VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

IMAGE CAPTIONING WITH RECURRENT NEURAL NETWORKS

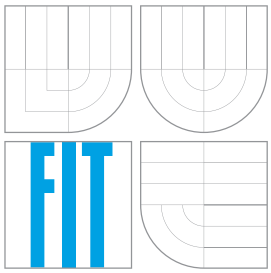
SEMESTRÁLNÍ PROJEKT
TERM PROJECT

AUTOR PRÁCE
AUTHOR

Bc. JAKUB KVITA

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

POPIS FOTOGRAFIÍ POMOCÍ REKURENTNÍCH NEURONOVÝCH SÍTÍ

IMAGE CAPTIONING WITH RECURRENT NEURAL NETWORKS

SEMESTRÁLNÍ PROJEKT
TERM PROJECT

AUTOR PRÁCE
AUTHOR

Bc. JAKUB KVITA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MICHAL HRADIŠ, Ph.D.

BRNO 2015

Abstrakt

Tato práce se zabývá automatickým generovaním popisů obrázků s využitím několika druhů neuronových sítí. Práce je založena na článcích z MS COCO Captioning Challenge 2015 a znakových jazykových modelech, popularizovaných A. Karpathym. Navržený model je kombinací konvoluční a rekurentní neuronové sítě s architekturou kodér–dekodér. Zakódovaný vektor obrázku se předává jazykovému modelu jako hodnota v paměti první vrstvy LSTM/GRU v síti. Práce zkoumá, jestli je model s takto jednoduchou architekturou schopen konkurovat ostatním současným modelům.

Abstract

In this paper we deal with automatic generation of image captions by using multiple kinds of neural networks. Study is based on the papers from MS COCO Captioning Challenge 2015 and character language models, popularized by A. Karpathy. Proposed model is combination of convolutional and recurrent neural network with encoder–decoder architecture. Vector with image representation is passed to language model as memory values of first LSTM layer of the network. This thesis investigate, whether model with such simple architecture can compete with other contemporary solutions.

Klíčová slova

rekurentní neuronové sítě, RNN, konvoluční neuronové sítě, CNN, popisování obrázků, LSTM, GRU, MS COCO, Torch, hluboké učení

Keywords

recurrent neural networks, RNN, convolutional neural networks, CNN, image captioning, LSTM, GRU, MS COCO, Torch, deep learning

Citace

Jakub Kvita: Image Captioning with Recurrent Neural Networks, semestrální projekt, Brno, FIT VUT v Brně, 2015

Image Captioning with Recurrent Neural Networks

Prohlášení

Prohlašuji, že jsem tento semestrální projekt vypracoval samostatně pod vedením pana Michala Hradiše, s uvedením všech literárních pramenů a publikací, ze kterých jsem čerpal.

.....
Jakub Kvita
February 27, 2016

Contents

Chapter 1

Introduction

When does a machine "understand" an image? One definition could be the following sentence: *A machine understand an image, when in can describe important content of the image.* This description should include present objects, their attributes and relation to each other. Determining the important content of the image can be quite difficult, even for humans, which have been trained for this task since they were born. However, deep learning techniques are proving to be quite successful in this kind of tasks. Similarly to people, these models require large amounts of training data, but later they can evaluate correctly even yet unseen situations.

Deep machine learning, sometimes under the name of neural networks[1], is branch of machine learning based on composing multiple non-linear functions to solve the task. As it is fundamentally different from the standard computer algorithms, it perform well on problems, which are unsuitable these algorithms. For example, neural networks have excellent performance in recognizing speech and images, writing stories and composing music. This work focuses on generating image descriptions in regular English sentences, which is also a task suitable for deep learning.

First, in chapter 2, I will first introduce neural networks and several key concepts which are used later in this work. In chapter 3 current state-of-the-art in the field of image captioning and summary of the key papers will be presented. I will list most popular frameworks and libraries for implementation of deep learning models and mention my experiments with them in the chapter 4. Using knowledge of previous chapters I will propose an image captioning model in chapter 5. Expectations of this proposal will be discussed in the concluding chapter 6.

[1] Terms *deep learning* and *neural networks* will be used in this work interchangeably.

Chapter 2

Neural networks

General idea of artificial neural networks emerged after World War II. Perceptron, as a single neuron unit, was created in 1958 by Frank Rosenblatt [41], but became popular only after combination with the backpropagation algorithm [4, 46]. At that time neural nets have not reached massive popularity, not because they are not working, but due to small computing power of machines back then and lack of datasets. Recently (after 2000) neural nets became popular again, under the name of "deep learning", to emphasize the use of several layers stacked on top of each other to create deep architectures, which are far more practical than shallow ones. During this reinvention, neural nets have been successfully applied in multiple fields like computer vision [17], speech recognition [15] and natural language modeling [34].

Various useful architectures and algorithms are now introduced almost every month. In this chapter, I will describe only a handful – recurrent neural networks with the LSTM and GRU units, and basics of convolutional neural nets.

2.1 Recurrent neural nets

Feedforward neural nets are extremely powerful models, which can be highly parallelized. Despite that, they can be only applied to problems with inputs and outputs, which have fixed dimensionality (e.g. one-hot encoding vectors). This is a serious drawback, as many of the real-world problems are defined as sequences with lengths that are unknown to us in beforehand. Soon recurrent neural networks were introduced and they proved to be very useful to this kind of task. There is vast amount of recurrent neural networks, many not suitable for sequential tasks like Hopfield network, which are very successful in specific tasks, but nevertheless not useful for us now.

Apart from classification, which can be more precise when using sequences, one of the most important tasks is next value prediction. This core task can be then extended very simply to predict arbitrary number of future values. Prediction problems are all around us, from the weather forecast and stock market prediction to the autocomplete in smartphones or web browsers.

We can understand recurrent neural networks as very deep forward nets with shared weights. It is called RNN unrolling and it is described in figure 1. Layers of this very deep net spread in time, together with the input sequence. This is very innovative idea, which enabled training RNN with backpropagation through time. It also shows that, as very deep networks, they have vanishing or exploding gradient problem, which means that

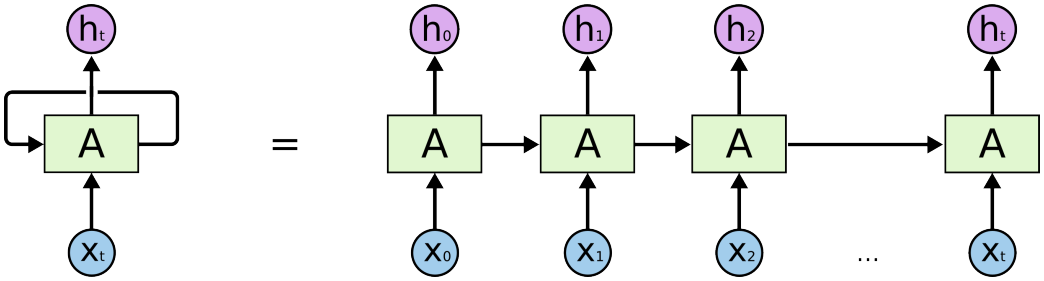


Figure 1: Unrolling of the recurrent neural net. [38]

the network is not able to learn long-term dependencies, even though in theory it should. This is a serious issue, which is caused by iterating many times over the weights and the activation function with derivatives > 1 (exploding gradient) or < 1 (vanishing gradient). Gradient then dies out and learning stops for distant dependencies. Among others this problem has been solved by the LSTM unit described in part 2.1.1, which is most popular now and following research resulting in GRU described in part 2.1.2.

2.1.1 LSTM – Long Short-Term Memory

Long Short-Term Memory nets are special kind of recurrent network, capable of learning long-term dependencies. This architecture was introduced by Hochreiter & Schmidhuber [19] after prior research of vanishing gradient problem [18]. Later architecture was refined and popularized by other researchers [12, 13] and nowadays LSTM is most popular RNN architecture used.

The LSTM unit was designed to remember a value for an arbitrary length of time. It contains gates that determine when the input is significant enough to remember, when it should keep or forget the value, and when it should output the value. To understand the flow of data, see the diagram of a simplified LSTM unit on the figure 2. All the gates can be described by the following series of equations.

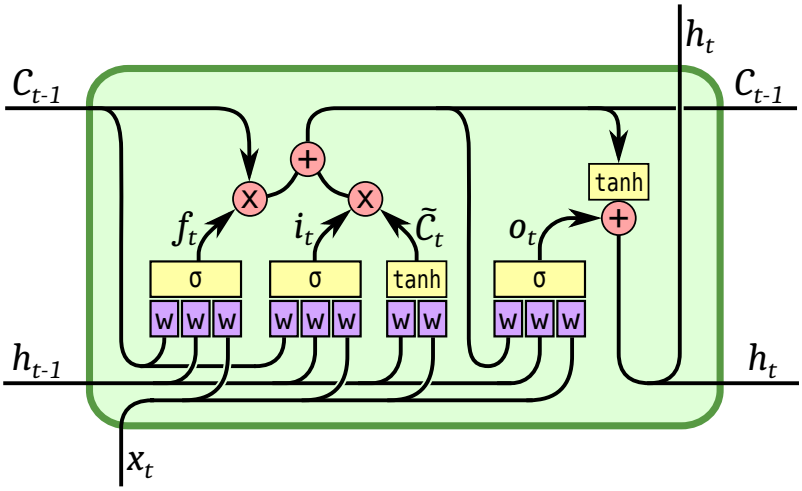


Figure 2: Variation of the LSTM unit.

$$
\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}C_{t-1} + b_i) && (1)\\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}C_{t-1} + b_f) && (2)\\
\widetilde{C}_t &= \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) && (3)\\
C_t &= f_t \odot C_{t-1} + i_t \odot \widetilde{C}_t && (4)\\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}C_t + b_o) && (5)\\
h_t &= o_t \odot \tanh(C_t) && (6)
\end{aligned}
$$

$$
\sigma(x) = \frac{1}{1 + e^{-x}} \qquad (7)
$$

In each time slice the unit is using current input x_t, last stored value c_{t-1} and unit output h_{t-1} to compute next state c_t and output h_t. Variables i_t, f_t, o_t denotes value of input, forget and output gates which are used to control the information flow.

LSTM based on these equations is using total of 11 weight matrices and 4 bias vectors for computations and sigmoid function σ defined in the equation (7) and the operation $\odot$ denotes the element-wise vector product. Equations described in this work are not the only way how to create an LSTM unit, but they will be used later while implementing the proposed model. Some of the versions are omitting "peephole connections", which allows gates to look at stored value C_{t-1}, C_t or include only some of them.

Training of the LSTM based network can be performed effectively by standard methods like stochastic gradient descend in the form of backpropagation through time. Major problem with vanishing gradients during training described earlier is not an issue as backpropagated error is fed back to each of the gates.

2.1.2 GRU – Gated Recurrent Unit

Gated Recurrent Unit [6] is slightly more dramatic variation on the LSTM theme. It combines hidden state of the unit h_t with the saved value C_t, merges input and forget gates into one update gate and removes peephole connections. These changes are simplifying standard LSTM models, but not at the expense of performance, and cause rapid growth in popularity. Diagram of the GRU unit is on the figure 3.

$$
\begin{aligned}
r_t &= \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r) && (8)\\
z_t &= \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z) && (9)\\
\widetilde{h}_t &= \tanh(W_{xh}x_t + W_{hh}(h_{t-1} \odot r_t) + b_h) && (10)\\
h_t &= (1 - z_t) \odot \widetilde{h}_t + z_t \odot h_{t-1} && (11)
\end{aligned}
$$

$$
\sigma(x) = \frac{1}{1 + e^{-x}} \qquad (12)
$$

Equations describe a version of GRU unit used in this work, with sigmoid function σ defined in equation (12). The operation $\odot$ again denotes the element-wise vector product.

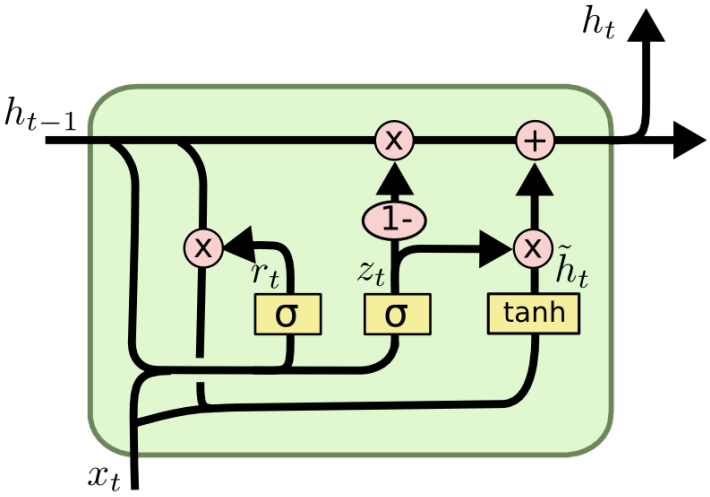


Figure 3: Variation of a GRU unit. [38]

While it is using only 4 weight matrices, 3 biases and just 1 state variable, researchers studied whether this can achieve at least same performance as previous LSTM unit.

Last year, study by Chung [7] was done, where different types of recurrent units were compared on the polyphonic music datasets. In this task LSTM and GRU were significantly better than all the other architectures, with GRU slightly in the lead. Generally, researchers agree that most of the LSTM variations, including GRU, are roughly on the same performance level. According to [16] GRU is an average variation, slightly better than vanilla LSTM, with much simpler architecture.

In paper [24], which emphasized variety of tasks and the data, GRU outperformed LSTM unit on all tasks with the exception of language modeling. There are multiple approaches to model languages and in this work I will explore different type than the one mentioned in Jozefowicz's paper [24]. More will be explained in following chapters. Interestingly they also found that LSTM nearly matched the GRU's performance, when its forget gate bias was initialized to 1 and not to naive initialization around 0. It is also worth mentioning that Jozefowicz in his paper discovered several architectures similar to GRU, but with slightly better general performance. They were found by evolutionary algorithm working on candidate architectures represented by the computational graph.

2.1.3 Language modeling and word embeddings

With the addition of LSTM units, recurrent neural nets quickly showed good performance in many different types of sequence processing like speech recognition from sound waves, signal prediction and language modeling. These result were further improved when researchers started stacking LSTMs on top of each other like pancakes.

Text is represented by discrete values and is usually presented to network in form of input vectors with one-hot encoding[1]. If we have a task with K classes, class i will be represented by a vector V of length K. All the values of V will be switched off to 0, except V_i, which will be 1. Vector V is simultaneously a degenerated multinomial probability distribution of the current input. If the output has the same shape as input, it can be

[1]One-hot encoded vector has exactly one high ('1') value and all the others low ('0').

simply created by softmax function at the output layer. Result will be proper multinomial distribution of next value, given current value.

At this point it is necessary to decide what will classes and defined vectors represent. In most cases, text prediction is performed at the word level. K is hence the number of words in the dictionary. This can cause some problems, as in bigger tasks dictionary often exceeds 100 000 records. This many classes require huge amount of training data to properly cover all the cases and high computational cost of the softmax layer is also an issue. This text representation cannot be used for texts not containing separate words, like multi-digit numbers. Nevertheless, state-of-the-art models have been using word-level representation. One of the advantages is no need to teach the net proper forms of the words. The net does not have to remember, how to spell the words properly and can learn other, more useful, features.

To solve the problem with extremely long input vectors, set of techniques called *word embedding* were developed. They map words from the vocabulary to suitable vectors of real numbers in high dimensional space (around 50–1000 dimensions). Chosen vectors cannot be random, they are meaningful in order of performing some following task. For example Skip-gram model [33] mapped 783 millions words to vectors of 300 real numbers, while creating reasonable relationships between them.

Character level modeling has been considered and used as an alternative to word-level, but so far had slightly worse performance. Regardless, it is still considered as an option, because it has much simpler representation of input and output. Consider roughly 45 characters in English text and over 50000 words created from them. Character level network is also more suited for Czech or Russian and other fusional[2] languages, which heavily use prefixes and suffixes to create new words. This is also an ability, which cannot be overlooked, as it is not available for word level. Character level models have usually smaller vocabulary size and have to be trained longer, as they need to learn spelling of the words on top of the same features of word level. With the properly trained character level model we can benefit from its much greater generative abilities, than we can achieve with word-level.

2.2 Convolutional neural nets

Feed-forward neural nets together with backpropagation algorithm showed very useful for range of tasks and it has even been proven [8, 21] they can approximate any continuous function. However, they were not very good in recognizing objects presented visually. As every unit is connected to large amount of units in the next layer (or all of them in fully-connected layers), the number of weights grows rapidly with the size of the problem and even more with the dimensionality. All these problems are manifesting even in image processing with only two dimensions. Convolutional neural nets (CNN) were introduced as a solution to reduce the number of parameters involved, while exploiting spatial constraints in the input.

Ideas of convolutional nets took inspiration from neurobiology, more precisely from the organisation of neurons in visual cortex of the cat. They were first used in the work of Homma [1] to process temporal signal. Later their design was improved by LeCun et al. [31]. Different CNN architecture was proposed by Graupe [14] for decomposition of one-dimensional EMG signals. Convolutional nets can be also used to natural language

[2]Fusional language is a type of language distinguished by its tendency to overlay many morphemes to denote grammatical, syntactic, or semantic change.

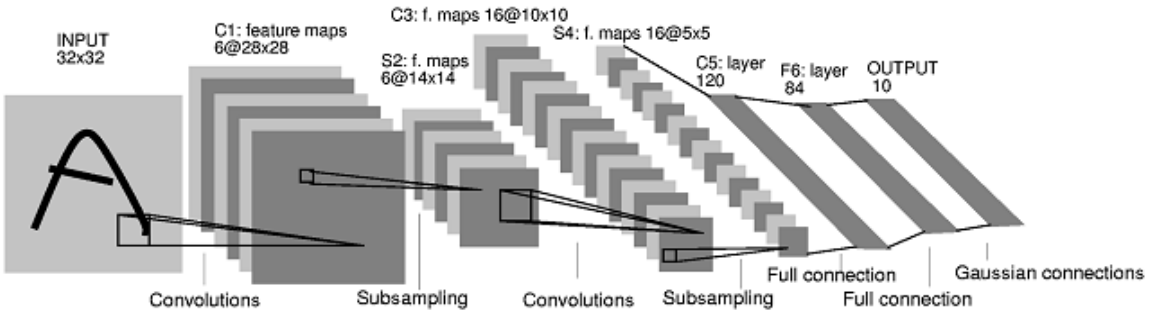


Figure 4: LeNet-5 architecture. [31]

processing [28] and to analysis of three-dimensional data like videos [23] or volumetric data (eg. 3D medical scans), but that is not as common as image processing.

Basic architecture of CNN can be described as the following process:

1. Convolve several small filters on the input.
2. Subsample this space of filter activations.
3. Repeat steps 1 and 2 until you are left with a sufficiently high level features.
4. Use a standard feed-forward neural net to solve the task, using the features as input.

Convolutional layer, which is most important and gave CNN their name is essentially the same as mathematical convolution used elsewhere. Here it means to apply a 'filter' over an input at all possible offsets. This filter - in image processing and computer vision called kernel - has a layer of connection weights with the same dimensionality as the input, but with much smaller size. Despite the fact that there is many connections, which are even overlapping, the weights are tied together and during training only handful of parameters per filter need to be updated. It is also possible to stack filters on top of each other to create more powerful architectures, but as they do not reduce dimensionality significantly, output has the same size as the input. Second type of a layer has been introduced to improve it - subsampling.

Subsampling, or max pooling in this version, is a simple operation that takes small non-overlapping grid of the input tensor and outputs the maximum value of each part. By putting this operation in between the convolutional layers, we can detect higher level features than without it.

Last type of unit commonly used in CNN is rectified linear unit (ReLU), which provides nonlinearity and improves overall performance [22, 36]. One of the first and most famous examples of convolutional neural net is LeNet[3] [31], which recognize handwritten digits from the MNIST database[4]. Architecture of LeNet-5 is on the figure 4.

[3]Demos and examples of LeNet: http://yann.lecun.com/exdb/lenet/
[4]MNIST database website: http://yann.lecun.com/exdb/mnist/

Chapter 3

Image caption generation

Scene understanding is one of the fundamental and most difficult tasks of computer vision. Being able to automatically generate image or video captions in regular text could have great effect. However, it is much more complicated than simple classification or object recognition, because the model also need to understand relations between the recognized objects and capture that correctly in the captions.

In this chapter I will do an overview of approaches to this task and more closely describe latest papers on which is this work based (section 3.1). Following parts cover datasets (3.2) and evaluation procedures (3.3) most commonly used for this task.

3.1 Related Work

Currently, neural networks are most heavily used to generate captions. Before them two main approaches were common. The first one used caption templates, which were filled by detected objects and relations. Second was based on retrieving similar captions from database and modifying them to fit current image. Question of similarity ranking has been addressed by many papers, which are based on the idea of joint embedding vector space for both images and captions [27]. Similar descriptions are in this space close to each other.

Both approaches above usually included generalization step to remove information relevant only to current image, for example names. They are quite successful in describing images, but they are heavily hand-designed and their text-generation power is fixed on the database/embedding and is not able to describe previously unseen compositions of objects. Over time these approaches fell out of favor to now dominant neural network methods.

Many of the methods using neural nets are inspired by successes in training of recurrent nets for machine translation. It is worth mentioning Sutskevers work [42], which studied general sequence to sequence mapping by converting input sequence to vector of fixed length. Vector is then decoded to output sequence. This encoder–decoder architecture is closely related to autoencoders and work of Kalchbrenner and Blunsom [25], who were first to map the entire input sequence to vector.

The introduced encoder–decoder architecture is important to the captioning task, because we can interpret image description problem as a translation from an image to a sentence. In this case, encoder part of the model is usually convolutional neural net, as they are excellent in the image classification [43]. Decoder part is similar as in machine translation models – type of a RNN or LSTM, as the output for both tasks is essentially same.

One of the most interesting event in this field is MS COCO Captioning Challenge[1] in which many of the state-of-the-art researchers compete directly against each other. Most of the works described further have participated in this challenge.

3.1.1 Show and Tell: A Neural Image Caption Generator

Model from this paper [45], made by Google researchers, tied for the first place in MS COCO Captioning Challenge with the following Microsoft model 3.1.2. The main idea for this work is to use recent advancements in machine translation and apply them for image captioning. Model uses encoder–decoder architecture, with CNN for the encoder part and RNN for the decoder part, as described earlier, which is trained to maximize the likelihood $p(S|I)$ of producing a target sequence of words $S = \{S_1, S_2, ...\}$ given an input image I.

Used convolutional neural net has been pre-trained for an image classification task and last hidden layer of this network has been used as an input to the RNN decoder that generates word sequences. The decoder part of the network is made of LSTM units based on the following equations:

$$
\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1}) && (1)\\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1}) && (2)\\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1}) && (3)\\
c_t &= f_t \odot c_{t-1} + i_t \odot \tanh(W_{xc}x_t + W_{hc}h_{t-1}) && (4)\\
h_t &= o_t \odot c_t && (5)
\end{aligned}
$$

Notation is same as in the chapter 2, σ is the sigmoid function and the operation $\odot$ denotes the element-wise vector product.

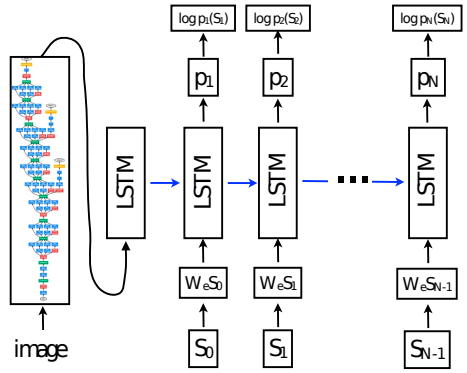


Figure 1: 'Show and Tell' image captioning model. [45]

RNN produces output on the word level, using word embedding vectors [33], one at a time. Complete procedure and architecture of the model with the unrolled RNN, word embeddings and *softmax* layer is on the figure 1. Computing procedure of the model equal to the image is on the following equations:

[1]MS COCO Challenge: http://mscoco.org/dataset/#captions-challenge2015

$$x_{-1} = CNN(I) \quad (6)$$
$$x_t = W_e S_t, \quad t \in \{0 \dots N-1\} \quad (7)$$
$$p_{t+1} = LSTM(x_t), \quad t \in \{0 \dots N-1\} \quad (8)$$

Each word is represented as a one-hot vector S_t of the size equal to size of the dictionary. Special start word S_0 and stop word S_N designated to mark start and end of the sequence are also present. Both the image and the words are mapped to the same space. Input image embedding is fed to the LSTM only once, at $t = -1$.

The CNN component of the model has been initialized to an ImageNet trained model, which helped quite a lot in terms of generalization. Word embeddings were left uninitialized (initialized randomly) as they did not observed significant gains while using large corpus. Dropout and ensebling used during training gave minor improvements. Model has been trained using stochastic gradient descent with fixed learning rate and no momentum. For the embeddings and the LSTM memory 512 dimensions were used. During the inference, beam search has been used to improve the results.

3.1.2 From Captions to Visual Concepts and Back

This paper [11] took quite a different approach than a previous one, however both tied for the first place in the competition. This model is not end-to-end, it has three stages. First, it learns to extract nouns, verbs and adjectives from regions in the image. These words come from the vocabulary constructed by using 1000 most common words in the training captions. By running detector on the image regions, model is able to produce bag of bounding boxes. Each bounding box represent location of word in the image. Network pretrained on ImageNet is used for initialization.

Second, these extracted words guide a language model to generate text, which include these words. The maximum entropy (ME) language model estimates the probability of a word w_i conditioned on the preceding words, as well the words with high likelihood detections, yet to be mentioned. This encourages all the words to be used, while avoiding repetitions. A left-to-right beam search similar to [40] is used during generation. After extending each sentence with a set of likely words, the top N sentences are retained and the others pruned away. The process continues until a maximum sentence length L is reached.

Third, candidate captions are re-ranked using MERT [37] (Minimum Error Rate Training) and the best one is selected. MERT uses linear combination of features computed over the sentence, for example log-likelihood of the sequence or its length. One of the features is Deep Multimodal Similarity Model (DMSM) score, which measures similarity between images and text. The DMSM is model proposed in this paper, which learns two neural networks that map images and text fragments to a common vector representation. These vectors are used to compute the cosine similarity score, which is one of the features fed to MERT.

Direct comparison of this approach with the first one, presented in the paper description 3.1.1, is in paper [9] by the same authors. They examine the issues of both approaches and achieve state-of-the-art performance by combining key aspects of RNN and ME methods.

3.1.3 Show, Attend and Tell: Neural Image Caption Generation with Visual Attention

Work [47] of researchers from universities in Toronto and Montreal was inspired by recent work in machine translation and introduced an attention based model. Attention is one of the most interesting parts of the human visual system. Rather than compressing an entire image into a static representation, attention allows for salient features to dynamically come to the forefront as needed. Proposed model has encoder-decoder architecture. Encoder part use a convolutional neural network to extract set of feature/annotation vectors (not just one). Each of the vectors correspond to a part of image. Features from a lower convolutional layer are used to obtain a correspondence between them.

Decoder part is a LSTM network working of the word level, which generates, apart from the word of the output, a context vector - a dynamic representation of the relevant part of the image at time t. The paper explored two attention mechanisms computing the context vector from the annotation vectors. First is the stochastic "hard" mechanism, which interprets the values in the context vector as the probability that corresponding location is the right place to focus for producing the next word. Second is deterministic "soft" mechanism introduced in [2]. Which gives the relative importance of the location by blending values for all annotation vectors together. This method is fully trainable by standard back-propagation methods.

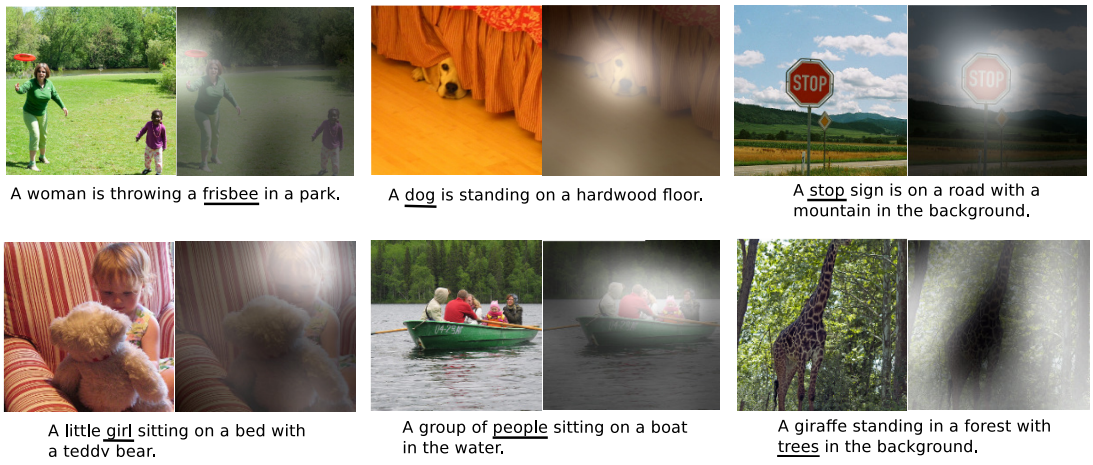


Figure 2: Examples of attending the correct object while generating the word. [47]

Paper is also showing how we can gain insight and interpret the results of the model by visualising where and what the attention was focused on. Examples of the correct visualisations is on the figure 2 and the wrong ones on the figure 3. Visualisations show that model can attend even "non-object" regions. This adds an extra layer of interpretability to the output. The model learns alignments that correspond very strongly with human intuition. Especially in the examples of mistakes, we can understand why those mistakes were made.

CNN trained on ImageNet without finetuning was used for the decoder part. Model was trained with several algorithms and researchers found that for Flickr8k dataset, RMSProp worked best, while for Flickr30k and MSCOCO datasets, Adam [29] algorithm was used. Performance during training was also improved by creating minibatches of sentences with same length, which greatly improved convergence speed.

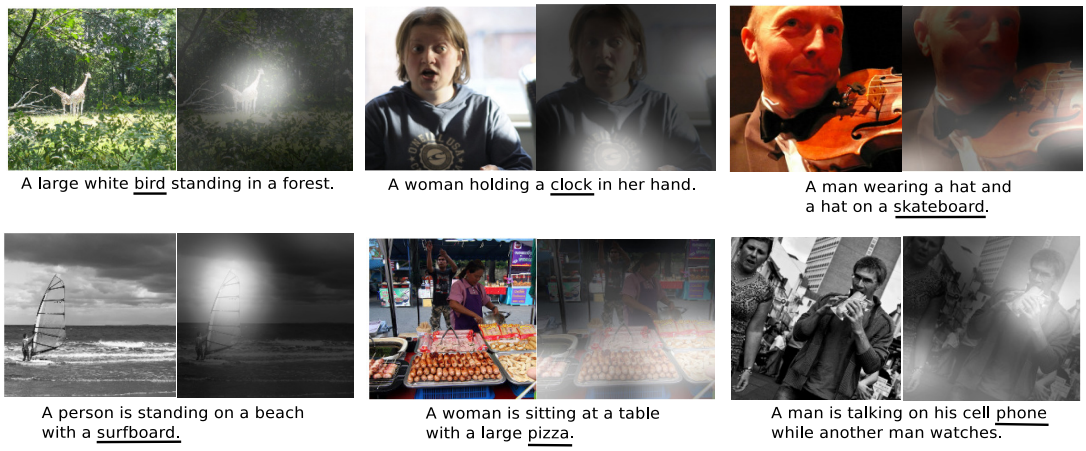


Figure 3: We can use attention to understand what the model saw while creating wrong caption. [47]

3.1.4 Long-term Recurrent Convolutional Networks for Visual Recognition and Description

The research group from Berkeley in paper [10] presented long-term recurrent convolutional network (LRCN), which combines convolutional and long-range temporal layers for several tasks. It is possible to apply the introduced network to recognize activity performed on the video (sequential input $\mapsto$ fixed output), generate description of the image (fixed input $\mapsto$ sequential output) or describe video (sequential input $\mapsto$ sequential output).

Architecture of the proposed model is similar to the one from part 3.1.1, only output of the CNN is fed to the LSTM in each time step. According to the task specification model can use separate convolutional networks, different for each time step, each with specific input or single CNN through all the time steps. Example is on the figure 4.

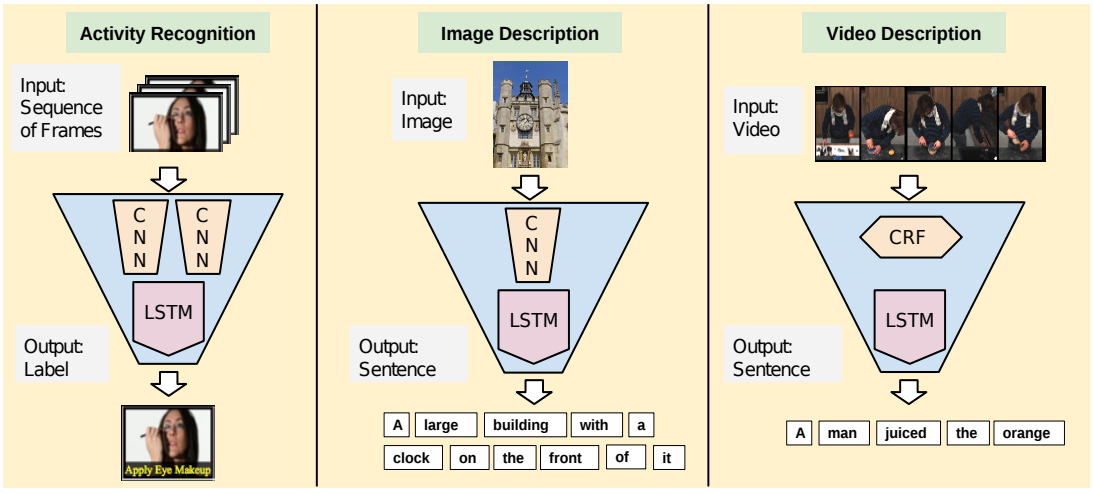


Figure 4: Task-specific instantiations of the LRCN model. [10]

3.2 Datasets

Big datasets are necessary requirement in training recurrent neural nets, together with sufficient computing power. As access to machines and hardware suitable for training has been made extremely easy, obtaining enough data become the biggest problem. All the descriptions in the image captioning datasets have to be human generated, which is very expensive. This is one of the reasons, not many specialized datasets are created.

There are two main options how to get images and captions. First, using user-generated data from an online service, most commonly Flicker. However, captions are not made specifically for the task and could be prone to error. Second option is to create captions directly for use in the dataset. Amazon Mechanical Turk[2] is heavily used for this task. All datasets mentioned here are created this way.

Flickr8k [20] was one of the first datasets created for this purpose. It has been later expanded into Flickr30k [48]. MS COCO [5] is dataset created by Microsoft for their captioning challenge. CIDEr [44] datasets PASCAL-50S, ABSTRACT-50S are youngest mentioned, designed specifically for evaluation with the CIDEr metric.

Table 1: Image captioning datasets.

Name	**Images**	**Captions per image**	**Note**
Flickr8k[3]	8 092	5	Focused on people or animals (mainly dogs) performing some specific action.
Flickr30k[4]	31 783	5-6	An extension of Flickr8k dataset.
MS COCO[5]	120 000	5	Images are divided - 80 000 for training and 40 000 for testing purposes.
PASCAL-50S[6]	1 000	50	Built upon images from the UIUC Pascal Sentence Dataset.
ABSTRACT-50S[7]	500	50	Built upon images from the Abstract Scenes Dataset. No photos.

[2]Amazon Mechanical Turk is crowdsourced Internet marketplace to perform tasks that computers are currently unable to do.

[3]Flickr8k project: http://nlp.cs.illinois.edu/HockenmaierGroup/8k-pictures.html

[4]Flickr30k project: http://shannon.cs.illinois.edu/DenotationGraph/

[5]MS COCO project: http://mscoco.org/dataset/

[6]PASCAL-50S and ABSTRACT-50S: http://ramakrishnavedantam928.github.io/cider/

[7]See footnote 6.

3.3 Evaluation

Recent progress in fields like machine translation, which are very similar to image captioning, caused spike of interest in evaluating regular text output accuracy. Although it is sometimes not clear if a description of an image is best option available, some degree of assessment is possible. The best results can be obtained by asking live raters to give a score on the usefulness of each description. Subjective scores can vary, but it can be averaged by giving same description to multiple raters. However this method consumes tremendous amount of time and usually external raters are necessary. Tools like Amazon Mechanical Turk can be used to great extent, but need for automated tools is evident.

3.3.1 Automated metrics

Assuming that one has access to human generated captions, which is ground truth in our case, completely automated metrics exists. Even though all of them compute how alike are generated to human descriptions, different approaches are used. One metric can use several different settings with slight changes in the algorithm. This raises the question, how can we compare results of different works, despite using the "same" evaluation method. Microsoft group of researchers addresses this issue in [5]. They created an evaluation server[8] which has many automated metrics, with several configurations, including all mentioned here. It will serve as a reference point for comparing image captioning models.

The most commonly used metric has been BLEU (Bilingual Evaluation Understudy) [39], which was created in 2002 to evaluate quality of machine translated text from one language to another. Scores are computed on individual segments, usually sentences. BLEU has high correlation with human judgments and is still highly popular even for captioning tasks. However, it is becoming outdated as automatic methods are now outperforming humans. Four different variations of BLEU are used in MS COCO evaluation server.

METEOR (Metric for Evaluation of Translation with Explicit ORdering) [30] is another metric for the evaluation of machine translation from 2007. It was designed to fix some problems of the BLEU metric and it can also look for synonyms and perform stemming on input words.

Last year, metric designed directly to caption evaluation called CIDEr (Consensus-based Image Description Evaluation) [44] was introduced. This is still new metric, but with growing popularity as it correlate well with human judgment. Main idea of this metric is that given enough captions for the same image, metrics perform better. This can be seen in datasets introduced with it (see part 3.2).

[8] MS COCO evaluation server: http://mscoco.org/dataset/#captions-upload.

Chapter 4

Tools and Experiments

In this chapter I will describe tools and frameworks, which are used to implement deep learning models. *Torch* will be discussed separately in its own section as it was a tool I used for experimenting with language modeling - predicting next character in text, which is second part of this chapter.

One of the tools most academic researchers in deep learning rely on is *Theano*[1] [3], which is Python library that works with mathematical expressions and matrices. It is built upon *NumPy* to handle multidimensional arrays and compiles expressions before use for efficient computation. Theano can be quite intimidating and non-intuitive for some people, as it is focused on researchers and creating new architectures. For this reason, many packages and libraries has been created on top of Theano to simplify and streamline development of standard models. Among most popular are *Keras*[2], *Lasagne*[3], *Blocks*[4], which are open-source and available on GitHub, and *PyLearn2*[5].

Next Python library, independent of Theano, is *TensorFlow*[6], a tool from Google, released last year. It is used to process symbolic data flow graphs on many different types of machines, ranging from multiple GPU computers to smartphones. Interesting feature is ability to perform partial subgraph computation, which allows distributed training of the neural network. TensorBoard is a tool worth mentioning, as it provides visualisations of training and evaluation of the model, tool missing in most of the other libraries.

Python tools with the engine implemented in C/C++ are not the only ones. *Caffe*[7] is well-known and widely used library with API in C++. It performs very well in image classification with neural nets and can be used as a source of many pre-trained models hosted on the Model Zoo[8] site. It is also possible to use *Deeplearning4j*[9], which is deep-learning library written for Java and Scala, and a lot of others. Nowadays, libraries are introduced almost every month as this field is very live, which also means not everything is implemented in every framework and users need to follow news about their tools.

[1] Theano: http://deeplearning.net/software/theano/
[2] Keras: https://github.com/fchollet/keras
[3] Lasagne: https://lasagne.readthedocs.org/en/latest/
[4] Blocks: https://github.com/mila-udem/blocks
[5] PyLearn2: http://deeplearning.net/software/pylearn2/
[6] TensorFlow: https://www.tensorflow.org/
[7] Caffe: http://caffe.berkeleyvision.org/
[8] Caffe Model Zoo: https://github.com/BVLC/caffe/wiki/Model-Zoo
[9] Deeplearning4j: http://deeplearning4j.org/

4.1 Torch

Torch[10] is an open source scientific computing framework and machine learning library for the Lua. Underlying implementation is using extremely fast LuaJIT and C, but no need to code in C is required. Torch is not as popular in academic environment as Theano, but it is used by several large companies including Google & DeepMind, Facebook and IBM, which also contribute to the project. Apart from the companies, Torch has a large ecosystem of community-driven packages[11] with almost every tool needed for machine learning, computer vision and signal processing, and wide range of utilities. In the rest of this section I will describe fundamental Torch packages, which are necessary for my research.

The core package of Torch is *torch*, which is installed together with the library. It contains data structures for multi-dimensional tensors and operations over them. This is the most important part, as almost every package depends on them. Additionally, it provides many utilities for accessing files, serializing objects, processing command-line parameters and other useful utilities.

4.1.1 nn, nngraph

The base Torch provides necessary math structures, but the *nn* package allows simple creation of neural networks with a common `Module` interface. Module represents a layer of the network, which is the building block of the nets in Torch. Layers have `forward()` and `backward()` method and can be joined together by module composites `Sequential`, `Parallel` and `Concat`. These components allows creation of arbitrary graphs.

The *nn* package also contains loss functions, which are subclasses of `Criterion`. Classes `ClassNLLCriterion` and `CrossEntropyCriterion` contain common cross-entropy classification criterion. Other regression and embedding criterions are also available together with simple method to train the network with stochastic gradient descent.

Creating networks with complex graphs is quite complicated with the *nn*. To make it easier, *nngraph* package has been introduced. *nngraph* bundles *nn* modules into graph nodes, which are linked together by specifying inputs and outputs. Graphs can be visualized by `dot()` method and exported to vector graphics.

Both packages are sufficient and provide even advanced features like weight-sharing or weight-tying. They are mainly focused on feed-forward and convolutional networks. Creating RNNs is possible, however it is very complicated and labor-intensive.

4.1.2 rnn

Torch's *rnn* [32] package extends *nn* can be used to build recurrent neural nets, LSTM and GRU layers, and so on. The package handles the unrolling of a network and provides several options how to train a network. One of the ways is to use `backwardOnline()`, call `forward()` method repeatedly and then go `backward()` in the opposite order. Other option is to decorate model with `Sequencer` and feed the sequence to the network with only one `forward()` and one `backward()` call.

Package also provide module for implementing attention model [35]. It is worth mentioning that creating complex recurrent networks with the *nngraph* package might be difficult, as both packages are altering *nn* functionality and may collide.

[10] Torch: http://torch.ch/

[11] Wikipage with list of packages: https://github.com/torch/torch7/wiki/Cheatsheet

4.1.3 Other packages

As the number of community packages is enormous, I will list just a few, which are commonly used and related to the topic of this work:

- *optim* — Optimization library with SGD, AdaGrad, RMSProp and more.
- *nninit* — Weight initialisation schemes for nn modules
- *image* — Routines to load/save and manipulate images as *torch* tensors.
- *word2vec* — Pre-trained word embeddings and the distance metric.
- *cunn*, *clnn* — CUDA and OpenCL backends for *nn* package.
- *loadcaffe* — Method for loading Caffe models in Torch.

4.2 Predicting next character in sequence

As an application of the Torch library, I created a model predicting next character in a text, as described in part 2.1.3. Compilation of all Shakespeare's works have been used as an input data, as in Karpathy's post [26].

Models used one-hot encoding for the input and probability distribution of the next character in sequence as an output. Networks have four recurrent layers (LSTM or GRU), each with 300 nodes, followed by linear and logarithmic softmax layer. Architectures are on the images 1 and 2.

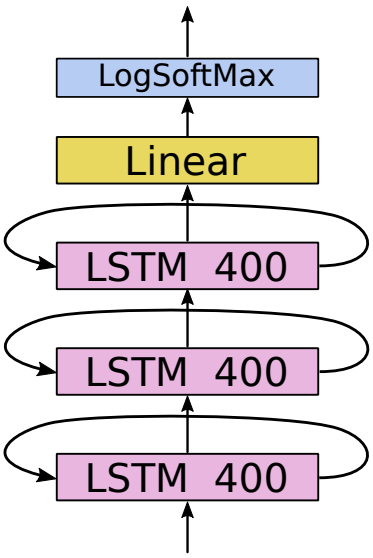


Figure 1: Architecture of the LSTM model.

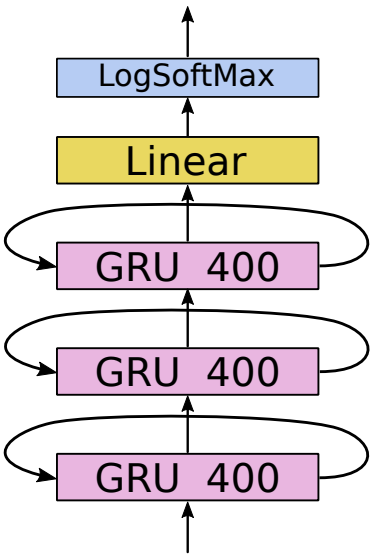


Figure 2: Architecture of the GRU model.

LSTM model was trained with stochastic gradient descent with Nesterov's momentum on minibatches of 20 subsequences, each with 40 characters. GRU model was trained with SGD without momentum, however, weight decay was used. After couple of hours, models were producing following samples:

Table 1: Predictions of the models.

LSTM model	QUEEN ELIZABETH: Whyou art me yough to hell three, ?gecT blswance'd wv reign Mantuas arm onB could hold thy anurrem tranioh newhful heart chnermwer than you are devnrcewMrnistCman yeT Norusezood with my chance? CAMILLO: Sirrah, that Is, my state now a haaaticaply. PETRUCHIO: Fellow me to never ae my tru sin someth
GRU model	GLOUCESTER: An ambatch the rest persure of methink Be father, a damned, skish a thy daughed, let tites an oar that your angpules dis cur,'t, my pl easons now many he tnoon king not what you must his duke, on to jello comforts of the welceful are whyself. Ty succemence, conves, our son, to choot this nimneat? First Muldiagoud Regerand: We'll not of a hates, I not have naturing at worthy Dore on Euncus, let my gental till rap it l ake, against thee, who was the labour tillows Henry: welcome. DUKE OF YORK: It is not amiss, is that close, shows do not, agoing told his cold and with their torety, you and then, your minds, as I waily low a jewel

It is obvious that the models are not perfectly trained, but they have been able to pick up the form of text (names of speakers in upper case, followed by colon) and most of the short English words. Training was using only single desktop CPU and only for several hours. As I am arranging access to the GPU nodes, models are expected to improve simply by prolonging the training times, without changing the parameters.

Chapter 5

Model

As a result of the previous research, I propose a model for the image captioning, which has a encoder–decoder architecture. Image is encoded by a convolutional neural network into a vector, which is passed to a first recurrent layer. The decoder is created by several LSTM or GRU layers. Image vector will be saved as value in memory of the LSTM unit. Sentence will be generated on character level, without embeddings. Architecture is on the figure 1.

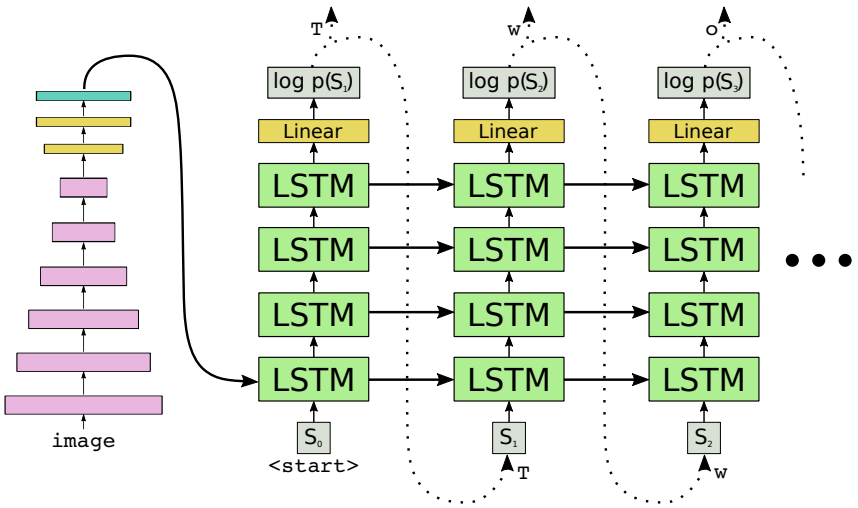


Figure 1: Architecture of the proposed model.

First question, this architecture is supposed to answer, is, whether model without the support of embeddings can compete with other contemporary solutions. Second is, how much will the model improve, if LSTM units are set with a large bias, as described in [24] and in part 2.1.2.

Model will work with one-hot character level encoding, with two extra special characters `<start>` and `<stop>`, which respectively mark beginning and the end of sequence. Size of the first LSTM layer will be between $100 - 1000$ items, number similar to the image embedding models mentioned in [45, 11].

Training will be performed on MS COCO and Flickr datasets, mentioned in part 3.2. I would also like to explore Adam [29] and several other optimization algorithms.

Chapter 6

Conclusion

Image captioning is problem more difficult than simple classification of images. Nowadays, deep neural networks are dominating the field almost exclusively. In this work I explained several features of neural networks, which are necessary for creation of image captioning models, and created an overview of state-of-the-art approaches to this problem. Deep learning is research area with a great need of data, therefore I also listed several biggest and most commonly used datasets and evaluation methods.

Character level prediction models were created as a building block for creating my own image captioning model, based on the information compiled. I proposed an architecture of the model based on several other approaches, with interest in character level language models instead of word embedding. I explore whether this simple architecture can compete with contemporary solutions focused on word level language models.

In several following months I am planning to work on the details of the convolutional network architecture, implementation and training of the proposed model. Project should be finished by May 2016.

Bibliography

[1] Les E. Atlas, Toshiteru Homma, and Robert J. Marks II. An Artificial Neural Network for Spatio-Temporal Bipolar Patterns: Application to Phoneme Classification. In *Neural Information Processing Systems*, pages 31–40. American Institute of Physics, 1988. 2.2

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR*, abs/1409.0473, 2014. 3.1.3

[3] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: A CPU and GPU Math Compiler in Python . In *Proceedings of the 9th Python in Science Conference*, pages 3 – 10, 2010. 4

[4] Arthur Bryson and Yu-Chi Ho. *Applied Optimal Control: Optimization, Estimation and Control*. Halsted Press book'. Taylor & Francis, 1975. 2

[5] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO Captions: Data Collection and Evaluation Server. *CoRR*, abs/1504.00325, 2015. 3.2, 3.3.1

[6] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *CoRR*, abs/1406.1078, 2014. 2.1.2

[7] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *CoRR*, abs/1412.3555, 2014. 2.1.2

[8] G. Cybenko. Approximation by Superpositions of a Sigmoidal Function. *Mathematics of Control, Signals, and Systems*, 2:303–314, 1989. 2.2

[9] Jacob Devlin, Hao Cheng, Hao Fang, Saurabh Gupta, Li Deng, Xiaodong He, Geoffrey Zweig, and Margaret Mitchell. Language Models for Image Captioning: The Quirks and What Works. *CoRR*, abs/1505.01809, 2015. 3.1.2

[10] Jeff 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. *CoRR*, abs/1411.4389, 2014. 3.1.4, 4

[11] Hao Fang, Saurabh Gupta, Forrest N. Iandola, Rupesh K. Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C. Platt, C. Lawrence

Zitnick, and Geoffrey Zweig. From Captions to Visual Concepts and Back. *CoRR*, abs/1411.4952, 2014. 3.1.2, 5

[12] Felix A. Gers and Jürgen Schmidhuber. Recurrent Nets that Time and Count. In *IJCNN (3)*, pages 189–194, 2000. 2.1.1

[13] Felix A. Gers, Jürgen Schmidhuber, and Fred A. Cummins. Learning to Forget: Continual Prediction with LSTM. *Neural Computation*, 12(10):2451–2471, 2000. 2.1.1

[14] D. Graupe, Ruey wen Liu, and George S. Moschytz. Applications of neural networks to medical signal processing. In *Decision and Control, 1988., Proceedings of the 27th IEEE Conference on*, pages 343–347 vol.1, December 1988. 2.2

[15] Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton. Speech recognition with deep recurrent neural networks. *CoRR*, abs/1303.5778, 2013. 2

[16] Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. LSTM: A Search Space Odyssey. *CoRR*, abs/1503.04069, 2015. 2.1.2

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *CoRR*, abs/1512.03385, 2015. 2

[18] Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen Netzen. *Master's thesis, Technische Universität München*, 1991. 2.1.1

[19] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, November 1997. 2.1.1

[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. 3.2

[21] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4:251–257, 1991. 2.2

[22] Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. What is the Best Multi-Stage Architecture for Object Recognition? In *Proceedings of the International Conference on Computer Vision (ICCV'09)*. IEEE, 2009. 2.2

[23] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3D Convolutional Neural Networks for Human Action Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):221–231, 2013. 2.2

[24] Rafal Józefowicz, Wojciech Zaremba, and Ilya Sutskever. An Empirical Exploration of Recurrent Network Architectures. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 2342–2350, 2015. 2.1.2, 5

[25] Nal Kalchbrenner and Phil Blunsom. Recurrent Continuous Translation Models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709. Association for Computational Linguistics, 2013. 3.1

[26] Andrej Karpathy. The Unreasonable Effectiveness of Recurrent Neural Networks. http://karpathy.github.io/2015/05/21/rnn-effectiveness/. [Accessed: 2015-12-30]. 4.2

[27] Andrej Karpathy and Fei-Fei Li. Deep Visual-Semantic Alignments for Generating Image Descriptions. *CoRR*, abs/1412.2306, 2014. 3.1

[28] Yoon Kim. Convolutional Neural Networks for Sentence Classification. *CoRR*, abs/1408.5882, 2014. 2.2

[29] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980, 2014. 3.1.3, 5

[30] Alon Lavie and Abhaya Agarwal. METEOR: An Automatic Metric for MT Evaluation with High Levels of Correlation with Human Judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, pages 228–231, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. 3.3.1

[31] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998. 2.2, 4, 2.2

[32] Nicholas Léonard, Sagar Waghmare, Yang Wang, and Jin-Hwa Kim. rnn : Recurrent Library for Torch. *CoRR*, abs/1511.07889, 2015. 4.1.2

[33] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. *CoRR*, abs/1301.3781, 2013. 2.1.3, 3.1.1

[34] Tomas Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, September 26-30, 2010*, pages 1045–1048, 2010. 2

[35] Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. Recurrent Models of Visual Attention. *CoRR*, abs/1406.6247, 2014. 4.1.2

[36] Vinod Nair and Geoffrey E. Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. In Johannes Fürnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010. 2.2

[37] Franz Josef Och. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, volume 1, pages 160–167. Association for Computational Linguistics, 2003. 3.1.2

[38] Christopher Olah. Understanding LSTM Networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Accessed: 2015-12-20]. 1, 3

[39] 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*, ACL '02, pages 311–318, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. 3.3.1

[40] Adwait Ratnaparkhi. Trainable Methods for Surface Natural Language Generation. *CoRR*, cs.CL/0006028, 2000. 3.1.2

[41] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958. 2

[42] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to Sequence Learning with Neural Networks. *CoRR*, abs/1409.3215, 2014. 3.1

[43] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. *CoRR*, abs/1409.4842, 2014. 3.1

[44] Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. CIDEr: Consensus-Based Image Description Evaluation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 3.2, 3.3.1

[45] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and Tell: A Neural Image Caption Generator. *CoRR*, abs/1411.4555, 2014. 3.1.1, 1, 5

[46] P.J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. Harvard University Press, 1974. 2

[47] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. *CoRR*, abs/1502.03044, 2015. 3.1.3, 2, 3

[48] Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78, 2014. 3.2