A neural network approach to image captioning

University of Amsterdam
MSc. Artificial Intelligence

Luca Simonetto (luca.simonetto@student.uva.nl)
Heng Lin (heng.lin@student.uva.nl)
Tushar Nimbhorkar (tushar.nimbhorkar@student.uva.nl)
Mircea Mironenco (mircea.mironenco@student.uva.nl)

Abstract

We explore recent machine learning techniques for connecting image contents with natural language. Image captioning is a task that has attracted increasing interest from researchers as the progress in the field of image recognition and language has accelerated, and state of the art techniques are being constructed every year. We aim to understand how the application of deep learning techniques changed both Computer Vision and Natural Language Processing paradigms, as well as what architectures would be most proficient given the Flickr8 dataset. In this research project, we construct a model consisting of an embedding layer for the input captions, which is then concatenated with the image features before passing through the primary LSTM layer. Our results show that approximately 77% of the captions produced were grammatically correct, 86% of the captions produced were syntactically correct, and 86% were semantically correct.

1 Introduction

The ability to understand and describe the contents of an image is a challenge connecting two fundamental areas of Artificial Intelligence. Recognizing the content of an image is a core task in Computer Vision, and being able to describe its contents while preserving and expressing the underlying relationships requires natural language understanding. The task's difficulty is evidenced by the necessity to have a formal understanding of the semantic and inter-modal relationships between the objects within the image.

As humans, we find it easy to capture the immense amount of details present in an image. Our prior knowledge of various object constructs, our scene understanding ability and the complex architecture of the visual cortex allow us to disseminate information and high-level concepts that require inference and selective attention. However, in a computer, the image is deconstructed into a matrix of pixels, with each value representing the intensity of that pixel within the image. It is from these pixels and their correlations that a computer must infer enough salient information to be able to both discern between different classes of objects but also derive what is occurring in the image.
The motivation behind constructing an image captioning system is characterized both by its long term research importance and its short-term real-world applicability. Humans are immersed in a sensorimotor stream that is driven by visual information and semantic understanding of the world. Enabling computers to acquire knowledge through vision and language means constructing agents that gain an understanding of the world in a similar fashion as we do, and is a critical step towards the development of more natural interactions.

The short-term applicability of the task, is characterized by the modality through which it is able to express information in natural language. Because we use the same modality to transfer information, its consumption by a human counterpart is enabled and as such the task has attracted increasing amount of interest from the research community with the practical goal in mind of constructing a system which can describe scenes or answer questions. Such practical applications have been constructed, and while still in an incipient phase, their performance has increased greatly in the last few years.

Recently, image captioning systems have demonstrated the ability to describe meaningful information present in images. The increasing performance of such system is strongly correlated with several advances in machine learning and

Computer Vision. Advances in our ability to train neural networks (A.Krizhevsky et al., 2012), having large dataset with human-annotated feature-rich images representing thousands of categories of objects (A.Krizhevsky et al., 2012), and state of the art image recognition models based on deep Convolutional Neural Network (Y. LeCun et al., 2010) have greatly contributed to the increase in performance of such systems. Specifically, Convolutional Neural Network have demonstrated the ability to correctly distinguish thousands of object categories, and has become the state of the art approach to other vision tasks such as image segmentation and scene understanding.

Our goal with this project was to study the state of the art image captioning systems and understand how they use natural language. An abstract and succinct description of modern models is one where given an image ***I*** we infer the most likely sentence ***S*** represented as a sequence of words, using the conditional probability **P(S|I)** to define the inference procedure. Furthermore, we can easily identify two tasks that are necessary for an image captioning system: analyzing the information present in the image to understand the objects and activities as well as their relationships, and describing them as linguistic constituents (nouns and verbs) sequentially ordered as a sentence. While there are models that tackle these 2 tasks individually and then construct a system to align them, the approach we present is an end-to-end model where a Convolutional Neural Network is used as an 'encoder' to identify and construct high-level features of the images which are then decoded by a Recurrent Neural Network under the assumption of direct correspondence between objects and their textual representation. This work is based on the "Show and Tell" image captioning approach (Vinyals et al., 2014).

2 Problem

For the task of connecting images with natural language several obstacles can be identified.

- An image recognition model strong enough to capture local and invariant information in images, with the purpose of identifying objects accurately is needed.

- Images and sequences are represented in different dimensional spaces. Our model would need to either embed both types of information in a multimodal space or be able to use them separately.

- As we are modeling sequences, the model should have the capacity to capture long-term dependencies. Furthermore, grammatical and syntactic rules should be captured by the language component and be correlated with the information in the images.

- If we modeled sequences with a Recurrent Neural Network, our approach would need to account for the vanishing and exploding gradient problem. The vanishing/exploding gradient issue is a consequence of the multiplicative process imposed by back-propagation when training a RNN (R. Pascanu et al., 2013). Depending on the initialization of our parameters, gradients may essentially be clipped by the sigmoid/tanh non-linearities, causing the layers at the beginning of the network to receive no updates, therefore prohibiting any form of learning.

- The model needs to account for data sparsity, essentially factoring out words that appear infrequently in our vocabulary, as well as limit the number of words allowed in a sentence.

- Once the model is trained on both images and sequences, generating the captions for an image can be done in several ways. Generally, the model should try and maximize the probability $P(S|I)$ where S is the sentence or caption that would fit the image I.

3 Approach

3.1 Data preprocessing and preparation

In order to train our language model, we need to collect the captioning text the model is learning from. There are 4-6 captions for each image. We initially removed the words with low frequency of occurrence, due to the fact that there will be insufficient context for the language model to learn how to use the infrequent word correctly. We removed approximately 22,000 words that occurred less than four times in the training captions, out of approximately 37,000 distinct words.

Apart from removing the infrequent words, we added a number of special tokens into the vocabulary, such as the start of sentence token, $< S >$; end of sentence,$< /S >$; and a span token $<$

$/SPAN >$ that is used to fill the caption to a specified length. We also removed the caption and its respective image feature in the case that the caption length is over the maximum specified length.

3.2 Learning and inference algorithms

Feedforward neural networks are function approximators who map an input **x** to a category ***y***. The mapping which can be represented as **y** = $f(x; \Theta)$, is part of a learning algorithm where the goal is to find the value of the parameters Θ that result in the best function approximation. The Convolutional Neural Networks are a specialized kind of feedforward networks. The neural network model is essentially a composition or chain of functions, which put together using a structure defined by an acyclic graph that can be used for various inference tasks. In the model $f(x) = f^3(f^2(f^1(x)))$ we call $f^{(1)}$ the first layer $f^{(2)}$ the second layer, etc. The input **x** is also sometimes called the **input layer**. They overcome the deficiencies of linear models, by having intermediate layers which apply non-linear activation functions to the inner representations of the network (which are outputs of previous layers).

In a neural network, information from the input layer flows forward until it reaches and output layer and is fed into the error function to produce a scalar cost. This cost is then propagated backwards through the network using the *back-propagation* algorithm (Rumelhart and Hinton, 1986), which is essentially a continuous application of the chain rule of calculus. Backpropagation moves backward from the error of the network through the outputs, weights and inputs of each hidden layer, assigning those weights responsibility for a portion of the error by calculating their partial derivatives $\frac{\partial J}{\partial \Theta}$. Those updates are then used to adjust the values of the parameters using stochastic gradient descent.

Recurrent neural networks are architectures which can be used to model sequences of data over time. As mentioned earlier, we want to model information received at time step ***t*** by also taking into account information from previous time steps. RNNs address this issue by having recursive loops which allows information to persist. Sequential information is preserved in the networks hidden state:

$h_t = \phi(Wx_t + Uh_{t-1})$, where h_t is the hidden state at time step *t*, x_t is the input, and h_{t-1} is the previous state. We can observe that the previous state is used to update the current state, and that both the input and the previous state are modified by a weight matrix W and U respectively. The matrices correspond the parameters that we want to learn, and can be thought of as 'templates' or indicators to how much important we should accord to the input and previous state. In a language model a practical example would be the inference of the gender when trying to predict the next word: '*She dropped **her** bag.*'.

RNNs are also trained using an extension of the backpropagation algorithm, which propagates the gradients backwards for each time step. However, as RNNs are generally much longer than traditional neural networks, so in practice we use a shortened version of backpropagation called *truncated backpropagation through time* (P. Werbos, 1990). As mentioned earlier, this comes at the cost of losing information about dependencies over longer sequences.

This downside of this approach is that there are cases where more context is required than the 'vanilla' RNN is able to capture. As we shorten the number of steps we backpropagate, the gap between the relevant information and where it would be needed increases. In theory, RNNs could capture information from a very large amount of previous time steps, however as mentioned earlier, due to the vanishing/exploding gradient problem another variant of the RNN is used.

Convolutional Neural Networks (LeCun and Bengio, 1995) are architectures that make the explicit assumption that our inputs are images. As with normal neural networks they are composed of layers that transform volumes of activations to another through differentiable functions. In convolutional architectures the neurons in a layer are connected only to a small portion of the previous layer. We construct these intermediate layers using a *filter* which we *convolve* on the previous layer (taking also the volume into account) while sharing the parameters. The filters parameters of our network, that are initially randomly initialized and that we learn as the network trains. With them we perform dot product operations with the values of the previous layers and obtain an activation map for each filter. This process allows the network to construct higher level representations (learned later in the network) from spatially combining

lower level representations (learned earlier in the network). A CNN architecture is composed of the following types of layers:

1. Input - Raw pixel values of the image.
2. Conv - Neurons connected to local regions of the input, constructed by taking a *filter* and applying it at different locations.
3. Nonlinear - A layer to apply a non-linear activation functions to the previous layer.
4. Pool - Performs down-sampling in spatial dimension.
5. Dense - A normal fully connected layer.

4 Model architecture

In this project, we implemented an end-to-end neural network framework for the generation of image captions. We divided our model into three different components, that worked together to generate the caption for a given image. The three components are: the image model, that was used to extract image features and to feed them into the main model; the language model, that does an initial elaboration of the caption; and the main LSTM model, containing the recurrent neural network that will output the final caption. The output of the image model and language model are concatenated to create a single tensor containing all the feature representations which are then fed into the LSTM model.

4.1 Image Model

In order to extract the feature vector of a given image (224x224 pixels), we used a pre-trained VGG-16 Convolutional Neural Network (CNN) model, initially used to perform classification on 1000 different object classes, that has been altered in order to output the second-to-last layer activations. The output of the CNN produces a vector of size 4096, that is used to create a matrix with max_caption_length number of repeated features.

4.2 Language Model

This model enables us to transform the input caption (given as a vector of indexes) to an embedded matrix, where each input index is embedded into a 300-dimensional dense vector.
Knowing that a smart initialization of the embedding layer could speed up the learning, we decided

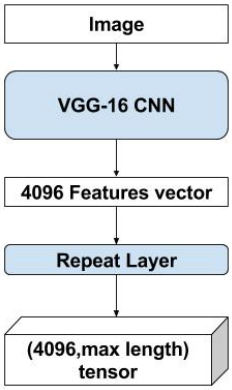

Figure 1: Image model architecture.

to create a custom embedding matrix, using existing data. For this task we used GloVe embeddings (Pennington, 2014), a collection of vectors for over 6 billion words, encoded to 300-length vectors and freely available. With this data we simply created a matrix of vocabulary_size rows and 300 columns, later used in the initialization of the embedding later.
Although already trained, we allowed the embedding layer to be fine-tuned, in order to enable the Language model to propagate the incoming gradients and adjust the embedding to the model convergence.

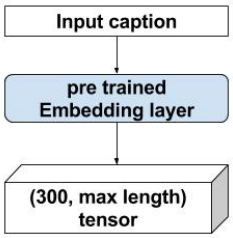

Figure 2: Language model architecture.

4.3 LSTM Model

We use a Long Short-Term Memory (Hochreiter and Schmidhuber, 1997) network as the building block of our language model. As a particular form of Recurrent Neural Networks, the LSTM is able to deal with vanishing and exploding gradients, which is the most common challenge for vanilla RNNs (R. Pascanu et al., 2013).
The core of the LSTM is a memory cell c that encodes knowledge at every time step of what inputs have been observed up to this step. The behavior of the cell is controlled by "gates" layers which are applied multiplicatively and thus can ei-

ther keep a value from the gated layer if the gate is 1 or zeros this value if the gate is 0. More specifically, three gates are being used that control whether to forget the current cell value (forget gate f), if it should read its input (input gate i) and whether to output the new cell value (output gate o).

When receiving a new input, namely the tensor created from the two above models, the LSTM is sequentially fed with it. At each time-step the network receives the image features vector concatenated to one word of the input caption, and it proceeds to output a 512-length vector that encodes the next word. The steps are repeated until the end of the input tensor is reached. Each LSTM output is then passed through a dense layer composed by vocabulary_size neurons, in order to get the index of the predicted word. Each word is then concatenated and the resulting sentence, during the training phase, is compared with the training desired output, allowing to calculate and push the computed gradients into the model.

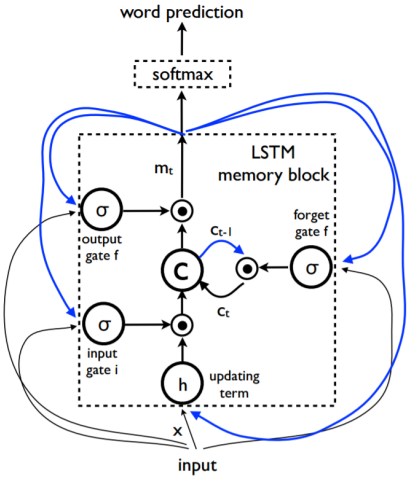

Figure 3: LSTM: the memory block contains a cell c which is controlled by three gates. In blue we show the recurrent connections the output m at time t 1 is fed back to the memory at time t via the three gates; the cell value is fed back via the forget gate; the predicted word at time t 1 is fed back in addition to the memory output m at time t into the Softmax for word prediction. .

$$i_t = \sigma(W_{ix}x_t + W_{im}m_{t-1}) \quad (1)$$

$$f_t = \sigma(W_{fx}x_t + W_{fm}m_{t-1}) \quad (2)$$

$$o_t = \sigma(W_{ox}x_t + W_{om}m_{t-1}) \quad (3)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot h(W_{cx}x_t + W_{cm}m_{t-1}) \quad (4)$$

$$m_t = o_t \odot c_t \quad (5)$$

$$p_{t+1} = softmax(m_t) \quad (6)$$

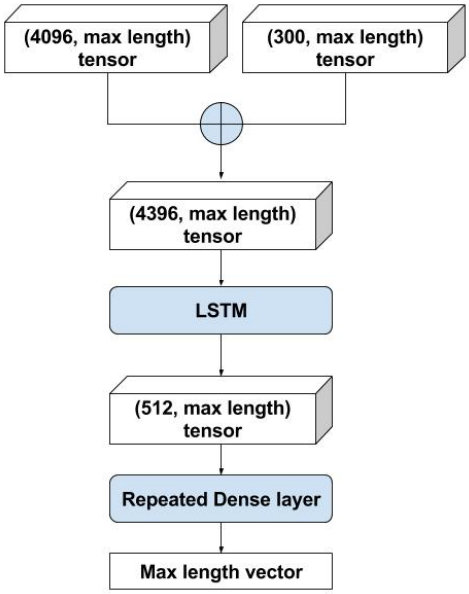

Figure 4: LSTM model architecture.

4.4 Caption generation

To generate captions, we simply take the output of each LSTM cell, and feed it back into the LSTM, to produce the next word in the caption. This is done until either the end of sentence token is reached or we have reached the maximum sentence length. This sampling procedure can be further improve by using the Beam search algorithm, which would selectively choose sequence which have a higher probability of occurrence, and prune outputs which happen less often together.

5 Experiments

In this research, we experimented with a number of different configurations for the model in order to find which one delivers the best performance, such as: (1) using an extra LSTM layer in the language model and (2) using an encoder to perform dimensionality reduction for the image model.

Regarding the use of an encoder to reduce the dimension of the input features, the increase in performance was very limited in our experiment. We first tried with a pre-trained encoder, created using the entire training dataset features and frozen during the main model training. The internal weights were not trainable in order to stop them from changing especially during the initial phase, where every prediction is not correct. No major improvements in the training phase have been recorded, instead we saw an increase of the loss value, so the idea has been discarded. We also tried adding an encoding layer that was not pre-trained, hoping that the added complexity would help the performance. This approach didn't work as expected as it increased the training times by a substantial amount and did not manage to improve the performance even after one full day of training. The potential issue with this approach is that some important information is lost with the dimensionality reduction of the input, and would require a very long training before a good configuration is met. With this results in mind we decided to completely remove the encoder approach from our model, speeding up the training times while having the same performance.

With regards to the analysis of the input caption, we tried to include a pre-processing layer after the embedding phase in the language model, composed by a LSTM block that would output a vector with the same length as the caption but with extracted information, helpful for the main LSTM block. This inclusion brought a substantial increase in training time and, like the encoder approach, failed to increase the performance of the model.

6 Dataset

In order to produce results in reasonable time, we used image feature vectors from an existing training dataset, consisting of 140000 different images, each of them annotated by hand with four to six different captions. Each caption briefly describes the content of the image in a concise way, in order to capture the general meaning of the input. No actual pictures have been provided along with the captions, as they were replaced by a vector of 4096 extracted features. Two examples of captions taken from the dataset are shown below.

An empty bench in front of a building.

A young man is tilting a skateboard up with his feet.

7 Model training

The training of the model uses the standard approach of:

- Computing the outputs of the network, which were produced by each LSTM cell.
- Computing the error of the network, by comparing the embedding of the generated word with the correct subsequent word in the caption.
- Computing the gradients of the error with respect to the updatable network parameters.
- Backpropagating the gradients through the network at each time step.
- Using stochastic gradient descent to update the network parameters.

First, instead of using a one-hot encoding for each word in the target vectors to compute the training cross entropy, we kept the caption structure that we had for the input, consisting of one word index for each word. To achieve this we used sparse cross entropy, that allowed us to reduce the memory footprint of the model and obtain faster training times. Instead of loading in memory the full training dataset (that would require more than 10GB of RAM), we split it into 7 randomly shuffled subsets of data, and trained the model sequentially on each of them.

The training process required an extended period of time, as going through the entire dataset one single time takes as long as 15 hours (using an i7 3537u 2.5GHz processor). As computational resources were scarce, all training was done on a single laptop CPU which would run for almost 24 hours, several days on end. Further performance improvements can be gained by switching to a GPU with extensive resources, which is efficient when doing matrix multiplications.

8 Model testing

In order to test the performance of the model, that accomplishes a relatively new task, seen in the literature only in the past few years, we decided to adopt a testing approach that uses human feedback to tell whether the model performs good or

not. Other types of testing, such as measuring loss, BLEU score (Papineni, 2002) etc. have been discarded, as analyzing the caption directly given the low amount of training data was considered more relevant for our experiment, and would give a better indication of the syntactic and semantic similarity we would like to capture. An example of this behavior can be seen when testing the model with the image of a group of people walking: the generated caption was ”a woman is holding a white shirt in a street .”. While grammatically and syntactically correct (meaning a high score for the standard metrics), it doesn't capture the sense of the image. Figure 5 shows the difference between the input image and the output caption.

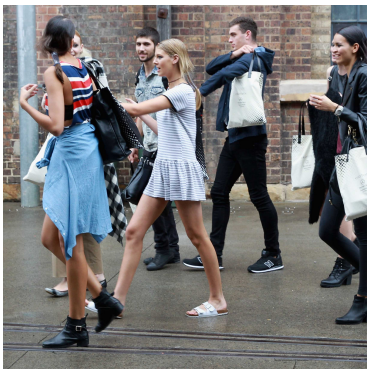

Figure 5: The generated caption is ”a woman is holding a white shirt in a street .”. While it conveys the wrong meaning, the sentence is grammatically and syntactically correct.

Having defined the testing metric, we downloaded 50 representative images from the internet that are similar to the ones that the model has already seen, along with other 15 images consisting of what we thought would be new for it, and proceeded to analyze the captions produced. Each caption is first analyzed grammatically and syntactically. After this, it is checked whether the semantic is aligned with the general sense of the image. An example of the images used can be seen in Figure 6.

9 Results

In this section the results of our work are analyzed. First, it has to be noted that we will present what the model is capable after having been trained on the full training data set two times, with an elapsed time of about 28 hours. A lot of time was spent tackling the already mentioned architectural choices, and each experiment accounted for around the same number of hours.

Figure 6: Four example images from the test dataset.

As said in the previous section, we tested the model with two different sets of images, the first being composed of ”usual” images and the second consisted of more unseen ones. The two following subsections, Grammar analysis and Syntax analysis, show the results for both sets, while the last one, Semantic analysis, is divided in two different parts. We graded the results in two or three different categories, namely *wrong*, *mildly accurate* and *accurate* (depending on the task), indicating the performance of the model for the specific input image.

9.1 Grammar analysis

Even if tested after only two full training cycles, our model managed to produce grammatically correct captions for 77% of the total testing data. It was able to put a dot at the end of the sentence for all captions apart from three, and most of the mistakes were made by repeating the same one or two words until the maximum caption length was encountered. Here are two examples, of a grammatically correct caption and a wrong one.
a giraffe standing in a field with trees in the background .
a colorful colorful colorful [...] colorful

9.2 Syntax analysis

In this evaluation, our model had difficulties producing captions that would be considered correct. Starting from the correct captions of the previous stage, 50 out of 65, the resulting percentage of syntactically correct captions was 86%. Although not very high, this results are still promising and will have a substantial boost if more training is

done. Two examples, a syntactically correct and wrong caption are shown below.
a bathroom with a sink and toilet .
a bunch of scissors and other colorful bears .

9.3 Semantic analysis

Using the remaining captions, that have both a correct grammar and a correct syntax (43 out of 50, 30 in the usual category and 13 in the unusual one), we conducted a direct confrontation between each testing image and the respective caption. In this case, we labeled each caption as *wrong*, *mildly accurate* and *accurate*.
Using Table 1 as a reference, we can see that if the caption is identified as correct grammatically as well as syntactically, approximately 86% of the captions were identified as semantically correct. Interestingly, it was found that the proportion of semantically correct caption for the 'usual' images is 90%. On the other hand, only 76% of the captions in the 'unusual' images were identified as semantically correct. This is a reasonable result due to the fact that there are more exposure of the captioning word in the corresponding image in the 'usual' category, such that resulting in higher accuracy at dependency mapping in the sequence. However, this assumption is made based on assuming the pre-trained Convolutional Neural Network model is capable of extracting features correctly for both categories. The definition of 'usual' and 'unusual' categories is made only based on the frequency of occurrence of a particular captioning word, regardless of the feature extraction model. Figure 7 illustrates the correct caption for an image in the 'usual' category. Figure 8 illustrates the incorrect caption for an image in the 'unusual' category. An interesting note is that the missile in the image is misclassified as an airplane, potentially because of the word 'missile' only occurred for 6 times in the training captions, whereas the word 'plane' occurred for 5116 times, and the two object shared similar geometry. In addition, the occurrence of military vehicle in the training image might be also very limited.

	Grammar	**Syntax**	**Semantic**
Acc. "usual"	75.55%	88.23%	90.00%
Tot. "usual"	34/45	30/34	27/30
Acc. "unusual"	80.00%	81.25%	76.92%
Tot. "unusual"	16/20	13/16	10/13
Accuracy	76.92%	86.00%	86.05%
Total	50/65	43/50	37/43

Table 1: Performance of the trained model on both "usual" and "unusual" images used in the testing phase.

Figure 7: The generated caption for an image in the 'usual' category: "a giraffe standing in a field with trees in the background.", which corresponds to the entities in the image, such as giraffe, field and trees, as well as identified correct relationships between them, such that the trees are behind the giraffe.

Figure 8: The generated caption for an image in the 'unusual' category: "a plane parked on a runway.", which is grammatically and syntactically correct, but does not corresponds to the objects and their relationships in the image.

10 Discussion and Conclusions

We have presented Neural network approach to Image Captioning, an end-to-end neural network system that can automatically view an image and generate a reasonable description in plain English. Image captioning is based on a Convolutional

Neural Network that encodes an image into a compact representation, followed by a Recurrent Neural Network that generates a corresponding sentence. The model is trained to maximize the likelihood of the sentence given the image. It was found that the model is capable of generating approximately 90% of semantically correct captions given the generated caption have correct grammatical and syntactical structure, for those images with object occurred frequently throughout the training caption; approximately 77% for those with object occurred less frequently. In other words, the model is capable of producing captions that identifies the entities and relationships among the entities in an image providing there are sufficient diversity in the training data.

10.1 Related work

As mentioned earlier, still image captioning has attracted a large amount of interest from researchers, and other approaches have been created over time. An approach similar to the presented here (Karpathy et al., 2015), uses bi-directional RNNs to infer the latent alignments between segments of sentences and regions of images. (Kiros et al., 2013) use a feedforward neural network to predict the next word based on the previous and the image.

Another interesting research area is one which involves generative models, which attempt to flip the problem and generate images based on text descriptions (Reed et al., 2016). by using a GAN (Goodfellow et al., 2014).

10.2 Future work

Considering our limited computational power, we only have the data as a result of training on the relatively small Flickr8k dataset. It would be interesting to see the result if we use a larger dataset for the training. We hypothesized that our model will perform better with the increase of the training set. It would be able to capture the dependency of sequences more accurately. Another interesting topic for further research is to investigate the model architecture that is capable of handling unlabeled data, for both the image and the text.
As this work only used a LSTM RNN implementation, a suggested continuation would be implementing the same architecture using a GRU (Gated Recurrent Unit) RNN. It offers a slightly different internal structure that could better suit this task, and as a smaller number of gates is used, could speed up the training process.
Lastly, more encoder configurations could be analyzed, in order to check if a particular encoding size in the input can increase the model performance.

References

A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. NIPS,2012.

Y. LeCun, K. Kavukcuoglu, and C. Farabet. Convolutional networks and applications in vision. In Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on, pages 253256. IEEE, 2010.

Pascanu, R., Mikolov, T. and Bengio, Y. On the difficulty of training recurrent neural networks. In Proc. 30th International Conference on Machine Learning 1310 – 1318 (2013).

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. Nature, 323, 533–536.

P. Werbos. Backpropagation through time: what it does and how to do it. Proceedings of IEEE,1990

Y. LeCun and Y. Bengio. Convolutional networks for images, speech, and time-series. In M. A. Arbib, editor, The Handbook of Brain Theory and Neural Networks. MIT Press, 1995.

S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural Computation, 9(8), 1997.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning.

GloVe: Global vectors for word representation. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2014), pages 15321543, Doha, QA.

Oriol Vinyals, Alexander Toshev, Samy Bengio and Dumitru Erhan. Show and Tell: A Neural Image Caption Generator.

Papineni, Kishore and Roukos, Salim and Ward, Todd and Zhu, Wei-Jing. BLEU: A Method for Automatic Evaluation of Machine Translation.

Karpathy, Andrej and Fei-Fei, Li. Deep Visual-Semantic Alignments for Generating Image Descriptions.

Kiros, Ryan and Zemel, Richard S. and Salakhutdinov, Ruslan. Multimodal Neural Language Models

Reed, Scott and Akata, Zeynep and Yan, Xinchen and Logeswaran, Lajanugen and Schiele, Bernt and Lee, Honglak. Generative Adversarial Text to Image Synthesis

Goodfellow, Ian and Pouget-Abadie, Jean and Mirza, Mehdi and Xu, Bing and Warde-Farley, David and Ozair, Sherjil and Courville, Aaron and Bengio, Yoshua Generative Adversarial Networks