Neural Image Captioning for Intelligent Vehicle-to-Passenger Communication

Fredrik Gustafsson
Department of Electrical Engineering
Stanford University
fregu856@stanford.edu

Abstract

Motivated by the future need of intelligent systems for vehicle-to-passenger communication in autonomous vehicles, we implement two models for automatic image caption generation based on the neural encoder-decoder framework. Following an extensive hyperparameter search, our non-attention based model achieves performance on the MSCOCO dataset highly comparable to that of its reference model. Our attention based model does however fail to exceed this performance, despite the attention mechanism appearing to function as intended. This leads to a discussion of possible causes and what architecture modifications that might be needed.

1 Introduction and related work

As the automotive industry continues to push heavily toward fully autonomous vehicles, promising significant improvements in terms of both safety and general convenience, one big question still remains to be answered: How do we build the general public's trust in these systems? For regular people to feel safe riding in a self-driving vehicle, it is crucial that it exhibits aspects of true intelligence. We believe one key aspect of this is the vehicle's ability to process its information about the world and convert it into natural language, enabling intuitive vehicle-to-passenger communication. Therefore, in this project we study the problem of automatically generating descriptive captions for images.

Our work follows in the footsteps of recent successful models for this task, which in turn draw inspiration from the neural encoder-decoder framework for machine translation. In this framework, an encoder Recurrent Neural Network (RNN) *reads* the source sentence and transforms it into a compact vector representation. This vector is then fed into a decoder RNN which *generates* a translated sentence. In the modified framework for image captioning, the encoder RNN is instead replaced by a Convolutional Neural Network (CNN) which encodes the image into a compact feature representation. This feature representation can either consist of a single vector representing the entire image, or multiple vectors each representing a certain spatial region of the image. In the first case, this single vector is simply fed into the decoder RNN which generates a sentence describing the image. In the latter case, one builds a model which can learn to focus its attention at different image

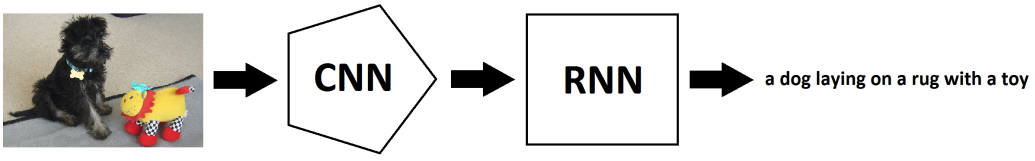

Figure 1: Overview of the neural encoder-decoder framework for image captioning.

regions as it is generating the sentence word by word. Figure 1 above shows an overview of the image captioning framework. In this project, we implement both a model utilizing a single image vector, referred to as *the standard model* in all subsequent discussion, and a model utilizing multiple spatial image vectors, referred to as *the attention model*. A complete implementation of our work is available on GitHub[1].

The original inspiration for this project was the work on image captioning done by Andrej Karpathy and Li Fei-Fei in [1]. Our approach is however mainly modeled after the work in [2], because of their straightforward architecture and their success at the COCO 2015 Captioning Challenge[2]. Specifically, the standard model is closely modeled after the architecture in [2]. A natural extension of this architecture to also include an attention mechanism is described in [3], and thus the attention model is closely modeled after this work.

The current state-of-the-art for the problem of image captioning, as judged by performance on the Microsoft Common Objects in Context (MSCOCO) [4] dataset, is the work in [5]. In this work, they utilize reinforcement learning techniques to directly maximize different quality metrics of their generated captions. A novel model architecture that exhibits close to state-of-the-art performance on MSCOCO is described in [6]. In this architecture, the model not only learns to focus its attention at different image regions when generating different words, as is the case in conventional attention models, but also *when* to pay attention to the image. This architecture seems promising and intuitively makes a lot of sense. Because of time constraints however, it is not explored any further in this project. Instead, the goal of our work is to implement the standard model with performance comparable to [2], and to beat this performance by utilizing the attention mechanism described in [3] in the attention model.

2 Technical approach

In this section we describe the neural encoder-decoder framework for image captioning in general terms, before proceeding to describe in detail the two specific models used in this work. We also define our training objective and describe the procedure used to generate a new caption for a provided image.

2.1 Overview

Given a dataset containing a number of images each with a corresponding caption, we train a well performing model by following a straightforward approach. We model the conditional probability of a caption S given an image I, $p(S|I)$, and attempt to directly maximize the assigned probabilities of the correct captions. If we let θ denote all parameters in our model, the objective is thus to find

$$\theta^* = \arg\max_\theta \sum_I \log p(S^I|I;\theta), \tag{1}$$

where S^I is the caption corresponding to the image I and the summation is over all images in the dataset. Since S^I in theory could be any caption, its length is not fixed. Instead we have $S^I = \{S_1^I, ..., S_{N^I}^I\}$, where S_k^I is word number k in the caption of length N^I. In all captions S^I, S_1^I is a special *START* token and $S_{N^I}^I$ is a special *END* token. Using the chain rule, the log probabilities in (1) above can thus be rewritten as

$$\log p(S^I|I;\theta) = \sum_{t=1}^{N^I} \log p(S_t^I|S_1^I, ..., S_{t-1}^I, I;\theta). \tag{2}$$

Thus, what we actually model is $p(S_t^I|S_1^I, ..., S_{t-1}^I, I)$, the conditional probability of the next word in the caption given the image and the previous words. To model this conditional probability, we use an RNN with a fixed length hidden state vector $\mathbf{h}_t \in \mathbb{R}^{D_h}$. The hidden state $\mathbf{h}_t$ will be updated every time we obtain a new model input $\mathbf{x}_t \in \mathbb{R}^{D_x}$ according to a non-linear function f_{RNN}, where the definition of f_{RNN} depends on the specific type of RNN model being used. In this project, we

[1] `https://github.com/fregu856/CS224n_project`

[2] `http://mscoco.org/dataset/#captions-leaderboard`

consider two different RNN models: Long Short-Term Memory (LSTM) [7] and Gated Recurrent Units (GRU) [8]. The function f_{LSTM} is defined according to:

$$\begin{aligned}
\mathbf{i}_t &= \sigma_\circ(\mathbf{W}_i\mathbf{x}_t + \mathbf{U}_i\mathbf{h}_{t-1}),\\
\mathbf{f}_t &= \sigma_\circ(\mathbf{W}_f\mathbf{x}_t + \mathbf{U}_f\mathbf{h}_{t-1}),\\
\mathbf{o}_t &= \sigma_\circ(\mathbf{W}_o\mathbf{x}_t + \mathbf{U}_o\mathbf{h}_{t-1}),\\
\tilde{\mathbf{c}}_t &= \tanh_\circ(\mathbf{W}_c\mathbf{x}_t + \mathbf{U}_c\mathbf{h}_{t-1}),\\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t,\\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh_\circ(\mathbf{c}_t) \triangleq f_{LSTM}(\mathbf{x}_t, \mathbf{h}_{t-1}),
\end{aligned} \tag{3}$$

where $\mathbf{W}_i, \mathbf{W}_f, \mathbf{W}_o, \mathbf{W}_c \in \mathbb{R}^{D_h \times D_x}$, $\mathbf{U}_i, \mathbf{U}_f, \mathbf{U}_o, \mathbf{U}_c \in \mathbb{R}^{D_h \times D_h}$, $\odot$ denotes element-wise multiplication, $g_\circ$ denotes element-wise application of the scalar function g and σ is the sigmoid function. f_{GRU} is in turn defined according to:

$$\begin{aligned}
\mathbf{z}_t &= \sigma_\circ(\mathbf{W}_z\mathbf{x}_t + \mathbf{U}_z\mathbf{h}_{t-1}),\\
\mathbf{r}_t &= \sigma_\circ(\mathbf{W}_r\mathbf{x}_t + \mathbf{U}_r\mathbf{h}_{t-1}),\\
\tilde{\mathbf{h}}_t &= \tanh_\circ(\mathbf{W}_h\mathbf{x}_t + \mathbf{r}_t \odot (\mathbf{U}_h\mathbf{h}_{t-1})),\\
\mathbf{h}_t &= \mathbf{z}_t \odot \mathbf{h}_{t-1} + (\mathbb{1} - \mathbf{z}_t) \odot \tilde{\mathbf{h}}_t \triangleq f_{GRU}(\mathbf{x}_t, \mathbf{h}_{t-1}),
\end{aligned} \tag{4}$$

where $\mathbf{W}_z, \mathbf{W}_r, \mathbf{W}_h \in \mathbb{R}^{D_h \times D_x}$, $\mathbf{U}_z, \mathbf{U}_r, \mathbf{U}_h \in \mathbb{R}^{D_h \times D_h}$ and $\mathbb{1}$ is the vector in $\mathbb{R}^{D_h}$ with all elements equal to 1. Once we have the hidden state $\mathbf{h}_t$ at time t, we obtain our modeled conditional probability $p(S^I_{t+1}|S^I_1, ..., S^I_t, I; \theta)$ by computing a vector $\hat{\mathbf{p}}_{t+1} \in \mathbb{R}^{D_V}$, containing a probability distribution over all words in our vocabulary of size D_V. $\hat{\mathbf{p}}_{t+1}$ is computed according to:

$$\hat{\mathbf{p}}_{t+1} = \mathrm{softmax}(\mathbf{W}\mathbf{h}_t + \mathbf{b}), \tag{5}$$

where $\mathbf{W} \in \mathbb{R}^{D_V \times D_h}$ and $\mathbf{b} \in \mathbb{R}^{D_V}$. Thus, if we by $\hat{\mathbf{p}}_{t+1}(S^I_{t+1})$ denote the element of $\hat{\mathbf{p}}_{t+1}$ corresponding to the word S^I_{t+1}, we have

$$p(S^I_{t+1}|S^I_1, ..., S^I_t, I; \theta) = \hat{\mathbf{p}}_{t+1}(S^I_{t+1}). \tag{6}$$

The other major design decision that needs to be made is to specify what the model input $\mathbf{x}_t$ should be. The exact definition of $\mathbf{x}_t$ depends on the model being used and is specified in detail in section 2.2 and 2.3 below. In general terms, $\mathbf{x}_t$ will be constructed to represent either the image or the current word in the caption, or a combination of the two. To represent an image, we use a pre-trained and fixed CNN which maps the image to a compact feature representation. Specifically, we use the Inception-v3 [9] architecture. This CNN model has been trained for image classification and exhibits close to state-of-the-art performance on the ImageNet ILSVRC [10] dataset. Given an image I, we feed it through the CNN and extract $\mathbf{I}^{CNN} \in \mathbb{R}^{2048 \times 64}$, where each column of $\mathbf{I}^{CNN}$ is a vector in $\mathbb{R}^{2048}$ representing one spatial region of the image specified by a 8×8 grid. To represent words, perhaps the most straightforward approach would be to use the one-hot representation. That is, to represent word number k in the vocabulary, S_k, by the vector $\mathbf{s}_k \in \mathbb{R}^{D_V}$, where $\mathbf{s}^k_k = 1$ is the only non-zero element. Instead, we use the standard approach of recent years and utilize an embedding matrix $\mathbf{W}_e \in \mathbb{R}^{D_w \times D_V}$, which maps the one-hot representation $\mathbf{s}_k \in \mathbb{R}^{D_V}$ to a dense vector $\mathbf{w}_k = \mathbf{W}_e\mathbf{s}_k \in \mathbb{R}^{D_w}$. The embedding matrix $\mathbf{W}_e$ is a model parameter, which in our experiments is initialized both using pre-trained GloVe [11] vectors and a random initialization.

2.2 The standard model

The standard model is the most straightforward implementation of a CNN-RNN based architecture for image captioning. In this case, we feed the image I as input to the RNN only at the first time step, and then input the words of the caption S^I in order. Thus, we first have to map our image representation $\mathbf{I}^{CNN} \in \mathbb{R}^{2048 \times 64}$ to the same space as our embedded word representations. We do this by averaging the columns of $\mathbf{I}^{CNN}$ and using a simple projection. If we by $\mathbf{I}^{CNN}_k \in \mathbb{R}^{2048}$ denote column number k of $\mathbf{I}^{CNN}$, the model input $\mathbf{x}_t$ and thus the complete model is specified

according to:

$$\begin{aligned}
&\mathbf{I}_{avg}^{CNN} = \frac{\sum_{k=1}^{64} \mathbf{I}_k^{CNN}}{64}, \\
&\mathbf{x}_0 = \sigma_\circ(\mathbf{W}_I \mathbf{I}_{avg}^{CNN} + \mathbf{b}_I), \\
&\mathbf{x}_t = \mathbf{W}_e \mathbf{s}_t^I, \quad t \in \{1, 2, ..., N^I - 1\}, \\
&\mathbf{h}_t = f_{RNN}(\mathbf{x}_t, \mathbf{h}_{t-1}), \\
&\hat{\mathbf{p}}_{t+1} = \text{softmax}(\mathbf{W}\mathbf{h}_t + \mathbf{b}),
\end{aligned} \tag{7}$$

where $\mathbf{W}_I \in \mathbb{R}^{D_w \times 2048}$, $\mathbf{b}_I \in \mathbb{R}^{D_w}$ and $\mathbf{h}_{-1} = \mathbf{0}$. Thus in this case, $D_x = D_w$.

2.3 The attention model

The attention model is a more complicated model where the image I is fed to the RNN not only once, but at every time step. At each time step, we also enable the model to focus at different spatial regions of the image by computing an attention distribution over the 64 image regions specified by $\mathbf{I}^{CNN} \in \mathbb{R}^{2048 \times 64}$. This attention distribution at time t, which we denote by α_t, is computed according to:

$$\begin{aligned}
&\hat{\mathbf{I}}_k^{CNN} = \sigma_\circ(\mathbf{W}_I \mathbf{I}_k^{CNN} + \mathbf{b}_I), \\
&\mathbf{a}_t^k = \mathbf{W}_a \tanh_\circ(\mathbf{W}_{aI} \hat{\mathbf{I}}_k^{CNN} + \mathbf{W}_{ah} \mathbf{h}_{t-1} + \mathbf{b}_a), \\
&\alpha_t = \text{softmax}(\mathbf{a}_t + \mathbf{b}_\alpha),
\end{aligned} \tag{8}$$

where $\mathbf{W}_a \in \mathbb{R}^{1 \times D_a}$, $\mathbf{W}_{aI} \in \mathbb{R}^{D_a \times D_w}$, $\mathbf{W}_{ah} \in \mathbb{R}^{D_a \times D_h}$, $\mathbf{b}_a \in \mathbb{R}^{D_a}$ and $\mathbf{b}_\alpha \in \mathbb{R}^{64}$. Because of limited storage, $\hat{\mathbf{I}}_k^{CNN}$ is computed in a pre-processing step using $\mathbf{W}_I$ and $\mathbf{b}_I$ taken from the best performing standard model. Once we have the attention distribution α_t, we obtain the image input at time t, $\mathbf{I}_t$, as the weighted sum

$$\mathbf{I}_t = \sum_{k=1}^{64} \alpha_t^k \hat{\mathbf{I}}_k^{CNN}, \tag{9}$$

and construct the model input $\mathbf{x}_t$ by concatenating $\mathbf{I}_t$ with the embedded vector of the current word in the caption. That is, $\mathbf{x}_t$ and thus the complete model is specified according to:

$$\begin{aligned}
&\mathbf{x}_t = \mathbf{I}_t \oplus \mathbf{W}_e \mathbf{s}_t^I, t \in \{1, 2, ..., N^I - 1\}, \\
&\mathbf{h}_t = f_{RNN}(\mathbf{x}_t, \mathbf{h}_{t-1}), \\
&\hat{\mathbf{p}}_{t+1} = \text{softmax}(\mathbf{W}\mathbf{h}_t + \mathbf{b}),
\end{aligned} \tag{10}$$

where $\oplus$ denotes vector concatenation and $\mathbf{h}_0 = \mathbf{0}$. Thus in this case, $D_x = 2D_w$.

2.4 Training

As outlined in section 2.1 above, our goal is to maximize the conditional probability our model assigns to the ground truth caption S^I of each image I in the training dataset. Combining equations (1), (2) and (6), we thus get that our final objective is to find

$$\theta^* = \arg\max_\theta \sum_I \sum_{t=1}^{N^I} \log \hat{\mathbf{p}}_t(S_t^I). \tag{11}$$

Thus, we define the per-image loss $L(I, S^I)$ according to:

$$L(I, S^I) = -\sum_{t=2}^{N^I} \log \hat{\mathbf{p}}_t(S_t^I), \tag{12}$$

where the sum starts at $t = 2$ since S_1^I is the special *START* token which there is no need to predict. The per-image loss $L(I, S^I)$ is in practice summed over a certain number of images to obtain a per-batch loss, and then minimized with respect to all model parameters θ using the Adam optimizer [12].

2.5 Caption generation

In the setting where we wish to generate a new caption S for an image I, we do not have a complete ground truth caption $S^I = \{S_1^I, ..., S_{N^I}^I\}$ to feed as input to the model. Instead, we initialize S with $S = \{S_1\}$ where S_1 is the *START* token, we set S_2 to the most likely next word as predicted by the model and then continue with this procedure until the predicted next word equals the *END* token. At this point, we will have generated a sentence $S = \{S_1, ..., S_N\}$ of length N which we hope is a descriptive caption of the provided image I.

3 Experiments

In this section we start by presenting the dataset being used for training and evaluation, we briefly describe the used evaluation metrics and describe in detail the performed tuning of hyperparameters. We then present the quantitative performance of both our models and finally perform a qualitative analysis of generated captions.

3.1 Dataset

We use the MSCOCO [4] dataset for both training and measuring the performance of our models. MSCOCO contains a total of 123000 images, each with five human generated captions. We use the same data split as in [1], [3], [5] and [6], setting aside 5000 random images each for validation and testing. This thus leaves us with a training set containing 113000 images. Figure 2 below shows two example training images together with their corresponding ground truth captions. We pre-process the captions in the training set by removing all non-alphanumeric characters, making all words lower case, prepending each caption with a special *START* token and appending each caption with an *END* token. We then proceed to create a vocabulary of all words that occur at least five times in the training set, which results in a vocabulary size of $D_V = 9855$ words. Finally, we replace every word not in the vocabulary with an *UNKNOWN* token. To evaluate the performance

A large black semi parked in a parking lot.
A very large semi truck without any load is parked.
A shiny, new, low riding semi trailer truck.
An eighteen wheeler truck is parked in a lot.
A large black truck in a parking lot.

A white horse drawn carriage in front of a yellow building.
A very pretty horse pulling a fancy carriage.
A white horse pulling a white carriage.on the street.
A white horse pulling a wagon while stopped on the side of the road.
A horse and carriage parked beside a building.

Figure 2: Two example images from the training dataset and their ground truth captions.

of our models, we use four different automatically computable metrics: BLEU4 [13], CIDEr [14], METEOR [15] and ROUGE-L [16]. These metrics can all be easily computed using a provided evaluation toolkit[3] and have become the standard evaluation metrics for image captioning. When tuning hyperparameters, we do so primarily based on the CIDEr scores, since it has been empirically shown that optimizing the CIDEr score leads to good overall performance [5].

3.2 Tuning of hyperparameters

We started the experiments on the standard model using a 1-layer LSTM as the RNN model type, applied dropout with a keep probability of $p_{dropout} = 0.5$, set the hidden state dimension to $D_h = 200$ and used a training batch size of 256 images. The embedded word dimension was $D_w = 300$ across all experiments. During training, we evaluated the performance by computing the CIDEr score on 1000 validation images after each finished epoch. The CIDEr score was then plotted against the

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

epoch number, enabling simple comparison of different model variations. We started studying the effect of increasing D_h and found that a larger value always resulted in at least a slightly higher CIDEr score. Eventually however, the gain became almost undetectable and we thus fixed the hidden state dimension to $D_h = 400$. We then studied the effect of varying the dropout keep probability $p_{dropout}$. Figure 3 (left and center) below shows the plotted CIDEr curves together with the corresponding curves for the per-epoch loss, for $p_{dropout} \in \{0.25, 0.5, 0.75, 1\}$. In the plots, we observe the classic behavior where an increase in $p_{dropout}$ always leads to a smaller loss, but that there is a tipping point when the model starts to overfit which hurts CIDEr performance. Based on figure 3, we thus fixed $p_{dropout} = 0.75$. Using the same method, we also fixed the number of layers to 1 (2 layers had slightly worse performance), the batch size to 256 (varying the batch size had essentially no effect) and the RNN model type to an LSTM (using a GRU slightly lowered performance). Finally, we studied the effect of word embedding initialization. Figure 3 (right) below shows a comparison of initializing $\mathbf{W}_e$ using GloVe [11] vectors and with a random matrix. We observe that the

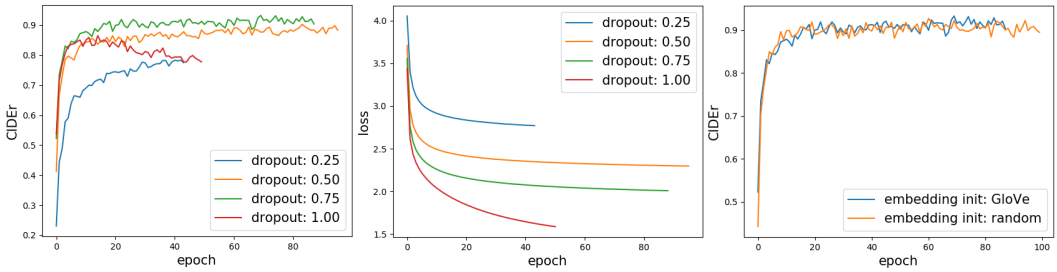

Figure 3: Left and center: The effect of varying $p_{dropout}$ as measured by the CIDEr score and the per-epoch loss. Right: Initialization of $\mathbf{W}_e$ using GloVe [11] vectors versus with a random matrix.

two methods in fact have very similar performance, the prior information encapsulated in the GloVe vectors thus seems to be of minor use for this particular task. Then, using the fixed parameter values from above, we also studied the effect of varying the size of D_a in the attention model. Similar to D_h, an increase always led to improved performance but with diminishing gain. Thus, we chose to set $D_a = 500$.

3.3 Results

In table 1 below we report the performance of both our models, using the parameter values set in section 3.2, on the test set containing 5000 images. We also include a number of benchmark results, including our own implementation of a nearest neighbor method, the methods presented in the papers [1] - [3] which served as the main inspiration for this work, and the current state-of-the-art [5]. BRRN and Hard-Attention used less powerful CNNs, and does thus not serve as fair

Table 1: Results on a test set of 5000 images. Σ indicates use of ensembling, † a smaller test set.

Model	CIDEr	BLEU4	METEOR	ROUGE-L
The standard model	**87.5**	**27.3**	23.3	**51.1**
The attention model	87.0	26.8	**23.5**	51.0
Nearest neighbor	44.7	10.6	17.1	36.9
BRNN† [1]	66.0	23.0	19.5	-
Hard-Attention [3]	-	25.0	23.0	-
NIC$^{\Sigma\dagger}$ [2]	85.5	27.7	23.7	-
NICv2$^{\Sigma\dagger}$ [2]	99.8	32.1	25.7	-
Att2in-SCST [5]	**111.4**	**33.3**	**26.3**	**55.3**

comparison. NIC is the most relevant benchmark since it is the architecture the standard model was modeled after. Compared to NIC, we use a more powerful CNN but does not use an ensemble model. These modifications were however showed in [2] to boost the performance by a similar amount, and thus the performance is highly comparable. One might thus expect the performance of the standard

model to approach that of NICv2 if we were to implement the full set of modifications[4] to NIC described in [2]. What is disappointing however, is that we not were able to boost performance by using the attention model. We discuss this further in section 3.5 below. We also studied whether the model generated captions are taken directly from the training data, or if they in fact are novel captions. We found that generated captions are present in the training data roughly 40% of the times, with no significant difference between our two models. For comparison, in [2] they found that for their top 15 candidate captions the corresponding number was roughly 50%. If they only considered the top candidate caption, it was 80%.

3.4 Qualitative analysis of generated captions

In figure 4 below, we show captions generated by the standard model on example images from the test set. The images were chosen to be representative of the model output, and are sorted into three different categories. The two images on the left are captioned without error and with a relatively high degree of detail. The captions for the center images are mostly correct but are missing a key detail, for instance that the man with bananas is in fact wearing them as a hat. This example demonstrates the limitations of our model. It recognizes that a man and a bunch of bananas are in the image, but fails to correctly describe the relationship between them when combined in an unconventional way. The images to the right are captioned incorrectly, but the captions are not completely unrelated to the images. Almost all generated captions seem to fall in to one of these categories, with the majority being similar to the center images. Completely incorrect captions in which you find absolutely no trace of the corresponding images are very rare. Figure 5 below shows how the generated captions

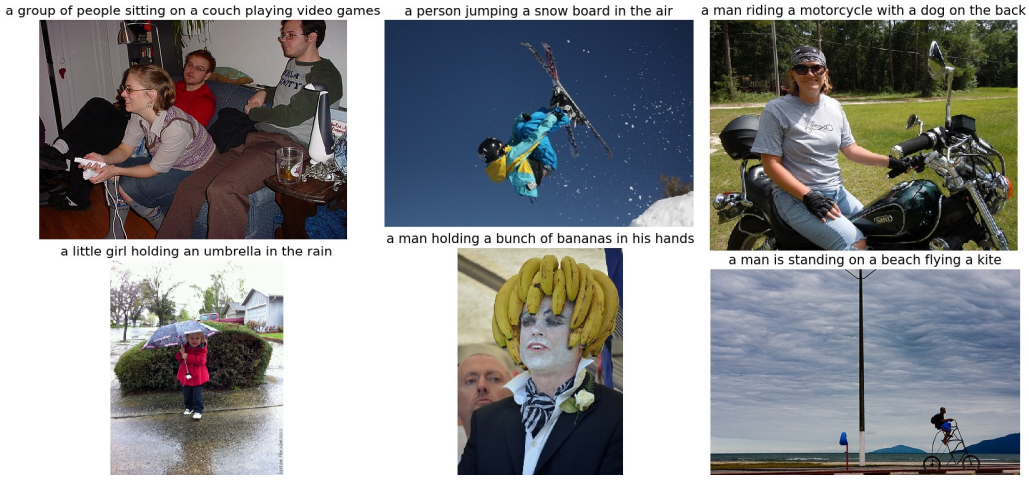

Figure 4: Example of generated captions on unseen test images. Left: Correctly describes the image. Center: Mostly correct, but missing key detail. Right: Incorrect, but somewhat related to the image.

for three validation images evolved during training, where captions higher up were generated at earlier epochs. We observe that the models are able to capture increasingly detailed image features, for instance replacing "a zebra" and "grass" with the more correct "two zebras" and "tall grass" in the left image. We also observe that the general language model works well. Even after just one epoch of training, the model will output valid and reasonable English sentences.

3.5 Analysis of the attention mechanism

As mentioned in section 3.3, we were disappointed to find that the attention model did not outperform the standard model. Contrary to our initial belief, it is not difficult to find example images where the attention mechanism seems to function exactly as intended. Figure 6 below shows a visualization of the attention distribution at the time of prediction of certain words. We observe that the model is able to attend to specific objects, for instance to the cat in the upper right corner of the image on the right, and that these objects seem to guide the caption generation. One could thus think

[4]Ensembles, scheduled sampling [17], fine-tuning of the CNN and beam search.

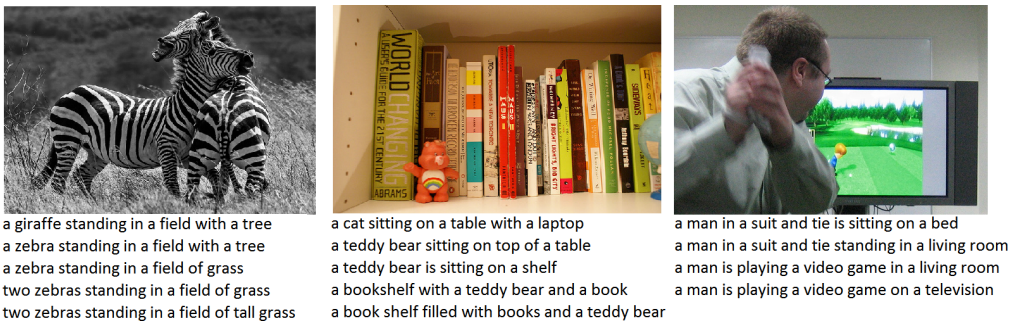

Figure 5: Evolution of generated captions. Captions higher up were generated at earlier epochs.

this would provide additional information to the model and help boost the performance, but that is apparently not the case. One possible explanation is that while the attention mechanism seems to function as intended, it might not actually provide any information not already captured internally by the standard model. It was also shown in [2] that feeding the entire image as an extra input at each time step, that is setting $\mathbf{x}_t = \hat{\mathbf{I}}_{avg}^{CNN} \oplus \mathbf{W}_e \mathbf{s}_t^I$, yielded lower performance. Considering that we use an image representation with only 64 spatial regions (for comparison, in [3] they used 196) it thus seems likely that we are partially experiencing the same issues. If one studies the current state-of-the-art models one also finds that both [5] and [6], who both use ResNet [18] as the image encoding CNN, use more advanced attention architectures. In [5], they compute $\mathbf{I}_t$ using the same

Figure 6: Attention visualization. Bright rectangles indicate areas of high attention.

method as described in section 2.3, but then feed $\mathbf{I}_t$ into their LSTM only in the update equation for $\mathbf{c}_t$. In [6], as described in section 1, their model also learns when to look at the image and when to only rely on the language model to generate the next word. Our belief is thus that the straightforward attention implementation used in this project not is quite capable of boosting the performance, at least not of a non-attention model using a CNN architecture as powerful as the Inception-v3 or ResNet. Finally, it is worth noting that we started performing hyperparameter tuning on the standard model. With reversed order, the results might have been slightly different.

4 Conclusion

We implemented two models for automatic image caption generation based on the neural encoder-decoder framework. Our implementation of a non-attention model, the standard model, met our set goal as it achieved performance highly comparable to that of [2]. The attention model did however fail to exceed this performance, despite the attention mechanism appearing to function as intended. We thus conclude that a modified attention model likely is needed, and future work would study the architecture proposed in [6] where the model also learns *when* to focus on the image. Our work was motivated by the future need of intelligent systems for vehicle-to-passenger communication. While a perfect image captioning model would be a good start, a system based solely on image captioning would be fundamentally flawed as it would be incapable of two-way communication. Future work would thus also study the related problem of visual question answering. For this task, the work in [19] based on the Dynamic Memory Network (DMN) should be a good starting point.

References

[1] Andrej Karpathy and Li Fei-Fei. *Deep Visual-Semantic Alignments for Generating Image Descriptions*. CVPR, 2015.

[2] Oriol Vinyals et al. *Show and Tell: Lessons learned from the 2015 MSCOCO Image Captioning Challenge*. IEEE Transactions on Pattern Analysis and Machine Intelligence, volume: PP, issue: 99, 2016.

[3] Kelvin Xu et al. *Show, Attend and Tell: Neural Image Caption Generation with Visual Attention*. Proceedings of the 32nd International Conference on Machine Learning (ICML), 2015.

[4] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. EECV, 2014.

[5] Steven J. Rennie et al. *Self-critical Sequence Training for Image Captioning*. arXiv:1612.00563, 2016.

[6] Jiasen Lu and Caiming Xiong et al. *Knowing When to Look: Adaptive Attention via A Visual Sentinel for Image Captioning*. CVPR, 2017.

[7] Sepp Hochreiter and Jurgen Schmidhuber. *Long short-term memory*. Neural Computation, volume: 9, issue: 8, 1997.

[8] Kyunghyun Cho et al. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. EMNLP, 2014.

[9] Christian Szegedy et al. *Rethinking the Inception Architecture for Computer Vision*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016.

[10] Olga Russakovsky and Jia Deng et al. *ImageNet Large Scale Visual Recognition Challenge*. IJCV, 2015.

[11] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. *GloVe: Global Vectors for Word Representation*. 2014.

[12] Diederik P. Kingma and Jimmy Ba. *Adam: A method for stochastic optimization*. ICLR, 2015.

[13] K. Papineni et al. *BLEU: A method for automatic evaluation of machine translation*. ACL, 2002.

[14] Ramakrishna Vedantam, C. Lawrence Zitnick and Devi Parikh. *CIDEr: Consensus based image description evaluation*. arXiv:1411.5726, 2015

[15] Satanjeev Banerjee and Alon Lavie. *Meteor: An automatic metric for MT evaluation with improved correlation with human judgments*. Proceedings of the ACL workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization, vol. 29, 2005.

[16] Chin-Yew Lin. *Rouge: A package for automatic evaluation of summaries*. Text summarization branches out: Proceedings of the ACL-04 workshop, vol. 8, 2004.

[17] Samy Bengio et al. *Scheduled sampling for sequence prediction with recurrent neural networks*. Advances in Neural Information Processing Systems, NIPS, 2015.

[18] Kaiming He et al. *Deep residual learning for image recognition*. CVPR, 2016.

[19] Caiming Xiong, Stephen Merit and Richard Socher. *Dynamic Memory Networks for Visual and Textual Question Answering*. ICML, 2016.