Czech Technical University in Prague

Faculty of Electrical Engineering

Department of Cybernetics

Bachelor Project

Image Captioning with Convolutional Neural Networks

Michal Najman

Supervised by

Dr. Juho Kannala

and

prof. Ing. Jiří Matas, Ph.D.

Submitted in January, 2017

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

BACHELOR PROJECT ASSIGNMENT

Student: Michal N a j m a n

Study programme: Cybernetics and Robotics

Specialisation: Robotics

Title of Bachelor Project: Image Captioning with Convolutional Neural Networks

Guidelines:

1. Familiarize yourself with neural networks and the problem of image captioning, in particular [1] and [2].
2. Reproduce some of the results of [2], evaluate the performance of the method and analyze its weaknesses.
3. Suggest improvements and evaluate them experimentally.
4. Discuss the results, their impact and potential topics for future research.

Bibliography/Sources:

[1] Ian Goodfellow, Yoshua Bengio and Aaron Courville - Deep Learning - Book in preparation for MIT Press, 2016, http://www.deeplearningbook.org
[2] Johnson, Justin and Karpathy, Andrej and Fei-Fei, Li - DenseCap: Fully Convolutional Localization Networks for Dense Captioning - Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016
[3] Karpathy, Andrej and Fei-Fei, Li - Deep Visual-Semantic Alignments for Generating Image Descriptions - CoRR, 2014

Bachelor Project Supervisor: Dr. Juho Kannala

Valid until: the end of the winter semester of academic year 2017/2018

L.S.

prof. Dr. Ing. Jan Kybic
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, May 24, 2016

Abstract

In this thesis, we elaborate on image captioning concerning especially dense image captioning. We present technical fundamentals of a model striving to solve such a task. Concretely, a detailed structure of DenseCap and Neural Image Caption is discussed. Experimentally, we examine results of DenseCap and analyse the model's weaknesses. We show that 92% of the generated captions are identical to a caption in the training set while the quality of those and the novel ones remains the same. We propose a criterion that significantly reduces a set of captions addressing an image whilst SPICE score of the set is maintained.

Keywords: image captioning, dense captioning, convolutional neural networks, long short-term memory

Abstrakt

Tato bakalářská práce se zaměřuje na automatickou tvorbu popisu obrázků (angl. image captioning), konkrétně na tzv. dense captioning. Problematika je ukázána ve světle současných modelů se zaměřením na stavbu DenseCap a Neural Image Caption. DenseCap zejména je prodoben experimentům, díky nimž jsou identifikovány nedostatky modelu. Pokusy ukazují, že 92 % generovaných popisků je identických vzorkům v trénovací množině. Je zjištěno, že jejich kvalita v porovnání s těmi, které v trénovací množině nejsou, je stejná. V neposlední řadě je navrženo kritérium, pomocí něhož lze významně zmenšit množinu popisků vztahujících se ke konkrétnímu obrázku, kdy SPICE skóre této menší množiny zůstává stejné.

Klíčová slova: automatická tvorba popisu obrázků, dense captioning, konvoluční neuronové sítě, long short-term memory

Český název: Automatická tvorba popisu obrázku pomocí konvolučních neuronových sítí

Author statement for undergraduate thesis:

Acknowledgements

I gratefully thank my supervisor Dr. Juho Kannala for his wise and critical comments as well as for his enriching attitude. Kiitos! Also, my appreciation goes to prof. Ing. Jiří Matas, Ph.D. who joined the thesis meetings and contributed immensely with novel ideas.

Last but not least, I thank my family for their support and my girlfriend Barbora for selecting the most pleasant shade of orange advisedly and for reviewing this text thoroughly.

Contents

1 Introduction

For most computer vision researchers the classification task has always been dominant in the field. Either it was a scene understanding in the pioneer 1960s or a traffic sign detection in the modern days, the task has been rooted in the soil of computer vision. It is not surprising that one of the most significant competition in the field comprises the image classification task among others.

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) awards annually the algorithm which is most successful at predicting the class of an image in its five estimates (known as top-5 error). For the record, the lowest top-5 classification error reached 28.2% at the ILSVRC2010 and 25.8% a year later, respectively [1].

Nonetheless, an unxpected breakthrough came in the year 2012 when Krizhevsky et al. [2] presented decades old algorithms [3, 4] enhanced by novel training techniques achieving so-far-not-seen results. In particular, the top-5 classification error was pushed to 16.4%. At the latest contest in 2015, the lowest top-5 error was brought to 3.5%, drawing on the work of Krizhevsky et al.

After this success, neural networks has revolutionised the field and brought in new challenges that had not been merely considerable before. One of those newly feasible techniques – image captioning – is discussed in this thesis. In fact, as an arising discipline with promising potential, image captioning still is an active area of research nowadays, striving to answer unsolved questions. Consecutively, since the field has not been entirely established yet, one must rely mainly on recently published papers and on-line lectures only.

Considering recent work, we define image captioning as a task in which an algorithm describes a particular image with a statement. However, it is expected that the statement is meaningful, self-contained and grammatically and semantically correct. In other words, the caption shall describe the image concretely, shall not require or rely on additional information and, last but not least, be consisted of a grammatically correct sentence that semantically corresponds to the image.

The reader is advised to fully appreciate the complexity of this problem. Note that an image can be thought of as an vector in a million dimensional space. For a three-channel 1024×768 image, its dimensionality reaches nearly 2.4 million. This extensively long vector is then mapped into a space of statements, expecting to meet requirements set above. The notion is illustrated more concretely in Fig. 1.

Based on full-image captioning research, in their work Johnson et al. [5] propose that the image captioning task can be viewed as weak classification since a caption is a label expressing richer concepts. Recent work in object detection enabled correct identification of multiple salient regions in an image and, similarly, rapid progress in image captioning extended the complexity of generated statements referring to an image. Therefore, Johnson et al. argue the recently seen fruitful advances in image captioning were excited along two axes — label density and label complexity. Naturally, a new unifying task they called dense captioning has arised [5]. See Fig. 2 elaborating on the notion of dense captioning.

In dense captioning, an algorithm detects salient regions in an image and captions each of those regions separately as in image captioning. Thus, the dense captioning task consists of two joint subtasks: object detection and captioning of those objects.

Although the dense captioning task raises new challenges, and thus is examined in this thesis, only Johnson's work regards it as the paper presenting the task was published only recently. Therefore, when comparing, this thesis takes into account full-image captioning mostly.

In conclusion, recent development in research utilizing accomplishments of neural networks creates a challenging opportunity of following the most up-to-date advances in image captioning while putting those notions into perspective with previously published results.

In this thesis, we examine and elaborate on the recent development in image captioning, especially concerning the latest work of Johnson et al. [5], the DenseCap model, and Vinyals et al. [6], Neural Image Caption (NIC).

First, we elaborate on neural networks and deep learning, yearning to describe thoroughly the notions in relation to image captioning. In Sec. 3, influential work as well as state-of-the-art models are put into perspective with available datasets and evaluation functions. As already mentioned, the thesis regards mainly the models DenseCap and NIC whose detailed examination can be found in Sec. 4.

The results of DenseCap are discussed in Sec. 5 together with a detailed analysis of the model's drawbacks. Following the findings, we propose experiments improving the model's results and evaluate them.

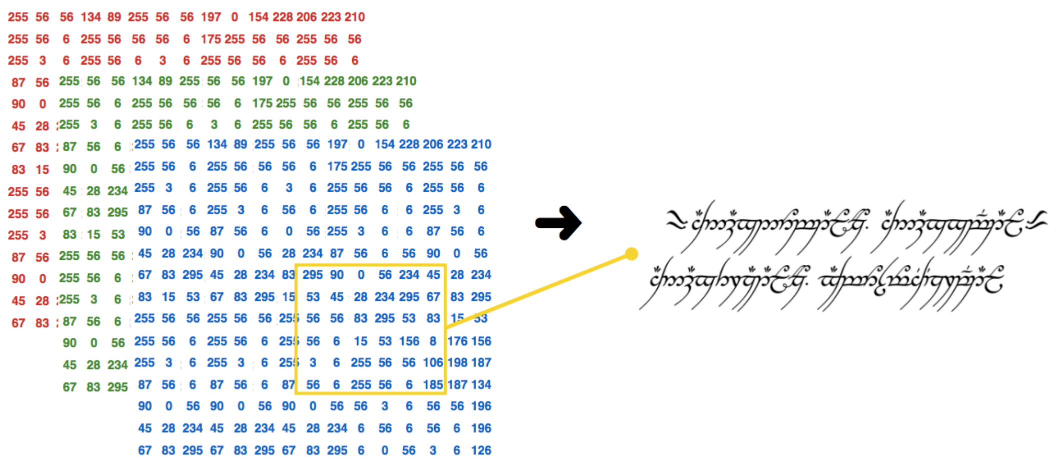


Figure 1: When developing an automatic captioner, the desired behaviour is as follows: an image, which to a computer is a $3 \times W \times H$ tensor containing integers in range from 0 to 255, is described with a sentence, which is just an ordered sets of pre-defined tokens. To illustrate the difficulty of the task, we use the famous inscription written in Black Speech that is engraved in the One Ring from the Lord of the Rings trilogy. Assuming the reader does not speak fluent Black Speech, this motivates the notion of translating a tensor into a sequence without further understanding of any of the two concepts. Essentially, we argue, this is the image captioning task.

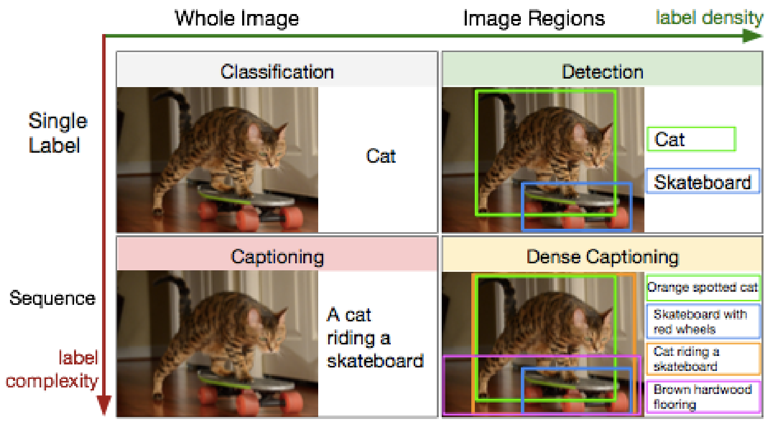


Figure 2: In this picture, recent progress in image description is shown as improvement along two axes. From classical one-label classification new tasks have emerged — salient object in the image are labeled independently and, in contrast, labels have been extended into sentence-like captions. We define dense captioning by combining the two. *Image taken from [5]*

2 Neural Networks

In order to tackle the image captioning task, recent work shows it is in one's interest to utilize neural networks [7]. This frequently used term dates back to 1950s when notions such as the Perceptron Learning Algorithm were introduced [8]. Modern neural networks draw on notions discovered in the era of a Perceptron. In this section, we first define a neuron as a fundamental part of modern neural networks. Then we elaborate on Convolutional Networks and Recurrent Networks.

2.1 Perceptron

For the purposes of this work, a perceptron is defined generally as it became a fundamental part of modern neural networks and the notation is utilized further on. Thus, a perceptron is compounded of one neuron. The neuron's output, known as the activation a, is mapped by $\Phi : \mathbb{R}^N \to \mathbb{R}$ as follows:

$$a = \Phi(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + b) \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^N$ is a feature vector, $\mathbf{w} \in \mathbb{R}^N$ and $b \in \mathbb{R}$ are weights and $\sigma(\cdot)$ is a non-linear function. In case of the Perceptron, $\sigma(\cdot)$ stands for

$$\sigma(z) = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

In other words, a perceptron is a non-linear function separating data into two classes each associated with either 1 or 0. A perceptron is parametrized by weights $\mathbf{w}$ and b. By setting proper weights, one effects the output and the perceptron's behaviour for a given feature vector. Therefore, such weights trimming is essential, yet non-trivial task. In order to find the weights, a learning algorithm was introduced, named the Perceptron Learning Algorithm [8]. This algorithm has a limiting property such that successful learning is achieved if and only if the data are linearly separable which is a major drawback pointed out in by Minsky and Papert in 1969 [9]. For example, there is no vector $\mathbf{w}$ and bias b that would make a perceptron mimic the XOR function.

2.2 Multi-Layer Neural Network

Taking the perceptron as inspiration, the XOR problem can be overcome by aligning neurons into layers and interconnecting those layers. This function is called a Feedforward Neural Network, an Artificial Neural Network or simply a Neural Network.

In a neural network, each layer comprises N neurons processing inputs coming from the previous layer and producing activations used later in the following layer. For the sake of simplicity, it is now assumed that the number of neurons N is same for all layers. However, this varies very often, for example, usually the output layer consists of fewer neurons corresponding to the nature of the problem being solved. To conclude, the activations of the k-th layer $(a_1^{(k)}, a_2^{(k)}, \ldots, a_N^{(k)}) = \mathbf{a}^{(k)} \in \mathbb{R}^N$ are each a function of the activations of the previous layer $k-1$, noted as $\mathbf{a}^{(k-1)} \in \mathbb{R}^N$:

$$\begin{aligned} a_1^{(k)} &= \Phi_1^{(k)}(\mathbf{a}^{(k-1)}) \\ a_2^{(k)} &= \Phi_2^{(k)}(\mathbf{a}^{(k-1)}) \\ &\vdots \\ a_N^{(k)} &= \Phi_N^{(k)}(\mathbf{a}^{(k-1)}) \end{aligned} \tag{3}$$

where $\Phi_i^{(k)}(\cdot)$ is a function defining the properties of the i-th neuron in the layer k, often having the form similar to Eq. (1). However, there are exceptions that proved to be essential to modern neural networks designs [2,7,10–12]. We discuss these in the later sections.

In practise, to lower the complexity of a network, in the given layer k all the functions $\Phi_i^{(k)}(\cdot)$ are always of the same form, only distinct in weights. Therefore, it is convenient to use vector notation. This is done by simplifying Eq. (3) into the following form:

$$\mathbf{a}^{(k)} = \mathbf{\Phi}^{(k)}(\mathbf{a}^{(k-1)}) \tag{4}$$

As an example, we show a neural network with one hidden layer. The network takes in a vector that is propagated forward into the hidden layer. The processes in the hidden layer are noted here as $\mathbf{\Phi}^{(1)}$. Further, the activations of the hidden layer are again propagated, analogically, into the second layer $\mathbf{\Phi}^{(2)}$ whose activations are the output of the network. The network's design is shown in Fig. 3. Formally, the network is fed with a feature vector $\mathbf{x} \in \mathbb{R}^N$ producing a vector $\mathbf{y} \in \mathbb{R}^M$:

$$\mathbf{y} = \mathbf{\Phi}(\mathbf{x}) = \mathbf{\Phi}^{(2)}(\mathbf{\Phi}^{(1)}(\mathbf{x})) \tag{5}$$

where $\mathbf{\Phi}^1 : \mathbb{R}^N \to \mathbb{R}^H$ is called the hidden layer and $\mathbf{\Phi}^2 : \mathbb{R}^H \to \mathbb{R}^M$ is called the output layer. Note that the number of neurons in the hidden layer H is a hyper-parameter.

Although the structure of the network in the example has been defined, there are still other hyper-parameters to be determined. For example, the form of the layer mappings $\mathbf{\Phi}^{(1)}$ and $\mathbf{\Phi}^{(2)}$ needs to be specified. A layer with the most simple form of its mapping is called a Fully Connected Layer and is discussed in the following subsection.

2.2.1 Fully Connected Layer

In the most basic neural network — a feedforward neural network comprising fully-connected layers only – each neuron processes activations of all neurons in the previous layer and is activated using $\Phi(\cdot)$ defined in Eq. (1). Thus, based on vector notation in Eq. (4), the activations of the k-th fully-connected layer are defined as

$$\mathbf{a}^{(k)} = \sigma(\mathbf{W}^{(k-1,k)}\mathbf{a}^{(k-1)} + \mathbf{b}^{(k-1,k)}) \tag{6}$$

where $\mathbf{W}^{(k-1,k)} \in \mathbb{R}^{N \times N}$ is a weights matrix with weights vectors aligned in rows, $\mathbf{b}^{(k-1,k)} \in \mathbb{R}^N$ is a vector of biases and $\sigma : \mathbb{R}^N \to \mathbb{R}^N$ is a non-linear function. Note that, since in practise the elements $\sigma_i(\cdot)$ are identical single-variable functions, we refer to them as simply $\sigma(\cdot)$.

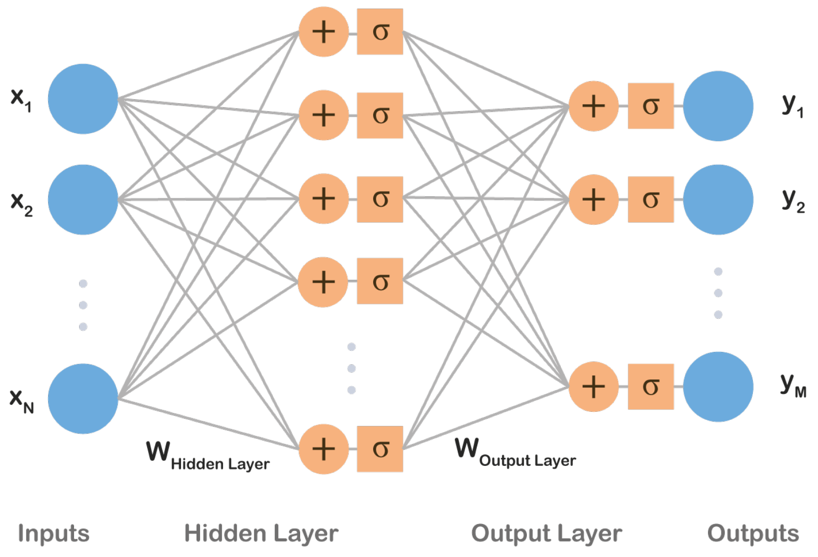


Figure 3: Inputs $x_1, \ldots, x_N$ are processed by a hidden layer and, consecutively, by an output layer producing outputs $y_1, \ldots, y_M$. Biases $\mathbf{b}$ were omitted for the sake of simplicity.

In contrast to a perceptron, $\sigma(\cdot)$ is generally required to be differentiable due to the nature of learning algorithms used in the field. For example, $\sigma(\cdot)$ used to be set to a sigmoid curve as similar to the perceptron's activation fuction shown in Eq. (2). Most commonly, tanh() or the logistic function (Eq. (7)) were used.

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

Nevertheless, when used in deep learning sigmoids suffer from problems such as vanishing or exploding gradients, therefore those were replaced with a Rectified Linear Unit (ReLU) [13]:

$$\sigma(\mathbf{z}) = \max(0, \mathbf{z}) \tag{8}$$

In modern networks, it is recommended to use ReLUs as they proved to provide better results and are thus the most common activation function used nowadays [7].

Drawing on the example presented above, we now assume that both layers are fully connected, meaning that layer mappings have a form of Eq. (6). Then Eq. (5) can be rewritten as follows:

$$\mathbf{y} = \sigma(\mathbf{W}^{(1,2)}\sigma(\mathbf{W}^{(0,1)}\mathbf{x} + \mathbf{b}^{(0,1)}) + \mathbf{b}^{(1,2)}) \tag{9}$$

2.2.2 Number of Parameters

Let us now assume $\mathbf{x} \in \mathbb{R}^N$, the activations of the hidden layer $\mathbf{a} \in \mathbb{R}^H$ and $\mathbf{y} \in \mathbb{R}^M$. Then we can calculate the number of parameters as $N + N \times H + H + H \times M$. Considering a small gray-scale image, 28×28, of a hand-written number taken from the MNIST dataset [14], that is classified as 0-9 digit, $N = 784$ and $M = 10$. Then the number of parameters, needed to be learned, is $795 \times H + 784$ where H, the number of hidden layers, is a hyper-parameter. For a hidden layer having the same width as the input vector, i.e. $H = 28$ in this example, the number of parameters reaches $23,044$.

Truly, this is a large number for such a shallow network suggesting that fully connected layers extensively increase the number of parameters.

2.2.3 Hornik's Theorem

The network mentioned above is a common design that in past was believed to yield promising results. It was shown by Hornik [15], a multilayer feedforward neural network is able to approximate any continuous function that is bounded. Yet, a possibly great number of hidden neurons might be needed in order to do so because the theorem does not quantify this important hyper-parameter.

2.2.4 Name Origin

As shown in Fig. 3, the structure is called a network since it can be drawn as a directed acyclic graph. Wondering about the name's background, one may notice the word *neural* and mistakenly assume a relation to biological neurons. However, as stated for example in [7], it is a common misconception as the name, neural networks, was derived in 1950s from former biological models serving as motivation. Those models are now, however, considered outdated, and conversely, modern neural networks are not designed to be realistic models, rather going beyond neuroscience perspective. Since that, endeavours to infer an algorithm from the brain's functioning have not faded. For example, Jeff Hawkins et al. developed Hierarchical Temporal Memory (HTM) [16] as a strict mathematization of human neocortex based on current neuroscience. Internally, HTM as a biologically inspired algorithm is distinct from deep learning and, admittedly, its results have not been as fruitful as those obtained by deep nets [17], which are discussed int the following section.

2.3 Deep Learning

In spite of former beliefs, it was found that [7] it is more efficient to insert several hidden layers one by one and propagate information sequentially creating a deep structure, instead of utilizing a shallow network given in the example. This concept is called deep learning and, surprisingly, has its roots already in the pioneer 1960s as it was assumed that an intelligent algorithm solving complex problems shall work with hierarchy of concepts [7] that was rather deep. This is why we get the name deep learning.

The notion was later found in the idea of modern neural networks which, as stated above, consist of numerous nested layers each extracting more abstract and complex features as information is propagated forward the network. Therefore, the modern neural networks and techniques used for learning them are usually nowadays referred to as deep learning.

Deep neural networks were introduced already in 1998 [18] and the optimization algorithm (back-propagation) was known by then and used frequently [3]. Yet, the deep nets were found too complex to be trained. In their books, Ian Goodfellow et al. list those reason that enabled the boom of neural networks in 2012: firstly, more data were available as well, therefore, the deep nets have started to outperform other models. Secondly, deeper models require decent architectures both in software and hardware and those had become available. Then on, promising results enabled advent of neural networks, especially in their deep form. Models such as convolutional neural networks or recurrent nets are considered state-of-the-art building blocks nowadays. Their detailed design is discussed in the following subsections.

2.4 Convolutional Neural Networks

In image analysis, many of recent advances in deep learning are built on the work of LeCun et al. [18] who introduced a Convolutional Neural Network (CNN) which had a large impact on the field. A CNN is a type of a neural network that is designed to process an image and represent it with a vector code. The architecture of CNNa draws on fully-connected neural networks. Similarly, a convolutional neural network is a compounded structure of several layers processing signals and propagating them forward.

However, in contrast to a vector activation in a fully-connected layer, activations in CNNs have a shape of three-dimensional tensors. Commonly, this output tensor is called a feature map. For instance, an input image of shape $3 \times W \times H$ is transformed by the first convolutional layer into a feature map of shape $C \times W' \times H'$, where C is the number of features. In other words, a convolutional layer transforms a volume into a volume.

A typical CNN consists of several convolutional layers and, at the top, fully connected layers that flatten convolutional volumes into a vector output. In the field's terminology, this vector code of an image is often called `fc7` features as it used to be extracted from the seventh fully connected layer of AlexNet [2]. Even though AlexNet has already been outperformed by many and the state-of-the-art designs are different from AlexNet, the term maintained its popularity. Additionally, depending on a problem the network is supposed to solve, an additional layer, such as soft-max, can be added on top of `fc7` features. A common design of a CNN is depicted in Fig. 4

2.4.1 Receptive Field

As mentioned above, a convolutional layer takes a tensor on input and produces a tensor, too. Note that these tensors have two spatial dimensions W and H, and one feature dimension C as they copy the form images are stored in. The context conveyed by the spatial dimensions is utilized in the CNN design which takes into account correlations in small areas of the input tensor called receptive fields. Concretely, in contrast to a neuron in a fully connected layer that processes all activations of the previous layer, a neuron in a convolutional layer "sees" only activations in its receptive field. Instead of transforming layer's activations it restraints to a specific small rectangularly shaped subset of the activations. When mentioning a receptive field, it is often expected only spatial dimensions of the input volume are referred to, i.e. a receptive field defines an area in the $W \times H$ grid. The shape of the receptive field is a hyper-parameter and varies across the models.

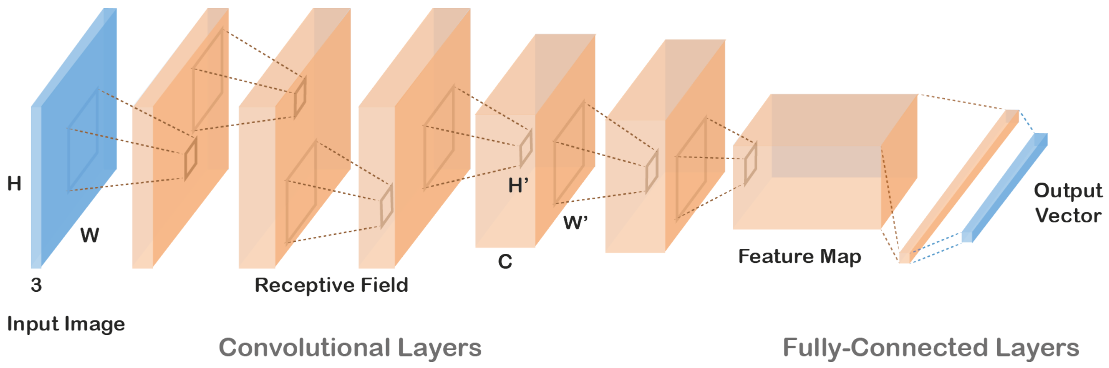


Figure 4: A convolutional neural network takes an image on input (in blue) and transforms it into a vector code (in blue). Convolutional Neural Networks are characteristic for processing volumes. An output of each layer is illustrated as an orange volume. Each neuron process only activations in the previous layer that belong to its receptive field. The same set of weights is used for neurons across the whole grid. On top of convolutional layers, fully-connected layers are commonly connected.

2.4.2 Convolution in CNNs

A neuron's receptive field is processed similarly to fully connected layer neurons. The values below the receptive field along the input tensor's full depth are transformed by a non-linear function, typically ReLU (Eq. (8)).

However, in contrast to fully connected layer neurons, the same set of weights (referred to as a kernel) is used for all receptive fields in the input volume resulting into a transformation that has a form of convolution across the input. A kernel is convolved across W and H spatial dimensions. Then, a different kernel is again convolved across the input volume producing another 2D tensor. Aligning up the output tensors into a $C \times W' \times H'$ volume assembles the layer's output feature map.

This is an important property of convolutional neural networks because each kernel detects a specific feature in the input. For example, in the first layer, the first kernel would detect presence of horizontal lines in the receptive fields, the second kernel would look for vertical lines, and similarly further on. In fact, learning such types of detectors in the bottom layers is typical for CNNs.

The design of CNNs has an immensely practical implication – since a kernel is convolved across the input utilizing the same set of weights and it covers only the receptive field, the number of parameters is significantly reduced. Therefore, convolutional layers are less costly in terms of memory usage and the training time is shorter.

2.4.3 Pooling Layer

Convolutional layers are designed in such a way the spatial dimensions are preserved and the depth is increased along the network flow. However, it is practical to reduce spatial dimensions, especially in higher layers. Dimensions reduction can be obtained by using stride when convolving, leading to dilution of receptive fields overlap. Nevertheless, a more straightforward technique was developed called a pooling layer. An input is partitioned into non-overlapping rectangles and the layer simply outputs a grid of maximum values of each rectangle. In practise, pooling layers are inserted often in between convolutional layers to reduce dimensionality.

2.5 Recurrent Neural Networks

Convolutional and fully connected layers are designed to process input in one time step without temporal context. Nonetheless, some tasks require concerning sequences where data are temporally interdependent. For that, a Recurrent Neural Network (RNN) – an extension of fully connected layers – has been introduced. RNNs are neural networks concerning information from previous time steps .

RNNs are used in a variety of tasks: transforming a static input into a sequence (e.g. image captioning); processing sequences into a static output (e.g. video labelling); or transforming sequences into sequences (e.g. automatic translation).

A simple recurrent network is typically designed by taking the layer's output from the previous step and concatenating it with the current step input:

$$\mathbf{y_t} = \mathbf{f}(\mathbf{x_t}, \mathbf{y_{t-1}}) \tag{10}$$

The function f is a standard fully-connected layer that processes both inputs indistinctly as one vector. Due to its simplicity, this approach is rather not sufficient and does not yield promising results [19]. Thus, in past years, a great number of meaningful designs have been tested [20]. The notion was advanced and designs have become more complex. For example, an inner state vector was introduced to convey information between times steps:

$$\mathbf{h_t}, \mathbf{y_t} = \mathbf{f}(\mathbf{x_t}, \mathbf{h_{t-1}}) \tag{11}$$

The most popular architecture nowadays is a Long Short-Term Memory (LSTM) [21] – a rather complex design, yet outperforming others [20].

2.5.1 Long Short-Term Memory

A standard LSTM layer is given as follows:

$$\mathbf{f_t} = \sigma_\mathbf{g}(\mathbf{W_f x_t} + \mathbf{U_f h_{t-1}} + b_f) \tag{12}$$
$$\mathbf{i_t} = \sigma_\mathbf{g}(\mathbf{W_i x_t} + \mathbf{U_i h_{t-1}} + b_i) \tag{13}$$
$$\mathbf{o_t} = \sigma_\mathbf{g}(\mathbf{W_o x_t} + \mathbf{U_o h_{t-1}} + b_o) \tag{14}$$
$$\mathbf{c_t} = \mathbf{f_t} \odot \mathbf{c_{t-1}} + \mathbf{i_t} \odot \sigma_\mathbf{c}(\mathbf{W_c x_t} + \mathbf{U_c h_{t-1}} + b_c) \tag{15}$$
$$\mathbf{h_t} = \mathbf{o_t} \odot \sigma_\mathbf{h}(\mathbf{c_t}) \tag{16}$$

where $\mathbf{x_t}$ is an input vector and $\mathbf{h_{t-1}}$ is an output vector. All matrices $\mathbf{W}$ and $\mathbf{U}$ and biases b are weights that together, with σ_g which is a logistic function Eq. (7), represent a standard neural network layer.

Thus, the forget gate vector $\mathbf{f_t}$, the input gate vector $\mathbf{i_t}$ and output gate vector $\mathbf{o_t}$ are outputs of three distinct one-layer neural nets each having its output between -1 and 1. $\mathbf{c_t}$ is a cell state vector that, as a hidden output, is propagated to the next time step. $\mathbf{h_t}$ is an output of the LSTM cell. $\odot$ stands for element-wise multiplication, σ_c and σ_h are usually set to tanh.

Note that $\mathbf{c_t}$ is a combination of the previous time step $\mathbf{c_{t-1}}$, element-wisely adjusted by the forget gate $\mathbf{f_t}$, and the output of a neural network, gated similarly by the input gate $\mathbf{i_t}$.

The output of LSTM $\mathbf{h_t}$ is a function of the cell state vector, first squashed between 0 and 1, and then adjusted by the output gate $\mathbf{o_t}$.

Connected to a network, LSTM consists typically of one layer only. LSTMs are known to preserve long-term dependencies, as shown for example by Karpathy [22].

3 Related Work

For a human, seeing an image for a short time is sufficient enough to describe it comprehensively [23]. Humans undertake this task intuitively and can produce immensely detailed description of the image. Nonetheless, producing similar results with a computer has proven to be a rather difficult task.

Image description was first tackled in a classification task where a label is assigned to an image [1, 24]. Although this approach have recently reached outstanding results as mentioned in Introduction, they cannot overcome the essential limitation as their set of outcomes is predefined and thus fixed. In real-world tasks a method yearning to compete with human abilities has to be capable of working with an unlimited set of outputs. In image captioning, such quality is especially desired as natural language, that captions are expressed with, is essentially unlimited. This fact has led to utilizing Recurrent Neural Networks in numerous recent papers [6, 25–29] as they are theoretically able to process and generate any sequence.

In Sec. 3.1, the relevant models leading to modern examples are discussed. Then, in Sec. 3.2 we elaborate on the most advanced algorithms that have been developed in image captioning. The detailed structure and principles are, however, extended on in Sec. 4.

3.1 Models History

The idea of describing an image with a statement was first unfolded using hard-coded rules that specified which visual context shall be used [30]. Some of the early approaches [31, 32] targeted videos as their visual input instead of nowadays more popular images. Nevertheless, the limitation of the fixed size output set still held true.

Challenging it, some authors came up with image-sentence embeddings [33] that bound an image and a textual statement in a common space [34, 35]. Also, it was very often to view the problem as document retrieval, meaning that for a given test image, a statement is retrieved from the training dataset, on the basis of image similarity [30, 36–38]. The latest mentioned approach uncovers the potential of a baseline method k-nearest neighbours in which the non-trivial task of finding convenient distance between images is tackled with CNNs [39]. In particular, the authors of [40] show that an image can be represented with a vector code obtained by a CNN. Image similarity is then measured in the space of those vector codes using, for example, euclidean distance. The caption is then chosen from the nearest images in the training set. In addition, some authors combine sentences stored in a database [41, 42].

After publishing their work referred to as AlexNet [2], Krizhevsky et al. were followed by many with applying convolutional neural networks to various problems. Recent work in image captioning, in particular, utilizes CNNs as image encoder [5, 6, 43, 44].

3.2 Current Models

3.2.1 Image Classification

In the most recent work in computer vision, variations of neural networks trained with stochastic gradient descent [2] are mostly used. These models are trained to classify an image, i.e. assign a class label to it [3, 10]. It proved to be very efficient to utilize a pre-trained image classification model in similar tasks. [5, 6, 25, 43, 44]. From the great variety of pre-trained models, let us mention VGG-16 [10] and Inception V3 [11] that set the state-of-the-art performance. Most recently, Inception V4 was introduced proposing a new framework of a convolutional layer that adds the feature map to its input and outputs this sum.

3.2.2 Object Detection

In the dense captioning task [5], an image is described with a set of statements, each corresponding to a salient region in the image. Thus, different phenomenons present in the image need to be detected and localized in order to describe them. The most advanced methods solving such a task (object detection) draw again on CNNs, such as YOLO [45] or Fast R-CNN [46].

Lately, Ren et al. [47] focused on generating boxes with a fully convolutional network by converting anchors into region proposals (Faster R-CNN), yet they do not propose end-to-end trainable architecture. Drawing on Faster R-CNN, a model entirely trained with back-propagation has been proposed by Johnson et al., called FCLN [5]. In terms of time performance, SSD by [48] outperforms others.

3.2.3 Recurrent Neural Networks

Assuming an image have been decoded into a feature map, recent publications adopt Recurrent Neural Networks (RNNs) [27, 49, 50] to construct a language model that generates text. Ability of RNNs to preserve long-term relations and generate sequences was studied by Bahdanau et al. [51–53].

Calling on these notions, a multimodal RNN was introduced [25,27,44,54]. Further work, on the other hand, simplified the language model and developed other RNN designs. Karpathy and Fei-Fei [43] exploited a Bidirectional RNN [55], whereas for example Kiros et al. [27] rather drew on the LSTM recurrence [21] by propagating the visual information from the decoder directly to the language model. Karpathy et al. [22] showed why LSTM in particular provides surprisingly good results despite their complex architecture. A certainly interesting idea was introduced by Bengio et al. [56] who took into account spatial attention – a map that highlights areas of an image that shall be described by the language model.

3.2.4 Metrics and Challenges

To measure performance of image captioning models, several metrics have been developed, such as CIDEr [57], BLEU [58] or METEOR [59] that each exploits different natural language characteristics. Yet it is a commonly shared opinion that each of them suffers from major drawbacks. To show that even a sufficient metric of image captioning is still an open problem, the reader is encouraged to consider 2015 MSCOCO Image Captioning Challenge [60] that introduced a unique human-evaluated judge system to overcome drawbacks of each automatic method. And in the post-contest evaluation, the submissions are judged with all the metrics mentioned above.

METEOR and BLEU are machine translation metrics adopted for image captioning, while CIDEr was designed to evaluate image captions. However, most recently introduced SPICE [61] is highlighted for its correlation with human judgements reaching 0.88 whereas CIDEr achieves 0.43.

Based on the results of 2015 MSCOCO Image Captioning Challenge, the best state-of-the-art performance was achieved equally by the Neural Image Caption model (NIC) [6] and the model presented by a team from Microsoft Research [28]. In terms of dense captioning, the best results are produced by the DenseCap model [5] which together with NIC is examined in this work further on.

4 Image Captioning Model

Image captioning models are complex systems that have been consisted of several modules from the very beggining. Presence of modularity holds true for modern captioning models as well. Framed by Vinyals et al. [6], modern image captioning models are generally compounded of two parts: an image encoder and a language model. In this section we adapt the framework and extend it to dense captioning. The notion of modularity is shown in Fig. 5.

In other words, an image encoder takes in a raw image an encodes it into a feature map using a CNN. As there has emerged numerous CNN models in past few years, we examine VGG-16 and Inception V3 in Sub. 4.1.

Having the feature map extracted, the most recent language models generated text sequentially, word by word, relying on inherited probabilistic dependency. Therefore, given an image, a caption S having the biggest probability is expected to paint the image most accurately, i.e. $S = \arg\max p(S_i|I)$.

To break this two-module scheme right away, in dense captioning a model generates text corresponding to regions in the image, therefore, such a model extends on the scheme and embeds a salient region detector in between the modules. Concretely, the region detector localizes objects in the full-image feature map and, for each, interpolates its own feature map that is further propagated into the language model. A scheme of a dense captioning model is shown in Fig. 6.

Regarding training captioning models, it is common in deep learning, models are constructed in such a way they can be trained in an ”end-to-end” fashion. End-to-end means only single loss function is used for optimizing the entire model which simplifies training significantly. This intriguing quality holds true for both examined models. Therefore, all modules of the models are optimized jointly utilizing a loss function operating over the output words. The error is back-propagated through the language model, optionally the salient region detector, down to the image encoder if fine-tuning is enabled. As a result, these modules are intertwined and non-interchangeable without further interventions.

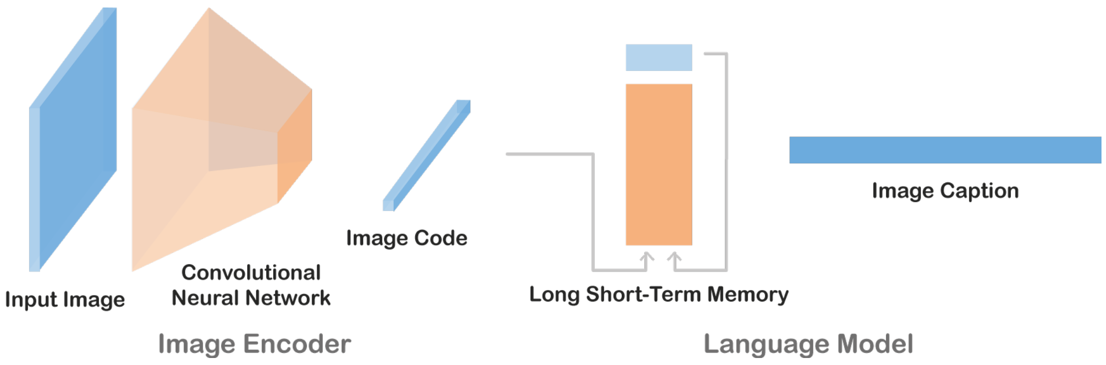


Figure 5: An image captioning model consists of an image encoder and a language model. The image encoder transforms an image into a vector code with a CNN. The language model generates sequentially a caption conditioned on the image code.

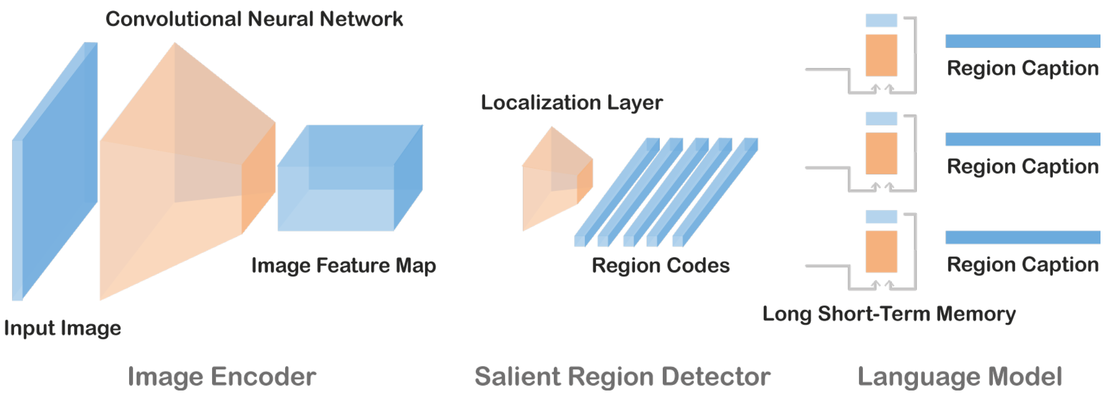


Figure 6: A dense captioning model consists of three modules: an image encoder, a salient region detector and a language model. The image encoder transforms an image into an image feature map (in blue) that is accepted by the salient region detector that localizes regions in the image using a localization layer. The language model generates sequentially a region caption conditioned on the region code (in blue) for each region separately.

In this section, we look at the architecture of image captioning models and explore and examine the most advanced two of them representing the state-of-the-art design, in particular DenseCap and NIC.

4.1 Image Encoder

Regarding an image encoder as a black box, it is fed with a raw image of shape $3 \times W \times H$ without additional pre-processing and it outputs a $C \times W' \times H'$ tensor denoted as a feature map.

4.1.1 Transfer Learning

Typically, the image encoder is a CNN architecture that has been pre-trained on a similar task concerning images, e.g. image classification. In practise, this means one chooses a particular CNN architecture from a great variety of them and removes some of the top layers of the network. In the case of AlexNet [2], it is very common to extract the features of the seventh layer (`fc7` features); and for example Jonhson et al. removed in DenseCap only the final pooling layer of the VGG design [10].

Since the encoder network has been trained on images, the required feature detectors at each layer has already been established. Although this is a useful trick decreasing training time, transfer learning [62] yields drawbacks — the nature of the image dataset, that the model was pre-trained on, determines what features are learnt. If the network was trained to describe significantly distinct images, the language model receives just partial information of what the given image captures. This is partially overcome by fine-tuning, which is a training phase in which the weights of the image encoder are adjusted as well, usually with lower learning rate. Concretely, in DenseCap, Johnson et al. fine-tuned all but the first four convolutional layers after 1 epoch.

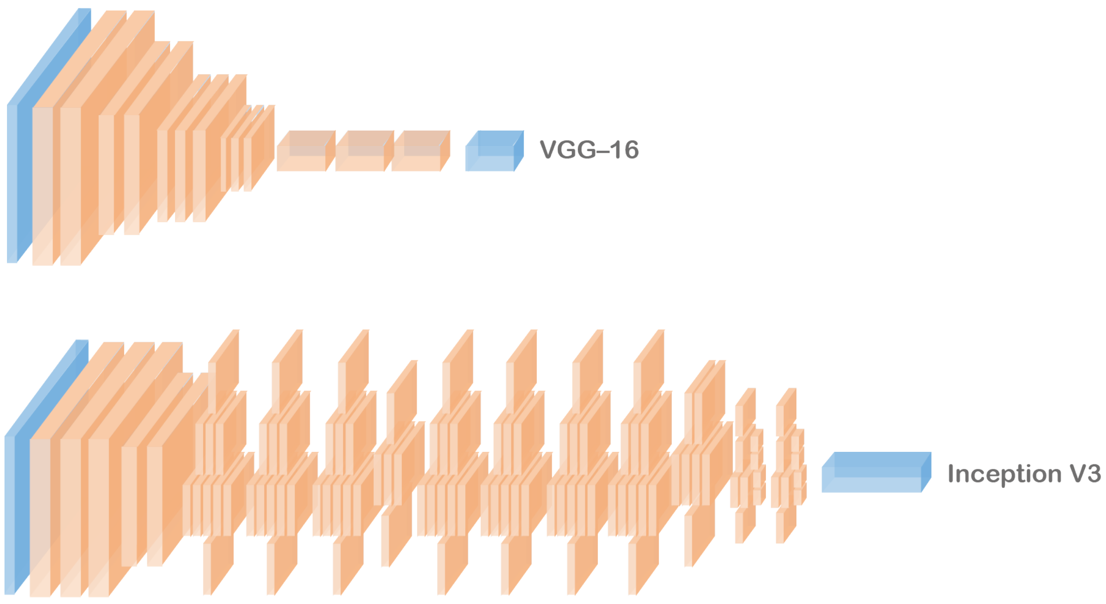


Figure 7: VGG-16 [10] and Inception V3 [11] are illustrated in the figure. Note that in contrast to VGG-16 that consists of traditional convolutinal layers, Inception V3 comprises Inception blocks that each is a convolutional network. It is underlined that Inception V3 has significantly deeper structure. The ratios of the output volumes (in orange) are not preserved.

4.1.2 VGG-16 and Inception V3

CNN models differ in depth and overall design. Nevertheless, a common framework was outlined by AlexNet proposing what type of convolutional layers shall be used. The image encoders in DenseCap and Show and Tell are based on AlexNet – VGG-16 [10], and respectively Inception V3 [11], both reach impressive performance [1,60] where the later, as of writing this thesis, is one of the most advanced CNN networks yielding in an ensemble 3.58% top-5 error on the validation set of the ILSVRC 2012 classification challenge. To compare, VGG-16 achieved 6.8% top-5 error on the same dataset.

Although detailed analysis of VGG-16 and Inception V3 goes beyond the scope of this work, we would like to point out that there is a fundamental difference between the networks. Whereas VGG-16 is constructed of traditional convolutional and max-pooling layers, Inception V3 goes beyond this framework and incorporates very deep structure of Inception blocks. The comparison of the two is shown in Fig. 7.

Having mentioned the design of the networks, the image encoder is treated in image captioning as a black box that extracts features from an image.

4.2 Salient Region Detector

This module is specific to dense captioning in terms of describing images. However, salient region detectors are essential for object detection. In this section, we describe Fully Convolutional Localization Layer (FCLN) introduced in DenseCap by Johson et al. FCLN is based on the Faster R-CNN model [47] built up for object detection. Fig.

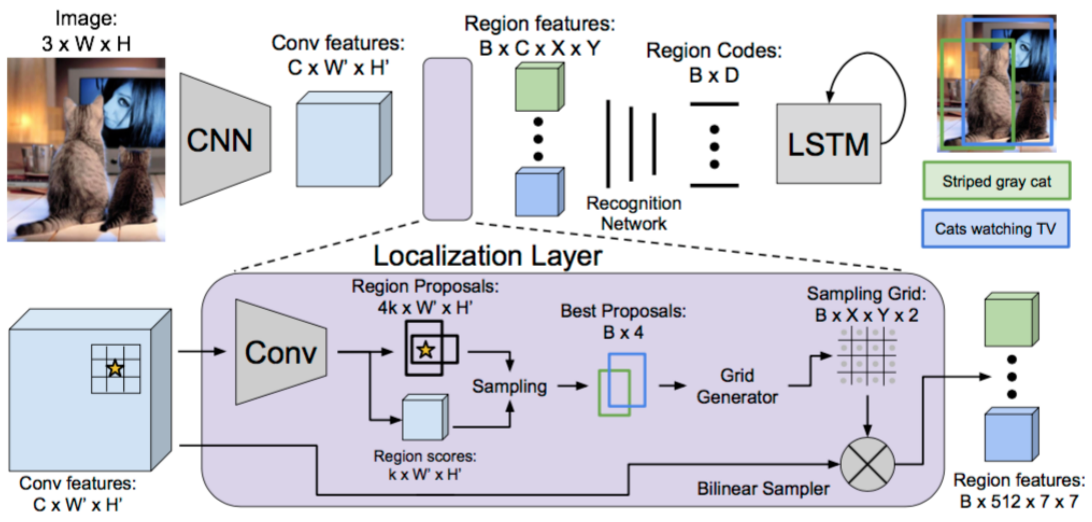


Figure 8: The scheme of the DenseCap model. *Taken from [5].*

8 shows that, as a black box, FCLN accepts an image feature map and returns B interpolated region codes of each region. B is a hyper-parameter setting the number of regions being described. In addition to the region codes, FCLN passes on four parameters denoting a rectangular bounding box and a region confidence into the language model.

Although it is theoretically not necessary, the salient regions are considered to be rectangular as it is more convenient to pass on four parameters defining a rectangle instead of a more complex structure marking a real shape of the region. Also, the later is much harder to detect since object segmentation is considered to be a more difficult task than detection and convenient datasets have been published just recently (e.g. Visual Genome [63]).

4.2.1 From Anchor to Region Proposal

In FCLN, an approach similar to Faster R-CNN is utilized to predict region proposals. For each point in the $W' \times H'$ grid, a corresponding point in the image is found, i.e. the grid $W' \times H'$ is stretched out onto the $W \times H$ grid. k default anchors of various aspects ratios are assigned to each mapped point in the stretched out grid, creating kWH' anchor boxes distributed regularly across the image in groups of k. Since FCLN regresses each region proposal on its anchor independently, this method is translation invariant to the actual position of objects in the image which, indeed, is a very convenient property.

The regression in particular is done as follows: the image feature map is transformed by a CNN into a $5k \times W' \times H'$ tensor containing four bounding box coordinates and a confidence score of each region proposal – five numbers per anchor, hence $5kW'H'$ in total.

In FCLN, the box coordinates parametrize a region with respect to its anchor's default position and aspects. In order to define the bounding box in the $0-1$ interval, the parametrization exploits normalized offset and log-space scaling transformation adapted from [46]. The confidence score denotes objectness of the captured area in the region and, thus, evaluates the content of the bounding box.

The extensive number of region proposals needs to be subsampled. Once the region proposals have been obtained, B of them are sampled using greedy non-maximum suppression (NMS) and the confidence score at test time.

To conclude, the CNN regresses bounding boxes and confidence scores from an image feature map. Due to the extensively large number of region proposals, only B of them are sampled.

4.2.2 Region Feature Map

Having the bounding box generated, a feature map for each individual region needs to be projected from the full-image feature map. To do so, Faster R-CNN sets in an RoI pooling mechanism. The RoI layer is proposed in such a way back-propagation is exploitable only for the weights of the image decoder, but not for the CNN that transforms coordinates and scores. In Faster R-CNN they overcome this with instead developed four-step optimization process which the detector is trained with.

To extend on that, the subtask is defined as follows: given the coordinates of the region and the $C\times W'\times H'$ image feature map, extract a $C\times X\times Y$ region feature map, where X and Y are fixed, thus independent on actual dimensions of the region. The given subtask is not trivial due to regularity of the grid, where its dimensions X, Y and the dimensions of the region proposal are generally incommensurable – we transform a variable rectangular area of $W' \times H'$ into a fixed grid $X \times Y$. In Faster R-CNN, they solve the problem with grounding which is a non-differentiable function, thus an obstacle for back-propagation.

In contrast, Johnson et al. overcame this method with bilinear interpolation which is the main contribution of FCLN, enabling the DenseCap model to be trained in an end-to-end fashion.

Intuitively, the full-image feature map is cropped at the real-valued area of the bounding box, stretched out and "averaged" so that only corresponding areas of the $W' \times H'$ grid are mapped into the $X \times Y$ grid.

Concretely, if we denote a full-image feature map $C \times W' \times H'$ by a tensor U and a region feature map $C \times X \times Y$ by V, FCLN interpolation $U \to V$ is defined as follows:

$$V_{c,i,j} = \sum_{i'=1}^{W'} \sum_{j'=1}^{H'} U_{c,i',j'} \cdot k(i' - x_{i,j}) \cdot k(j' - y_{i,j}) \tag{17}$$

where $x_{i,j}$ and $y_{i,j}$ are real-valued coordinates in U so that $V_{c,i,j}$ should be equal to U at $(c, x_{i,j}, y_{i,j})$. $x_{i,j}$ and $y_{i,j}$ are computed based on dimensions and position of the region proposal. Computation of $x_{i,j}$ and $y_{i,j}$ is an essential step in projecting the region proposal to U and stands for the "cropping and stretching" part of the process. The "averaging" part is done by bilinear interpolation where kernel $k(d) = \max(0,\, 1 - |d|)$ is used.

To sum up the projection step, the values of the $X \times Y$ grid are a linear combination of the corresponding values of the $W' \times H'$ grid whilst the number of features C is preserved.

4.2.3 Recognition Network

Lastly, to reshape the tensor into a vector and provide additional transformation, a Recognition Network is connected to the output of FCLN. This two-layer fully connected neural network serves as a utility reshaping the region feature map tensor into a D-dimensional code representing the region. Also, such transformation is applied to the bounding box parameters and the confidence score. Johnson et al. argue this refines the required values needed in the language model.

4.3 Language Model

For generating text, a language model conditioned on the region codes is used in both DenseCap and NIC (where a full-image code is accepted instead). In fact, these models are merely identical regarding text as a sequence of tokens from a pre-defined vocabulary. When working with sequences, recurrent neural networks [39], the LSTM recurrence yields best results in cases of processing language [22]. It is no wonder this rather complex architecture was chosen for its performance in both examined models.

The sentence S that is to be generated consists of T words creating a chain $s_1, \ldots, s_T$ where each word is an element of the vocabulary V. The generative process starts with the region code obtained from the previous module. The code is transformed additionally to a smaller sized vector and fed into the LSTM at the zero time step.

$$\mathbf{x_0} = W\,(\texttt{region code}) \tag{18}$$

$$\mathbf{y_0} = LSTM(\mathbf{x_0}) \tag{19}$$

where W is the down-scale transformation of the region code.

The zero time step initializes hidden states, therefore, the output of LSTM $\mathbf{y_0}$ is discarded. After it, $\mathbf{x_1}$ is set to a special `START` token that is passed on to the LSTM which, in contrast to the zero step, now generates the token the first word is chosen from.

$$\mathbf{y_1} = LSTM(\mathbf{x_1}) \tag{20}$$

Now on, at every time step, $\mathbf{y_t}$ is regressed to a word $s_t \in V$. After selecting the word s_t, the token $\mathbf{x_{t+1}}$ corresponding to this word is inputted again into the LSTM at time $t+1$, creating a loop. This process repeats on, until the `END` token, y_T, is obtained on the output of the LSTM. Note that every sentence has a different length, therefore, T just denotes the times step in which the `END` token was sampled.

Regressing s_t on $\mathbf{y_t}$ is not a trivial task. Both NIC and DenseCap utilize a softmax layer that estimates probabilities of each word from V given $\mathbf{y_t}$. In DenseCap, s_t is sampled greedily meaning that at each time step t, the word having the highest probability is chosen, i.e. $s_t = \arg\max_{s_i \in V} p(s_i)$ where $p(s_i)$ is the probability of the word s_i at the time step t.

Yearning for more sophisticated results, NIC extends the regression by maximizing the probability of the whole sentence S. It is assumed the optimal sentence S^* is given by $S^* = \arg\max_S p(S)$, and the probability of the sentence S is given by $p(S) = p(s_1) \cdot \ldots \cdot p(s_T)$. Sampling the words greedily, as it is done in DenseCap, does not guarantee bringing in the optimal solution. For that reason, NIC employs the beam search algorithm that, at every time step, keeps in l sequences of words that have the highest probability. Beam search does not guarantee finding an optimal solution either but surely can generate more likely sentences. In fact, in the latest NIC paper [6], they report that their model yields best performance when $l = 3$.

5 Experiments

Having stated the fundamentals of image captioning models, this section presents the experiments conducted to verify hypotheses regarding their performance. It is important to underline the fact that improvement is traceable as long as there is sufficiently defined evaluation function. In the problem of dense captioning, there are several aspects to be measured. Namely, a generated bounding box shall be positioned correctly and a caption shall be well-written and meaningful. Note that the later is defined vaguely and, thus, needs to be specified (caption quality is discussed in Sec. 5.2).

Although immense progress in the field has been seen lately, there is still notable dependency on computational power when applying changes to a model. Thus, when drawing up this thesis it was right away determined that all possible experiments triggering with weights in the model are, unfortunately, infeasible if one seeks to ehnance model's performance significantly using an average computer. To illustrate the fact, Vinyals et al. [6] state that training the NIC model took over three weeks with a high-end GPU and, similarly, Johnson et al. [5] report the DenseCap model generates 100 captions in $166ms$ using a single GPU whereas the same task is done on our machine in $31s$ running on CPU ($190\times$ slower). From that reason, experiments that require re-training, i.e. modifying a localization layer in DenseCap, were omitted.

Further, Hessel et al. show that the failing cases of image captioning have its roots in the language model [64]. They report that training the image encoder gradually, while keeping the same language model, improves the results at every step. Conversely, while fixing the image encoder, training the language model yields better captions to a certain level after which additional training does not improve the results. They argue the finding suggests that the current language models are the core limitation.

Therefore, we focused the experiments on quality of captions where we see a room for improvements. As an experimental model, DenseCap is used since its authors made their pre-trained model available on-line [65].

5.1 Dense Captioning Results

Given an image I, the DenseCap model generates a set of captions C where each caption $c \in C$ is associated with its bounding box $\mathbf{b} = (x_1, y_1, x_2, y_2)$ and a region confidence score s. During our experiments, we set the number of generated captions $B = 300$ while we consider only those of them whose $s > -4$, leading to $|C| \approx 150$ on average.

A typical result of DenseCap and NIC is shown in Fig. 9. The most significant distinctness of the models is their output – a caption in contrast to a set of captions. Therefore, the nature of generated text is different. Concretely, covering the example image, the NIC model captions the scene generally whereas DenseCap detects each salient region (the kite, the person, the sand, the blue sky and others) and captions them separately. More information is found using the later approach, for example, the color of the t-shirt, the sky, the trees on the hill.

The reader is encouraged to appreciate the quality of the generated captions as this performance was far from achieving no more than three years ago. Yet, in the following pages we examine drawbacks of the DenseCap method.

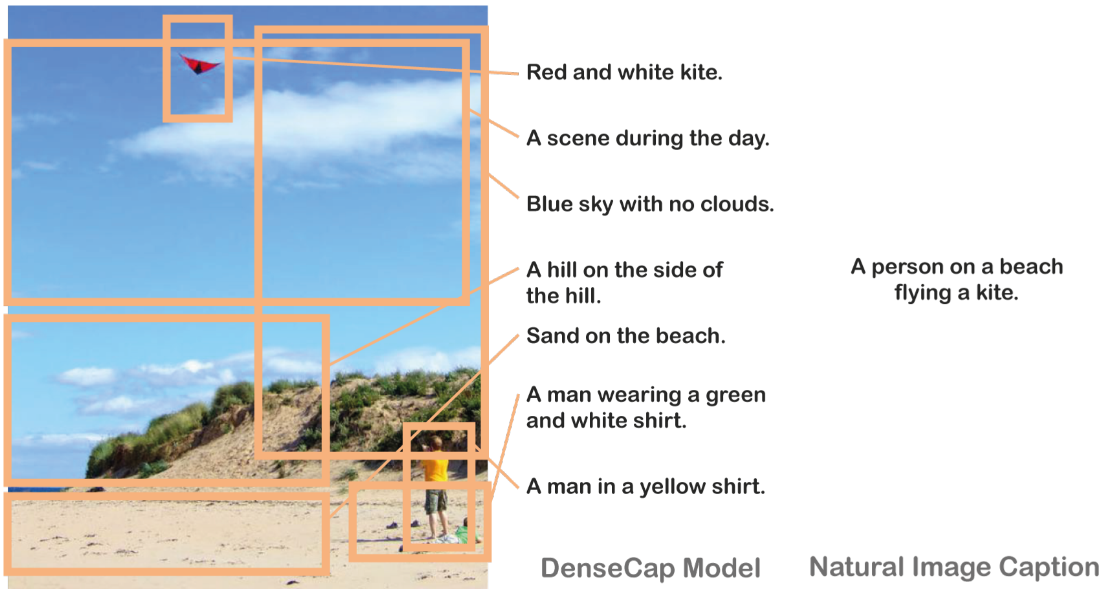


Figure 9: Note that NIC captions the image generally relating to important phenomenons in the image whereas each caption of DenseCap corresponds to a salient region in the image. The captions shown were cherry-picked from a set of 153 captions to illustrate the performance. *The image is taken from [66].*

Redundancy We found that the set of captions C is redundant and contains a relatively big number of identical captions in terms of textual similarity (shown in Fig. 10), thus it is relevant to reduce the set while maintaining the conveyed information, see Sec. 5.4.

Restricted Outcomes Concerning generated statements, we found that only a limited number of words from the training vocabulary is retrieved. Concretely, the model was trained on a vocabulary containing 11 thousand words, whereas the set of words generated on the test set (Pascal 50S) was significantly smaller, concretely 719 words. In addition, in Sec. 5.2.3 we show that 92% of the generated captions are identical to the training samples.

Scene Setting Dependency Moreover, we found that the generated captions represent the distribution of words in the training set but not tightly – this can be shown with articles. For example, we examined the training set distribution of the following terms: *a front wheel, the front wheel, front wheel.* The term *front wheel* is by far most common. Same applies to the distribution found in pictures of front wheels of a bicycle where the term *front wheel* is the most frequent one as well.

However, when the angle from which the image was taken is changed and typically the object does not face the camera directly, this distribution is affected leading to misclassification of an object. For example, we found that a bicycle that has its front wheel on the right-hand side is misclassified more often than when facing the left side.

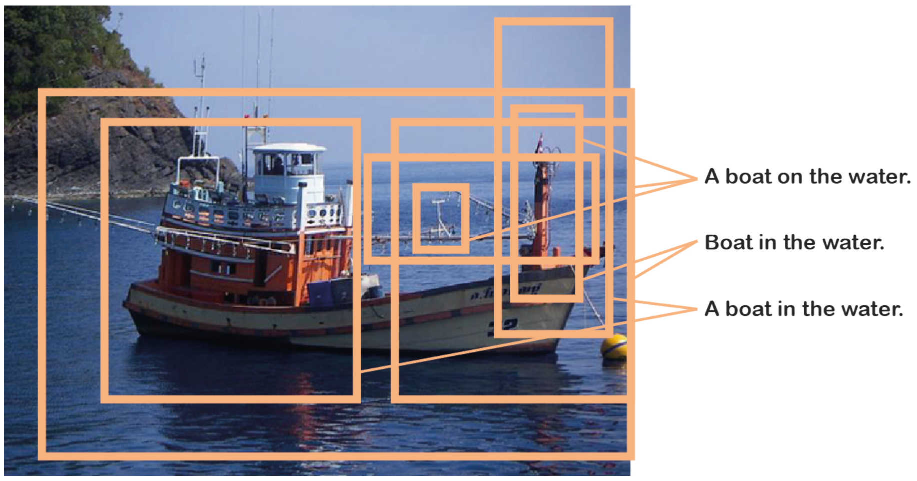


Figure 10: This figure illustrates redundancy of the set of the generated captions. Note that all of the bounding boxes depicted are described with factually identical captions, only articles and prepositions differ. Additionally, some of the boxes do not overlap thus it is not possible to determine how many boats there are in the picture. Although not depicted, given this image, similar redundancy was observed with the caption *Boat is white.*

Also, if the bicycle is taken from the forty-five angle position, the model captions the rear wheel as *front wheel* or generates meaningless captions (*bike on the bike, a black metal umbrella*).

The finding suggests that objects set up unusually in the scene are captioned incorrectly or not recognised at all; and that common phrases are linked with specific object positions whereas least common or rare settings of the same objects are captioned incorrectly (see Fig. 11). In conclusion, to find the most failing case and examine their source one needs to test the model on a dataset containing unusual scene settings which is presented in the following section.

5.2 Caption Quality

5.2.1 Pascal 50S Dataset

Unfortunately, the only dataset available for dense captioning is the Visual Genome dataset that was used for training the model – the test set used by Johnson et al. is not available. We decided to evaluate caption quality on the Pascal 50S dataset [57]. Pascal 50S contains 1000 images, each captioned with 50 statements covering to the full image. Also, Pascal 50S contains images which some of them are blurred or taken from an unusual angle.

Despite dense captioning is examined, we decided to use a dataset of full-image captions. Thus, we evaluate a set of captions C, generated given an image I, against a ground truth set C_{GT} that covers I.

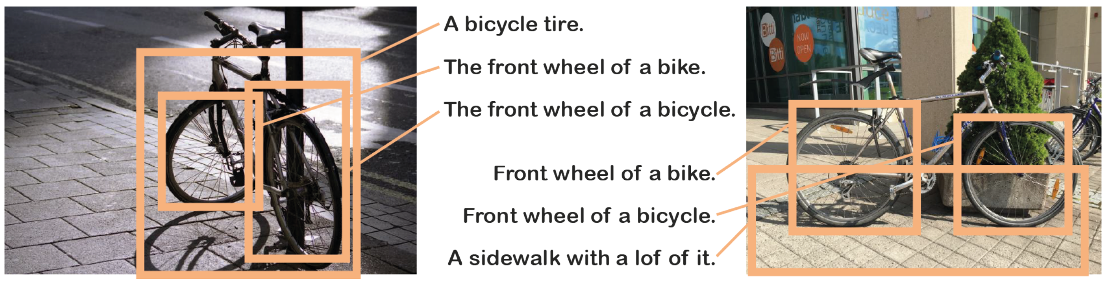

A bicycle tire.
The front wheel of a bike.
The front wheel of a bicycle.
Front wheel of a bike.
Front wheel of a bicycle.
A sidewalk with a lof of it.

Figure 11: The figure illustrates scene dependency of DenseCap. Taking a bicycle from an unusual angle leads to misclassification. The image on the left is taken from behind and the other depicts a bike facing right-hand side – both positions are considered unusual scene settings (if the same bike is taken facing left-hand side, wheel captions are correct). Note that all wheels are captioned as a front wheel. Also, the other captions are erroneous.

5.2.2 Evaluation Metrics

In this work, SPICE [61], Levenshtein distance and sets overlap are used.

As SPICE works with semantic propositional content of a caption converting it into a scene graph, it provides more transparent evaluation than other captioning metrics, in our view. More importantly, SPICE is adaptable to dense captioning due to its properties. We propose a trick that extends its capabilities from evaluating a statement to a set of statements. In practice, this means that the set C is concatenated into a single string in which captions are separated with a dot, as in real text.

To evaluate a caption, similarity of a candidate caption scene graph and a reference scene graph is computed. In SPICE, graph similarity is given by F-score of corresponding edges in the graphs. Working with graphs provides convenient implications: SPICE is invariant to order and duplicity of the concatenated statements.

In conclusion, SPICE covers factual accuracy taking into account linguistic variation.

Beside SPICE, Levenshtein distance and sets overlap are used in this thesis. Levenshtein distance denotes a minimum number of edits applied to change one statement into the other, i.e. inserting, substituting or deleting one word.

For fast evaluation, we use sets overlap. We create a set of words from each caption and exclude stop words, then we compute sets overlap as intersection over union. The set of stop words contains the most frequent words: {a, an, the, is, are, in, on, of, with, to, and}.

As shown later, it is convenient to find the nearest caption. This can be interpreted as looking for the most similar caption. This is done with both Levenshtein distance and sets overlap. In case of the former, given a set of captions E, the nearest captions $e^* \in E$ to a caption c is defined as follows:

$$e^* = \arg\min_{e \in E} l(e, c) \tag{21}$$

where $l(\cdot, \cdot)$ is Levenshtein distance. The nearest caption for sets overlap is defined as the caption e^* maximizing sets overlap:

$$e^* = \arg\max_{e \in E} s(e, c) \tag{22}$$

where $s(\cdot, \cdot)$ is word sets overlap of two captions. Further on, we denote maximal sets overlap as MSO.

5.2.3 Nearest Caption Analysis

Using the Pascal 50S dataset, it was found that the generated captions are very similar to the captions in the training set C_{train}. In fact, 91.94% of the generated captions are identical to a caption in C_{train}. To address the remaining 8.06% of captions, we utilize minimal Levenshtein distance over C_{train} to find the nearest caption. In Tab. 1, the frequency of minimal distances is shown. Note that more than 99% has a minimum distance one or zero. Examples of one-distanced captions is shown in Tab. 2.

Minimal Levenshtein distance does not provide sufficient details for captions further than 1, hence for additional analysis, we used maximal sets overlap (MSO). This measure is utilised especially when looking for the nearest caption in C_{GT} where Levenshtein reaches values beyond 5 or 6 which for a short caption often means entire substitution.

Levenshtein Distance	Frequency
0	91.94%
1	7.32%
2	0.69%
3	0.05%

Table 1: The generated captions are very similar to the examples in the training set. Nearly 92% of them are identical to some caption in the training set. Minimal distance of 1, i.e. only one word was inserted, substituted or deleted, has frequency of 7.32%. Less than 1% has a nearest distance 2 or 3. Genereated by DenseCap [5] on Pascal 50S [24].

Looking at the nearest captions in C_{GT}, we found that 35.59% of the generated captions has MSO equal to 0. On the other side of the spectrum, the captions with MSO greater than 0.5 has frequency of 1.29%. Examples of such captions and their nearest samples in C_{GT} are shown in Tab. 3.

To compare the sets, average MSO over C_{train} is 0.98 whereas over C_{GT} it is only 0.14. The findings suggest that the generated captions and the ground truth captions are significantly distinct. An example of C_{GT} and C is shown in Fig. 14.

The following findings in this section are comprehensively depicted in Fig. 12, in which captions are compared with C_{train} and C_{GT}, and the histograms of their evaluation scores are shown.

We found that SPICE and MSO has correlation of 0.77 when evaluating against C_{GT}. Therefore, we use MSO when evaluating against C_{train} as this operation would be very costly using SPICE. Certainly, on the other hand, SPICE as a more complex evaluation function provides more accurate results.

As stated above, 92% of the generated captions are retrieved from C_{train}. However, when stop words are not taken into account, this number rises to 95.0%. Each generated caption can be sorted into one of two sets, a "non-creative" set and a "creative" set –

the former containing captions that has a textual identity in C_{train}, the later denoting the opposite.

To analyse the quality of the sets of "creative" and "non-creative" captions, we plot MSO over C_{train} and SPICE against C_{GT} for each generated caption (see Fig. 12 for more details).

Also, we found the distributions of scores are merely the same proving that in terms of factual accuracy there is no difference between a caption from the training set and a "creative" caption.

Note that nearly half of the captions has SPICE score of zero. Partially, this is caused by the nature of the PASCAL 50S ground truth captions that is different from what is generated by DenseCap. However, a great portion of the zero-scored captions is wrong. Interestingly enough, this can be put into perspective when compared with the frequency of captions having MSO over C_{GT} equal to 0. As stated above, 35% of the captions does not relate to ground truth at all. But 15% of the captions has non-zero MSO and yet still SPICE score of 0. Thus, some words in those captions are correctly retrieved but they are factually wrong, e.g. wrong colour of an object is predicted.

In the following subsections, we try to improve caption quality with beam search.

5.3 Beam Search

Vinyals et al. implemented beam search to enhance the results of the NIC model. They report beam size $l = 3$ produced the most advanced results [6]. Adapting their approach, we implement beam search to DenseCap as both models use LSTM and softmax layer to generate the probability of the next word. (In fact, the DenseCap repository [65] already contained implementation of beam search that was labeled experimental and needed to be completed and debugged.) The results of beam sizes $l = 3$ and $l = 20$ are depicted in Fig. 13. In spite of expectations, the generated captions are worse when using beam search. The model produced captions that are shorter and incomplete, for example the same salient region generated *cat looking at the window* without beam search, *the cat* with both $l = 3$ and $l = 20$. Similarly, in other examples captions without beam search are better-looking, therefore, we did not perform quantitative analysis as the result is obvious. The reason for beam search worsening the DenseCap results, despite in NIC those were improved, remains an open problem.

Generated Caption	Nearest Caption in Training Set
a decorative design on the wall	a design on the wall
the design on the wall	a design on the wall
a white plane in the background	a second plane in the background
the boy is holding a baby	the boy is holding a bat
a dark green hair	a dark green hedge

Table 2: Example of generated captions that have a minimal distance equal to 1. Note that some differences are meaningless (e.g. articles) while some changed the meaning entirely (e.g. a *bat* to a *baby*)

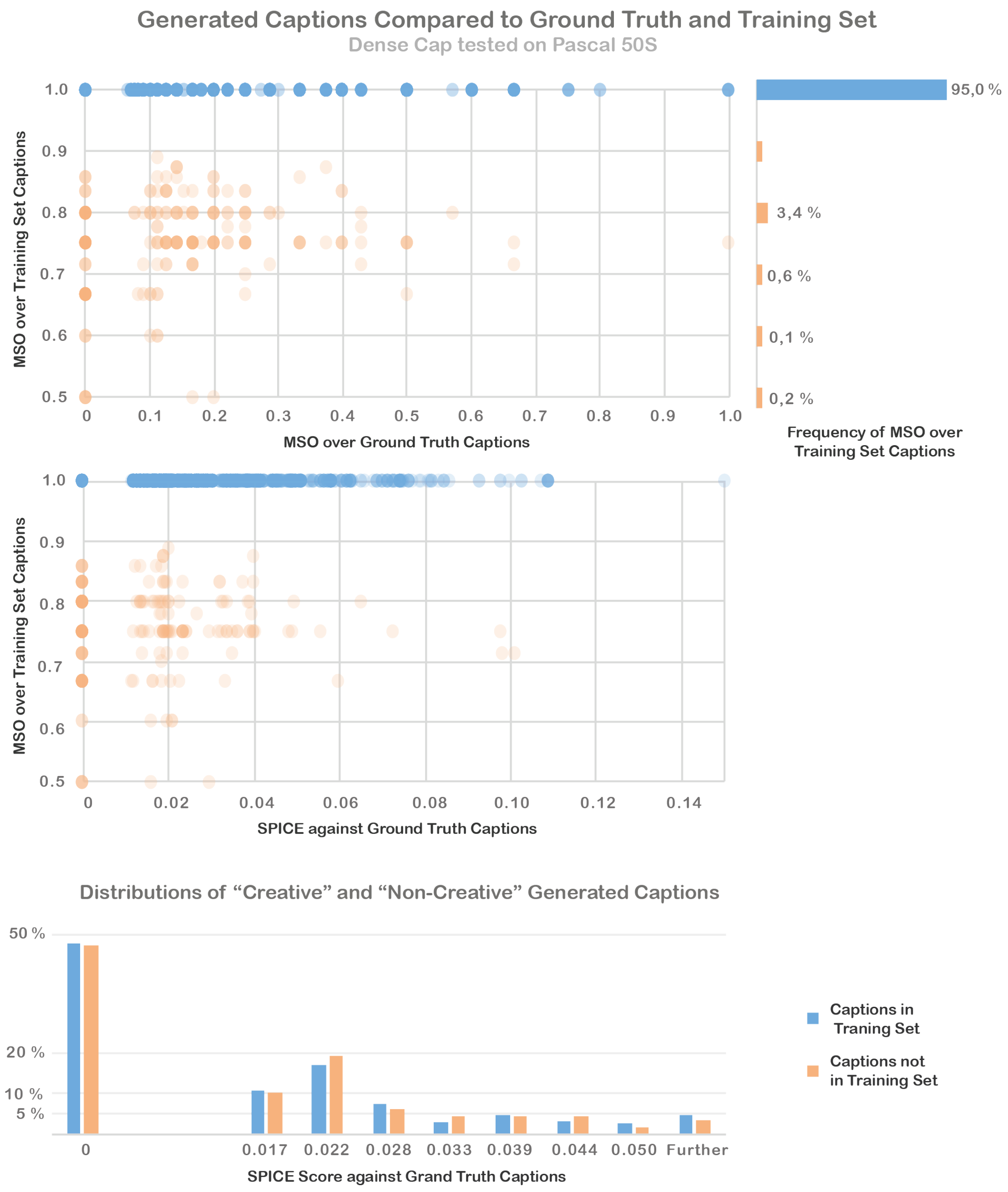
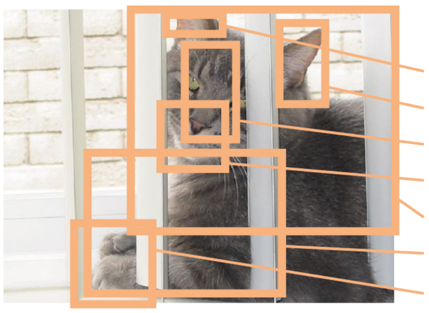


Figure 12: *Scatter Plots:* Each point in a scatter plot represents an individual caption generated on the Pascal 50S images. Point's darkness illustrates the number of captions having the same coordinates. Note that the colors in all graphs match. Both scatter plots depict the same set of all generated captions (i.e. all images together). The y-axis shows similarity to the nearest caption in the training set (MSO over C_{train}). In case of the upper plot, the x-axis represents caption's similarity to its ground truth (MSO over C_{GT}) and in case of the bottom plot, the x-axis stands for the same but using SPICE. Note that both distributions are very similar, concretely, correlation of SPICE and MSO is 0.77. *Histograms:* The upper histogram depicts that 95% of all generated captions are identical to a sample in the training set in terms of sets overlap. The bottom histogram shows the distribution of SPICE scores of the generated captions that are in the C_{train} and that are not in C_{train} are very similar.



Generated Greedily	Beam Size 3	Beam Size 20
The ear of a cat.	The left ear.	Ear on the cat.
The ear of a cat.	The left ear.	The left ear.
The cat's eye are green.	The on the cat.	The on the cat.
The cat has a nose.	The on a cat.	The on a cat
Cat lookoing at the window.	The cat.	The cat.
Cat is looking at the window.	A window.	A window.
A cat in the background.	The cat.	This is a cat.

Figure 13: Therefore, the nature of generated text is different. Concretely, covering the example image, the NIC model captions the scene generally and comprehensively whereas DenseCap detects each salient region (the kite, the person, the sand, the blue sky and others) and captions them separately.

5.4 Distinct Caption Set

In the recommended setting by the authors of DenseCap, the number of generated captions per image B is 300. As already mentioned, the set of captions C is redundant and partially contains false members. Thus, we look for a subset of the generated captions $C^* \subseteq C$ that describes the image as well as C while using only distinct captions. As we define it, a distinct caption $c \in C$ provides semantic information about the image which is not conveyed by any of $C/\{c\}$. In other words, such a caption brings in unique facts about the image.

5.4.1 SPICE Score Delta

The notion of caption distinctness can be expressed with its contribution to the overall score. To address distinctness, we define $\Delta(c)$ as follows:

$$\Delta(c) = f(C) - f(C/\{c\}) \qquad w.r.t. \quad C_{GT} \tag{23}$$

where f is an evaluation function that computes score with respect to ground truth. We use the SPICE score.

From definition, caption distinctness is dependant on C, and thus the same text could have different Δ for different images and different methods of generating C.

Generated Caption	Nearest Caption in Ground Truth	MSO
cat laying on a bed	a cat laying in a bed	1
car parked on the side of the road	white car is parked on the side of a road	0.8
man and woman sitting on couch	man sitting on a couch	0.75
a body of water	a barge is on a body of water	0.67
wooden table top	a cat sits on top of a wooden table	0.6

Table 3: Examples of captions that has MSO greater than 0.5.

Generated Captions

0.00946	A train on the tracks.
0.00000	Train on the tracks.
-0.00099	A man carrying a backpack.
-0.00099	A yellow line on the platform.
-0.00049	Trees behind the train.
-0.00049	Train tracks on the ground.
0.00000	People waiting for the train.
-0.00049	Trees growing on the side of the road.
0.00946	The train is yellow.
-0.00099	Front windshield of train.
0.00000	A clear blue sky.
0.00000	People walking on the platform.
-0.00149	A man wearing a black shirt.
-0.00049	Train tracks on the ground.
0.00000	A large green tree.
-0.00200	Person wearing black pants.
-0.00049	Lights on the train.
0.00946	People walking on the train platform.
-0.00099	A wooden door.
0.00000	Power lines above the train.
-0.00049	Green trees in the background.
-0.00149	Green bushes in the background.
-0.00049	Lights on the train.
-0.00049	Train tracks on the ground.
0.00000	A large tree in the distance.
0.00000	People waiting for train.
-0.00049	A platform on the platform.
0.00000	Gravel between the tracks.
-0.00049	Power lines on the pole.
-0.00049	Windows on the train.
0.00000	Train on the tracks.
-0.00049	A black bag on the ground.
-0.00049	Grass growing on the side of the road.
0.00000	Gravel between the tracks.
-0.00049	Train tracks on the ground.
-0.00049	Grass growing on the ground.
0.00000	Gravel on the ground.
-0.00049	Front of train is yellow.
-0.00099	Lights on the front of the train.
-0.00049	A black bag on the floor.
-0.00149	The woman is wearing black shoes.
-0.00049	A metal fence.
-0.00149	White clouds in blue sky.
-0.00099	Green grass on the ground.
0.00000	Power lines above the train.
-0.00099	A long white train.
-0.00049	A brown roof.
...	(40 more)

Ground Truth Captions

A man awaits a train as it approaches.
A man waits patiently for the train to come.
A man waiting on a platform as a train approaches.
A train is approaching a platform with people.
A train is approaching the station.
A train is coming into a station.
A train is pulling in the station for the passengers.
A train pulls up to the station.
A man is waiting for the train.
A yellow train is on it's tracks.
Man waiting on the train.
People wait as a trolley approaches the station.
A man is waiting for the train to pass.
A train stopped at a train stop.
People waiting for the train.
A yellow train is at a station.
A man is standing on a platform near a train.
An incoming train arriving at a station.
A train pulling into a station.
Two people are waiting for a train to arrive.
(30 more)

Distinct Caption Set

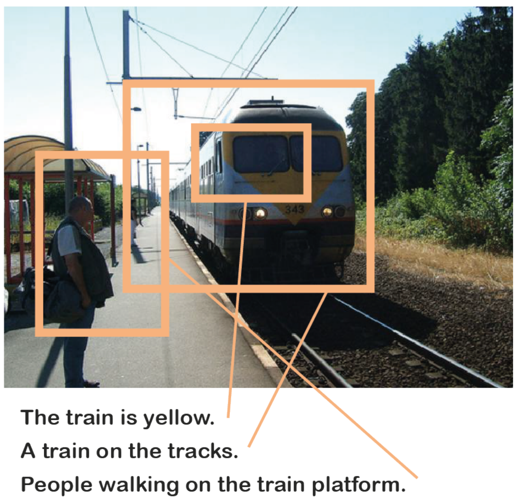


Figure 14: The figure depicts the results for the picture taken from Pascal 50S [57]. On the left, the generated captions $c \in C$ with their $\Delta(c)$ are shown. Top right, the Pascal 50S ground truth set C_{GT} is presented. Bottom right, we present the distinct caption set C^* for this image. This picture was cherry-picked to illustrate the following. C contains captions that are correct but not present in C_{GT}, incorrect captions and redundant captions (all in blue). Note that one could argue even C^* contains incorrect captions as the people do not seem to be walking.

A caption c is distinct if $\Delta(c) > 0$ which, in other words, means it enriches the scene graph of C with at least one true positive edge that was not retrieved by any of captions in $C/\{c\}$. Conversely, a caption having $\Delta(c) < 0$ brings in an false positive edges. If $\Delta(c) = 0$, caption's semantic propositional content is duplicate in the scene graph. Note that $\Delta(c)$ is equal to 0 for all of those duplicates.

Taking into account $\Delta(c)$, we extend the definition of the distinct caption set C^* as follows:

$$C^* = \{c \mid c \in C,\ \Delta(c) > 0\} \tag{24}$$

It turns out that only a few captions in C are distinct. We found that using the same setting as in the previous experiments, the average cardinality of C^* is 10.24, the median is 4. Rarely, the number of distinct captions reaches higher numbers, concretely, the maximum is $|C^*| = 107$. An example of a low cardinality set C^* is shown in Fig. 14.

Despite relatively low cardinality of C^*, we found the score maintains its value, $f(C^*) \simeq f(C)$. Concretely, we found that on average:

$$\frac{f(C^*)}{f(C)} = 1.011 \tag{25}$$

We arrive at the conclusion that the average distinct caption subset C^* has slightly better SPICE score than the set of generated captions C, yet having significantly lower cardinality. In other words, the outcomes of DenseCap can be reduced to a smaller amount of factually diverse captions.

5.4.2 Caption Distinctness Predictor

In the experiment above, C^* was found using PASCAL 50S ground truth C_{GT}. In this section, we present an experiment yearning to predict whether a caption is an element of C^*. Such a predictor shall be independent on ground truth, thus when predicting distinctness of c, we utilize the parameters of a caption's bounding box $\mathbf{b}$ and its region confidence score s. Formally, a predictor h is defined as follows:

$$h(\mathbf{b}, s) = \begin{cases} 1 & \text{if } c \in C^* \\ 0 & \text{otherwise} \end{cases} \tag{26}$$

We use the support vector machine method to learn $h()$ using a RBF kernel and soft-margin. Searching the hyper-parameter space while using 10-folded cross validation, the highest average test accuracy we obtained is 0.891 ± 0.008. Since the relative frequency of non-distinct captions is 0.88, a successful predictor was not found using this method:

$$p(\text{c is correctly classified}) \simeq p(c \notin C^*) \tag{27}$$

5.5 Discussion

We found that 92% of the generated captions are identical to those in the training set. Together with scene setting dependency it seems the DenseCap model is over-fitted and retrieves caption from the training set instead of generating novel captions for unseen region codes. However, we found that once it does the quality remains the same. A common example of one word substitution is color exchange. This can be viewed positively, for instance in medical use, it is desirable to make diagnosis non-creatively. On the other hand, in real-world image captioning, unseen settings require the model's ability to interconnect concepts present in the training set, e.g. colours or sizes.

Despite its fruitfulness in the NIC model, it seems that beam search cannot be transferred easily to DenseCap without further development and analysis. For example, it shall be examined what is the cause of higher probability of extremely short captions that we have observed. We hypothesize that there is no fundamental restraint that precludes successful adaptation of beam search to DenseCap as both models use a very similar loss function in their language models. Significant cause of this, however, might have its roots in the nature of captions that are used for full images and for salient regions only.

Regarding the caption distinctness predictor, we found a predictor based on a bounding box and confidence score cannot be trained. However, we hypothesize that such a predictor exists. Yet it might need more variables such as for instance the set C itself. Note that the localization layer of DenseCap generates a rather extensive number of bounding box proposals that are subsampled using a confidence score, a value obtained by a neural net. In future work, this approach to score captions shall be tested.

Further, we would like to underline the fact that dense captioning can be extended by introducing additional criteria to the set of generated captions. Above, we defined a criterion minimizing the cardinality of the set. A possible criterion might cover, for example, restriction to value adding captions. In our criterion we assume $\Delta(c) > 0$, excluding $\Delta(c) = 0$. This means information that is represented by more than one caption is not present in C^*. Future work shall find a way of extending our C^* with captions $\Delta(c) = 0$.

Lastly, the results are biased by the fact Pascal 50S is a dataset designed for full-image captioning whereas we used it unsuitably for the dense captioning task. However, our approach shows that this trick is feasible and yields promising results; and the set of generated captions can be optimized to correspond to ground truth of different nature. In further work, optimizing the set of generated captions to a natural language text describing the image shall be tested.

6 Conclusions

In this thesis, we elaborated on image captioning, underlining especially characteristics of dense image captioning. We showed Convolutional Neural Networks and Reccurent Neural Nets are fundamental parts of modern Deep Learning that successfully provides solutions to the discussed tasks. It was demonstrated the most recent models draw primarily on CNN-based image encoders and the LSTM recurrent method. Concretely, the models DenseCap and Neural Image Caption were compared and examined.

Taking drawbacks of available evaluation techniques into account, we showed the qualities of the SPICE score when evaluating a dense captioning model using a full-image dataset. Concretely, SPICE can be extended in such a way it accepts a set of captions referring to an image and evaluates it thoroughly against ground truth.

In addition, we showed that SPICE correlates highly with maximal sets overlap and therefore this function is applicable to cases where big data are concerned.

When analysing the outcomes of DenseCap, it was found the set of generated captions is redundant, restricted and dependant on scene setting. In particular, the set contains identical captions in terms of textual similarity and consists of greatly reduced vocabulary compared to the training set. Most importantly, the language model in DenseCap can be viewed as a sentence retrieval system since 92% of the generated captions are within the training examples. However, the quality of those and the novel ones remains the same. Together with the scene setting dependency problem, we arrived at the conclusion the DenseCap model is over-fitted in those terms.

Also, we found that improving such a complex system is not straightforward. By adapting approaches successful in full-image captioning, the results are worsen in case of dense captioning. Similarly, a SVM predictor was proposed to estimate the set of distinct captions. The chosen methods illustrates that generation of high-quality set of captions that together describe the image comprehensively and non-redundantly, remains an open problem.

References

[1] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks.," in *NIPS* (P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1106–1114, 2012.

[3] Y. LeCun, L. Bottou, G. Orr, and K. Muller, "Efficient backprop," in *Neural Networks: Tricks of the trade* (G. Orr and M. K., eds.), Springer, 1998.

[4] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1," ch. Learning Internal Representations by Error Propagation, pp. 318–362, Cambridge, MA, USA: MIT Press, 1986.

[5] J. Johnson, A. Karpathy, and F. Li, "Densecap: Fully convolutional localization networks for dense captioning," *CoRR*, vol. abs/1511.07571, 2015.

[6] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: Lessons learned from the 2015 MSCOCO image captioning challenge," *CoRR*, vol. abs/1609.06647, 2016.

[7] I. Goodfellow, Y. Bengio, and A. Courville, "Deep learning." Book in preparation for MIT Press, 2016.

[8] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, pp. 65–386, 1958.

[9] M. Minsky and S. Papert, "Perceptrons : an introduction to computational geometry," 1969.

[10] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[11] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *CoRR*, vol. abs/1512.00567, 2015.

[12] C. Szegedy, S. Ioffe, and V. Vanhoucke, "Inception-v4, inception-resnet and the impact of residual connections on learning," *CoRR*, vol. abs/1602.07261, 2016.

[13] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS-11)* (G. J. Gordon and D. B. Dunson, eds.), vol. 15, pp. 315–323, Journal of Machine Learning Research - Workshop and Conference Proceedings, 2011.

[14] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010.

[15] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Netw.*, vol. 4, pp. 251–257, Mar. 1991.

[16] D. Fan, M. Sharad, A. Sengupta, and K. Roy, "Hierarchical temporal memory based on spin-neurons and resistive memory for energy-efficient brain-inspired computing," *CoRR*, vol. abs/1402.2902, 2014.

[17] M. Otahal, M. Najman, and O. Stepankova, "Design of neuromorphic cognitive module based on hierarchical temporal memory and demonstrated on anomaly detection," *Procedia Computer Science*, vol. 88, no. Complete, pp. 232–238, 2016.

[18] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, pp. 2278–2324, 1998.

[19] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," *Journal of Machine Learning Research*, 2015.

[20] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *CoRR*, vol. abs/1503.04069, 2015.

[21] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, pp. 1735–1780, Nov. 1997.

[22] A. Karpathy, J. Johnson, and F. Li, "Visualizing and understanding recurrent networks," *CoRR*, vol. abs/1506.02078, 2015.

[23] L. Fei-Fei, A. Iyer, C. Koch, and P. Perona, "What do we perceive in a glance of a real-world scene?," *Journal of Vision*, vol. 7, no. 1, p. 10, 2007.

[24] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results." http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html.

[25] J. Mao, W. Xu, Y. Yang, J. Wang, and A. L. Yuille, "Explain images with multimodal recurrent neural networks," *CoRR*, vol. abs/1410.1090, 2014.

[26] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," *CoRR*, vol. abs/1411.4389, 2014.

[27] R. Kiros, R. Salakhutdinov, and R. S. Zemel, "Unifying visual-semantic embeddings with multimodal neural language models," *CoRR*, vol. abs/1411.2539, 2014.

[28] H. Fang, S. Gupta, F. N. Iandola, R. K. Srivastava, L. Deng, P. Dollár, J. Gao, X. He, M. Mitchell, J. C. Platt, C. L. Zitnick, and G. Zweig, "From captions to visual concepts and back," *CoRR*, vol. abs/1411.4952, 2014.

[29] X. Chen and C. L. Zitnick, "Learning a recurrent visual representation for image caption generation," *CoRR*, vol. abs/1411.5654, 2014.

[30] A. Farhadi, M. Hejrati, M. A. Sadeghi, P. Young, C. Rashtchian, J. Hockenmaier, and D. Forsyth, "Every picture tells a story: Generating sentences from images," in *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ECCV'10, (Berlin, Heidelberg), pp. 15–29, Springer-Verlag, 2010.

[31] R. Gerber and H.-H. Nagel, "Knowledge representation for the generation of quantified natural language descriptions of vehicle traffic in image sequences.," in *ICIP (2)*, pp. 805–808, 1996.

[32] B. Z. Yao, X. Yang, L. Lin, M. W. Lee, and S. C. Zhu, "I2t: Image parsing to text description.," *Proceedings of the IEEE*, vol. 98, no. 8, pp. 1485–1508, 2010.

[33] R. Socher and F.-F. Li, "Connecting modalities: Semi-supervised segmentation and annotation of images using unaligned text corpora.," in *CVPR*, pp. 966–973, IEEE Computer Society, 2010.

[34] C. L. Zitnick, D. Parikh, and L. Vanderwende, "Learning the visual interpretation of sentences," in *2013 IEEE International Conference on Computer Vision*, pp. 1681–1688, Dec 2013.

[35] A. Karpathy, A. Joulin, and F. Li, "Deep fragment embeddings for bidirectional image sentence mapping," in *Advances in Neural Information Processing Systems 27* (Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, eds.), pp. 1889–1897, Curran Associates, Inc., 2014.

[36] M. Hodosh, P. Young, and J. Hockenmaier, "Framing image description as a ranking task: Data, models and evaluation metrics," *J. Artif. Int. Res.*, vol. 47, pp. 853–899, May 2013.

[37] R. Socher, A. Karpathy, Q. Le, C. Manning, and A. Ng, "Grounded compositional semantics for finding and describing images with sentences," *Transactions of the Association for Computational Linguistics*, vol. 2, pp. 207–218, 2014.

[38] V. Ordonez, G. Kulkarni, and T. L. Berg, "Im2text: Describing images using 1 million captioned photographs," in *Neural Information Processing Systems (NIPS)*, 2011.

[39] J. Devlin, H. Cheng, H. Fang, S. Gupta, L. Deng, X. He, G. Zweig, and M. Mitchell, "Language models for image captioning: The quirks and what works," *CoRR*, vol. abs/1505.01809, 2015.

[40] M. Kolár, M. Hradis, and P. Zemcík, "Technical report: Image captioning with semantically similar images," *CoRR*, vol. abs/1506.03995, 2015.

[41] P. Kuznetsova, V. Ordonez, A. C. Berg, T. L. Berg, and Y. Choi, "Collective generation of natural image descriptions," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, (Stroudsburg, PA, USA), pp. 359–368, Association for Computational Linguistics, 2012.

[42] S. Li, G. Kulkarni, T. L. Berg, A. C. Berg, and Y. Choi, "Composing simple image descriptions using web-scale n-grams," in *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, CoNLL '11, (Stroudsburg, PA, USA), pp. 220–228, Association for Computational Linguistics, 2011.

[43] A. Karpathy and F. Li, "Deep visual-semantic alignments for generating image descriptions," *CoRR*, vol. abs/1412.2306, 2014.

[44] J. Mao, W. Xu, Y. Yang, J. Wang, Z. Huang, and A. L. Yuille, "Learning like a child: Fast novel visual concept learning from sentence descriptions of images," *CoRR*, vol. abs/1504.06692, 2015.

[45] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015.

[46] R. B. Girshick, "Fast R-CNN," *CoRR*, vol. abs/1504.08083, 2015.

[47] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015.

[48] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector," *CoRR*, vol. abs/1512.02325, 2015.

[49] P. J. Werbos, "Generalization of backpropagation with application to a recurrent gas market model," *Neural Networks*, vol. 1, pp. 339–356, 1988.

[50] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.

[51] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *CoRR*, vol. abs/1406.1078, 2014.

[52] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *CoRR*, vol. abs/1409.0473, 2014.

[53] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *CoRR*, vol. abs/1409.3215, 2014.

[54] R. Kiros, R. Salakhutdinov, and R. Zemel, "Multimodal neural language models," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)* (T. Jebara and E. P. Xing, eds.), pp. 595–603, JMLR Workshop and Conference Proceedings, 2014.

[55] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *Trans. Sig. Proc.*, vol. 45, pp. 2673–2681, Nov. 1997.

[56] K. Xu, J. Ba, R. Kiros, K. Cho, A. C. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," *CoRR*, vol. abs/1502.03044, 2015.

[57] R. Vedantam, C. L. Zitnick, and D. Parikh, “Cider: Consensus-based image description evaluation,” *CoRR*, vol. abs/1411.5726, 2014.

[58] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “Bleu: A method for automatic evaluation of machine translation,” in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL ’02, (Stroudsburg, PA, USA), pp. 311–318, Association for Computational Linguistics, 2002.

[59] A. Lavie and A. 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, (Stroudsburg, PA, USA), pp. 228–231, Association for Computational Linguistics, 2007.

[60] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft COCO: common objects in context,” *CoRR*, vol. abs/1405.0312, 2014.

[61] P. Anderson, B. Fernando, M. Johnson, and S. Gould, “Spice: Semantic propositional image caption evaluation,” in *ECCV*, 2016.

[62] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?,” *CoRR*, vol. abs/1411.1792, 2014.

[63] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, M. Bernstein, and L. Fei-Fei, “Visual genome: Connecting language and vision using crowdsourced dense image annotations,” 2016.

[64] J. Hessel, N. Savva, and M. J. Wilber, “Image representations and new domains in neural image captioning,” *CoRR*, vol. abs/1508.02091, 2015.

[65] J. Johnson and A. Karpathy, “Densecap repository.” `https://github.com/jcjohnson/densecap`, April 2016.

[66] C. Shallue, “Show and tell: image captioning open sourced in tensorflow.” `https://research.googleblog.com/2016/09/show-and-tell-image-captioning-open.html`, September 2016.

List of Figures

List of Tables