the nature of language will always include only a small fraction of all the possible combinations of words and the different syntactical ways to express the same meaning. In Figure 7, this can be observed nicely. The captions describe the images well, but they all sound similar and as is they were following some underlying template.

In order to create even more human like sentence with more diversity and creativity, a

(Donahue et al., 2017)

Figure 7: Descriptions/Captions created by a Vec2Seq model. The captions describe the images correctly, but they lack diversity and naturalness. They all have a similar structure and wording.

slightly different model can be used. A system like that would distance itself further from the idea of calculating its error by comparing the generated sentence directly to a ground truth sentence. This *teacher forcing* is limiting the naturalness and diversity of the generated language. With the introduction of Generative Adversarial Network (GAN) by Goodfellow et al. in 2014, a possible solution to this problem was created.

It should be noted here, that Vec2Seq models are normally still used to pretrain the generative system. Adversarial methods can then be used to further increase the quality of the generated text. This should emphasis the continuing importance of Seq2Seq and Vec2Seq models.

5 GAN based models

5.1 Generative Adversarial Networks

A GAN is less of a model and more of a general idea of how to train two networks in an adversarial fashion (Goodfellow et al., 2014). The two networks are a generator G and a discriminator D. The task of the generator is to replicate some real and existing data in a way to fool the discriminator, whose task in turn is to determine whether some input was provided from the set of real data, or was created by the generator. The generator is unaware of the distribution, and usually uses random noise as a starting point to learn to replicate the real data using the feedback from the discriminator. When the generator gets meaningful input that is needed for the generation, it is often called a conditional GAN, since the generator creates samples, conditioned on a certain input (Mirza & Osindero, 2014). The discriminator is simply trained to map real data samples to *true* and generated samples to *false*.

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.

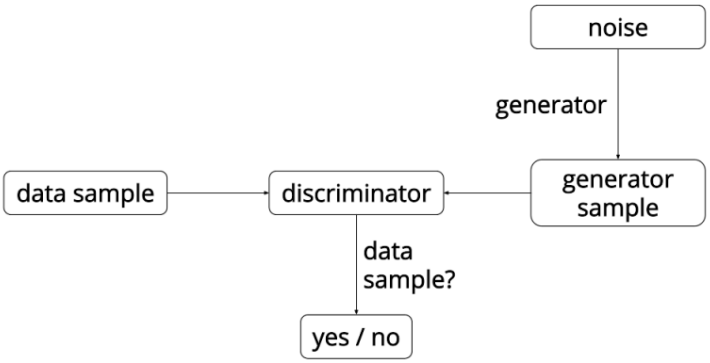
After successful training, this should result in two expert systems: The generator that can generate samples very similar to the actual data samples and the discriminator that is trained to spot any difference between real and generated sample. This has been proven to work well on images, where the smallest shifts in the parameter space lead only to small changes in the color of single pixels (Reed et al., 2016; Radford, Metz, & Chintala, 2015).

A large advantage of the GAN is the fact that it can be trained in an unsupervised fashion (Goodfellow et al., 2014), which allows to employ large bodies of training data without corresponding labels. For many sorts of data, images and text in particular, unsupervised training is preferred, since huge amounts of data are available, but labeling is expensive.

At its core, both components of the GAN can consist of NNs of different complexity and overall structure. Depending on the properties of the input and its structure, different types of NNs can be used. When it comes to training, the classically used min-max game, based on maximum likelihood, is not the only option. The next chapter shows, how the idea of GANs can be used in a Reinforcement Learning (RL) framework.

5.2 Reinforcement learning and General Adversarial Networks

The generator in a GAN is often a NN that adapts its weights in order to make the discriminator produce positive (real data) output. This generator does not have to be a NN, though.



(Glover, 2016)

Figure 8: The general idea of the GAN framework. A generator produces samples to fool the discriminator. The discriminator is trained to detect generated samples.

Especially, when it comes to generating sequences, using NNs can become impractical. Sentence generation is the sequential process of a discrete selection of tokens. It is this sequential nature, that makes GANs with a NN as generator not a smart choice for sentence generation.

The first reason is that they are designed to output real-valued, continuous data (like images). When doing this, the generator will make small changes in order to improve the quality of its output gradually. This is hardly possible when creating discrete tokens like words (Goodfellow, 2016; Huszar, 2015).

The second problem is that the discriminator in a GAN can only score the whole sentence and not parts of the generated sequence. A much faster convergence can be reached when parts of the sentence can be scored as unfinished sequences.

In order to avoid these two disadvantages, L. Yu, Zhang, Wang, and Yu (2016) propose a different GAN architecture for sentence generation with a discriminator. Instead of using a NN for generation, they used RL to generate sequences and the output of the discriminator is interpreted as reward for the system. This idea was previously proposed by Bachman and Precup (2015) and Bahdanau et al. (2016). In an extension of this work, Dai, Fidler, Urtasun, and Lin (2017) used the same approach, but created a situation, where the generated text was also conditioned on an image vector that was extracted from a pretrained CNN. How this RL-GAN looks like and how it works are explained in the Section 5.3. Before explaining this framework, a short introduction to general RL is given here.

RL models consist of an agent in an environment that is executing different actions and receiving reward from the environment for these actions (Figure 9). The goal of the system is

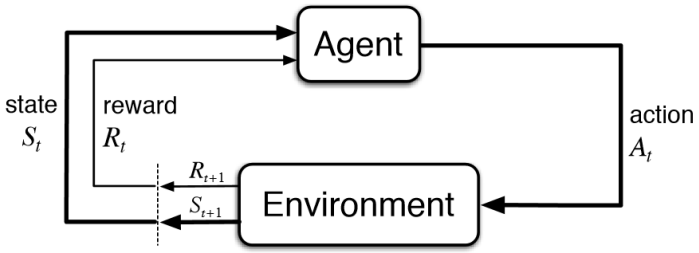


Figure 9: High level view on the idea behind RL. The agent picks a action A at timestep t. This action is chosen based on a certain policy (e.g. acting greedy with regard to the expected reward). The environment returns the new state of the world back to the agent and the reward for the executed action. Based on this, the agent tries to maximize the overall future reward.

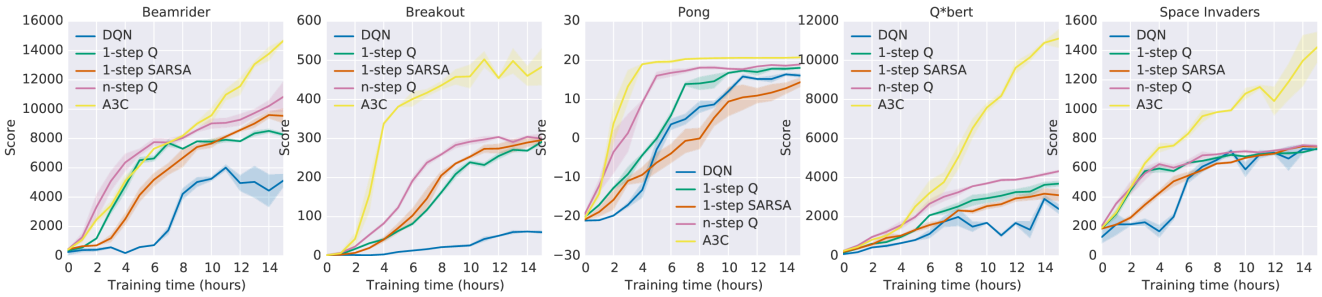
to maximize the overall reward, by choosing the best action in each state the agent finds itself in. Overall reward means that not only the immediate reward for the next action is important, but the whole sequence of all future rewards for all future actions is to be maximized. Gilman (2018) offers a highly intuitive and quick understanding of RL in his blog. Sutton and Barto (1998) give detailed explanations about the principle ideas behind RL and Arulkumaran, Deisenroth, Brundage, and Bharath (2017) offer a quite recent overview over RL combined with deep learning.

There are many different ways to implement RL algorithm. While Q-Learning (Watkins & Dayan, 1992) was very popular for quite some time, policy gradient (Sutton, McAllester, Singh, & Mansour, 1999) methods are getting the most attention right now. Policy gradient methods try to maximize the future reward by directly estimating the best action to chose in every state (F. Yu, 2017), while Q-Learning tries to estimate the value or the future reward of a state. Policy based methods return a probability distribution over all existing actions, whereas value iteration methods (Q-Learning) return the expected reward for all the actions or future states.

From the policy gradient family, the Actor-Critic framework (Witten, 1977; Barto, Sutton, & Anderson, 1983) is currently getting a lot of attention. One of the newest approaches is called Asynchronous Advantages Actor-Critic (A3C) and has shown great performance when it comes to Atari games (Mnih et al., 2016). This is due to two main benefits.

(1) Asynchronous training on different threads allows a faster training and offers a bigger exploration of the environment.

(2) Advantage refers to the fact that A3C algorithms are not trained on the reward from the environment directly, but train a critic to estimate the future reward from the environment. By subtracting the actual reward from the expected reward given by the critic and using this difference as so called advantage value, the estimated reward from the critic acts as



(Mnih et al., 2016)

Figure 10: Performances of different RL algorithms on different Atari games. Deep Q-Learning (DQN), $n(1)$-step Q-Learning and SARSA are all value iteration based algorithms. This is a selection from some of the classic Atari games, where A3C clearly outperforms other RL algorithms.

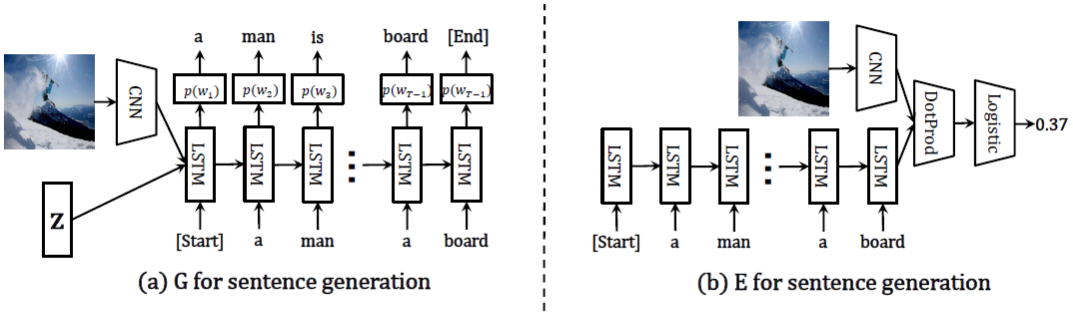
a bias. In contrast to standard policy gradient methods, the system is not trained on the rewards directly, but is trained on the information, if the given reward is bigger or smaller than expected. Together, this allows for a faster and more stable training process than many other RL algorithms (Mnih et al., 2016).

5.3 State-of-the-art model: RL-GAN

The conditional RL-GAN framework, proposed by Dai et al. (2017), consists of a generative model and a discriminator model. The generator is made up by a NN that has three different inputs: The image-vector from a CNN; the history of already produced words in this sentence; a random input vector for more variability. This network creates one token after another, until either the END_SYMBOL was produced, or a fixed sentence length is reached. Figure 11a shows this.

The generated sentence is given to the discriminator. The discriminator itself is a NN that has two inputs: An image-vector and a sentence. Depending on whether the image-sentence pair is a real example (form the human-annotated dataset) or a fake example (sentence from the generator), the discriminator is trained binary to output a 1 or a 0, respectively. This means the discriminator is trained to produce a low reward for image-sentence pairs that differ far form the real data. Figure 11b shows this.

The value returned by the discriminator is representing its believe that the given image-sentence pair is form the real dataset. The higher this believe is, the more the discriminator was fooled and the higher is the reward for the generator. The generator is trained, to maximize overall reward over all actions. L. Yu et al. (2016) and Dai et al. do this by using the REINFORCE algorithm (Williams, 1992). Zhang et al. (2017) use A3C (Mnih et al., 2016),



(Dai et al., 2017)

Figure 11: (a) The generator gets three inputs: image, noise (z) and history of produced words. The output is the next token. (b) The discriminator gets two inputs: image and sentence. The output is a measurement of how well the two fit together.

which allows for parallel, asynchronous training.

By repeatably training both discriminator and generator for one step, both will gradually become better at their designated task, ultimately resulting in generated descriptions that are indistinguishable from the ones from the original dataset.

To come back to the earlier mentioned problem of unfinished sequences, Monte Carlo search (Tesauro & Galperin, 1997) is proposed as solution by L. Yu et al. (2016) and also used by Dai et al. (2017). Monte Carlo search describes the process of sampling possible future events in order to estimate the future reward. In this concrete case, this means, after every produced token that did not end the sequence, the sequence is finished with the same or a simplified generator for n times. The finished sequences can be scored by the discriminator and averaged to result in an estimation of future reward. By doing this, unfinished sequences can be scored and this can greatly decrease training time.

Liang, Hu, Zhang, Gan, and Xing (2017) also propose a similar framework, with additional focus on creating paragraphs with consistent and smooth topic-transitions.

6 Creating a Baseline Model

The very recent RL-GAN models are said to be able to create even more human like descriptions and sentences, with more naturalness, than Vec2Seq models. For this reasons, RL-GAN models can be seen as the current state-of-the-art image description techniques. But since these models need to be pretrained by a 'simpler' model in oder to work, the creation and validation of a baseline model is necessary in the first place. The fact, that these models, when tuned correctly, create very impressive descriptions for images should also not be neglected.

In order (1) to strengthen the evidence for Vec2Seq models, (2) to create a baseline model and to create a system that can be used for pretraining RL-GAN systems, a Vec2Seq model was implemented and tested. The implementation is oriented on Chen and Zitnick (2015). Chollet (2017a) is also a great reference regarding the implementation in Keras (Chollet et al., 2015).

6.1 Implementation Details

6.1.1 Dataset

The dataset MS COCO 2017(Chen et al., 2015) was used for training and validation purposes. In the 2017 version of the dataset, the training corpus contains 118.287 images, and the validation corpus 5.000 images. The dataset for the final testing is made up by 41.000 images.

Every image is labeled with 5 (sometimes 6) ground truth sentences, describing what is primarily happening in the image. These sentences can be called captions, since they do not describe every detail of the image, but rather the most dominant feature or action in the image. This dataset is widely used, since it offers a big variety of images with multiple descriptions/captions. The ground truth annotations for the testing dataset are hidden and are only used for the final evaluation after submitting one's results.

6.1.2 System

As described in Section 4.2, the system consists of one or multiple LSTM layers. The input for the first LSTM layer at every timestep is one word, embedded in a certain way, and a representation of the image in form of an image vector. The image vector is retrieved through a pretrained CNN and the embedding through a pretrained dense vector embedding model. Figure 6 shows the architecture from a related work, that is very similar to the one used in this work.

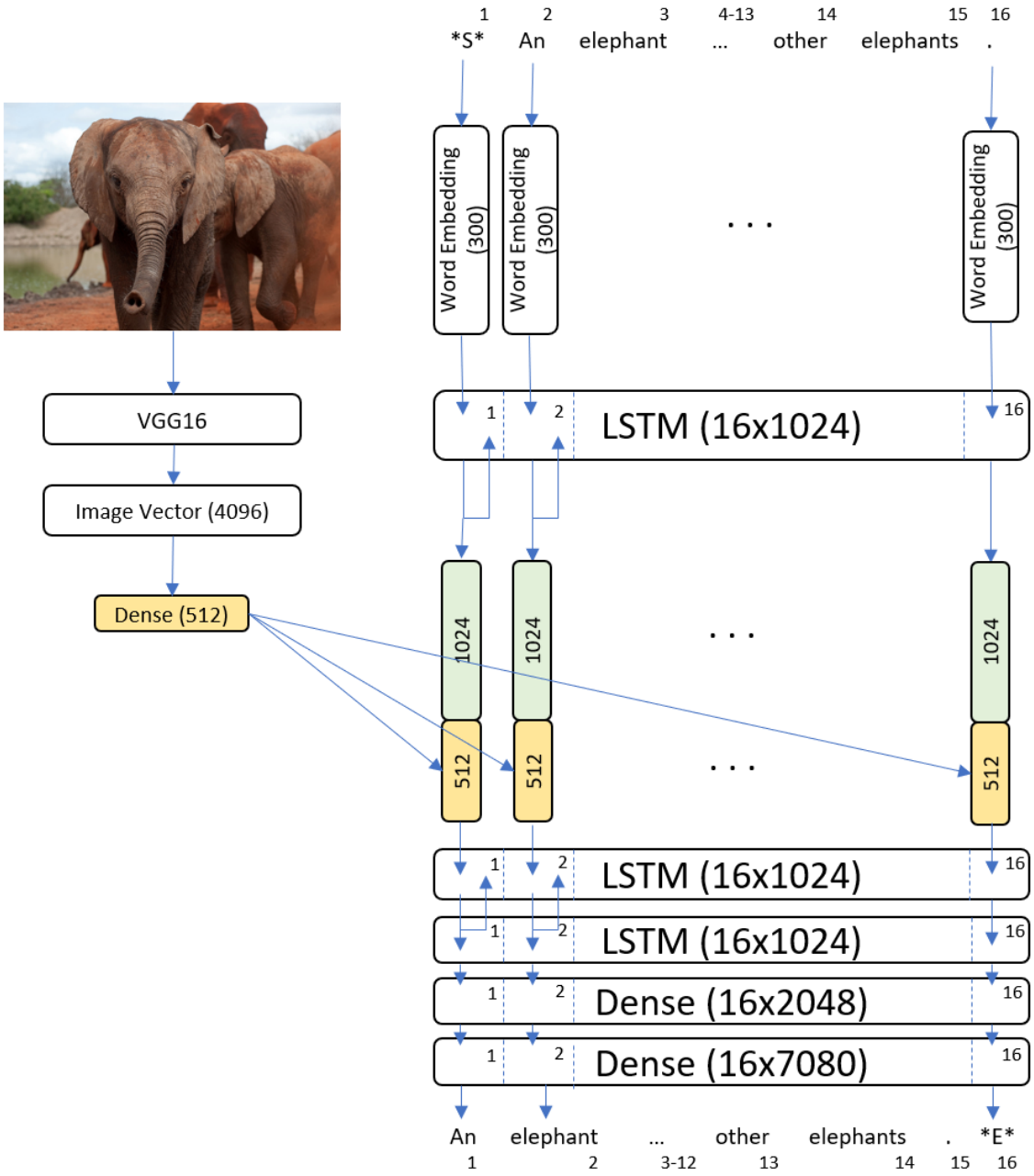


Figure 12: The architecture of the model as implemented in this work in detail. An image and a ground truth sentence is taken for one learning iteration. Image and sentence are embedded and merged. The resulting matrix is fed into stacked LSTMs followed by dense layers, followed by a categorization layer. The same sentence, shifted one step into the future, is used as target sentence.

The image vectors and embedded captions are created in a separated preprocessing step. The image vectors are created with the pretrained VGG16 (Simonyan & Zisserman, 2014) taken directly from Keras Applications (Chollet et al., 2015). CNNs are normally pretrained on large classification corpora and can then be used for various image tasks. The normal procedure is to remove the last layer, the classification layer, and to max- or average-pool the second last layer into a new vector. This new vector is normally called context or image vector. In this work it contains 4096 float values and max pooling is used to create it. There are other CNN models that outperform VGG16 in classification tasks and in parameter needed to do so. A lot of the literature still use VGG16 for their experiments, though. This is probably due to its prestige as being one of the first deep CNNs that could perform extraordinary well on the image classification task. It was used in this work, partly for the same reason, partly to create a valid comparison to other papers. Further experiments should include testing different models like Xception (Chollet, 2017b) or Inception-v4 (Szegedy, Ioffe, Vanhoucke, & Alemi, 2017).

For the word embedding, the English fasttext model (Mikolov, Grave, Bojanowski, Puhrsch, & Joulin, 2018) is used. Fasttext (Bojanowski, Grave, Joulin, & Mikolov, 2017) does not assign a word vector to every single word, but instead uses sub-informations of the words to create the embedding vectors. This basically means using character n-grams to make up words. By default, n-grams of sizes between 3 and 6 are used to build all words. For example: 'he' could have an embedding, but 'she' is embedded as 's' + 'he'. By doing this, more words can be stored while keeping the model considerably small.

Every caption gets a unique START_SYMBOL as first token and sentences with more than 16 words are truncated. This is simple to fasten up the training. Also, all words are made non-capital. This is necessary, since it reduces the size of the vocabulary drastically, while the meaning of the sentences stays the same, in most of the cases. All captions are embedded in this fashion in a preprocessing step.

At training time the model gets two inputs: One embedded caption as a list of embedding vectors, where each word (or embedding) represents on timestep; the respective image vector.

The caption is fed into a LSTM layer with 1024 units that returns not only the output after the last timestep, but the full sequence of outputs after every timestep. This means the output of this LSTM layer is a 16x1024 matrix, representing the output of the LSTM at each timestep/word. The image vector is fed into a 512 unit dense layer for faster computation in the next step.

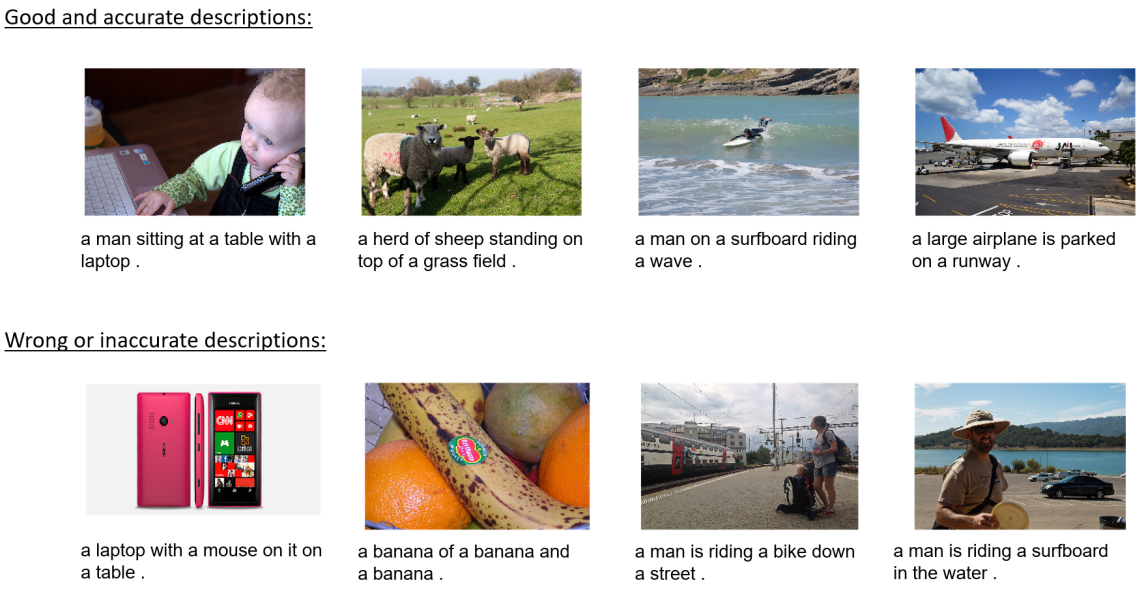


Figure 13: A selected sample of images and their captions, generated by the system described in Section 6.1.2

The new image vector, or the dense layer, is now replicated n times, where n is the number of timesteps (16 in this case) and appended once at each of the 16 LSTM output vectors. By doing this, it is ensured, that the image information is present at every prediction step. This step results in a 16x1536 (1024+512) input matrix.

This matrix is then fed into two stacked layers of 1024 LSTM cells. There is a 0.3 dropout layer after each LSTM layer. After that, there is one dense layer with 2048 units, followed by the categorization layer with v units, where v is the size of the vocabulary. Adam with default learning rate (0.001) was used as optimizer. The whole model has 41,128,591 parameters in total and is shown in Figure 12. The architecture of the model is inspired by Vinyals et al. (2014), and was adapted based on the authors experience and on empirical results.

The output of the network is a probability distribution over all possible words. This is achieved by using the softmax activation function on the last layer. The vocabulary is determined by using any word, that occurs at least 8 times in all of the 591.753 training captions. This yields a vocabulary size of 7820. The final output of the complete network is a 16x7820 matrix, representing 16 distributions over the vocabulary.

This matrix is then compared to the true caption. The true caption is the same sentence that went into the network, embedded, not with a dense vector embedding model, but hot one encoded within the 7820 vocabulary categories. The important difference between input and target is that the target sentence is shifted one step into the future. This is simply done by

only giving the input sentence a START_SYMBOL. Target sentences on the other hand have a END_SYMBOL added after the last word. This way input and target sentence have the same length while at the same time any word at position *t* in the target sentence can be found at position *t+1* in the input sentence. Figure 5 and Figure 6 both show this in slightly different abstraction levels.

In order to validate the system after training, the same process is iteratively repeated until the END_SYMBOL is produced by the system or the maximum sentence length is reached.

To start the process, the input sentence only contains the START_SYMBOL on the first position, and the rest of the sentence is padded. This sentence along with the image vector is fed into the network. The system now creates a probability distribution for the first token.

Now one word has to be chosen out of this distribution. This choice can be a simple greedy selection, where the token with the highest probability is chosen or a heuristic method like beam search can be used to find the best sequence of words. The token is then added after the START_SYMBOL (or the last generated token) and the new sentence is re-fed into the network.

6.2 Experiment

The system was trained for 50 epochs with a mini batch size of 64. Most of the literature advises to use 30-50 epochs for convergence and to use a batch size of 64 or 128. The complete dataset could not be loaded at once, since it was too big to keep in memory. Therefore, the data was partitioned into 20.000/8.000 (depending on the word embedding model) images, equal to 100.000/40.000 captions. Every partition was trained for 50 epochs, then the system was validated with images from the validation dataset and then the next partition was loaded and used for training. Training was done on a Geforce GTX 1080 and took around 15 hours without preprocessing. The whole system can be found on `https://github.com/fabudlx/Image2Sequence`.

6.3 Evaluation and Discussion of the Results

The goal of this experiment was (1) to create a baseline model that shows that Vec2Seq models can successfully describe images with captions and (2) to create a system that can be used to pretrain a RL-GAN system in further studies.

To validate if the system can satisfy these two conditions, a qualitative analysis of the generated descriptions of the images was done. Figure 13 shows a small selection of well

Table 2: The best scores for different models on partly different datasets. Results from this study show slightly lower scores, but the trend is similar to other works.

Model	dataset	BLEU-1	BLEU-2	BLEU-3	BLEU-4	CIDEr-D	METEOR	ROUGE-L
This work (2018)	COCO-val2017	0.634	0.467	0.326	0.216	0.809		0.469
Donahue et al.(2017)	COCO-val2014	0.714	0.543	0.402	0.297	0.889	0.242	0.524
Donahue et al.(2017)	COCO-test2014	0.895	0.804	0.695	0.585	0.934	0.335	0.678
Karpathy and Fei-Fei(2017)	COCO-test2014	0.828	0.701	0.566	0.446	0.692	0.280	0.603
Devlin et al.(2015)	COCO-test2014	0.907	0.819	0.710	0.601	0.937	0.339	0.680

and badly described images that should represent how well the system generates captions. From a subjective standpoint, good and mostly correct grammar can be observed. The fit of the description to the images is often accurate or at least the topic of the image is correctly described. In some cases the description is completely wrong and the generated sentence fits a different image. The occurrences of cases like this could be an evidence for an overfitted model that tries to detect the 'category' of the image and uses a similar sentence for each of these categories. The sentences still show some diversity, but their structure is mostly very similar. This was to be expected from a Vec2Seq model and can also be observed in similar studies (Kiros et al., 2014a, 2014b; Devlin et al., 2015; Donahue et al., 2017; Chen & Zitnick, 2015; Vinyals et al., 2014; Karpathy & Fei-Fei, 2017).

To further evaluate the model, a qualitative analysis with different measurements can be used. The official evaluation script (Chen et al., 2015) from MS COCO includes the scores for BLEU 1-4 (Papineni et al., 2002), CIDEr-D, METEOR, Rouge-L and SPICE. Using these scores, and comparing them to similar studies yields a clear numeric analysis. It should be noted here that most of these scores work on n-grams and do not automatically represent the quality of the generated sentence. The fact that the generated sentences can be compared to five ground truth sentences increases the validity of these scores being a good representation for quality of captions, but does still not make them perfect measurements. It should also be noted that a perfect BLEU score, for example, shows a copying of one of the ground truth sentences, which is not the goal of this study. Quoting Papineni et al.: "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."

Nonetheless, these scores offer a numeric way of analyzing and comparing different models and very low scores represent a high probability for wrong or unsuited captions. The results for the COCO 2017 validation dataset, produced with the official evaluation script (Chen et al., 2015) are reported in Table 2. The Meteor score could not be obtained

because of flaws in the evaluation script.

Almost all other studies use and report on the MS COCO 2014 validation dataset or only report on their scores on the official test dataset. Even though a direct comparison to the results of the 2014 dataset is not possible, the results from similar studies are also shown in Table 2. This has only been done to put the results of this study in some context.

It can be observed that the scores obtained in this study are not as high as in similar papers. The trend of the scores from this and from other studies are the same, however. This is due to very little hyperparameter tuning in this study and quite a lot of hyperparameter tuning in similar studies. All of the named studies took part in the MS COCO competition and adapted their models in order to maximize their scores. This study simply wants to support the claim of Vec2Seq models being able to create appropriate descriptions for images, but was not optimized for the scores of the MS COCO challenge.

The qualitative and quantitative analysis both show that Vec2Seq models can perform the image description task considerably well without human guidance in a end-to-end fashion. It is assumed that the same model can be used for the pretraining of more sophisticated models like the early described RL-GAN model (Section 5.3).

7 Conclusion

AID has been a highly researched topic for many years. Starting from heavily engineered systems that extract features in order to fill templates or to fuel n-gram models to produce the most likely sentences, to the use of CNNs and RNNs that need less or no hand crafted rules but instead make use of huge amounts of training data, to the most recent ideas of using GANs in order to create more diverse and natural descriptions.

This study shows that computer vision and understanding visual input is, to this date, one of the most interesting and challenging tasks in computer science. Huge amounts of human labeled data and the steady rise of computational power allow for more complex and heavier models to tackle the task. It is hard to define, when the task of visual description is fully solved, since there is no fixed measure to define the quality or the completeness of a description.

In the opinion of the author, Vec2Seq models offer a solid baseline model that can be used for many tasks and produces well usable descriptions or captions. The stiffness and similarity of the generated sentences and the sometimes wrongly generated descriptions however still leaves room for improvement. These improvements could be the introduction of more randomness to the Vec2Seq model, with the goal of increasing the diversity of its generated sentences. Adding a random input vector next the image and text vector could achieve this. The increase of dropout between layers could also result in more divers sentences. At the same time it would hinders the model from overfitting and trying to implicitly categorize the images. Trying out different pretrained CNN models and different word embedding models as well as adapting the architecture could also increase the performance of the system.

In order to further improve the naturalness and human-likeness of the generated descriptions, the recently proposed RL-GAN framework should be implemented and researched further. RL and GANs are two very active and new fields of research, especially regarding language generation. To this date only two studies support the claim of a combined RL-GAN system to produce more realistic sentence than Vec2Seq models. More studies to strengthen or weaken this claim are definitely needed.

References

Anderson, P., Fernando, B., Johnson, M., & Gould, S. (2016). SPICE: Semantic Propositional Image Caption Evaluation. In *European conference on computer vision 2016* (Vol. 9909, p. 382-398).

Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017). Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, *34*(6), 26-38. doi: 10.1109/MSP.2017.2743240

Bachman, P., & Precup, D. (2015). Data generation as sequential decision making. In *Advances in neural information processing systems* (pp. 3249–3257).

Bahdanau, D., Brakel, P., Xu, K., Goyal, A., Lowe, R., Pineau, J., ... Bengio, Y. (2016). An Actor-Critic Algorithm for Sequence Prediction. *Computing Research Repository*, *abs/1607.07086*. Retrieved from `http://arxiv.org/abs/1607.07086`

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. *Computing Research Repository*, *abs/1409.0473*. Retrieved from `http://arxiv.org/abs/1409.0473`

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

Bernardi, R., Cakici, R., Elliott, D., Erdem, A., Erdem, E., Ikizler-Cinbis, N., ... 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. Retrieved from `http://dl.acm.org/citation.cfm?id=3013558.3013571`

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, *5*, 135–146.

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

Chen, X., & Zitnick, C. L. (2015). Mind's eye: A recurrent visual representation for image caption generation. In *IEEE conference on computer vision and pattern recognition 2015* (p. 2422-2431). doi: 10.1109/CVPR.2015.7298856

Chollet, F. (2017a). A ten-minute introduction to sequence-to-sequence learning in Keras. Retrieved July 18, 2018, from `https://blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-in-keras.html`

Chollet, F. (2017b). Xception: Deep Learning with Depthwise Separable Convolutions. *IEEE*

Conference on Computer Vision and Pattern Recognition 2017, 1800-1807.

Chollet, F., et al. (2015). *Keras*. `https://github.com/fchollet/keras`. GitHub.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. In *Conference on neural information processing systems 2014 workshop on deep learning*.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2015). Gated Feedback Recurrent Neural Networks. In *Proceedings of the 32nd international conference on international conference on machine learning* (Vol. 37, pp. 2067–2075). Retrieved from `http://dl.acm.org/citation.cfm?id=3045118.3045338`

Dai, B., Fidler, S., Urtasun, R., & Lin, D. (2017). Towards Diverse and Natural Image Descriptions via a Conditional GAN. *IEEE International Conference on Computer Vision*, 2989-2998.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *Conference on computer vision and pattern recognition 2009* (p. 248-255).

Denkowski, M., & Lavie, A. (2014). Meteor Universal: Language Specific Translation Evaluation for Any Target Language. In *Proceedings of the ninth workshop on statistical machine translation*.

Devlin, J., Cheng, H., Fang, H., Gupta, S., Deng, L., He, X., ... Mitchell, M. (2015). Language Models for Image Captioning: The Quirks and What Works. In *Association for computer linguistics 2015, volume 2: Short papers* (pp. 100–105). Retrieved from `http://aclweb.org/anthology/P/P15/P15-2017.pdf`

Donahue, J., Hendricks, L. A., Rohrbach, M., Venugopalan, S., Guadarrama, S., Saenko, K., & 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. Retrieved from `https://doi.org/10.1109/TPAMI.2016.2599174` doi: 10.1109/TPAMI.2016.2599174

Farhadi, A., Hejrati, M., Sadeghi, M. A., Young, P., Rashtchian, C., Hockenmaier, J., & Forsyth, D. (2010). Every Picture Tells a Story: Generating Sentences from Images. In *Proceedings of the 11th european conference on computer vision: Part iv* (pp. 15–29). Springer-Verlag. Retrieved from `http://dl.acm.org/citation.cfm?id=1888089.1888092`

Gilman, R. (2018). *Intuitive RL: Intro to Advantage-Actor-Critic (A2C)*. Retrieved July 18, 2018, from `https://hackernoon.com/intuitive-rl-intro-to-advantage-actor-critic-a2c-4ff545978752`

Glover, J. (2016). *An introduction to Generative Adversarial Networks*. Retrieved July 18,

2018, from http://blog.aylien.com/introduction-generative-adversarial-networks-code-tensorflow/

Goodfellow, I. (2016). *Generative Adversarial Networks for Text.* Retrieved July 18, 2018, from https://www.reddit.com/r/MachineLearning/comments/40ldq6/generative_adversarial_networks_for_text/

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems* (pp. 2672–2680).

Haykin, S. (1994). *Neural networks: a comprehensive foundation.* Prentice Hall PTR.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. In *Conference on computer vision and pattern recognition* (pp. 770–778).

He, X., & Deng, L. (2017). Deep Learning for Image-to-Text Generation: A Technical Overview. In *IEEE signal processing magazine* (Vol. 34, p. 109-116).

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. Retrieved from http://dx.doi.org/10.1162/neco.1997.9.8.1735 doi: 10.1162/neco.1997.9.8.1735

Hodosh, M., Young, P., & Hockenmaier, J. (2013). Framing Image Description As a Ranking Task: Data, Models and Evaluation Metrics. *Journal of Artificial Intelligence Research*, *47*(1), 853–899. Retrieved from http://dl.acm.org/citation.cfm?id=2566972.2566993

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

Jaimes, A., & fu Chang, S. (2000). A Conceptual Framework for Indexing Visual Information at Multiple Levels. In *Proceedings of SPIE internet imaging 2000* (pp. 2–15).

Karpathy, A., & Fei-Fei, L. (2017). Deep Visual-Semantic Alignments for Generating Image Descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(4), 664–676. Retrieved from https://doi.org/10.1109/TPAMI.2016.2598339 doi: 10.1109/TPAMI.2016.2598339

Kiros, R., Salakhutdinov, R., & Zemel, R. (2014a). Multimodal Neural Language Models. In E. P. Xing & T. Jebara (Eds.), *Proceedings of the 31st international conference on machine learning* (Vol. 32, pp. 595–603). PMLR. Retrieved from http://proceedings.mlr.press/v32/kiros14.html

Kiros, R., Salakhutdinov, R., & Zemel, R. S. (2014b). Unifying Visual-Semantic Embeddings with Multimodal Neural Language Models. *Computing Research Repository*, *abs/1411.2539*. Retrieved from http://dblp.uni-trier.de/db/journals/corr/

corr1411.html#KirosSZ14

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems 25* (pp. 1097–1105). Retrieved from http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

Kulkarni, G., Premraj, V., Dhar, S., Li, S., Choi, Y., Berg, A. C., & Berg, T. L. (2011). Baby talk: Understanding and generating image descriptions. In *Conference on computer vision and pattern recognition 2011* (p. 1601-1608). doi: 10.1109/CVPR.2011.5995466

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation, 1*, 541-551.

Li, S., Kulkarni, G., Berg, T. L., Berg, A. C., & Choi, Y. (2011). Composing Simple Image Descriptions Using Web-scale N-grams. In *Proceedings of the 15th conference on computational natural language learning* (pp. 220–228). Retrieved from http://dl.acm.org/citation.cfm?id=2018936.2018962

Liang, X., Hu, Z., Zhang, H., Gan, C., & Xing, E. P. (2017). Recurrent Topic-Transition GAN for Visual Paragraph Generation. In *International Conference on Computer Vision* (pp. 3382–3391).

Lin, C.-Y. (2004). ROUGE: A package for automatic evaluation of summaries. In *Association for computational linguistics* (p. 10). Retrieved from http://research.microsoft.com/~cyl/download/papers/WAS2004.pdf

Luong, M., Brevdo, E., & Zhao, R. (2017). Neural Machine Translation (seq2seq) Tutorial. *https://github.com/tensorflow/nmt*.

Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., & Joulin, A. (2018). Advances in pre-training distributed word representations. In *Language resources and evaluation 2018*.

Mirza, M., & Osindero, S. (2014). Conditional Generative Adversarial Nets. *Computing Research Repository, abs/1411.1784*. Retrieved from http://arxiv.org/abs/1411.1784

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., ... Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning* (pp. 1928–1937).

Ordonez, V., Kulkarni, G., & Berg, T. L. (2011). Im2Text: Describing Images Using 1 Million Captioned Photographs. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems 24* (pp. 1143–

1151). Retrieved from http://papers.nips.cc/paper/4470-im2text-describing-images-using-1-million-captioned-photographs.pdf

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th annual meeting on association for computational linguistics* (pp. 311–318). Retrieved from https://doi.org/10.3115/1073083.1073135 doi: 10.3115/1073083.1073135

Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *Computing Research Repository, abs/1511.06434*.

Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., & Lee, H. (2016). Generative Adversarial Text to Image Synthesis. In M. F. Balcan & K. Q. Weinberger (Eds.), *Proceedings of the 33rd international conference on machine learning* (Vol. 48, pp. 1060–1069). PMLR. Retrieved from http://proceedings.mlr.press/v48/reed16.html

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in neural information processing systems 28* (pp. 91–99). Retrieved from http://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-networks.pdf

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

Shatford, S. (1986). Analyzing the Subject of a Picture: A Theoretical Approach. *Cataloging & Classification Quarterly, 6*, 39 - 62. Retrieved from http://www.informaworld.com/10.1300/J104v06n03_04

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

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. In *Proceedings of the 27th international conference on neural information processing systems - volume 2* (pp. 3104–3112). MIT Press. Retrieved from http://dl.acm.org/citation.cfm?id=2969033.2969173

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction* (Vol. 1) (No. 1). MIT press Cambridge.

Sutton, R. S., McAllester, D., Singh, S., & Mansour, Y. (1999). Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Proceedings of the 12th international conference on neural information processing systems* (pp. 1057–1063). MIT

Press. Retrieved from http://dl.acm.org/citation.cfm?id=3009657.3009806

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the 31st conference on artificial intelligence 2017* (pp. 4278–4284). Retrieved from http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14806

Tesauro, G., & Galperin, G. R. (1997). On-line policy improvement using Monte-Carlo search. In *Advances in neural information processing systems* (pp. 1068–1074).

Vedantam, R., Zitnick, C. L., & Parikh, D. (2015). CIDEr: Consensus-based image description evaluation. In *Conference on computer vision and pattern recognition* (p. 4566-4575). IEEE Computer Society. Retrieved from http://dblp.uni-trier.de/db/conf/cvpr/cvpr2015.html#VedantamZP15

Vinyals, O., & Le, Q. V. (2015). A Neural Conversational Model. *Computing Research Repository, abs/1506.05869*. Retrieved from http://dblp.uni-trier.de/db/journals/corr/corr1506.html#VinyalsL15

Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2014). Show and Tell: A Neural Image Caption Generator. *Computing Research Repository, abs/1411.4555*. Retrieved from http://dblp.uni-trier.de/db/journals/corr/corr1411.html#VinyalsTBE14

Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. *Machine Learning*, *8*(3), 279–292. Retrieved from https://doi.org/10.1007/BF00992698 doi: 10.1007/BF00992698

Williams, R. J. (1992). Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, *8*(3-4), 229–256. Retrieved from https://doi.org/10.1007/BF00992696 doi: 10.1007/BF00992696

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

Yang, Y., Teo, C. L., Daumé, H., III, & Aloimonos, Y. (2011). Corpus-guided Sentence Generation of Natural Images. In *Proceedings of the conference on empirical methods in natural language processing* (pp. 444–454). Association for Computational Linguistics. Retrieved from http://dl.acm.org/citation.cfm?id=2145432.2145484

Yu, F. (2017). *Deep Q Network vs Policy Gradients - An Experiment on VizDoom with Keras*. Retrieved July 18, 2018, from https://flyyufelix.github.io/2017/10/12/dqn-vs-pg.html

Yu, L., Zhang, W., Wang, J., & Yu, Y. (2016). SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. *Computing Research Repository, abs/1609.05473*. Retrieved from http://dblp.uni-trier.de/db/journals/corr/corr1609.html#YuZWY16

Zeiler, M. D., & Fergus, R. (2014). Visualizing and Understanding Convolutional Networks. In

D. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars (Eds.), *European conference on computer vision 2014* (pp. 818–833). Springer International Publishing.

Zhang, L., Sung, F., Liu, F., Xiang, T., Gong, S., Yang, Y., & Hospedales, T. M. (2017). Actor-critic sequence training for image captioning. *Computing Research Repository, abs/1706.09601*. Retrieved from `http://arxiv.org/abs/1706.09601`