Image Captioning

Severin Hußmann, Simon Remy, Murat Gökhan Yigit
Seminar Information Systems WS 17/18
Humboldt University Berlin
severin.hussmann@hu-berlin.de
simon.remy@hu-berlin.de
murat.goekhan.yigit@hu-berlin.de

Abstract

Image Captioning becomes an increasingly popular discipline in machine learning. It can support visually impaired people, improve search engines or facilitate online marketing. There are several proposals introducing better and better performing models. This paper aims to offer the reader elementary background knowledge about the field and an instruction how to develop a simple image caption model by herself. After evaluating different approaches how to build the model architecture we provide the blueprint of the most prominent one, explain it in detail, exemplify how to train it on the Flickr8k dataset, evaluate it and discuss possible enhancements. For a non optimized model the results show that it can adequately keep up with the state of the art models.

1 Introduction

Image captioning aims for automatically generating a text that describes the present picture. In the last years it became a topic with growing interest in machine learning and the advances in this field lead to models that can score even higher than humans do [47]. But how does this research field improve our daily lives? Image captioning can for instance help visually impaired people to grasp what is happening in a picture. Furthermore, it could enhance the image search of search engines, simplify SEO by automatically generating descriptions for pictures or improve online marketing and customer segmentation by identifying customer interests through interpreting their shared images via social media platforms. Nevertheless, image captioning is a very complex task as it goes beyond the sole classification of objects in pictures. The relation between objects and attributes have to be recognized. Finally, these information must be expressed in a natural language like English [10]. Beyond that, this challenge is also interesting because it links the fields computer vision and natural language processing with each other which are both two major fields in machine learning [49].

Our paper aims at offering an academic support of our Image Captioning blog entry[1] in the context of the "Information Systems" seminar at the Humboldt University of Berlin. The goal of this blog and paper is an introduction to image captioning, an explanation of a comprehensible model structure and finally a tutorial to implement that model. We focus on the basic understanding of the topic and the application of a basic model rather than generating a fine tuned model to compete with the state of the art models.

There are several approaches in theory and practice how to design an image caption generation model. We follow the commonly used approach of an encoding convolutional neural network (CNN) in combination with a recurrent neural network (RNN) or, more precisely a long short-term memory net (LSTM). Encoding CNN means that the neural network is used for the generation of a meaningful representation of the input image to an internal fixed-length representation. For that purpose we use a pre-trained CNN for image classification and removed the last layer to obtain image features. The

[1] https://humboldt-wi.github.io/blog/research/seminar/07imagecaptioning/

LSTM processes the linguistic information of the image captions. This approach is well documented and researched in a broad amount of literature, blogs or tutorials and therefore serves very well as a first step into automatically generating descriptions for images [9, 46, 44, 35, 11, 22, 14, 34].

Our paper is structured as follows. First, we introduce our methodology in section 2 by explaining the basics of CNNs, RNNs, how to approach the right model architecture and the BLEU score. Subsequently, in our approach in section 3 we explain our dataset and our final architecture. In section 4 we present our results, discuss our model critically and point out possible improvements. Finally, section 5 concludes our paper with a brief summary.

2 Methods

In this section we introduce our used methods for the underlying neural network structures, our architecture and our evaluation model. These are in the following divided into the subsections CNN, RNN and LSTM, Model Architecture and BLEU score.

2.1 CNN

In an artificial neural network (ANN) input is passed through a net of neurons which typically outputs a classification or regression of input data. Neurons are designed as functions that need a certain threshold to pass information through to the next neuron or the output layer. These neurons are trained in an iterative process by adjusting the functions' weights through backpropagation. In backpropagation errors from the classification are fed back into the network in order to improve performance in the next iteration. This step is repeated until sufficiently satisfying results are achieved. Thus, neural networks identify patterns in a progressive way [29].

From the simple feedforward neural network, many complex variations have been developed. One of them is Convolutional Neural Network (CNN), that is commonly used in the field of visual pattern recognition and has been widely popular and successful in the past years [42]. Visual patterns in images are recognized in a way that local combinations of edges form motifs, motifs assemble into parts and parts into objects [30]. The typical goal of a CNN in a visual analysis is to classify these images. CNNs have a multistage architecture and consist of an input layer, hidden layers and an output layer. The hidden layers are either convolutional, pooling or fully-connected [29]. There are several successful CNN architectures, like AlexNet, ZF Net or GoogLeNet, which provide superior results in image classification [26, 43, 50]. One of them is the VGG Oxford Net (i.e. VGG16), which was developed by Simonyan and Zissermanwhich, that we are using in our approach [40]. Although VGG 16 didn't perform as well as the GoogLeNet in the ImageNet competition 2014, the generalization, comprehensibility, simplicity and accessibility of the model makes it suitable as a basis for our work [2].

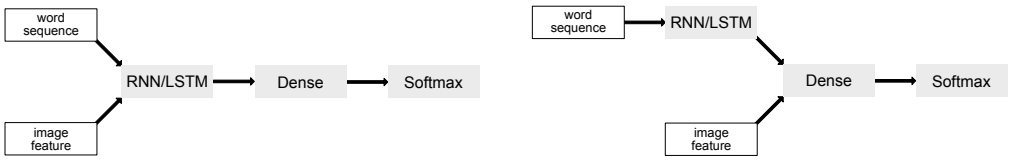
2.2 RNN and LSTM

Another extension of ANN is Recurrent Neural Network (RNN). RNN can handle variable length of input by having a recurrent hidden state whose activation on each time is dependent on that of the previous time [12]. Therefore, recurrent networks are very powerful in their ability to represent context and outperforming static networks. Nevertheless, Bengio et al. have observed, that RNNs are hard to train to capture long term dependencies [8]. Long Short-term Memory (LSTM) is an RNN architecture, which uses purpose-built "memory cells" to store information. This seems to be better at finding and exploiting long range dependencies in the data [17]. Both RNNs and LSTMs are used in the fields of speech and text recognition.

2.3 Model Architecture

LeCun et al. define a deep-learning architecture as a stack of multiple simple modules. Most of these modules map non-linear inputs to outputs and are part of the learning process, whereas others only decrease the data dimensions like pooling layers [30]. Whenever developing an ANN the overall goal is to find an architecture with consistently good performance [7]. P.G. Bernardos and G.-C. Vosniakos point out that the most common method to find a good architecture is a repetitive trail-and-error process. In this process many different architectures are examined and compared to each other.

This approach is very time-consuming and requires human knowledge based on past experience and intuition [7]. However, the authors describe four different general approaches to find best performing architectures in a more systematic way: (i) use empirical and statistical methods to analyze an ANN's internal parameter and choose the best performing one; (ii) interpret ANNs as an adaptive fuzzy system, like fuzzy inference [32]; (iii) use pruning algorithms to add or remove neurons from an initial network and evaluate the ANN's performance with previously specified criterions [15]; (iv) build topology space by varying the number of hidden layers and hidden neurons and use genetic algorithms to evaluate the different architectures [5].
Recent works follow the last approach. They automate the process of architecture engineering by learning a model architecture directly on the dataset [52]. As an example the Neural Architecture Search (NAS) framework has to be named, which was developed by Zoph and Le. The core of the framework is a controller RNN which samples child networks that are trained on a given dataset. The resulting accuracy can then be used to update the controller using policy gradient. This way the controller generates better architectures over time [51]. This approach was developed and tested for image classification but can also be applied to other problems.
In contrast, Tanti et al. focused on models that are specifically used for image captioning. They follow the first approach defined by Bernardos and Vosniakos and point out that in recent work many different model architectures were developed but have not been systematically compared yet. In order to do so, Tanti et al. use a standard dataset for image captioning and compared several quality metrics like (i) probability metrics which indicate how well an architecture predicts words for a given image feature and text prefix; (ii) caption quality by comparing the similarity of a generated caption and its true value using standard methods like the BLEU score [38]; (iii) diversity of the used vocabulary and; (iv) the ability to find the correct image to a given caption. Additionally, they recommend not to optimize hyperparameters of each architecture in order to not give one architecture an advantage over the others, because it is difficult to find one optimal set of hyperparameters for all models. In total, they compared 16 different architectures in their experiment [44].
As previously mentioned many different architectures have been developed in recent research. Most of them are inject type architectures. Besides that there also exist merge architectures, both types will be explained in the following and how they are compared by Tanti et al.. Regardless of the specific architecture, RNNs and LSTMs are used. As already mentioned in section 2.2 these networks perform very well for text prediction.
A high-level distinction between inject and merge architectures can be made by the point in time at which image features are included in the model. In general, merge architectures integrate image features after the caption prefix is processed. In contrast to that, most of the inject architectures process image features before the last word of the prefix is processed. In other words, in any inject architecture the image information always influences the textual feature processing.



(a) inject architecture (b) merge architecture

Figure 1: General structure of inject and merge architectures as proposed by Tanti et al. [44].

In literature, inject architectures (see Figure 1a) are divided into four different types, which again differ in the point of time at which image informations are passed to the model as described below:

- *Init-inject*: The image feature vector is used to initialize the RNN or LSTM.
- *Pre-inject*: The image feature vector is injected into the model before the caption prefix, so that the image information is treated like the first word of the prefix.
- *Par-inject*: The image feature vector and the caption prefix are fed into the model at the same time. They can either be combined or serve as two independent inputs.
- *Post-inject*: The image feature vector is fed to the model after the caption prefix and will be treated like the last word of the prefix.

In contrast, merge architectures (see Figure 1b) combine image information and caption prefix after processing the latter one. The merge step can be done in three ways:

- *merge-concat*: Concatenate the image feature vector and the caption to one vector. The resulting vector has the length of the sum of length of the input vectors.
- *merge-add*: Add element-wise image feature vector and caption prefix vector together. The resulting vector has the same dimensions as the input vectors.
- *merge-mult*: Multiply element-wise the image feature vector and caption vector together. The resulting vector has the same dimensions as the input vectors.

A clear downside of the concatenate approach is the increase of dimensions of the resulting vector, which results in more complex layer and requires more computational power in the end. It is also possible to combine both approaches in a mixed model. A more detailed overview can be found in [44].

Tanti et al. group different architectures with respect to the binding time into two groups: early binding architectures with init-inject and pre-inject and late binding architectures with merge and post-inject. They came to the conclusion that models with late binding architectures perform better than models with early binding architectures for all metrics. Overall *merge-add* performs best, even slightly better than *merge-mult* which has the highest standard deviation in median perplexity. In the end all architectures have almost the same word frequency distribution. No model used more than 7% of the known vocabulary. For more details on other metrics see Tanti et al. [44].
The authors came to the conclusion that keeping text informations and image informations apart results in the best performance. A possible explanation to this may be the fact that treating the image feature as a word in early binding architectures would increase the vocabulary size. Based on this results we will implement a merge based architecture in the following as described in section 3.2.

2.4 BLEU Score

In order to evaluate the quality of our model we apply the BLEU score. BLUE stands for *bilingual evaluation understudy* and was developed by Papineni et al. to improve the evaluation of machine translations [38]. The manual human evaluation could take up several months and thus they developed an automatic, fast, language-independent and high correlated to human evaluation approach to access the translation quality. The central idea behind BLEU is that "the closer a machine translation is to a professional human translation, the better it is" [38]. That means we need to measure the closeness of the candidate translation to the reference translation. Measuring this closeness works by means of comparing the n-grams of the candidate to the n-grams of the reference sentence. In the following we explain the modified n-grams scores on the basis of our example candidate and reference sentences:

- `Candidate 1`: The dog is running over the green field.
- `Candidate 2`: The cat is running over the blue field.
- `Reference`: The dog is running over the green field.

	Unigram: 1-gram	Bigram: 2-gram	Trigram: 3-gram	Tetragram: 4-gram
Candidate 1	$8/8 = 1$	$7/7 = 1$	$6/6 = 1$	$5/5 = 1$
Candidate 2	$6/8 = 0.75$	$3/7 = 0.43$	$2/6 = 0.33$	$1/5 = 0.2$

Table 1: n-gram scores of example sentences

Modified n-grams means that repetitions are not taken into account, i.e. the sentence *"the the the the the the the the"* would result in a unigram score of $2/8 = 0.25$ instead of 1. Furthermore, the position of the n-gram is not of importance. The different n-gram scores account to different aspects of the the candidates. A higher score of the 1-gram stands for a better adequacy and the 2, 3 and 4-grams scores account for better fluency. BLEU combines the evaluation of the different aspects by using the geometric mean of the modified n-gram scores. In our example sentences this looks as follows:

Longer or shorter candidate sentences would be problematic as they would score higher. Longer candidate sentences are penalized by the modified n-grams. In order to handle shorter candidate

	Candidate 1	Candidate 2
BLEU 1	$1 \cdot \frac{8}{8} = 1$	$1 \cdot \frac{6}{8} = 0.75$
BLEU 2	$0.5 \cdot \frac{8}{8} + 0.5 \cdot \frac{7}{7} = 1$	$0.5 \cdot \frac{6}{8} + 0.5 \cdot \frac{3}{7} = 0.59$
BLEU 3	$0.33 \cdot \frac{8}{8} + 0.33 \cdot \frac{7}{7} + 0.33 \cdot \frac{6}{6} = 1$	$0.33 \cdot \frac{6}{8} + 0.33 \cdot \frac{3}{7} + 0.33 \cdot \frac{2}{6} = 0.50$
BLEU 4	$0.25 \cdot \frac{8}{8} + 0.25 \cdot \frac{7}{7} + 0.25 \cdot \frac{6}{6} + 0.25 \cdot \frac{5}{5} = 1$	$0.25 \cdot \frac{6}{8} + 0.25 \cdot \frac{3}{7} + 0.25 \cdot \frac{2}{6} + 0.25 \cdot \frac{1}{5} = 0.43$

Table 2: BLEU scores of example sentences

sentences, BLEU makes use of a multiplicative brevity penalty factor (BP). When the candidate's sentence/ corpus length c is smaller or equals to the length of a reference sentence/ corpus r, i.e. $c \leq r$, the geometric average of the modified n-gram precisions is multiplied with $e^{(1-r/c)}$. When there are multiple reference sentences, r is set to the closest reference sentence length, the *"best match length"*.

The BLEU score ranges from 0 to 1, whereas it is unlikely to score 1, because this means that the candidate is identical to the reference. It is important to emphasize that the more reference sentences there are for a candidate sentence, the higher the score is.

3 Approach

In the following, we will describe the dataset on which we used to train and test the model. Subsequently, we explain how we developed the model and how to generate the training data from the dataset in order to train the model.

3.1 Dataset

There is a wide variety of datasets to choose from when training a model for image captioning. In this section, we describe which factors determined our choice in the dataset and which properties are important in such a choice.

In order to be able to train the model, it is required to have a description of an image. We did not choose one of the datasets that have a description, which is generated automatically or harvested from the web [16]. On the one hand, automatically generated descriptions tend to not understand the underlying image well enough to use them as a basis for predicting new captions [28]. In order to evaluate such captions expensive human judgments to consider the quality of content selection is required. This is important to answer the question, whether the image is correctly described by the caption. On the other hand, descriptions that are harvested from the web do not describe what is depicted in the image but rather give additional information that can not be seen in the image. This does not lead to descriptions, that can be used to train a model also with regard to the Gricean maxims of relevance and quantity [37, 18].

Therefore, the dataset of choice needed to be one with images that are described by actual people. In this context we found three datasets, that fulfill this requirement. The first one is the IAPR-TC12 dataset from Grubinger et al. (2006). This dataset with 20000 images contains descriptions of what can be recognized in an image. Unfortunately, it has too detailed descriptions with an average of 23 words, which contains irrelevant elements for the image understanding. This leads to confusing caption generation and is thus not suitable for our work [19]. Another dataset, MSCOCO, with 330,000 images is also a good option to consider when setting up a complex fine tuned model, but for our simple model case, the Flickr 8k dataset is sufficient. [1]

The dataset Flickr8k by Hodosh et al. (2013) has shorter descriptions with an average length of 12 words per descriptions than the IAPR-TC12. It provides five different descriptions per image of the noticeable events and entities of an image. The multiple descriptions lead to a considerable degree of variance in the way many images can be described as it can be seen in the Figure 2. This leads to qualitatively better results because of different perspectives and interpretations through different people. The people describing the images were told to describe the people, objects, scenes and

1. A man is doing tricks on a bicycle on ramps in front of a crowd.
2. A man on a bike executes a jump as part of a competition while the crowd watches.
3. A man rides a yellow bike over a ramp while others watch.
4. Bike rider jumping obstacles.
5. Bmx biker jumps off of ramp.

Figure 2: Example image of Flickr8k with its five descriptions [20]

activities, that are shown in an image without having any further information about the context in which they were taken. The result were conceptual descriptions that focus only on the information that can be obtained from the image alone. While other description methods (e.g. perceptual description) capture low level properties of images or non-visual descriptions, which describe attributes that cannot be seen in the image, conceptual descriptions identify what is depicted in the image [21]. Since we are not interested in low-level attributes and also not in low-level properties, conceptual image descriptions are relevant for our work. While things in the images may be abstract, image understanding is mostly interested in concrete descriptions of the depicted scenes and entities, their attributes and relations as well as the events they participate in. Also the Flickr8k dataset has descriptions that are not too specific but rather generic in order to be able to generate a broad variety of captions. The dataset contains 8092 images of people and animals (mostly dogs) performing some action [20].

3.2 Image Caption Generation Model

As already mentioned in section 2.3, we will develop our model based on the study of Tanti et al.. Due to the fact that merge architectures are superior to inject architectures, we are going to implement a model based on this structure. We will also use a LSTM as part of our model since it seems to perform better than normal RNNs [44].
We use the functionality provided by the *Keras* package (version 2.0 or higher) with either TensorFlow or Theano backend and Python 3.5.

3.2.1 Model Structure

Our model takes two separate vectors as inputs: (i) image feature vector and (ii) word sequence vector. In section 3.2.2 we will explain in more detail how to generate the latter one.
The image feature vector needs to be generated for each image. To do so we use the VGG16 to perform object recognition [40]. The CNN has 19-layers and was initially trained on the ImageNet dataset [13]. Since we are not interested in image classification we removed the last layer of the network, which is a soft-max layer. As a result, the model returns a 4,096-element vector. Because of this, the first input layer needs the shape of 4,096 nodes. The VGG CNN will always return vectors with this size, therefore we do not need to care about changing dimensions at this point.
Srivastava et al. have shown in their research that random drops of a certain rate of the data in a neural net helps to avoid overfitting [41]. Tanti et al. report that applying a dropout directly to the

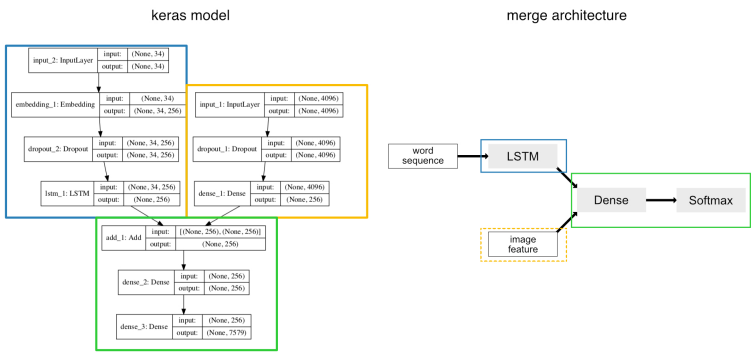


Figure 3: Comparison of model implementation in Keras and theoretic architecture.

input have a better effect than applying it to a layer of a higher level [44]. Because of this, a dropout layer with a dropout rate of 50% is applied to the input vector. In order to reduce the dimensions of the data, we apply a dense layer in the next step with a Rectified Linear Unit (ReLU) activation function. In our case, we reduce the feature vector with 4,096 elements to a vector with 256 elements. This previously described part corresponds to the yellow part in Figure 3.
The second input of our model takes a word sequence vector. Its size depends on the number of words of the longest possible caption which the model shall be able to predict in the end. At the same time, a lower bound n is given by the longest caption in the dataset. Therefore, a convenient way seems to be to set the size of the second input to n, the number of words of the longest caption in the dataset. As in section 3.1 explained, a good caption should be as short and precise as possible. Because of this, it would not make sense to train the model to predict longer captions as in the dataset, which represents the ground truth. The longest caption in the given training set has $n = 34$ words.
Nevertheless, our model should allow to generate shorter captions and needs to be able to handle word sequences which have less than n words. At the same time, each layer in a neural net has a fixed number of input nodes, which requires always an input of the same length. As a solution to this, shorter sequences are padded so that all are of the same length. We will explain this in more detail in section 3.2.2. In order to handle padded sequences, we add an embedding layer to the next step. This way we can tell the model to ignore the padding we just added to the sequence since it does not contain any information for the model to learn. Since the embedding layer can only be used as the first layer in a model in Keras, a dropout layer will be applied to its output and not directly to the model input as previously explained. Again a dropout rate of 50% is used to avoid overfitting. The last layer in this part of the model is an LSTM layer with 256 cells. This part of the model is depicted in the blue part of Figure 3.
After the word sequence vector got processed by the LSTM layer and the image feature vector also got compressed to the same dimensions, they get merged. This takes place in the first layer of the green part in Figure 3, which is a simple add layer. The layer adds both vectors element-wise into a single vector by preserving the original dimensions of the input vectors. Afterwards, the vector passes another dense layer with a ReLU activation function. The last layer in the model is also a dense layer. Instead of a ReLU function it uses a softmax function in order to create a probability distribution over the complete vocabulary. In other words, it predicts the word which has the highest probability to follow the given word sequence in combination with the image feature.
As suggested in literature, we use a categorical cross-entropy cost function, since our target variable is in categorical format (see 3.2.2 for details) as well as Adam as an optimizer method [44, 24].

3.2.2 Training Data Generation

In the following, we will explain how to generate a training set from the given dataset for our model. In general, neural networks work in a way that they learn to map some data X to some target Y, e.g. a label (in case Y is known in the training data, this is known as supervised learning). In our case X consists of two parts, X_1 the image data and X_2 the captions i.e. word sequences. As our goal is not to map an image to a specific caption but rather learn the relationship between image features and word sequences and between word sequences and single words our target Y is not a complete caption

in the dataset but a single word. Or, put another way, the model generates a new caption word by word based on a given image feature and a caption prefix. Thus, the caption generation process is iterative. After a new word got predicted by the model it will be appended to the existing prefix and it is fed into the model again. This raises two questions (i) how to choose the first word of a caption and (ii) how to determine if a caption is complete?
Both problems can be solved by adding an artificial start and end sequence to each caption. It is important that these sequences are not part of the original image captions in the dataset. An example is shown in Table 3 below.

original caption:	The dog is running over the green field
wrapped caption:	<start> The dog is running over the green field <end>

Table 3: Caption after wrapped into artificial start and end sequence.

To be able to learn the relationships between caption prefix, image feature and single word each caption has to be split up as illustrated in Table 4. The first column contains image feature vectors (X_1), the second column caption prefixes (X_2) and the last column contains the next word after the corresponding caption prefix in each row.

X_1	X_2	Y
vec(4,096)	0 0 0 0 0 0 0 0 <start>	the
vec(4,096)	0 0 0 0 0 0 0 <start> the	dog
⋮	⋮	⋮
vec(4,096)	<start> the dog is running over the green field	<end>

Table 4: Illustration of the training data generation process with padding = 8, before one-hot encoding and numeric transformation. Each row symbolizes one iteration in the model.

As explained in section 3.2.1, each caption has to be of the same length. Because of this, a zero padding will be added to each caption prefix which has fewer words than the number of nodes of the input layer. Due to the fact, that neural nets can only handle numerical inputs the caption prefixes (X_2) and the target variable (Y) have to be encoded. To encode the prefixes a tokenizer can be applied to the vocabulary. In order to do so, the tokenizer has to be fitted on the vocabulary, which works the following way: all captions get split up into single words (token). Then the tokenizer reads in the stream of tokens and any time it reads a new token, a counter will be increased and its number will be mapped to the corresponding token. After the tokenizer has read each word in the vocabulary, it can translate any word into an integer. Considering we add a zero padding to the prefix, it is important that the first word is mapped to 1 and not 0.
Encoding the target variable is more complex. As described before, the model should predict a probability for each word in the vocabulary to follow a given prefix. As a result, the model outputs a vector r with each element v as a probability of one word and n the total number of words in the vocabulary:

$$r = \{v_1, v_2, ..., v_n\}$$
$$\text{with} \sum_{i=1}^{n} v_i = 1$$

Therefore, we need a specific probability distribution for the vocabulary at each step in the learning process. In other words, the model should learn that a specific prefix has a 100% chance to be followed by a chosen word. This can be encoded using one-hot encoding. In the probability vector r each index corresponds to one word in the vocabulary. To teach the model that the j'th word should

follow a given prefix p, the target variable Y_{pj} can be in general expressed as:

$$Y_{pj} = \{v_1, v_2, ..., v_n\} \qquad \forall v_{i \neq j} \in Y_{pj} = 0$$
$$v_{i=j} = 1$$
$$\text{with } i = \{1, ..., n\}$$

In the end, each cell in the table contains a vector which is either an image feature, a transformed caption prefix or one-hot encoding of the next word in the caption.

3.2.3 Caption Generation Process

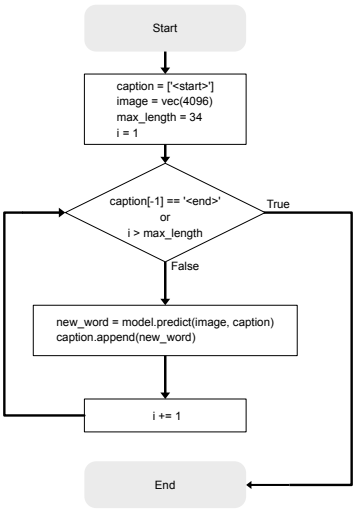


Figure 4: Iterative caption generation process

As described in the previous section and depicted in Figure 4, the caption generation process is an iterative one. In each iteration, one word will be predicted and appended to the caption prefix. To kick off this process the previously defined start sequence and an image feature will be passed to the model. The process will either end by predicting the artificial end sequence as the next word or by reaching the maximum number of words per caption. This upper bound is given by the number of nodes in the input layer for the caption prefix.

4 Results and Discussion

In this section we present the results of our *Simple-Merge* model and critically review our approach. As mentioned before the BLEU score was developed to automatically evaluate the quality of machine translations [38]. As the closeness of a candidate translation to a reference translation is measured this metric can also be used to evaluate the generation of image captions. Here, we compare the closeness of a generated image caption to the original caption. By means of the NLTK library in Python we calculated the BLEU scores for our Simple-Merge model on the Flickr8k testset [31]. Although the scores can not compete with state of the art image captioning results they come at least considerably close for a non-optimized model as can be seen in Table 5.

On other datasets such as Flickr30k and MSCOCO, other models score even higher on the BLEU score. On MSCOCO the top models score around as follows [1]:

- `BLEU 1`: 0.80
- `BLEU 2`: 0.63
- `BLEU 3`: 0.48
- `BLEU 4`: 0.36

	BLEU 1	BLEU 2	BLEU 3	BLEU 4
Simple-Merge	0.53	0.29	0.16	0.09
mRNN [35]	0.57	0.39	0.26	0.170
Google NIC [46]	0.63	0.41	0.27	-
Log Bilinear [25]	0.66	0.42	0.28	0.18
Soft-Attention [48]	0.67	0.45	0.30	0.20
Hard-Attention [48]	0.67	0.46	0.31	0.21
Fine-Grained Attention [10]	0.69	0.48	0.34	0.24

Table 5: BLEU scores tested on Flickr8k of Simple-Merge compared to other models

Besides the BLEU score there are also other evaluation metrics such as METEOR, ROUGE, CIDEr, SPICE or WMD [23]. Choosing a metric is a fundamental decision as it is not only important when comparing models with each other but also for developing new models [23]. Hence, it is important to critically review our metric of choice. BLEU has several deficiencies such as recall evaluation and the lack of explicit word matching. It fails to detect semantic similarity when dealing with scarce common words. METEOR handles this issue by introducing synonym matching [6]. Whereas BLEU or METEOR were developed to evaluate machine translation, CIDEr and SPICE where developed to evaluate image description generation. CIDEr introduces tf-idf weighted n-grams similarity [45] and SPICE a scene-graph synonym matching [3]. Word Mover's Distance WMD is a document distance measure that deals with the two problems that firstly captions can be similar although they do not share the same words or synonyms and secondly captions can describe similar objects although they may not be semantically similar. WMD makes use of the Earth Mover Distance on word2vec [27, 39, 36]. All these evaluation metrics are significantly different and there are even different versions and implementations for some of them. Although there is not the ultimate one (yet), authors in [23] propose to use WMD as it successfully computes the semantic similarity of sentences. The usage of another metric and the optimization of our model upon it could have improved our results.

Another way of improving our results would be the utilization of a different dataset. The first step would be a bigger dataset. Current papers introducing new models only rarely use Flickr8k. Flick30k or MSCOCO with 30,000, respectively 330,000 ($>$200,000 labeled), images are the datasets of choice. The latter comprises 91 different types of objects in their natural context and all in all 2.5 million labeled instances of these objects [33]. This implies that not only a larger but also a better and richer training set could improve our results.

A further possibility of improvement would be a broader training of our LSTM by feeding in other text sources to teach our model what word stands in which interrelation with other words. In this way existing relationships are augmented but at the same time we would not increase our vocabulary. If we for instance observe the word "BMX" only once in our vocabulary, our model would usually only learn one combination with that word. In other text sources "BMX" would possibly be used hundreds of times in other combinations. These combinations would not necessarily be all in our vocabulary but maybe a few of them, which is better than just one.

Yet another enhancement could be the utilization of word2vec instead of hot encoding. It facilitates a distributed representation of words in a vector space. Thereby, we could easily calculate how close different words in the vector space are to each other (in meaning or context) by calculating the distance of their points. Instead of a probability distribution the model would create a word vector [49].

State of the art models are applying attention mechanisms which typically produce a spatial map highlighting regions of the image that are relevant to each generated word [1, 4, 49, 48]. This mechanism allows to focus on important features from an image. Step by step, the regions of interest of the image are determined and fed into the RNN. These regions of interest are typically salient objects in the picture which results in more human-like captions [4].

Furthermore, BeamSearch would also increase the performance of our model. Thereby, the set of the k best candidate sentences are considered up to a time t to generate sentences with the size $t + 1$ and only the resulting best k of them are kept [4, 47, 22, 44].

Finally, we could also use a fine tuned CNN architecture on our dataset. The VGG16 that we used is trained on the ImageNet dataset [2].

5 Conclusion

In this paper we addressed the challenge of automatically generating image captions. It is a topic of growing interest in the field of machine learning and in practice it can help for instance visually impaired people of grasping the content of an image. First, we explained the foundational methods CNN, RNN and LSTM, on which our work is based on. After introducing how to approach the right architecture we explain how to build the model upon this architecture. Our model consists of a CNN, the pretrained VGG16, where we removed the last layer in order to obtain an image feature vector. We combined the CNN with a LSTM by means of a merge architecture that proves to perform best. We trained our model on the Flickr8k dataset that fulfills the necessary criteria of human generated conceptual descriptions. Even though our model can not keep up with the state of the art models, it scores adequately for a non-optimized model on the BLEU score. Furthermore, our focus was to provide an instruction for building this basic model as an entry into this research field. Finally, we discussed several options to enhance our basic model in order to compete with current ones. All in all, the result of our work is an easily understandable, reproducible and extendable model.

References

[1] COCO - Common Objects in Context. http://cocodataset.org/#captions-leaderboard.

[2] ImageNet Large Scale Visual Recognition Competition 2014 (ILSVRC2014). http://www.image-net.org/challenges/LSVRC/2014/.

[3] Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. SPICE: Semantic Propositional Image Caption Evaluation. *arXiv:1607.08822 [cs]*, July 2016.

[4] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering. *arXiv:1707.07998 [cs]*, July 2017.

[5] Jasmina Arifovic and Ramazan Gencay. Using genetic algorithms to select architecture of a feedforward artificial neural network. *Physica A: Statistical mechanics and its applications*, 289(3-4):574–594, 2001.

[6] Satanjeev Banerjee and Alon Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the Acl Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, 2005.

[7] P.G. Benardos and G.-C. Vosniakos. Optimizing feedforward artificial neural network architecture. *Engineering Applications of Artificial Intelligence*, 20(3):365–382, April 2007.

[8] Yoshua Bengio Y., P Simard, and P Frasconi. Learning long-term dependencies with gradient descent is difficult. Volume 5, pages 157–166, 1994.

[9] Jason Brownlee. How to Develop a Deep Learning Photo Caption Generator from Scratch, November 2017.

[10] Yan-Shuo Chang. Fine-grained attention for image caption generation. *Multimedia Tools and Applications*, 77(3):2959–2971, February 2018.

[11] Xinlei Chen and C. Lawrence Zitnick. Mind's eye: A recurrent visual representation for image caption generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2422–2431, 2015.

[12] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv:1412.3555 [cs]*, December 2014.

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference On*, pages 248–255. IEEE, 2009.

[14] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2625–2634, 2015.

[15] Scott E. Fahlman and Christian Lebiere. The cascade-correlation learning architecture. In *Advances in Neural Information Processing Systems*, pages 524–532, 1990.

[16] Yansong Feng and Mirella Lapata. Automatic Image Annotation Using Auxiliary Text Information. *Proceedings of ACL-08: HLT*, pages 272–280, 2008.

[17] Alex Graves. Generating Sequences With Recurrent Neural Networks. *arXiv:1308.0850 [cs]*, August 2013.

[18] H. P. Grice. *Logic and Conversation*. 1975.

[19] Michael Grubinger, Paul Clough, Henning Müller, and Thomas Deselaers. *The IAPR TC-12 Benchmark – a New Evaluation Resource for Visual Information Systems*. 2006.

[20] Micah Hodosh, Peter Young, and Julia Hockenmaier. Framing image description as a ranking task: Data, models and evaluation metrics. *Journal of Artificial Intelligence Research*, 47:853–899, 2013.

[21] Alejandro Jaimes and Shih-Fu Chang. Conceptual framework for indexing visual information at multiple levels. pages 2–15, December 1999.

[22] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137, 2015.

[23] Mert Kilickaya, Aykut Erdem, Nazli Ikizler-Cinbis, and Erkut Erdem. Re-evaluating automatic metrics for image captioning. *arXiv preprint arXiv:1612.07600*, 2016.

[24] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[25] Ryan Kiros, Ruslan Salakhutdinov, and Rich Zemel. Multimodal neural language models. In *International Conference on Machine Learning*, pages 595–603, 2014.

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. ImageNet Classification with Deep Convolutional Neural Networks. volume 25, 2012.

[27] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. From word embeddings to document distances. In *International Conference on Machine Learning*, pages 957–966, 2015.

[28] Polina Kuznetsova, Vicente Ordonez, Alexander C. Berg, Tamara L. Berg, and Yejin Choi. Collective generation of natural image descriptions. In *50th Annual Meeting of the Association for Computational Linguistics, ACL 2012 - Proceedings of the Conference*, 2012.

[29] Yann LeCun and Yoshua Bengio. The Handbook of Brain Theory and Neural Networks. pages 255–258. MIT Press, Cambridge, MA, USA, 1998.

[30] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.

[31] Chin Yee Lee, Hengfeng Li, Ruxin Hou, and Calvin Tanujaya Lim. NLTK library bleu score documentation. https://www.nltk.org/_modules/nltk/translate/bleu_score.html, September 2017.

[32] J. Leski and Ernest Czogala. A new artificial neural network based fuzzy inference system with moving consequents in if–then rules and selected applications. *Fuzzy Sets and Systems*, 108(3):289–297, 1999.

[33] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common Objects in Context. *arXiv:1405.0312 [cs]*, May 2014.

[34] Junhua Mao, Xu Wei, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan L. Yuille. Learning like a child: Fast novel visual concept learning from sentence descriptions of images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2533–2541, 2015.

[35] Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan Yuille. Deep captioning with multimodal recurrent neural networks (m-rnn). *arXiv preprint arXiv:1412.6632*, 2014.

[36] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[37] Vicente Ordonez, Girish Kulkarni, and Tamara L. Berg. Im2Text: Describing Images Using 1 Million Captioned Photographs. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 1143–1151. Curran Associates, Inc., 2011.

[38] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

[39] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121, 2000.

[40] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[41] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[42] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *arXiv:1602.07261 [cs]*, February 2016.

[43] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. *arXiv:1409.4842 [cs]*, September 2014.

[44] Marc Tanti, Albert Gatt, and Kenneth P. Camilleri. Where to put the Image in an Image Caption Generator. *arXiv preprint arXiv:1703.09137*, 2017.

[45] Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4566–4575, 2015.

[46] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164, 2015.

[47] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and Tell: Lessons Learned from the 2015 MSCOCO Image Captioning Challenge. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):652–663, April 2017.

[48] Kelvin Xu, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, Attend and Tell- Neural Image Caption Generation with Visual Attention. In *International Conference on Machine Learning*, pages 2048–2057, April 2016.

[49] Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. Image captioning with semantic attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4651–4659, 2016.

[50] Matthew D. Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. *arXiv:1311.2901 [cs]*, November 2013.

[51] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

[52] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning Transferable Architectures for Scalable Image Recognition. *CoRR*, abs/1707.07012, 2017.

A. Appendix

Example of a caption generated by our model. Table 6 shows the probability the model calculated for each word to follow the given prefix.

Figure 5: Test image. Source: `https://www.flickr.com/photos/jgirl4858/3285180819`

	0	1	2	3	4	5	6	7	8	9	10
caption	startword	football	player	in	red	uniform	be	tackle	the	ball	endword
probability	1.00	0.15	0.97	0.08	0.28	0.22	0.11	0.14	0.19	0.61	0.58

Table 6: Caption and word probabilities as generated by our model for the image shown in Figure 6.