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 Bc. JAKUB KVITA
AUTHOR

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 — Bc. JAKUB KVITA
AUTHOR

VEDOUCÍ PRÁCE — Ing. MICHAL HRADIŠ, Ph.D.
SUPERVISOR

BRNO 2015

Abstrakt

Výtah (abstrakt) práce v českém jazyce.

Abstract

Výtah (abstrakt) práce v anglickém jazyce.

Klíčová slova

Klíčová slova v českém jazyce.

Keywords

Klíčová slova v anglickém jazyce.

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.

....
Jakub Kvita
January 1, 2016

Poděkování

Zde je možné uvést poděkování vedoucímu práce a těm, kteří poskytli odbornou pomoc.

Contents

Chapter 1

Introduction

Klasicky popis toho co se tady bude dit, jak je to dulezite, atd.

Chapter 2

Neural networks

General idea of neural networks was slowly emerging after World War II. Perceptron, as a single neuron unit, was created in 1958 by Frank Rosenblatt[1], but became popular only after creation of backpropagation algorithm in 1975. 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, rebranded as 'Deep Learning', because researchers realized that it is possible and very useful to stack neural nets on top of each other and create deep architectures, which are more practical than shallow ones. During this reinvention neural nets have been successfully applied in multiple fields like computer vision, speech recognition and natural language processing.

Since then various useful architectures and algorithms are now introduced almost every month. There is vast amount of various architectures and algorithms, in this chapter, I will describe only a couple – those used in this thesis.

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,

[1]The perceptron: A probabilistic model for information storage and organization in the brain. Rosenblatt, F. Psychological Review, Vol 65(6), Nov 1958, 386–408.

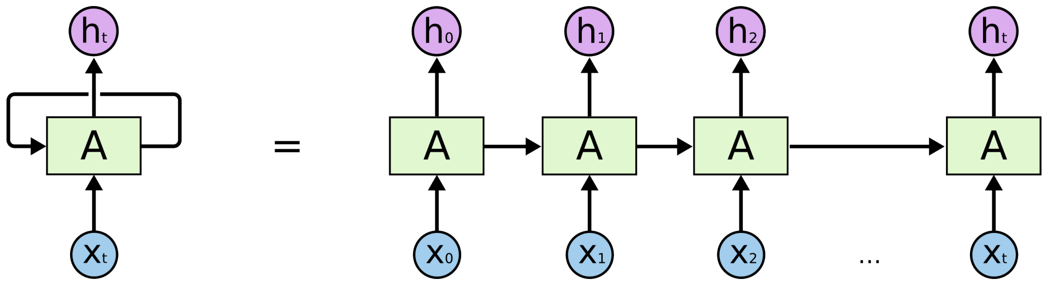

Figure 1: Unrolling of the recurrent neural net. (C. Olah 2015 [13])

Mam to tady rozebrat vice?

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 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 (1997) in [6] after prior research of vanishing gradient problem. Later architecture was refined and popularized by other researchers and nowadays LSTM is most used and popular RNN architecture used.

The LSTM unit designed that it can 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 is on the figure 2.

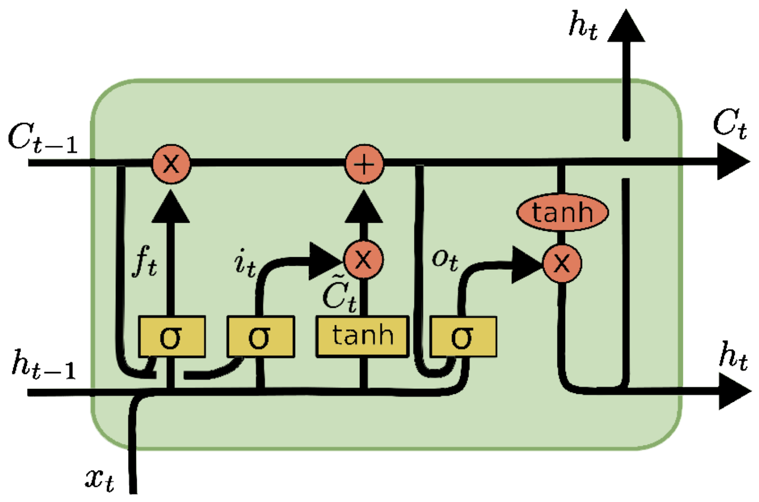

Figure 2: Variation of the LSTM unit. (C. Olah 2015 [13])

All these gates can be described by series of equations (1)→(6). 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.

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

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

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 is slightly more dramatic variation on the LSTM theme from 2014 paper [1]. 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.

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

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

Equations (8)→(11) 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

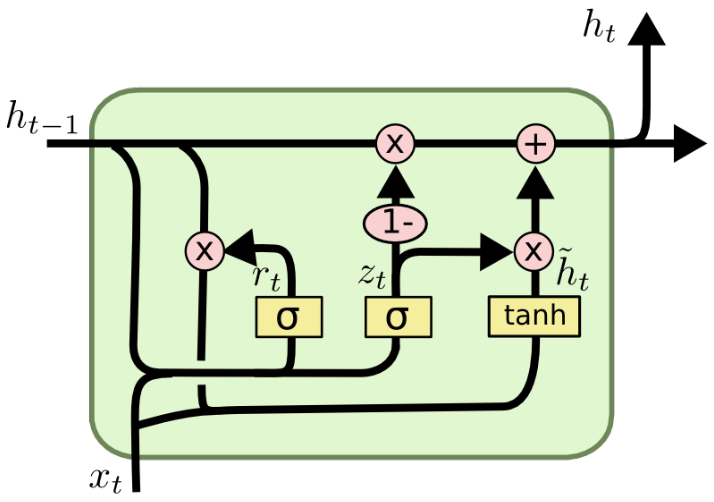

Figure 3: Variation of a GRU unit. (C. Olah 2015 [13])

vector product. 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 [2] 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. In [5] GRU is an average variation, slightly better than vanilla LSTM, with much simpler architecture.

In paper [8], 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 [8] paper. 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 – text generation

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[2]. If we have a task with K classes, class i will be represented by a vector V of length K. All the entries of V will be switched off to 0, except V_i, which will have the value of 1. Vector V is simultaneously a degenerated multinomial

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

probability distribution of the current input. If the output has the same shape as input, it can be simply created by softmax function at the output layer. Result will be proper multinomial distribution of next value, given current value.

Pridat ukazky vektoru a jak to odpovida tomu rozdeleni.

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.

Popsat word embeddings jako reseni problemu s hodne dimenzema. [12]

experimentovani s character level sitema, protoze to je jednodussi, vhodne i pro jine jazyky nez anglictinu (konjugace, deklinace - rustina)

krome jinych oblasti maji dobre vzsledky i v generovani textu

mozna nekde zminit lstm stacking

tohle kodovani je taky rozdeleni pravdepodobnosti o tom jaky je soucasny vstup.

roztahnout ukazkama vektoru, rovnicema pro multinomial. nepridavat referenci na ten prvni clanek

nejdrive character level, pak presun do word level, jake maji plusy a minusy

moc velky slovnik, reseni word embeddings.

Word level, character level, word embeddings.

Popis toho jak se pracuje s textem v rnn, ze to je taky sekvence. Character level, word level, embeddings. Popis rozdilu toho jak funguji preklady a generovani dalsiho prvku sekvence.

Character-level models are much worse for vocabulary size though. I can barely get my character-level models to learn more than 5000 words. Usually less.
[?]olBaa 2 body před 6 měsíců
The problem is the slow learning time, I suppose. Training a char-rnn should in theory yield vocabulary as rich, and possibly more rich as word-level model. I observed emergency on new word forms in the texts generated by character-level RNNs, so it may need some more computational resources to get the vocab.
[?]yowdge 1 bod před 6 měsíců
Well, yeah, you can also approximate any function with a single-layer NN, but that doesn't mean that deep NNs aren't useful. It could be the case that you need millions of hidden units to get a character-level NN to perform as well as a word-level NN (do you know of any empirical research on this?).

2.2 Convolutional neural nets

Kratky uvod do toho, kde se pouzivaji, jak se vyvinuly, jednoduchy popis toho jak funguji. Obrazek?

Asi neni potreba davat subsekce na vrstvy, staci popsat jak to funguje vsechno dohromady, jednotlive vrstvy ve vetach v jednom odstavci. Obrazek. V diplomce rozpracovat vic.

Chapter 3

Experiments

Kapitola jen na semestralni projekt. V diplomce ji odstranim.

Jak se to implementuje, jake knihovny se pouzivaji - Caffe, Theano, TensorFlow, Torch. Popsat ze Torch bude v tehle kapitole.

Budu popisovat veci co jsem zkousel implementovat v Torchi.

3.1 Torch

Torch se zrecykluje do diplomky.

Udelat tady tabulku o ruznych balicich co torch ma

Jak funguji rekurentni site v Torchi.

Nacitani modelu z Caffe, ukladani v Torchi…

3.1.1 nn, nngraph

Linky na knihovny v poznamkach pod carou.

3.1.2 rnn

3.1.3 Other packages

loadcaffe, optim,…

3.2 Predicting next character in sequence

Implementace sekce Language modeling, jak se to konkretne dela.

Jak jsem to udelal, co to dela, ukazky.

Karpathyho char-rnn [9]

Chapter 4

Image caption generation

Znovu uvod k tomu jak je to dulezite a tentokrat jak na tom lidi pracuji, co je potreba a jak se to hodnoti.

4.1 Related Work

Dat tomu nejake lepsi jmeno, clanky o popisovani obrazku ktere jsem cetl, pouzil.

4.1.1 Show and Tell

[17]

Clanek z Coco od Googlu.

[15]

Zminit i strojovy preklad (Sequence to Sequence Learning with Neural Networks), architektura encoder, decoder

4.1.2 Show, Attend and Tell

[18]

Clanek z Coco z Montrealu/Toronta

4.1.3 From Captions to Visual Concepts and Back

[4]

Clanek z Coco od Microsoftu, mrknout se i na pokracovani v druhem clanku

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

[3]

Clanek z Coco z berkeley

4.2 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[1] is heavily used for this task. All datasets mentioned here are created this way.

Flickr8k [7] was one of the first datasets created for this purpose. It has been later expanded into Flickr30k [19]. MS COCO [11] is dataset created by Microsoft for their captioning challenge. CIDEr [16] 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[2]	8 092	5	Focused on people or animals (mainly dogs) performing some specific action.
Flickr30k[3]	31 783	5-6	An extension of Flickr8k dataset.
MS COCO[4]	120 000	5	Images are divided - 80 000 for training and 40 000 for testing purposes.
PASCAL-50S[5]	1 000	50	Built upon images from the UIUC Pascal Sentence Dataset.
ABSTRACT-50S[6]	500	50	Built upon images from the Abstract Scenes Dataset. No photos.

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

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

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

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

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

[6]See footnote 5.

4.3 Evaluation metrics

BLEU, cIDER, jak se pouzivaji, co delaji…

4.3.1 BLEU

[14]

4.3.2 CIDEr

[16]

4.3.3 METEOR

[10]

Chapter 5

Model

Do semestralniho projektu nebo az na diplomku?

Design modelu, co chci pouzit, jake metody chci zkusit.

Polozit si principialni otazku a zjistit jestli to nejak pomuze, jak to funguje.

5.1 Architecture

Architektura modelu, jake matematicke modely jsem pouzil, bez implementacnich detailu.

5.2 Training details

Popis pomoci jakeho algoritmu jsme trenovali, s jakyma parametrama, minibatches, datasety.

Chapter 6

Conclusion

Udelat jeden zaver pro semestralni projekt, pak ho prepsat pro diplomku.

Bibliography

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

[3] 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.

[4] 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.

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

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

[7] 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.

[8] 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.

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

[10] 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.

[11] Microsoft. Common Objects in Context. http://mscoco.org/dataset/#download. [Accessed: 2015-12-29].

[12] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. *CoRR*, abs/1310.4546, 2013.

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

[14] 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.

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

[16] 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.

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

[18] 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.

[19] 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.