# Image captioning with Deep Neural Networks - Visualizing attention maps with Deep Taylor Decomposition

Master's thesis

for acquiring the degree of
Master of Science (M.Sc.)

in Statistics

at the School of Business and Economics
of Humboldt-Universität zu Berlin

submitted by
Christopher Lennan
Student no. 558204

Examiners:
Prof. Dr. Stefan Lessmann
Dr. Grégoire Montavon

Berlin, November 6, 2017

# Contents

# List of Figures



# List of Tables

# Abbreviations

**BPTT** Back-propagation thorugh time.

**CNN** Convolutional Neural Network.

**CRF** Conditional Random Field.

**DTD** Deep Taylor Decomposition.

**FFNN** Feed Forward Neural Network.

**HMM** Hidden Markov Model.

**ILSVRC** ImageNet Large Scale Visual Recognition Challenge.

**LRP** layer-wise relevance propagation.

**LSTM** Long-Short-Term Memory.

**MRF** Markov Random Field.

**MRNN** Multimodal Recurrent Neural Network.

**MSCOCO** Microsoft Common Objects in Context.

**NIC** Neural Image Caption.

**NLP** Natural Language Processing.

**RBM** Restricted Boltzmann Machine.

**ReLU** rectified linear unit.

**RNN** Recurrent Neural Network.

**VGG16** Oxford Visual Geometry Group 16-layer model.

# 1 Introduction

Neural Networks have in recent years achieved breakthrough performances in image recognition and natural language understanding tasks. A natural extension is to combine these two tasks and design a model that describes the content of images, referred to as image captioning. The goal is to train an end-to-end system that takes an image as input and outputs a written description of what the image contains.

Besides the incentive to replicate the fascinating human ability to process visual information into descriptive language, an image captioning system has multiple real world applications, e.g. designing systems for visually impaired people, transcribing entire movies, or labelling images for further image processing tasks.

An important skill of a successful image captioning system is its ability to focus on certain parts of the image to generate the next word in a caption, which is called attention mechanism. By visualizing the attention maps, i.e. the weights the system assigns to each image pixel, one gets feedback on what parts of the image were important for generating the next word. The attention mechanism thus serves as a sense checking feature, as it allows to visually observe the importance of inputs to the classification result, and contributes to the growing efforts of making neural networks more explainable.

This is where this thesis ties in - the relevance propagation method Deep Taylor Decomposition (DTD) is applied to an image captioning system with the goal of generating more precise and richer visualizations, called heatmaps, of the attention maps. Applied to convolutional neural networks it has been shown that DTD produces fine grained heatmaps with contours of objects that were important for a classification. In the context of an image captioning system, the idea is to use DTD to produce these rich heatmaps to better understand which image features where important for the attention mechanism and ultimately the system for generating the next word.

Deep Learning methods are only on the verge of being implemented into more and more products of everyday life. In order to rely on Deep Learning methods in safety-related settings, e.g. autonomous driving, it is crucial that these methods are better understood and scrutinized, for which this thesis provides new insights.

# 2 Literature Review

An image captioning system consists of three main building blocks: i) Convolutional Neural Network (CNN), ii) Recurrent Neural Network (RNN), and iii) attention mechanism. Each building block is a type of neural network and represents a dynamic research area that entails a vast range of publications. The following literature review focuses on CNNs in the object recognition domain, which is a subtask of computer vision that aims to identify objects in images and video. For RNNs the focus is on language modeling, where the goal is to predict the next word in a sentence given context. For a thorough literature review of neural networks with a detailed timeline see Section 5 in Schmidhuber [70].

## 2.1 Convolutional Neural Networks

A type of Neural Network that was specifically designed for visual recognition tasks and was inspired by the human visual cortex, is a CNN. CNNs were first introduced by Fukushima [37] in 1980, under the name of Neocognitron, and later improved to its current day structure by LeCun et al. [54] in 1998.

Many enhancements to CNNs were developed in the subsequent years, including [72, 26], and outperformed other recognition systems on medium sized datasets, e.g. the MNIST handwritten digit dataset [25]. The advent of more computing power, especially in the form of Graphical Processing Units (GPUs), allowed to train larger CNNs from around 2010 onwards. Paired with the creation of large scale datasets like ImageNet [27] and LabelMe [16], with more than 1 million training images, allowed to control overfitting while training these deep CNNs.

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) has become the standard benchmark for measuring the performance of large-scale image recognition systems [68]. The ILSVRC was launched in 2010 with $1,461,406$ images across 1000 object categories in the image classifications task (systems predict a list of object categories present in an image). Alongside the image classification task, ILSVRC features an object localization task (predict single object category and its position) since 2011, and an object detection task (predict multiple object categories in an image and their position) since 2013.

Krizhevsky et al. [51] in 2012 significantly improved existing classification accuracies on ImageNet by training a CNN with five convolutional layers, three max-pooling layers, and three fully connected layers. Two copies of this newtork were trained simultaneously on two GPUs, and shared weights at certain layers. The network achieved a 37.5% top-1 and 17.0% top-5 error rate in the 2010 ILSVRC [67] dataset. The previously best published results at that date were 45.7% top-1 and 25.7% top-5 error rates [69].

Table 1: ILSVRC results overview

| | Image classification | | Object localization | | Object detection | |
|---|---|---|---|---|---|---|
| Year | Team | Top-5 error | Team | Top-5 error | Team | MAP† |
| 2010 | NEC-UIUC [1] | 28.19% | - | - | - | - |
| 2011 | XRCE [2] | 25.77% | UvA & UoT [2] | 42.5% | - | - |
| 2012 | SuperVis. [51] | 16.42% | SuperVis. [51] | 34.19% | - | - |
| 2013 | Clarifai [89] | 11.74% | OverFeat [71] | 29.88% | UvA-Euv. [4] | 22.58% |
| 2014 | GoogLeNet [79] | 6.67% | VGG [74] | 25.32% | NUS [5] | 37.21% |
| 2015 | MSRA [82] | 3.57% | MSRA [82] | 9.02% | MSRA [82] | 62.07% |
| 2016 | Trimps-Sou. [7] | 2.99% | Trimps-Sou. [7] | 7.71% | CUImage [7] | 66.28% |

† Mean average precision
Results collected from [68, 82, 1, 2, 3, 4, 5, 6, 7]

Simonyan and Zisserman [74] secured the second place in the ILSVRC-2014 challenge with a 7.3% top-5 error rate, using an ensemble of seven models. The first place was secured by Szegedy et al. [79] with their GoogLeNet architecture, a 22 layers deep CNN, with a 6.7% error rate. Post submission Simonyan and Zisserman [74] managed to decrease the top-5 error rate to 7.0% with a single model, the best single model result among all competitors that year. In this thesis we use the Oxford Visual Geometry Group 16-layer model (VGG16) from [74] as the encoder of the image captioning system (see Section 3.3).

## 2.2 Recurrent Neural Networks

The first neural networks with recurrent connections were designed by Elman [32] and Fahlman [33], who successfully applied these networks to find semantic features for words and learning to recognize characters in Morse code. RNNs have since been applied to various different classification tasks, ranging from speech recognition (e.g. Mikolov et al. [59]), to machine translation (e.g.

Kalchbrenner and Blunsom [44], Cho et al. [23], Bahdanau et al. [13], Sutskever et al. [78]), and language modeling (e.g. Graves [39]).

A key feature of RNNs is to store information in hidden states and pass them on to the next iteration. They however struggle to store this information over many iterations and thus are not capable to model long term dependencies (see for example [42], p.1). Hochreiter [40] attributed this inability to the vanishing or exploding gradients problem, where the backpropagated error signals either decay exponentially in the number of layers, or explode ([41], p.4,5). The unstable gradients prohibit learning through oscillating weights or a prohibitive amount of time necessary for learning with vanishing gradients. Hochreiter and Schmidhuber [42] introduced a novel recurrent cell, called Long-Short-Term Memory (LSTM), which addressed the vanishing or exploding gradient problem by introducing an additional long term memory state (see Section 4.3 for further details).

Graves [39] used LSTMs to generate long ranging sequences by predicting each data point at a time. In the context of language modeling he showed that LSTMs outperform standard RNN structures in next word predictions. Xu et al. [84] use this kind of sequence generator for their image captioning system and its structure will be further discussed in Section 4.2.1.

## 2.3 Image captioning

The automatic generation of natural language image descriptions, called image captioning, has been a very dynamic research area over the past decade, and we follow [87, 57] in dividing this area into three approaches, template-based methods, retrieval-based methods, and language-based methods.

### 2.3.1 Template-based methods

This method generates sentences from predefined templates that comprise grammar rules. The algorithms are generally trained by splitting sentences into fragments, aligning them with predicted content in images, and then generate sentences based on the template.

Kulkarni et al. [52] used a Conditional Random Field (CRF) to predict the objects in an image (nouns), visual attributes of the objects (adjectives), and spatial relationships between objects (prepositions). They then organise this information into triples (the template), e.g. [(white, cloud); in; (blue, sky)],

and then explore several methods to find the most likely fill-words to generate a complete natural language sentence, e.g. "There is a white cloud in the blue sky" ([52], p.2896). In a similar fashion, Yang et al. [85] employed a Hidden Markov Model (HMM) to predict the most likely quadruplet of objects, actions, scenes, and prepositions in an image, an then used a template approach to genrate the sentence. Mitchell et al. [60] follow [52] for image feature extraction and extend the template to a syntactic decision tree, following Yao et al. [86].

The major drawback of this method is its dependence on the templates, resulting in low generalization ability.

### 2.3.2 Retrieval-based methods

This approach finds the best matching sentence for an image from a set of sentences. It can thus not generate novel captions, but achieve human level descriptions given that the set of sentences to choose from was generated by humans ([87], p.2). Retrieval-based models are generally encoder-encoder frameworks, in which images and sentences get encoded to a common space, and then an objective is minimized to align images with their corresponding sentences. The trained model then allows to map images to sentences that have a similar embedding in the common space.

Farhadi et al. [34] encode images and corresponding sentences into a common "triplet" space of $(object, action, scene)$. Images are projected by solving a Markov Random Field (MRF) and descriptive sentences are then retrieved from a set of sentences written to describe similar images ([34], p.3-6). Ordonez et al. [64] applied a similar approach on a large captioned photo collection with 1 million images.

Models in the spirit of [34] are limited to represent images and sentences with a triplet ([46], p.2). A variety of methods have been employed to achieve richer common representations, including Deep-Boltzmann machines [77], log-bilinear models [49], and topic models [14, 43].

Methods that are closer connected to this thesis are those that encode images and sentences with neural networks. Frome et al. [36] retrained lower layers of a pretrained CNN to match the vector representation of words associated with that image. Socher et al. [75] extended this model by mapping

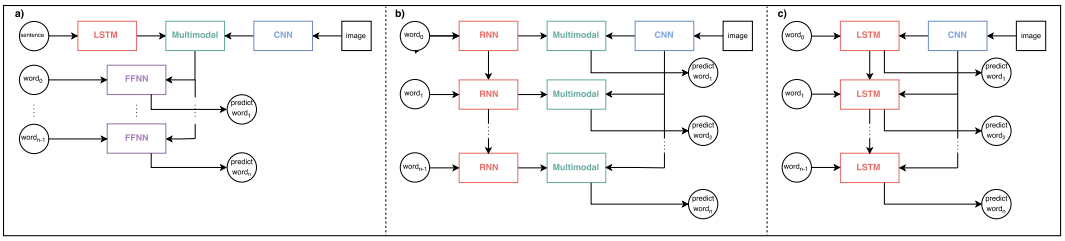



Figure 1: Evolution of NIC [50, 57, 81]

entire sentences into a common embedding space using a RNN. Devlin et al. [30] use a CNN to encode a test image, calculate the cosine similarity to each training image, and then create a set of candidate captions from the captions of the most similar images ([30], p.3). In one of their models they then use a K-means clustering configuration to retrieve the optimal caption, and find that it works similarly well, in terms of both automatic metrics and human judgements, to language-based models with a RNN decoder.

Karpathy and Fei-Fei [45] improved image-sentence retrieval benchmarks by embedding fragments of images and fragments of sentences into a common space, and thus achieve a more fine grained representation of each. Their approach also offers interpretable predictions as the decision to pick a certain sentence can be attributed to certain fragments of the image.

### 2.3.3 Language-based methods

Language-based models aim to learn the joint probability distribution of image and text content in a common space, in order to generate novel sentences that have richer syntactical structure than template based approaches ([87], p.2).

Just as Duygulu et al. [31] were inspired for image to word models by the machine translation literature in 2002, the main inspiration for tackling image captioning with neural networks came from recent machine translation advances with neural networks, e.g. [44, 23, 13, 78]. In these encoder-decoder neural networks, a RNN is used to encode sentences in one language, and, based on the hidden states of the encoder, a subsequent RNN decodes the hidden states and generates sentences in the target language.

As depicted in Figure 1(a), Kiros et al. [50] use a CNN to encode images and a LSTM to encode whole sentences, to embed both in a common multi-

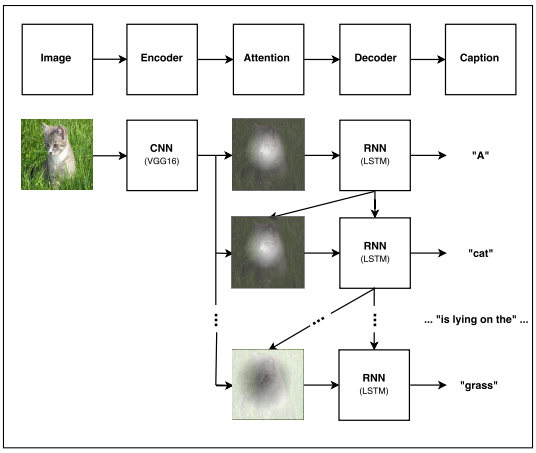



Figure 2: Architecture sketch of NIC with attention [84]

modal space. After applying a scoring function in multimodal space, its output serves as input to a Feed Forward Neural Network (FFNN), together with the previous word and the image encoded by the CNN. The FFNN then predicts the next word. In a previous paper, Kiros et al. [49] use a FFNN to predict the next word in a sentence, given the image and previous words.

Mao et al. [57] extended [50] by using a RNN to map each word into a multimodal space, rather than the whole sentence, from which the next word gets predicted (see Figure 1(b)). Vinyals et al. [81] simplified this structure by feeding the encoded image straight into the LSTM decoder (see Figure 1(c)), and was thus the first image captioning system in the pure encoder-decoder fashion which were pioneered in the machine translation domain. Feeding the encoded image straight into the decoder allowed it to keep track of the objects that have been described by the generated text already ([81], p.2). And the LSTM cell is a more capable decoder to memorise those long term dependencies than the vanilla RNN used in [57].

In 2016 Xu et al. [84] added a attention mechanism in the spirit of Bahdanau et al. [13] to the Neural Image Caption (NIC) from [81]. The attention mechanism is a FFNN that takes the previous hidden state and encoded image as inputs, and predicts a focus area, the attention weights, on the input image. The encoded image is then weighted by the attention weights and fed into the LSTM decoder to predict the next word (see Figure 2 for an architecture sketch). The ability of the decoder to focus on different parts of the image to generate each word, enabled richer and more detailed descriptions

of the system. It also allowed attributing certain areas of the image being responsible for parts of the caption, e.g. a nonsensical "blue dog" might have arisen from the attention mechanism focusing on a blue sky in the background.

Various enhancements have been developed recently for the NIC framework - Yao et al. [87] add high level semantic attributes as inputs to the LSTM decoder, whereas very recently Fortunato et al. [35] applied a Bayesian Recurrent Neural Network as the decoder and achieved improved results on the Microsoft Common Objects in Context (MSCOCO) dataset.

## 2.4 Attention mechanisms in Neural Networks

In human vision, the retina is a system that allows to focus with high resolution, the fovea, on parts of the optic input that are relevant for a given task, while the periphery is processed in low resolution (Larochelle and Hinton [53], p.1). By sequentially focusing on different parts of an optic input, fixational eye movements (see e.g. Martinez-Conde et al. [58]), the human visual system achieves translation invariance and a reduction in the number of pixels to process.

Inspired by human vision, Larochelle and Hinton [53] interpreted object recognition in images as a sequential process with retinal fixations. They trained a Restricted Boltzmann Machine (RBM) to use multiple task specific fixations with few pixels. Denil et al. [28] build a system for simultaneous object tracking and recognition based on gaze date. Their system learns online an attention strategy for fixation points and thus better estimate the location of the target object.

Mnih et al. [61] proposed a RNN for image recognition which adaptively selects regions to focus on and processing only the focus region at high resolution. This approach circumvents the disadvantage of CNNs, whose parameters increase linearly with an increase in image pixels.

Only recently have attention methods been applied to RNN architectures used in the Natural Language Processing (NLP) domain. Bahdanau et al. [13] proposed a novel neural machine translation architecture, in which the decoder RNN is enhanced with an alignment model, that focuses on certain parts of a source sentence while decoding a translation.

Xu et al. [84] adopted the attention mechanism from [13] in the soft attention approach of their image captioning system. Similarly to [13], they parametrized the attention model as a FFNN, which outputs weights over the input image, that allows the decoder RNN to focus on parts of the image for generating the next word in a caption. The analysis and visualization of the soft attention mechanism from [84] is the focus of this thesis and will be presented in more detail in Section 5.

## 2.5 Neural network decompositions

There exists extensive research in understanding nonlinear classifiers and, especially recently, in understanding deep neural networks. One branch of this research are pixel-wise decompositions, i.e. decomposing the classification outcome of an image classification neural network in terms of its input pixels. Bach et al. [12] introduced a general decomposition method called layer-wise relevance propagation (LRP), which assumes that a classifer can be decomposed into several layers of computation, and that relevances at each layer follow a set of contraints. For multilayer neural networks, they proposed back-propagating relevances to lower layer neurons proportionally to neuron pre-activation values.

Bazen and Joutard [15] framed a widely used linear decomposition method as a first order Taylor expansion, and based on this, formulated a unified approach for decomposing nonlinear models, called Taylor decomposition. Inspired by this approach, Bach et al. [12] conceptually introduced Taylor-type decompositions for multilayer neural networks, which was further developed by Montavon et al. [62] into the DTD framework, which propagtes relevances based on layer-wise Taylor decompositions. Montavon et al. [62] also provided practical implementation rules for applying DTD to FFNN and CNN.

The DTD method by Montavon et al. [62] is used for visualizing the attention maps in this thesis and will be further described in Section 6.

# 3 Convolutional Neural Networks

Similar to standard FFNNs, CNNs are hierarchical Neural Networks, with neurons that perform dot products between inputs and learnable weights, generally followed by a nonlinearity [9]. Their convolutional layers alternate with

subsampling layers, similar to simple and complex cells in human primary visual cortex ([26], p.1237). We will follow the notation from Goodfellow et al. [38] in the following theoretical presentation.

## 3.1 Convolution operation

Convolution is a mathematical operation on two functions $x$ and $w$ with argument $t \in \mathbb{R}$ defined as (p.331, [38]):

$$s(t) = \int x(a)w(t-a)da \tag{1}$$

In the CNN literature the function $x$ is usually referred to as the input, the function $w$ as the kernel or filter or receptive field, and the output $s$ as the activations or feature map.

The inputs and kernels in neural networks have finite dimensions and we thus work in discrete time, i.e. $t \in \mathbb{N}$ and

$$s(t) = \sum_{a \in \mathbb{N}} x(a)w(t-a). \tag{2}$$

Inputs and kernels are usually multidimensional arrays (tensors), so convolutions are applied along multiple axes. For a two-dimensional input $x$, and two-dimensional kernel $w$, the convolution operation is then

$$s(i,j) = \sum_{m=0}^{M} \sum_{n=0}^{N} x(m,n)w(i-m, j-n), \tag{3}$$

which is equivalent to

$$s(i,j) = \sum_{m=0}^{M} \sum_{n=0}^{N} x(i-m, j-n)w(m,n), \tag{4}$$

given the commutative property of the convolution operation. In many neural network implementations, convolution is actually implemented as a cross-correlation function ([38], p.333):

$$s(i,j) = \sum_{m=0}^{M} \sum_{n=0}^{N} x(i+m, j+n)w(m,n), \tag{5}$$

For real-valued inputs, cross-correlation and convolution are equivalent, except that the kernel is not rotated. Training an algorithm with convolution is equivalent to training it with cross-correlation, as the appropriate values for each filter will be learned in the same manner, but only stored rotated relative to one another. Going forward, with convolution we will refer to the operation in equation 5.

The convolution operation can intuitevly be understood as sliding a window, the kernel, over an image, and at each position, a dot product between the kernel weights and image pixel values is taken. Sliding the window across the entire image yields a map of activation values, the feature map. Any part of the image, that has a similar pixel value structure as the weights in the kernel, will result in a high activation value (in absolute value). To see this, note that in Euclidean space the dot product of vectors $u, v$, with fixed length $\|u\| \, and \, \|v\|$, is defined as:

$$u'v = \|u\| \, \|v\| \, cos(\theta), \tag{6}$$

with $\theta$ being the angle between $u$ and $v$. In order to maximize $u'v$, $cos(\theta)$ needs to be maximized, which is the case when $\theta = 0$. Thus $u'v$ is maximized if $u$ and $v$ are codirectional.

## 3.2 CNN specific attributes

The design of CNNs addresses three main objectives ([54], p.6):

1. scalability through shared weights and sparse interaction
2. cope with shift of objects through local receptive fields (filters)
3. distortion invariance through spatial subsampling.

### 3.2.1 Shared weights and sparse interaction

The kernels in CNNs are generally smaller than the input. This sparse interaction reduces the number of parameters to store and consequently fewer operations to compute the output ([38], p.335). Another feature of a convolutional layer is parameter sharing. The kernel weights are applied at every positon of the input, and are thus not tied to a specific input element, as is the case in a fully connected layer. A CNN therefore does not learn a separate

set of parameters for every location, but rather one set, that is shared across all locations. The number of parameters in a kernel are usually significantly less than the input or output dimension, and therefore significantly more parameter efficient than a fully connected layer ([38], p.338).

Consider the following example from Figure 9.6 in [38]: vertical edges in an image can be computed by substracting each pixel value with its left neighbouring pixel. Using a convolution kernel with entries $(-1, 1)$, this operation requires $h \times (w-1) \times 3$ floating-point operations ($h$ = heigth, $w$ = width, each convolution includes two multiplications and one addition). The same transformation in a matrix operation would require $h \times w \times h \times (w-1)$ calculations. On a $224 \times 224$ dimensional image, the convolution approach would require 149 856 calculations, versus 2 506 391 552 in the matrix multiplication approach. Convolutions are thus a very efficient method to apply small, local linear transformations across an entire input image.

### 3.2.2 Local receptive fields

Another positive consequence of parameter sharing is a translation equivariance property. The convolution operation is equivariant towards shifts in inputs, e.g. shifting an image one pixel to the right ([38], p.338). A function $f$ is equivariant towards function $g$ if:

$$f(g(x)) = g(f(x)). \tag{7}$$

Given that a shift is a linear transformation and the linearity of the convolution operation, this result follows immediately. This means that it does not matter whether an image is shifted first and then a feature map is calculated, or whether a feature map is calculated first, and then a shift is applied. For image recognition, this plays a vital role, as the first layer filters of a CNN generally learn shapes like edges, which need to be detected all over an image, so sharing parameters makes a lot of sense.

### 3.2.3 Spatial subsampling

Spatial subsampling is realised in neural networks through pooling layers, which compress the activations from the previous layer, with the objective to keep important features, while discarding irrelevant details (Boureau et al.

[18], p.1). More precisely, pooling methods aim to make activations invariant to small translations, e.g. rotations or scale, and consequently support the neural network in learning general features, rather than subtleties derived from individual images. Differently put, it is more important whether a feature is present, rather than what its exact position and shape is ([38], p.342). Pooling is thus a method to avoid overfitting in a neural network.

Popular pooling operations are a sum, an average, a max, and a $L^2$ norm, among others. The pooling operation is performed in pre-specified neighbourhoods, for CNNs generally in a square. For example, a max pooling operation with a $3 \times 3$ neighbourhood, called pooling window, would return the max value of the nine activation values in that neighbourhood. The pooling operation is repeated across the entire input dimensions, where the step size between subsequent operations is referred to as pooling strides. Generally, pooling is applied in non-overlapping subregions of the input, i.e. the pooling window dimensions equal the pooling strides.

By compressing the activations from previous layers, pooling reduces the number of parameters in the network and thus improves computational efficiency and memory requirements.

The appropriate pooling operation depends significantly on the classification task, but finding the theoretical optimal strategy is difficault, given many confounding factors ([18], p.1), e.g. the pooling window size, or the resolution of low-level features. In object recognition tasks, the precise location of the object is less important, and pooling is thus an efficient method to avoid overfitting and make the system scalable. Other tasks, that might require precise spatial features, pooling might lead to underfitting ([38], p.347). Szegedy et al. [79] for example use both max and average pooling layers in their Inception CNN, and skip some pooling layers to avoid underfitting. For Simonyan and Zisserman [74] the optimal pooling architecture was to use only max pooling layers, in intervals of two and three convolutional layers. Boureau et al. [18] provide some theoretical insights, stating that max pooling performs particularly well for features with a low probability of activation.

## 3.3 CNN encoder in image captioning systems

In this thesis the CNN designed by Simonyan and Zisserman [74], called VGG16, is used as the encoder for the image captioning system. An im-

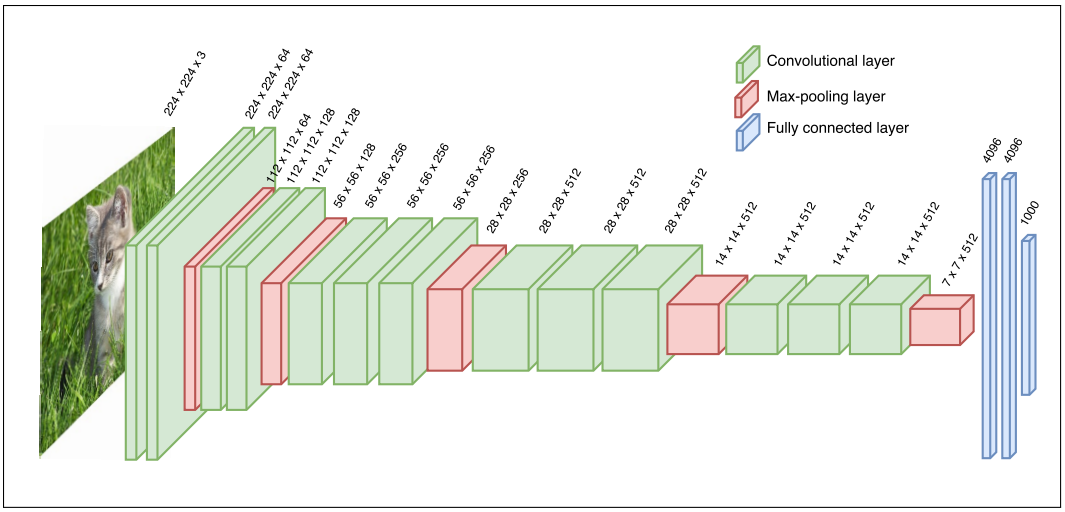


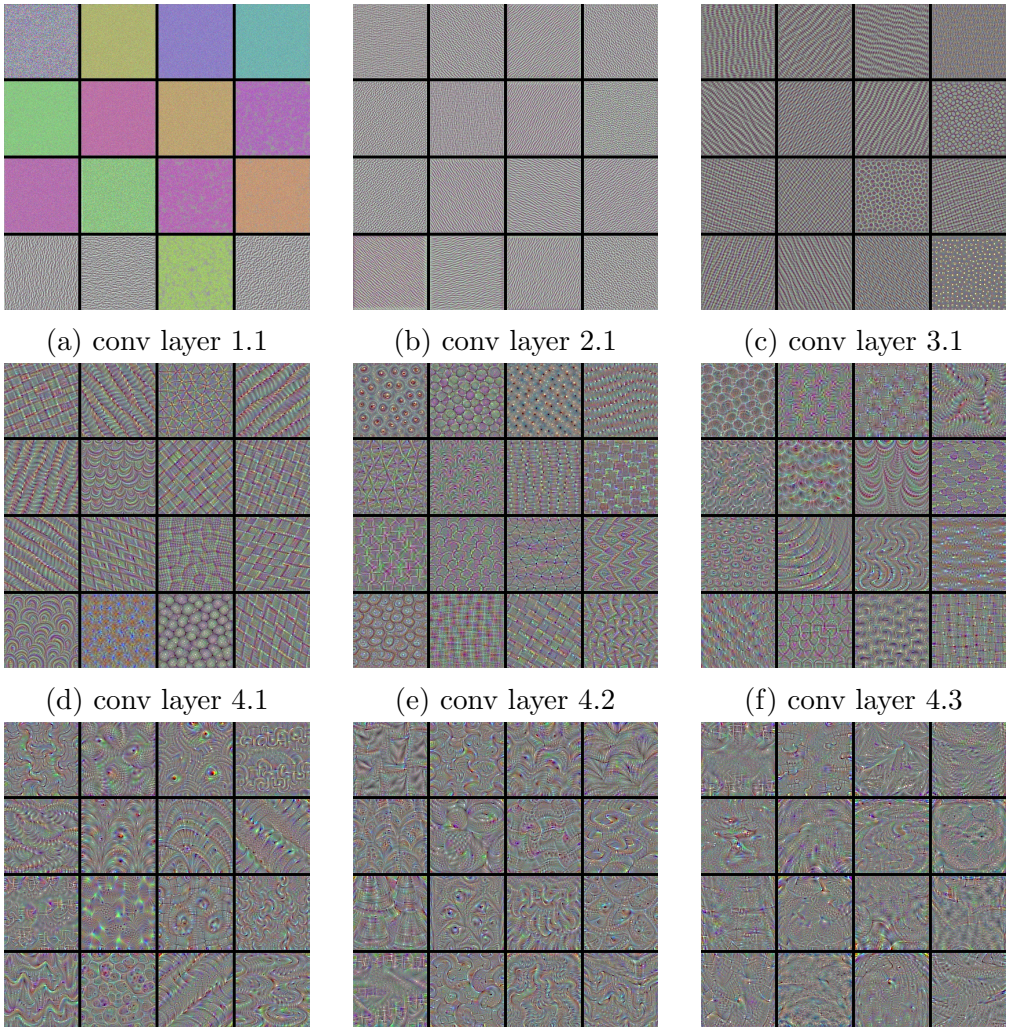

Figure 3: VGG16 architecture

VGG16 consists of five convolutional layer blocks (in green), where each block is followed by a max-pooling layer (in red). The convolutional layer blocks are followed by three fully connected layers (in blue), that map the filter activations from the convolutional layers to the 1000 object classes of the ImageNet challenge. For the image captioning system trained for this thesis the activations from the last convolutional layer are extracted.

portant feature of VGG16 is the use of filters with small receptive fields of $3x3$ pixels throughout the entire neural network, compared to e.g. first layer receptive fields of $11x11$ in Krizhevsky et al. [51] or $7x7$ in Zeiler and Fergus [89] (see Figure 3 for an architecture overview). Stacking several layers of small receptive fields allows for a more discriminative decision function, a reduction in parameter size, and consequently an implicit regularisation ([74], p.3).

A further advantage of VGG16 is its simple architecture, as it is a purely sequential model that does not feature multiple models trained in parallel and parameter sharing among models as for example in [51]. It should be noted however that in principle any CNN could be used as an encoder (Xu et al. [84], p.6).

The image captioning model trained for this thesis uses VGG16 as an encoder of images and extracts the activations from the third convolutional layer in the fifth block, i.e. the last convolutional layer in VGG16, referred to as convolutional extraction layer going forward. The activation values, after propagating an image through the CNN, in the convolutional extraction layer will be referred to as image annotations.



(a) conv layer 1.1 (b) conv layer 2.1 (c) conv layer 3.1

(d) conv layer 4.1 (e) conv layer 4.2 (f) conv layer 4.3

(g) conv layer 5.1 (h) conv layer 5.2 (i) conv layer 5.3

Figure 4: VGG16 convolutional filter visualisation (adapted from [24])

In CNNs the lower (i.e. later) convolutional layers generally correspond to more abstract features, whereas higher convolutional layers detect features like colours and simple patterns, e.g. edges (compare with Figure 4). See Section 5.2 for further details on the choice of the convolutional extraction layer.

# 4 Recurrent Neural Networks

RNNs are specialized neural networks to process sequential data, e.g. text, just as CNNs are specialized for handling grid like data, e.g. images ([38], p.373). In FFNNs and CNNs the assumption is that all inputs are independent of each other, and consequently the outputs are independent of each other. For NLP tasks, e.g. next word prediction, however, it is vital to exploit the sequential structure of the data and not treat the sequence elements as independent (wil [10]). This is what RNNs do, they repeat the same operation on each element of a sequence, and share information, the hidden state, between each operation. The hidden state serves as a memory that stores information of previous elements that might be of importance for the current state.

The following treatment of RNNs focuses on the language modeling class of RNNs and will follow the notation from Zaremba et al. [88] and Graves [39].

## 4.1 Standard RNN structure

Let $x = x_1, \ldots, x_T$ be an input sequence with $T$ elements, where each element $x_t \in \mathbb{R}^n$. Let $T_{n,m} : \mathbb{R}^n \to \mathbb{R}^m$ be an affine transformation of the form $Wx_t + b$, with some weight matrix $W \in \mathbb{R}^{m \times n}$, and bias array $b \in \mathbb{R}^m$. The hidden state at time point $t$, $h_t$, is defined as

$$h_t = f(T_{n,n}x_t + T_{n,n}h_{t-1}) \tag{8}$$
$$= f((Vx_t + b) + (Uh_{t-1} + c)) \tag{9}$$

where $V, W \in \mathbb{R}^{n \times n}$ are weight matrices and $f$ is a logistic sigmoid function

$$f(x) = \frac{e^x}{1 + e^x}, \tag{10}$$

or hyperbolic tangent function

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{11}$$

A RNN can then be defined as the deterministic state transition function

$$RNN : x_t, h_{t-1} \to h_t. \tag{12}$$

The hidden state $h_t$, which is generally zero initialized, is thus a function of the previous state's hidden state $h_{t-1}$ and the current input $x_t$.

It is important to note that the affine transformation $T_{n,n}$ does not depend on the state, i.e. the parameters for the weight matrices $V, U$ and bias terms $b, c$ are shared across time steps. The parameter sharing makes the RNN not dependend on the input sequence length $T$ and it can thus generalize to other lengths not seen in training ([38], p.373). It also allows RNNs to process important information regardless of its position ([38], p.373). For example, if the classification task is to extract the animal from the sentences

1. "A dog is running after a ball"
2. "Visually impaired people have been particularly helped by the service of guide dogs"

then a RNN trained on recognizing animals should pick up "dog" in both sentences. Given that each word in a sentence is processed by the same weight matrices, parameter sharing, the RNN can quite likely pick up "dog" regardless of its position at the beginning or end, and the differing sequence lengths.

## 4.2 RNN language model

In language models words are encoded as word vector embeddings, denoted as $x_t \in \mathbb{R}^E$, and one-hot encoded binary variables, $\tilde{x} \in \mathbb{R}^D$, where $E$ and $D$ are the word embedding and dictionary dimensions, respectively.

### 4.2.1 Training

Throughout training, the goal of the language model is to predict the next word in a sentence. Figure 5 illustrates the connections of the language model - at each time step $t$, the RNN gets as inputs the true word vector $x_t$, the previous hidden state $h_{t-1}$, that stores information of previous words seen by the model, and computes the current hidden state $h_t$. The output $y_t$ is then computed by two affine transformations

$$s_t = W_s h_t + b_s, \quad W_s \in \mathbb{R}^{E \times H}, \quad b_s \in \mathbb{R}^E,$$

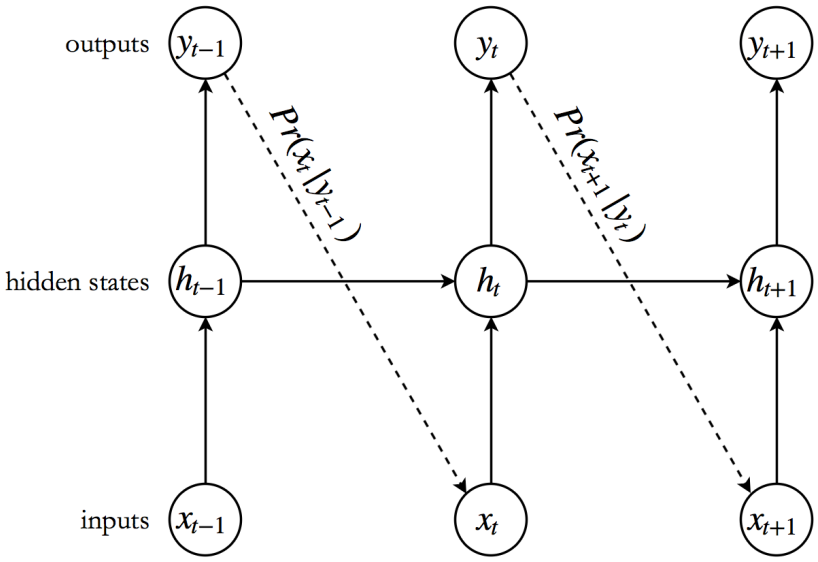



Figure 5: RNN language model (adapted from [39])

and

$$y_t = W_y s_t + b_y, \quad W_y \in \mathbb{R}^{D \times E}, \quad b_y \in \mathbb{R}^D.$$

The first transformation, where $E$ and $H$ are the word vector embedding and hidden state dimensions, can be understood as a decoding layer, where the hidden state $h_t$ gets projected to the word embedding space, and the output $s_t$ represents a word vector. The second transformation then maps the word vector $s_t$ to the dictionary space $D$, such that $y_t$ represents a non-normalized distribution over all words.

The probability of the next word is then given by the probability distribution $Pr(\tilde{x}_{t+1}|y_t)$, and the probability given by the model to the input vector sequence is

$$Pr(\tilde{x}) = \prod_{t=1}^{T} Pr(\tilde{x}_{t+1}|y_t). \tag{13}$$

$Pr(\tilde{x}_{t+1}|y_t)$ is highest when the learned probability distribution assigns a high probability to the true next word $\tilde{x}_{t+1}$, given the output $y_t$, that it has learned in the RNN. $Pr(\tilde{x})$ is thus maximized, if at each time step the predictive distribution assigns a high probability to the true next word.

The objective of the network is to maximize $Pr(\tilde{x})$, which is equivalent to minimising the negative log-likelihood, called loss or $L(\tilde{x})$ going forward

$$\begin{aligned}\max Pr(\tilde{x}) &= \max \prod_{t=1}^{T} Pr(\tilde{x}_{t+1}|y_t) \\ &\Leftrightarrow \max \log[\prod_{t=1}^{T} Pr(\tilde{x}_{t+1}|y_t)] = \max \sum_{t=1}^{T} \log[Pr(\tilde{x}_{t+1}|y_t)] \\ &\Leftrightarrow \min - \sum_{t=1}^{T} \log[Pr(\tilde{x}_{t+1}|y_t)] =: \min L(\tilde{x}).\end{aligned}$$

A dictionary has a finite amount of words and the words are generally one-hot encoded, i.e. $\tilde{x}_t$ is a zero vector $\in \mathbb{R}^D$, where $D$ is the dictionary size, and, if $\tilde{x}_t$ is the $k^{th}$ word in the dictionary, the $k^{th}$ element is one. $Pr(\tilde{x}_{t+1}|y_t)$ is then a multinomial distribution, for which the multinomial logistic function (or softmax) is a natural choice ([39], p.6)

$$Pr(\tilde{x}_{t+1} = k|y_t, W) = \frac{e^{y_t^k}}{\sum_{k'=1}^{D} e^{y_t^{k'}}},$$

where $y_t^k$ is the $k^{th}$ element of output vector $y_t$. The distribution is parametrised by $W$, which represents all weight matrices in the $RNN$. Intuitively, estimating $Pr(\tilde{x}_{t+1} = k|y_t, W)$ can be understood as finding the optimal weights in $W$, through Back-propagation thorugh time (BPTT), that maximises the probability assigned to the true word $\tilde{x}_{t+1}$, given the current word $\tilde{x}_t$ and previous hidden state $h_{t-1}$.

Plugging the softmax function into the loss $L(\tilde{x})$, we arrive at

$$\begin{aligned}L(\tilde{x}) &= -\sum_{t=1}^{T} \log \left( \frac{e^{y_t^k}}{\sum_{k'=1}^{D} e^{y_t^{k'}}} \right) \\ &= -\sum_{t=1}^{T} y_t^k + \underbrace{\sum_{t=1}^{T} \log \left( \sum_{k'=1}^{D} e^{y_t^{k'}} \right)}_{normalization} \\ \Rightarrow \min L(\tilde{x}) &\Leftrightarrow \min - \sum_{t=1}^{T} y_t^k\end{aligned}$$

Minimising $L(\tilde{x})$ can also be interpreted as minimizing the cross-entropy loss $H(p,q) = -\sum_{\tilde{x}} p(\tilde{x}) \log(q(\tilde{x}))$, where $p$ is the true distribution, and $q$ the estimated distribution. Note, that given the one-hot encoding of the input word vectors $\tilde{x}_t$, $p$ is a delta function that assigns all probability mass to the correct word position [9], so it is a one-hot vector itself. It follows that for each $\tilde{x}_t$

$$\begin{aligned} H(p(\tilde{x}_t), q(\tilde{x}_t)) &= -\sum_{k=1}^{D} p(\tilde{x}_t^k) \log(q(\tilde{x}_t^k)) \\ &= -\log\left(\frac{e^{y_t^k}}{\sum_{k'=1}^{D} e^{y_t^{k'}}}\right) \\ &= L(\tilde{x}_t) \end{aligned}$$

Given that the delta function has an entropy of zero, i.e. $H(p) = 0$, it follows immediately that the objective also minimizes the Kullback-Leibler Divergence $D_{KL}(p||q)$ between the true and predicted distribution

$$D_{KL}(p||q) = H(p,q) - H(p) = H(p,q).$$

As mentioned before, the cross entropy loss function thus tries to assign all probability mass of the predicted distribution $Pr(\tilde{x}_{t+1} = k|y_t, W)$ to the correct class.

### 4.2.2 Generating sequences

By training the network to predict the next word as described in section 4.2.1, the model can be used to generate new sentences by sampling from the output probabilities. The simplest approach is to pick at each time step $t$ the word with the highest probability, i.e.

$$\hat{x}_{t+1} = \underset{k=1,\ldots,D}{\arg\max}\, Pr(\tilde{x}_{t+1}^k|y_t, W).$$

## 4.3 Long-Short-Term Memory cells

Hochreiter and Schmidhuber [42] introduced a new RNN cell architecture called LSTM, which addressed the vanishing or exploding gradient problem that standard RNNs suffer from when modelling long term dependencies.

LSTMs have an explicit long term memory cell $c_t$, which allows it to memorize information for many time steps (Zaremba et al. [88], p.2).

$$LSTM : x_t, h_{t-1}, c_{t-1} \rightarrow h_t$$

$$\begin{aligned} i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1}) \\ f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1}) \\ o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1}) \\ g_t &= tanh(W_{xg}x_t + W_{hg}h_{t-1}) \end{aligned}$$

$$\begin{aligned} c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\ h_t &= o_t \odot tanh(c_t), \end{aligned}$$

where $\odot$ is an elementwise product and $\sigma$ a sigmoid function. The inputs $x_t$ and $h_{t-1}$ enter the LSTM cell through four gates - the input gate $i_t$ projects the inputs with the learned weight matrices $W_{xi}, W_{hi}$, and the input modulation gate $g_t$ learns weights $W_{xg}$ and $W_{hg}$ to modulate that contribution to the memory state $c_t$. The forget gate $f_t$ learns weights to control the contribution of the previous memory state $c_{t-1}$ to the current state $c_t$. Finally, the output gate then learns weight matrices $W_{xo}$ and $W_{ho}$ that determine how the memory state $c_t$ is transformed to the hidden state $h_t$.

Xu et al. [84] used a LSTM cell very similar to the one presented above, but added a context vector $z_t$ as input to the four gates (see Figure 6). The context $z_t$ is a weighted representation of low level image features extracted from the encoder CNN, which will be further explained in Section 5.2. The LSTM thus predicts the next word based on the current word $x_t$, the previous hidden state $h_{t-1}$, and a certain part of the input image, represented by the weighted context vector $z_t$

$$LSTM_{XU} : x_t, h_{t-1}, c_{t-1}, z_t \rightarrow h_t.$$

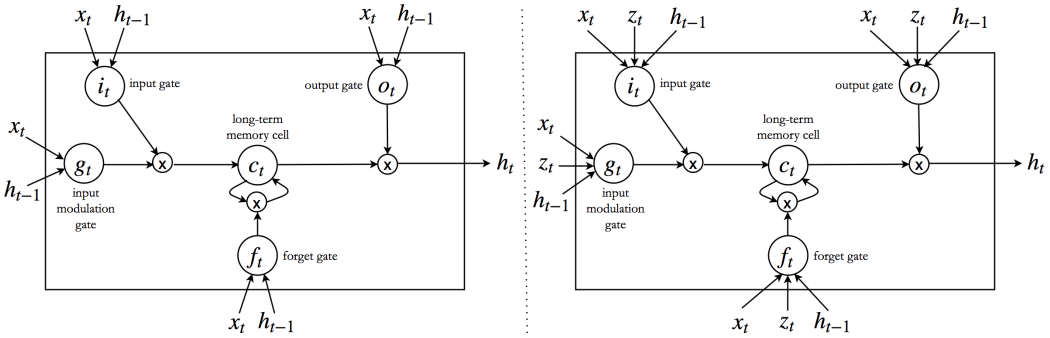



Figure 6: LSTM cells (adapted from [88, 84])

# 5 Attention mechanism

## 5.1 Attention in translation models

Neural machine translation has recently been very successfully applied to translation tasks, see e.g. [44, 23, 78]. These neural networks have "encoder-decoder" architectures, where a RNN is used to encode sentences in one language, and, based on the hidden states of the encoder, a subsequent RNN decodes the hidden states and generates sentences in the target language. These network need to compress all necessary information from the source language sentence into the hidden state vector, which makes it dificult for these networks to handle long sentences (see e.g. Bahdanau et al. [13], Cho et al. [22]).

To address this issue, Bahdanau et al. [13] introduced an attention mechanism, that proposes a set of words at each time step, that are the most relevant for generating the next word in the target language. This mechanism is called attention.

Again, let $x = x_1, \ldots, x_T$ be an input vector sequence in the source language, and $y = y_1, \ldots, y_T$ be the output vector sequence in the target language. Let the function $f$ be a RNN or LSTM, then the hidden state is $h_t = f(x_t, h_{t-1}) \in \mathbb{R}^n$ . Further, let $q$ be some nonlinear function that aggregates the hidden states $h_1, \ldots, h_T$ into a context vector $c = q(h_1, \ldots, h_T)$.

The decoder RNN is then usually trained to predict the next word $y_{t+1}$ given all previous words, the context $c$, and the hidden state from the decoder RNN $s_t$ (p.3, [13])

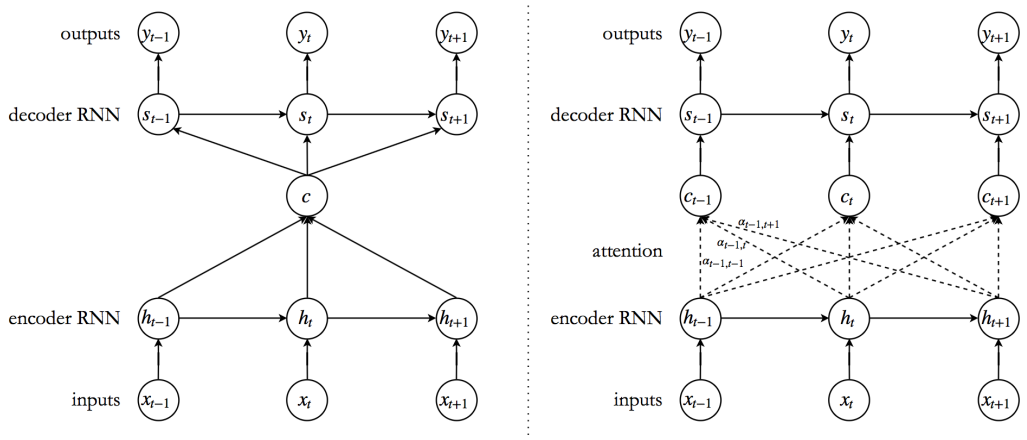



Figure 7: Neural translation model with attention (adapted from [13])

$$p(y) = \prod_{t=1}^{T} p(y_t | y_1, \ldots, y_{t-1}, c)$$

where in a RNN, each time step is modelled as

$$p(y_{t+1} | y_1, \ldots, y_t, c) = g(y_t, s_t, c).$$

Here $g$ can be any nonlinear function that models the probability of $y_t$, e.g. in section 4.2.1 the softmax function was used.

Bahdanau et al. [13] changed the above architecture such that each time step $t$, a new context vector $c_t$ is fed to the decoder RNN to compute the hidden state for the next period $s_{t+1} = f(s_t, y_t, c_t)$ (see Figure 7). The time dependent context vector is calculated in the following way [1] j

$$c_t = \sum_{i=1}^{T} \alpha_{t,i} h_t$$

[1]Note that Bahdanau et al. [13] use a bi-directional RNN as an encoder, a main motivation being to construct the hidden states $h_t$ to be a representation of previous and following words. This ensures that each hidden state captures its local environment better and thus aids the attention mechanism in finding relevant local states. For the image captioning attention mechanism this detail does not matter as the encoder RNN is replaced by a CNN, so we present the encoder as a simple RNN to make the presentation and notation more concise.

where the weight $\alpha_{t,i}$ for each hidden state $h_t$ is calculated as

$$\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_{k=1}^{T} \exp(e_{k,i})}$$

where

$$e_{t,i} = a(s_{t-1}, h_i).$$

Here $e_{t,i}$ is the output of an attention model $a(s_{t-1}, h_i)$, a FFNN, that evaluates how important the hidden state from the encoder at time step $i$, $h_i$, is for the output of the decoder at time step $t$, $y_t$ .

## 5.2 Attention in image captioning models

The soft attention mechanism in Xu et al. [84] is based on the attention architecture from [13], where the encoder RNN is replaced by a CNN. The attention mechanisms lets the model focus on certain parts of an input image, that it has learned to be important for generating the next word in a caption describing the image.

Xu et al. [84] extracted the activations from a lower convolutional layer, instead of fully connected layers ( Xu et al. [84], p.3), and were thus able to obtain a correspondence between the activation maps and portions of the image. This correspondence is due to the preservance of spatial locality of a CNN (see e.g. Chen et al. [19], p.4), i.e. the activations retrieved in a convolutional layer correspond to the same position in the input image, relatively speaking. The dimensions of a convolutional activation map after several convolutional and pooling layers will be smaller, but the proportions of the image are retained. Fully connected layers do not have this property, as they are flat and thus loose any spatial structure.

The encoder CNN produces $L$ annotations, $a = a_1, \ldots, a_L$, $a_l \in \mathbb{R}^D$, where $D$ corresponds to the number of filters in the respective convolutional layer that was chosen for extraction (each filter produces one activation map). $L$ is the dimension of the flattened activation map. For example, for VGG16 the convolutional extraction layer chosen for this thesis has dimensions $D = 512$ and $L = 196$. Each element in $a_l$ corresponds to an activation value from one of the $D$ filters, and it can thus be interpreted as a summary of features detected in that part of the image.

The attention model $f_{att}(a_l, h_{t-1})$ takes the previous hidden state $h_{t-1}$ and annotation $a_l$ as inputs, thus the attention weights are calculated at each time step based on the sequence the decoder RNN has already produced and stored in the hidden state. Also note that the same annotation vector $a_l$ is used at each time step.

Several projections happen in the attention model to compute a score $e_{t,l}$ for image region $l$ at time point $t$, $e_{t,l} = f_{att}(a_l, h_{t-1})$. Firstly, the hidden state $h_{t-1}$ gets projected to the annotation space $\mathbb{R}^D$ by $h^p_{t-1} = W_{hp}h_{t-1}$, where $W_{hp} \in \mathbb{R}^{D\times H}$, where $H$ is the dimension of the hidden state in the LSTM. Secondly, the same projection is applied to the annotations $a_l$, such that $a^p_l = W_p a_l + b_{ap}$, where $W_p \in \mathbb{R}^{D\times D}$ and $b_{ap} \in \mathbb{R}^D$.

Both $a^p_l$ and $h^p_{t-1}$ get aggregated in a combined representation

$$q_{t,l} = \tanh(a^p_l + h^p_{t-1}),$$

and the attention model $f_{att}$ then learns a linear combination of $q_{t,l}$ to compute a score $e_{t,l}$ for image region $l$ at time point $t$

$$e_{t,l} = W_{att}q_{t,l} + b_{att},$$

where $W_{att} \in \mathbb{R}^D$ and $b \in \mathbb{R}$. The attention mechanism can be summarized as

$$e_{t,l} = W_{att}\Big\{ \tanh\Big[ (W_{ap}a_l + b_{ap}) + (W_{hp}h_{t-1}) \Big]\Big\} + b_{att},$$

where $W_{ap}, W_{hp}, W_{att}, b_{ap}$, and $b_{att}$ constitute the trainable weights.

The scores $e_{t,l}$, $l = 1, \ldots, L$, then get normalized by a softmax function to produce the attention weights $\alpha_{t,l}$

$$\alpha_{t,l} = \frac{\exp(e_{t,l})}{\sum_{l'=1}^{L} \exp(e_{t,l'})}.$$

As was pioneered by Bahdanau et al. [13], the context vector $z_t$, that serves as the image representation input to the LSTM (see Section 4.3), is computed as a weighted sum of the annotations $a = a_1, \ldots, a_L$ and alpha weights $\alpha_t = \alpha_{t,1}, \ldots, \alpha_{t,L}$

$$z_t = \sum_{l=1}^{L} \alpha_{t,l} a_l.$$

The $\alpha$ weights from the soft attention mechanism (SA) are visualized as a grey heatmap, where light/white areas depict high weights, and dark areas low weights. The attention weights $\alpha_{t,l}$, that correspond to pixels $l = 1, \ldots, L$ in the CNN extraction layer which preserved spatial locality, get reshaped to a $14 \times 14$ two-dimensional representation, and are then upsampled by a factor of 16, to match the $224 \times 224$ input dimension of the images. This is the same visualization approach as in Xu et al. [84].

# 6 Deep Taylor Decomposition

Montavon et al. [62] pioneered a decomposition framework to explain nonlinear predictions of deep neural networks. We will first introduce Taylor decompositions, then describe its layer-wise application in the context of neural networks, and then focus on relevance redistribution rules, which depend on the input domain.

## 6.1 Taylor decomposition

Consider an arbitrary differentiable function $f : \mathbb{R}^P \longrightarrow \mathbb{R}$, the first-order Taylor expansion around the point $\tilde{x} \in \mathbb{R}^P$ is given by

$$\begin{aligned} f(x) &= f(\tilde{x}) + \left( \frac{\partial f}{\partial x}|_{x=\tilde{x}} \right)^T (x - \tilde{x}) + \epsilon \\ &= f(\tilde{x}) + \sum_{p=1}^{P} \frac{\partial f}{\partial x_p}|_{x_p=\tilde{x}_p} (x_p - \tilde{x}_p) + \epsilon, \end{aligned}$$

which equates to a linear approximation of $f(x)$.

In a neural network classification context, a sensible point at which to perform the Taylor expansion is a root point, i.e. $f(\tilde{x}) = 0$ ([62], p.213). A root point represents a point with zero neuron activation, which e.g. can be interpreted as no object detection in a final output layer with rectified linear unit (ReLU) nonlinearity. Assuming that $x_p \geq 0$, e.g. if the previous layer

had a ReLU nonlinearity, the partial derivative $\frac{\partial f}{\partial x_p}|_{x_p=\tilde{x}_p}$ picks up the positive impact of input $x_p$ on the functional value $f(x)$, which is the desired decomposition. Expanding around a root point and ignoring the Taylor residual $\epsilon$, the first-order Taylor expansion simplifies to

$$f(x) = \sum_{p=1}^{P} \frac{\partial f}{\partial x_p}|_{x_p=\tilde{x}_p}(x_p - \tilde{x}_p).$$

In order for the first-order approximation to be precise, the root point $\tilde{x}$ must lie in the vicinity of $x$. Strategies to find permissable root points in the neural network context will be discussed in Section 6.3.

## 6.2 Layer-wise Taylor decomposition

A deep neural network is composed of multiple layers of neurons, where each neuron is the output of a linear transformation of neurons from the previous layer, followed generally by a nonlinearity function, and in the context of CNNs, a pooling layer. A neural network is thus a structure of small subfunctions at each neuron, and through the interconnection of many neurons, deep neural networks achieve their high representational power (Montavon et al. [62], p.213).

DTD exploits this structure of small subfunctions by calculating the relevance for each neuron separately, in a top-down order (Montavon et al. [62], p.214). Consider a neuron $x_j$, whose relevance $R_j$ we want to decompose on the set of lower layer neurons $\{x_i\}$. The function $R_j(x_i)$ captures the functional relationship between $R_j$ and $\{x_i\}$, and can be decomposed by Taylor decomposition (see Section 6.1) as

$$R_j(\{x_i\}) = \sum_i \underbrace{\frac{\partial R_j(\{x_i\})}{\partial x_i}|_{x_i=\tilde{x_i}^j}(x_i - \tilde{x_i}^j)}_{R_{ij}},$$

where $R_{ij}$ is the redistributed relevance from neuron $x_j$ to $x_i$. The total relevance assigned to neuron $x_i$ comprises all redistributed relevances $R_{ij}$ to which $x_i$ contributed

$$R_i = \sum_j R_{ij}.$$

This leads to a relevance redistribution function, that expresses the lower layer relevance $R_i$ in terms of decomposed higher layer relevances $R_j$

$$R_i = \sum_j \frac{\partial R_j(\{x_i\})}{\partial x_i}|_{x_i=\tilde{x_i}^j}(x_i - \tilde{x_i}^j).$$

Montavon et al. [62] show that layer-wise relevance conservation holds and that the entire decomposition from output to input is consistent, i.e. (p.214, [62])

$$\sum_f R_f = \ldots = \sum_j R_j = \sum_i R_i = \ldots = \sum_p R_p,$$

and

$$\{R_f\}, \ldots, \{R_j\}, \{R_i\}, \ldots, \{R_p\} \geq 0,$$

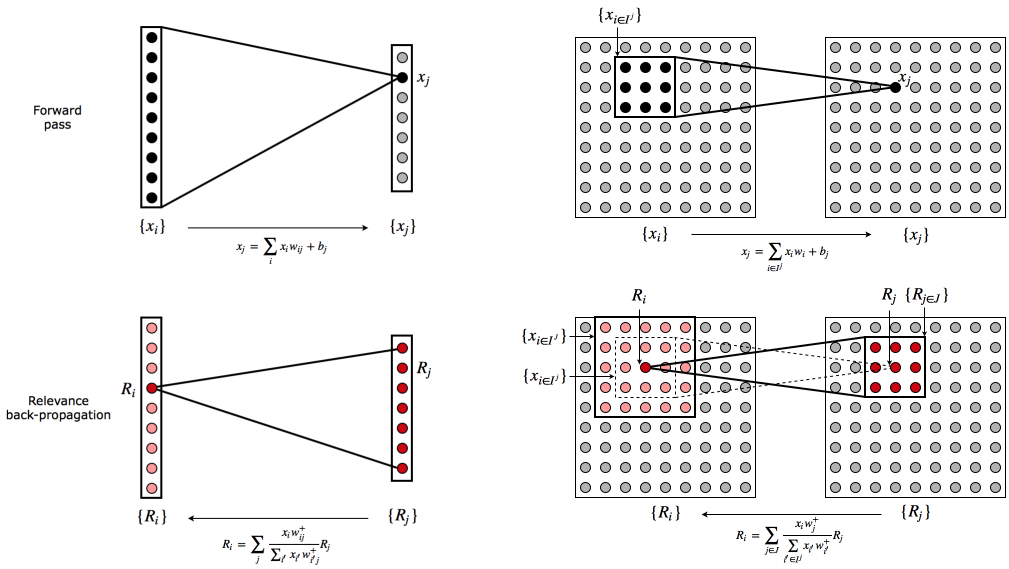
where $\{R_f\}$ and $\{R_p\}$ are the relevances assigned to the output and input nodes, respectively. If for example in the output layer of a CNN the activation value of a node representing the class "cat" is chosen as $R_f$, then the amount of relevance that is backpropagated through the CNN is conserved, such that the total relevance assigned to all image pixel, $\{R_p\}$, equals the activation value of the "cat" node.

## 6.3 Relevance redistribution rules

A sensible expansion point $\tilde{x}$ for a Taylor expansion in a neural network classification context is a root point that is close to $x$, the point that is being approximated. Montavon et al. [62] present three methods, the $w^2$-rule, $z^+$-rule, and $z^{\mathcal{B}}$-rule, for redistributing relevances, that each depend on the input domain and method to find a root point (Montavon et al. [62], p.216). In this thesis the $z^+$-rule and $z^{\mathcal{B}}$-rule are applied, so this section focuses on presenting their concepts and algorithmic implementation for both fully connected and convolutional layers. A further propagation rule that will be used is the $\alpha\beta$-rule introduced by Binder et al. [17], of which the $z^+$-rule is a special case.

### 6.3.1 $z^+$-rule

For a constrained input space, $x \in \mathcal{X} \subset \mathbb{R}^D_+$, the relevance propagation rule, $z^+$-rule, for a fully connected layer is




(a) Fully connected layer (b) Convolutional layer

Figure 8: Relevance propagation overview

$$R_i = \sum_j \frac{z_{ij}^+}{\sum_{i'} z_{i'j}^+} R_j,$$

where $z_{ij}^+ = x_i w_{ij}^+$ and $w_{ij}^+ = \max(0, w_{ij})$. Given that $\{x_i\}, \{w_{ij}^+\} \geq 0$, the redistribution of $R_j$ to $R_i$ is proportional to $x_i$'s positive impact on $x_j$, relative to the positive impact of all $x_i$'s on $x_j$. The total amount of relevance assigned to $x_i$ is then the sum of relative positive impacts to all nodes $\{x_j\}$ that $x_i$ contributed to.

In a convolutional layer the output $x_j$ is computed from a subset of input nodes $\{x_{i \in I}\}$, and not the entire set $\{x_i\}$ as in a fully connected layer. Further, the weights $w$ are shared in a convolutional layer (see Section 3.2.1) unlike in a fully connected layer. The $z^+$-rule in a convolutional layer is thus

$$R_i = \sum_{j \in J} \frac{x_i w_j^+}{\sum\limits_{i' \in I^j} x_{i'} w_{i'}^+} R_j, \quad w_j^+ = \max(0, w_j).$$

The index $j$ runs over all indices in index set $J$, which includes all output nodes $x_j$ to which $x_i$ contributed (see nodes in dark red in Figure 8(b)). The corresponding set of input nodes, that produced $\{x_j \in J\}$, is denoted as $\{x_i \in I^J\}$ (see nodes in light red in Figure 8(b)). Each receptive field is denoted as an index set $\{x_{i \in I^j}\}$, corresponding to the output $x_j$.

### 6.3.2 $z^{\mathcal{B}}$-rule

For a box-constrained input domain $\mathcal{B} = \{x_i : \forall_{i=1}^{D} l_i \leq x_i \leq h_i\}$, where $l_i \leq 0$ and $h_i \geq 0$ are the lowest and highest input values for each dimension $i = 1, \ldots, D$, the redistribution rule, $z^{\mathcal{B}}$-rule, for a fully connected layer becomes

$$R_i = \sum_j \frac{z_{ij} - l_i w_{ij}^+ - h_i w_{ij}^-}{\sum_{i'} z_{i'j} - l_i w_{i'j}^+ - h_i w_{i'j}^-} R_j,$$

where $z_{ij} = x_i w_{ij}$ and $w_{ij}^- = \min(0, w_{ij})$. The data independent terms $l_i w_{ij}^+$ and $h_i w_{ij}^-$ enforce positivity ([62], p.216) and we end up again with a proportional redistribution scheme as in the $z^+$-rule.

The $z^{\mathcal{B}}$-rule for convolutional layers follows the same principle as described for the $z^+$-rule. For an efficient computational implementation both redistribution rules were reformulated as a sequence of matrix operations, as described in [8], and implemented in TensorFlow for this thesis (available under [55]).

### 6.3.3 $\alpha\beta$-rule

The $z^+$-rule redistributes relevances to lower layers proportionally to the positive weight associated to a neuron, while ignoring the effect a neuron with a negative weight has on higher layer neurons. The $\alpha\beta$-rule, introduced by Binder et al. [17], also redistributes on negative weights, by controlling the magnitude of positive redistribution with an $\alpha$ parameter, and the magnitude of counter relevance with a $\beta$ parameter

$$R_i = \sum_j \Big( \alpha \frac{x_i w_{ij}^+}{\sum_{i'} x_{i'} w_{i'j}^+} - \beta \frac{x_i w_{ij}^-}{\sum_{i'} x_{i'} w_{i'j}^-} \Big) R_j.$$

Note that with $\alpha = 1$ and $\beta = 0$ the $\alpha\beta$-rule is equal to the $z^+$-rule. The $\alpha\beta$-rule does not comply with the DTD-framework, as its heatmaps can contain negative relevances. The rule has however shown to produce sparser

heatmaps and the negative relevances can be interpreted as areas that introduced uncertainty to the classification outcome [8].

## 6.4 Training-free relevance model

The layer-wise Taylor decomposition, with its top-down dependency structure, assumes that $R_j$ can be exclusively decomposed on the set of lower layer neurons $\{x_i\}$ ([62], p.214). Montavon et al. [62] present two relevance models that fulfill this dependency structure, one of which is the training-free relevance model, which will be used in this thesis ([62], p.217). A relevance model maps neuron activations to the relevance of a neuron in a higher layer, while its output can be decomposed onto its input variables ([62], p.216).

Consider a fully connected layer with ReLU nonlinearity ([62], p.216, and Montavon et al. [63], p.6)

$$x_j = \max\Big(0, \sum_i x_i w_{ij} + b_j\Big).$$

We assume for now that the relevance $R_j$ can be written as a product of the neuron activation $x_j$ and a positive constant $c_j$ as $R_j = x_j c_j$. Then the relevance neuron $R_j = \max(0, \sum_i x_i w'_{ij} + b'_j)$, with $w'_{ij} = w_{ij} c_j$ and $b'_j = b_j c_j$, can be decomposed to lower layer neurons $\{x_i\}$ by Taylor decomposition

$$R_j = \sum_i \frac{\partial R_j}{\partial x_i}(x_i - \tilde{x}_i^j).$$

The $z^+$-rule then applies to propagate the relevances from $R_j$ to $R_i$

$$R_i = \sum_j \frac{x_i w_{ij}^+}{\sum_{i'} x_{i'} w_{i'j}^+} R_j.$$

This procedure can be repeated one layer further down with neurons $\{x_k\}$. The relevance for neuron $x_k$ can be written as

$$\begin{aligned} R_k &= \sum_i \frac{x_k w_{ki}^+}{\sum_{k'} x_{k'} w_{k'i}^+} R_i \\ &= x_k \underbrace{\sum_i \frac{w_{ki}^+}{\sum_{k'} x_{k'} w_{k'i}^+} R_i}_{=c_k}, \end{aligned}$$

where $c_k$ is approximately constant and positive ([63], p.7). Note that treating $c_k$ as constant is important as it allows to represent the relevance $R_k$ as a product of activation value $x_k$ and a constant, which then allows to decompose $R_k$ again by Taylor decomposition to propagate relevances to the next lower layer.

For a deep neural network with ReLU nonlinearities, as is the case for VGG16, DTD is thus a repeated application of the $z^+$-rule up to the first layer ([62], p.218). For the first layer, with a bounded input domain in the context of image recognition CNNs, the $z^{\mathcal{B}}$-rule is applied to produce relevances $\{R_p\}$ at the input pixel level.

In CNNs each convolutional layer, or block of convolutional layers as e.g. in VGG16, is generally followed by a pooling layer. Montavon et al. [62] propose for pooling layers to redistribute relevances proportional to neuron activation in the pool to ensure explanation continuity ([63], p.7). Let the $L_q$-norm represent a pooling operation such that $x_k = \|\{x_j\}\|_q$, then the relevance is redistributed as

$$R_j = \frac{x_j}{\sum_{j'} x_{j'}} R_k,$$

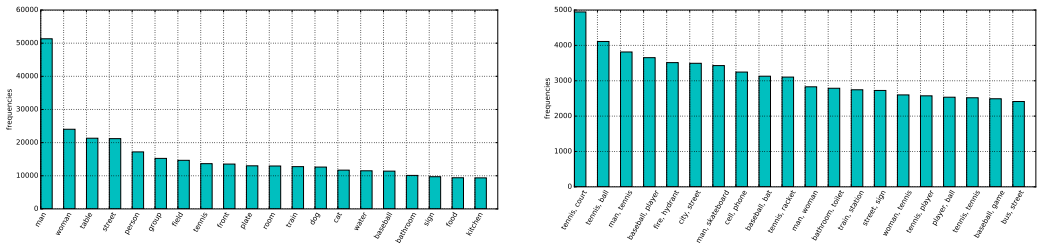
where the index $j^{'}$ runs over all nodes in the pool $\{x_j\}$.

# 7 Training an image captioning system

## 7.1 Software and hardware infrastructure

The architecture for the image captioning system trained for this thesis was adopted from Xu et al. [84], who also provided a Theano implementation of the model in [83], which proved very helpful for understanding details of the system, that do not get mentioned in the paper. The system consists of three main building blocks, the CNN, the attention model, and the decoder LSTM. For this thesis the model, training, and evaluation code was written in TensorFlow [11], using the Python application programming interface, and is available on GitHub under [55].

The system was trained using a Tesla K80 GPU on Google Cloud's ML Engine. The large working memory of 12 GB of the Tesla K80 allowed to increase the batch size to 128, which significantly improved training speed. Training the system for five epochs on the entire training set (414 113 samples) took 20




(a) (b)

Figure 9: Noun and noun tuple frequencies in train set

hours.

## 7.2 Dataset

The system was trained on the MSCOCO dataset [56]. The training set contains 82 783 images, with a total of 414 113 captions (82 586 images have five captions, 196 images have six captions). All captions were collected using human workers on Amazon's Mechanical Turk (AMT) (Chen et al. [21] p.2) and have a mean caption length of 10.5 words. In order to ensure the visual complexity of the images, they were collected by searching for pairs of 80 object categories on the web photo community Flickr.

Figure 9(a) gives an overview of the most frequent nouns in the training set, where person related nouns like man, woman, person, and group are among the most frequent. Other frequent nouns are related to sports, animals, and food. Figure 9(b) gives an indication of frequent noun tuples occuring in captions. Sports related captions have very frequent common tuples, as well as street scenes, and bathrooms. For each of those image categories we would expect to generate fewer novel and creative captions, as the training captions show lower variability.

## 7.3 Transfer learning

For the CNN we used the VGG16 architecture, for which ImageNet pre-trained weights are available for download from the authors [73]. Throughout training of the NIC, the weights in the CNN were not fine tuned, as the pre-trained weights are optimized for detecting a wide spectrum of objects, as required by

the NIC. The training time was also minimized by not optimizing the CNN weights, which was an important consideration given the size of the system (same approach as in Xu et al. [84], p.6).

For word embeddings several training attempts were done with pre-trained word embeddings from Pennington et al. [66], but no performance gain was found compared to training the embeddings from random initializations, so the latter approach was chosen. Unlike the VGG16 weights, the word embeddings had to be learned throughout training to achieve a well performing model. For the final system an embedding dimension of 512 was chosen, i.e. each word in the dictionary is represented by a 512-dimensional vector, whose representation gets learned throughout training.

## 7.4 Objective function

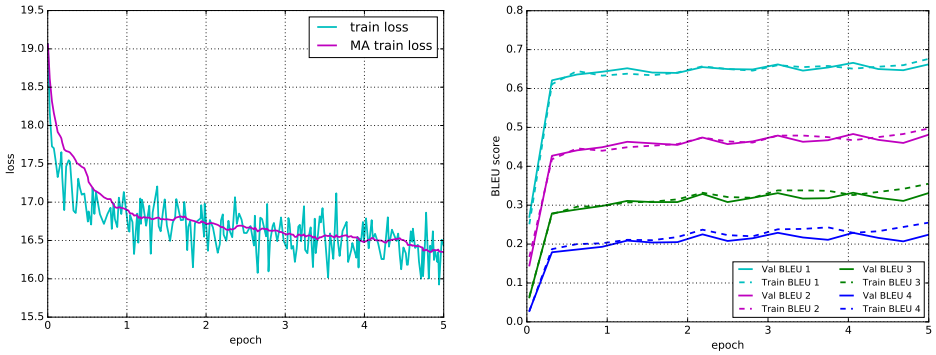
The whole system was trained end-to-end by minimizing a cross entropy loss function (Section 4.2.1) with the stochastic gradient descent algorithm Adam [48]. A regularization term was added to the loss function as proposed by Xu et al. [84], which encourages the model to explore the entire image while generating the caption. At each time step $t$ it holds that $\sum_i \alpha_{t,i} = 1$, as $\alpha_{t,:}$ is the output of a softmax function. To promote exploration of the entire image over time, we thus require that $\forall i : \sum_t \alpha_{t,i} = 1$, which is ensured by adding the squared deviation to the loss function. The final loss function that is minimized is

$$L(x) = -\sum_t \log[Pr(x_{t+1}|y_t)] + \sum_i (1 - \sum_t \alpha_{t,i})^2.$$

Figure 10(a) depicts the loss calculated on each batch of 128 images throughout training. The rate of loss decrease slows down significantly after one epoch, but continues over the entire training horizon. It is likely that the loss would have further improved when training would have continued beyond five epochs, however at the risk of overfitting and further computation time and costs.

## 7.5 Regularization

A major problem in training an image captioning system is overfitting, where the system learns to match images with the corresponding captions in the training set, rather than learning the structure of images and corresponding




(a) (b)

Figure 10: Loss and BLEU metrics

sentences, such that novel captions can be generated for new images (see e.g. Soh [76], p.5). The system trained for this thesis was regularized through dropout and early stopping (as in Xu et al. [84], p.6). Dropout was applied to the LSTM cell as described in Zaremba et al. [88] and several dropout probabilites were tested. Inspired by the experiments and Soh [76], a dropout probability of 75% was chosen.

For early stopping, the BLEU scores on a train and test set were monitored throughout training, depicted in Figure 10(b). Diverging BLEU scores between the train and test set would indicate overfitting, as the caption's quality only improves on the training set, and the system cannot generalize to previously unseen images in the test set. The BLEU scores did not diverge over the five epoch training horizon, and continued to improve slowly. Thus there was no reason to stop training before the end of five epochs (similar to findings in Soh [76], p.5).

## 7.6 Performance metrics

For this thesis the standard metrics BLEU [65], METEOR [29], and CIDEr [80] were calculated using an evaluation package provided by Lin et al. [56] under [20]. BLEU measures the similarity between two sentences by computing the geometric mean of n-gram precisions and a penalty term for short sentences (Kilickaya et al. [47], p.200). METEOR represents the harmonic mean of unigram precision and recall between two sentences, and also uses a synonym matcher to identify semantic similarities when words do not match

Table 2: BLEU, METEOR, and CIDEr metrics

| Metric | Thesis NIC | Xu et al. [84] |
|---|---|---|
| BLEU 1 | 66.7% | 70.7% |
| BLEU 2 | 48.4% | 49.2% |
| BLEU 3 | 34.1% | 34.4% |
| BLEU 4 | 23.9% | 24.3% |
| METEOR | 22.2% | 23.9% |
| CIDEr | 74.9% | - |

Results for Xu et al. [84] are from the soft-attention model

exactly between sentences. Both BLEU and METEOR were designed for machine-translation assessments, whereas CIDEr is a recent metric especially developed for evaluating image captions. CIDEr calculates co-occurences of n-grams, where frequent n-grams are down weighted.

Quantifying the quality of generated captions with metrics is a challenging problem, as human judgement on the quality of a caption is a complex and subjective process (see e.g. Kilickaya et al. [47] or Chen et al. [21]). There are many ways to describe the same scene, so an automatic evaluation will benefit if the set of validation captions for each image is large, such that the generated caption can be compared to many alternative descriptions. BLEU for example performs poorly when comparing sentences individually (Chen et al. [21], p.3). For the MSCOCO validation set there are five captions for each image to compare the generated captions against, therefore some comfort can be taken towards the quality of the metrics.

Chen et al. [21] collected an additional human caption for each image on a MSCOCO test set and treated this caption as the prediction. The "human predictions" achieved lower BLEU scores than the image captioning system from Xu et al. [84], and the METEOR score of 25.2% is slightly higher than the system generated captions. This indicates that human judgement and performance metrics correlate only to a certain extent.


Figure 11: Caption evolution

## 7.7 Results

### 7.7.1 Generated captions

The quality of generated captions in the model trained for this thesis (Thesis NIC) is close to the state-of-the-art performance by Xu et al. [84], with a BLEU 4 score of 23.9% and METEOR score of 22.2%, compared to 24.3% and 23.9% respectively in Xu et al. [84] (see Table 2). The reported results for Thesis NIC in Table 2 are based on the entire validation set, which contains 40 501 images. The mean length of generated captions is 9.8 words, slightly shorter than the captions in the train set with a mean length of 10.5 words.

For a human assessment of caption quality Figure 11 shows the evolution of captions throughout training. The captions are generally accurate in describing the images' content, and their quality improves as training continues. The captions achieve a good quality after one epoch already, in line with the high BLEU scores after one epoch depicted in Figure 10(b). The improvements in subsequent epochs are marginal, e.g. from "a large white airplane is parked

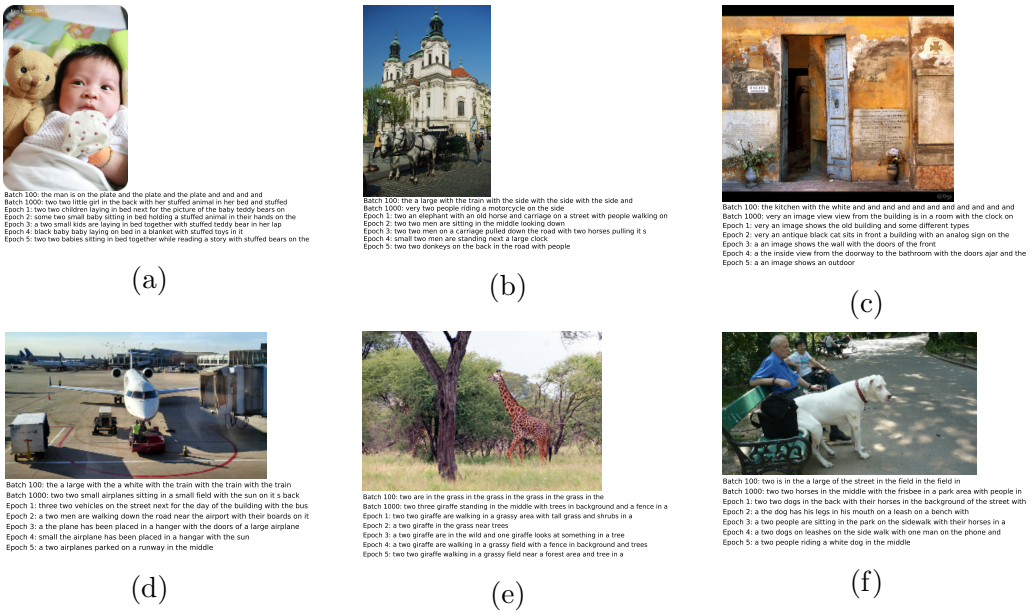



Figure 12: Caption evolution (second most probable word)

on the street" after one epoch, to "a white airplane sitting on top of a tarmac" after five epochs.

To further analyse the model's generative process and quality, Figure 12 lists captions generated from the second most probable word at each time step. It can be seen that the quality is significantly worse than when picking the most probable word at each time step, evidenced also by a low BLEU 4 score of 2.9%. Interestingly, the main objects in each image are recognized most of the time, which indicates that dominant objects, e.g. giraffe or baby, remain likely candidates with high probabilities over multiple time steps. Also the sentence syntax is accurate in most cases, indicating that a word's syntactic fit has high importance in the model.

### 7.7.2 Attention maps

The attention maps generated by the Thesis NIC are depicted in Figure 13. The gray heatmaps show the attention weights that were used to generate the word at each time step. The probability for each generated word is taken from the output of the LSTM, which forms a probability distribution over the dictionary of words used in the system. It is thus a representation of the classification confidence for that word.

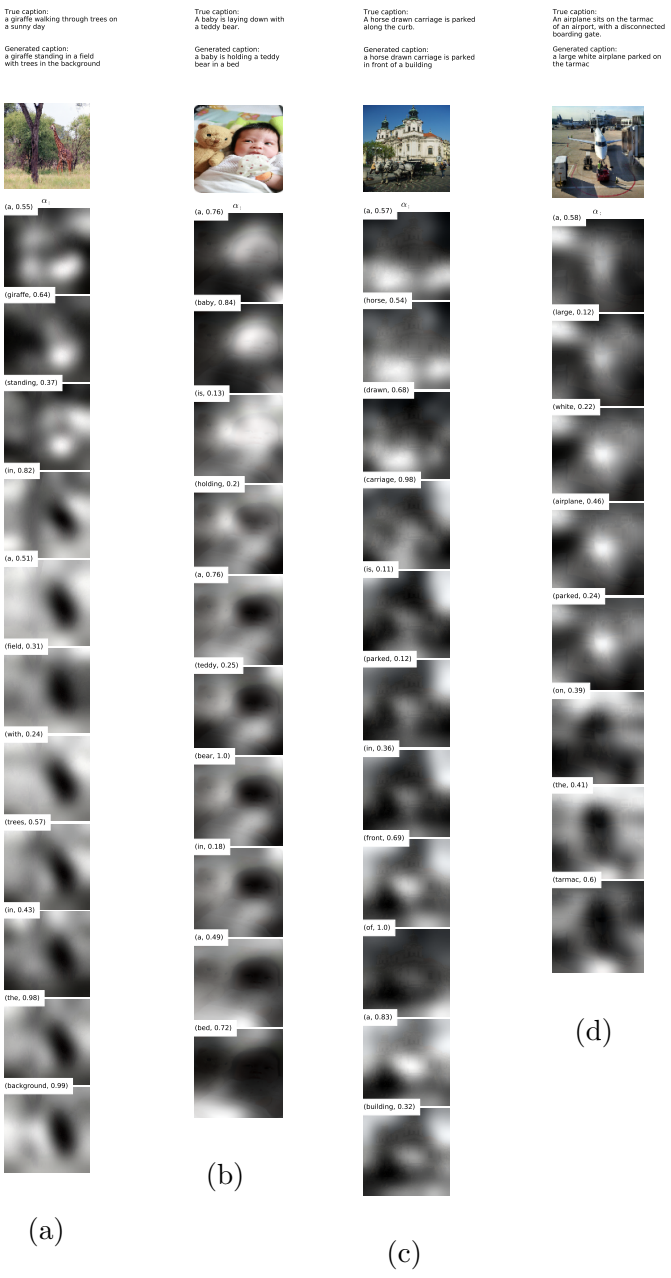



Figure 13: Attention maps generated by Thesis NIC

The foci of the attention model correspond well for most example images with the object that is being described in the caption at each time step. For

example, the baby's face in Figure 13(b) is in focus over the first two words of the caption ("A baby"), while the focus shifts to the teddy bear in the middle of the sentence, and then to the background, in line with content of the generated caption. Some further well performing attention maps examples are depicted in Figure 16.

Generally the quality of attention weight transitions and precisions correlate with the quality of the caption. Examples with inaccurate captions tend to have attention foci that are not informative and diffuse, as can be seen in Figure 19 in the Appendix. For examples where the quality of caption diverges from the quality of attention weights, i.e. a high quality generated caption but nonsensical attention weights, the system seems to have learned a mapping to a similar image in the training set and thus reproduces a training caption (overfitting), rather than focusing on different objects while generating the caption.

# 8 Relevance propagation in an image captioning system

The goal is to associate each pixel in the input image with a relevance score, which measures its impact on the output of the attention model $f_{att}$. We will denote the relevance scores for the input layer of the CNN as $R^x$, which has the same $224 \times 224 \times 3$ dimensions as the input image $x$. Similarly, $R^a$ denotes the relevances at the convolutional extraction layer with activation values $a_{l,d}$, i.e. $R^a \in \mathbb{R}^{196 \times 512}$ for $conv5\_3$ in VGG16. Relevances at the intermediate layers of the attention model are denoted as $R^{a^p}$, $R^q$, and $R^e$, corresponding to layers whose nodes have activation values $a^p$, $q$, and $e$ respectively. A relevance score is derived for each node in the neural network, so we will use the same subscripts for relevances, e.g. $R^a_{l,d}$ refers to the relevance assigned to node $l$ in channel $d$ in the convolutional extraction layer. The attention weights get calculated at each time step $t$, as they depend on the hidden state $h_{t-1}$, and therefore the relevance gets backpropagated at each time step. For notational simplicity the time index is dropped.

At first, the images get propagated through the CNN to produce the annotations $a = a_{1,:} \ldots, a_{L,:}$, with $a_{l,:} \in \mathbb{R}^D$. These annotations can be understood as a pixel contextualization, where each pixel $a_{l,:}$ is represented by a $D$-dimensional vector that spans all channels. At each time step $t$, the annotations get projected to $a^p$, and are then combined with the projected hidden

state $h^p$, to form the combined context $q_{l,:}$ (see forward pass in Figure 14). Then the attention weights $W_{att}$ project $q_{l,:}$ to the attention logits $e_l$, which get normalized to produce the attention weights $\alpha_l, l = 1, \ldots, L$

$$e_l = \underbrace{\tanh\left[(\underbrace{a_{l,:}W_{att} + b_{att}}_{a^p_{l,:}}) + (\underbrace{hW_h}_{h^p})\right] W_{att} + b_{att}}_{q_{l,:}}$$

$$\alpha_l = \frac{\exp(e_l)}{\sum_{l'=1}^{L} \exp(e_{l'})}.$$

First the relevance propagation in the CNN is presented, and then three approaches, DTD-A, DTD-B, and DTD-C, will be explored that differ in how the relevance is propagated from attention model to the CNN. In the DTD-A approach the attention model is by-passed and relevance propagation starts at the convolutional extraction layer. In the DTD-B approach the relevance is propagated through the attention model, which allows to distribute relevances selectively to channels of the convolutional extraction layer. The DTD-C approach equals DTD-B, but redistributes relevances with the $\alpha\beta$-rule instead of the $z^+$-rule.

## 8.1 CNN encoder

For the first convolutional layer in the VGG16 CNN the $z^{\mathcal{B}}$-rule is applied. The lower and upper bound in the $z^{\mathcal{B}}$-rule are set equal to the minimum and maximum pixel value in the image. For all other convolutional layers the $z^+$-rule is applied in the DTD-A and DTD-B approach, and the $\alpha\beta$-rule for the DTD-C approach. In each convolutional layer of VGG16 the convolution operation is followed by a ReLU activation function, which allows to apply the $z^+$-rule repeatedly as per the DTD framework.

Positive bias terms, that are not allowed in the DTD framework, were added to the denominator of both the $z^{\mathcal{B}}$-rule, $z^+$-rule, and $\alpha\beta$-rule, so that relevance can be redistributed on them (compare with [62], p.219).

For the max-pooling layers the relevance is redistributed proportionally to the neuron activations in the pool, as proposed in Montavon et al. [62] and described in Section 6.3, which promotes explanation continuity ([63], p.7).

Attributing the relevance solely to the neuron with the maximum activation value in the pool, as well as redistributing equally to all nodes in the pool, yielded inferior heatmaps to proportional redistribution. Please note that in the CNN forward pass max pooling was applied, as in the original VGG16 architecture.

## 8.2 Attention mechanism

The DTD framework requires that each node's relevance can be written as the product of the node's activation value $x_i$ and a constant $c_i$. For the output layer, where the relevance backpropagation begins, this constant needs to be pre-defined. For example, in image classification neural networks the constant is one for the node belonging to the class whose evidence we wish to backpropagate, and zero for all other nodes in the output layer. Defining $c$ as a one-hot vector, results in only backpropagating the evidence from the selected node, while ignoring all other nodes.

In all three approaches, DTD-A, DTD-B, and DTD-C, $c$ is set equal to the attention weights $\alpha$

$$c_l = \alpha_l, \quad l = 1, \ldots, L.$$

This means that we are backpropagating relevances from all nodes in the output layer, contrary to a one-hot vector, but put a higher emphasis on those nodes for which the attention mechanism generated a high $\alpha$ weight. We are thus backpropagating the evidence of nodes in the focus of the attention map, and the resultant heatmaps provide a visualization of the attention foci. Furthermore, the $\alpha$-weights are a natural choice as scaling constants, as they represent a probability distribution with $\forall l : \alpha_l \geq 0$ and $\sum_l \alpha_l = 1$.

### 8.2.1 DTD-A

In the DTD-A approach, the constant $c$ gets applied to the activation values $a$ of the convolutional extraction layer

$$R^a_{:,d} = a_{:,d} * c = a_{:,d} * \alpha.$$

Note that the same set of constants, $c = c_1, \ldots, c_L$, gets applied to each channel $d = 1, \ldots, D$. While the attention focus at each time step will pick up activations in each channel that lie in the focus, this approach does not use

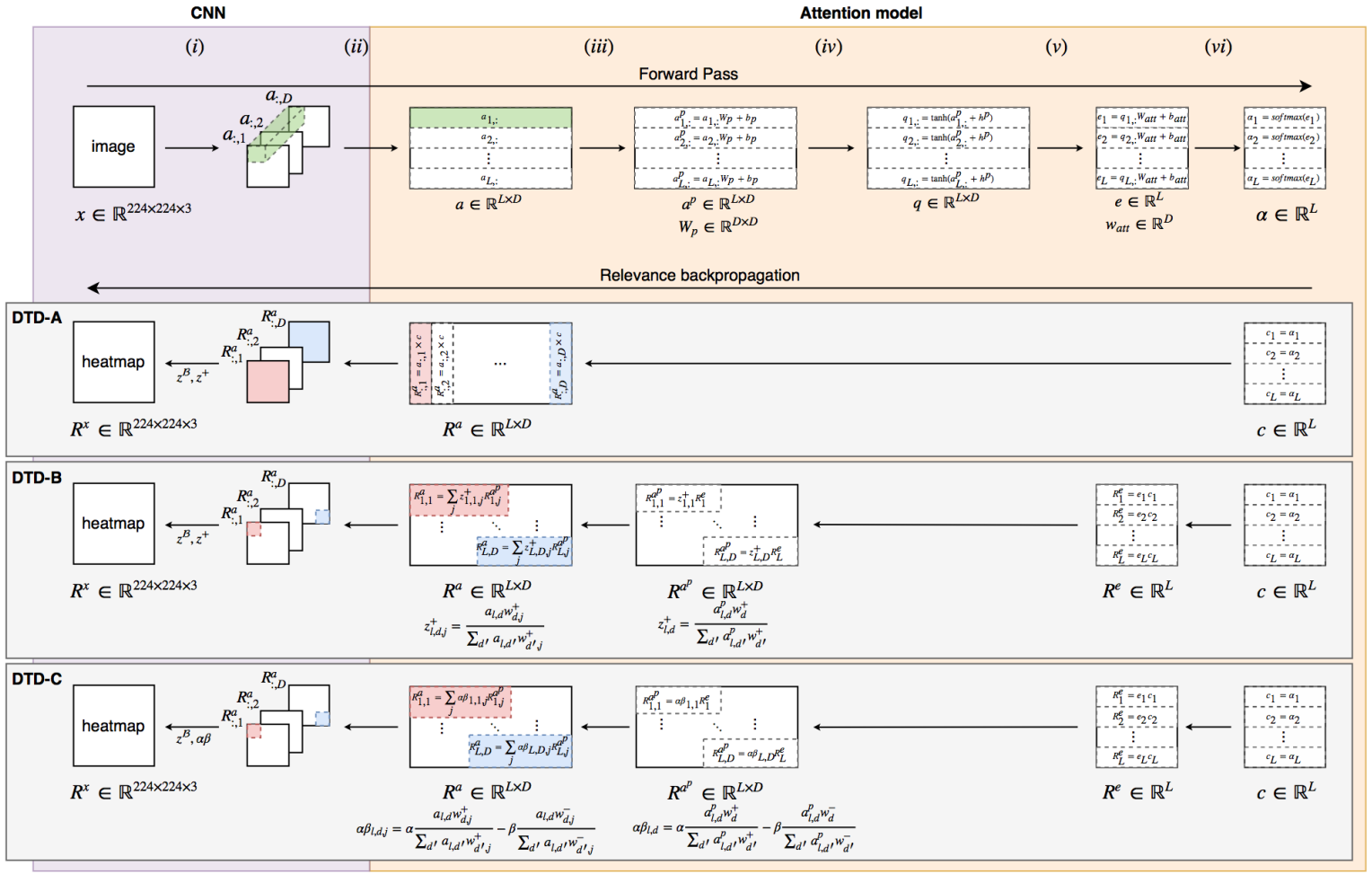



Figure 14: Relevance propagation in NIC

The diagram depicts steps $(i) - (vi)$ of the forward pass in the NIC and the corresponding relevance backpropagation steps for the DTD-A, DTD-B, and DTD-C approach. The CNN is VGG16, for which $L = 196, D = 512$ in the convolutional layer conv5_3, which is chosen as the extraction layer.

Forward Pass: (i) image $x$ gets propagated through CNN and annotations $a$ get extracted from convolutional extraction layer; (ii) $a$ is reshaped; (iii) $a$ is mapped to $a^p$; (iv) $a^p$ and $h^p$ are combined to $q$; (v) each $q_{l,:}$ is mapped by attention weights $W_{att}$ and bias $b_{att}$ to attention logit $e_l$; (vi) $e$ is normalized by softmax function to $\alpha$.

DTD-A: (vi)-(iii) $c$ is set equal to $\alpha$ and multiplied with each annotation channel $a_{:,d}$ to produce $R^a_{:,d}$; (ii) $R^a$ gets reshaped; (i) relevances get backpropagated to input layer by repeatedly applying $z^+$-rule for convolutional layers, activation proportional redistribution for pooling layers, and $z^{\mathcal{B}}$-rule for first convolutional layer.

DTD-B: (vi) $c$ is set equal to $\alpha$ and multiplied with attention logits $e$ to produce $R^e$; (v)-(iv) redistribute $R^e$ to $R^{a^p}$ by applying $z^+$-rule; (iii) redistribute $R^{a^p}$ to $R^a$ by applying $z^+$-rule; (ii)-(i) same as in DTD-A.

DTD-C: same as DTD-B, except for using $\alpha\beta$-rule instead of $z^+$-rule at each occurence.

any of the information from the attention weights $w_{att}$, which are applied to each channel in the attention mechanism to produce $\alpha$.

$R^a$ fulfills the requirements of a relevance neuron in the DTD framework, as $\alpha$ is constant, and $a$, $\alpha \geq 0$, and consequently $R^a \geq 0$. The relevance is then propagated back through the encoder CNN as described in Section 8.1.

### 8.2.2 DTD-B

Motivated by the shortcomings of the DTD-A approach, the goal for the DTD-B approach is to re-distribute $R^e$ as a function of the learned attention weights $w_{att}$, such that $R^a_{l,d}$ becomes a more accurate measure of how important the activation value $a_{l,d}$ was in producing the attention weight $\alpha_l$. The approach taken here is to backpropagate the relevance through the attention model, i.e. from $R^e$ down to $R^a$, using the $z^+$-rule repeatedly.

For each output neuron $e_l$ of the attention model, the total relevance to be redistributed is

$$R^e_l = \max(0, e_l) * c_l,$$

and we set $c_l = \alpha_l$, as in the DTD-A approach. The activations $e_l \in \mathbb{R}$ violate the positivity requirement in the DTD framework, and therefore all negative activations are set equal to zero.

The relevance $R^{a^p}$ is derived by redistributing $R^e$ with the $z^+$-rule. Again, $a^p \in \mathbb{R}^{L \times D}$ violates the positivity condition, so all negative values are set equal to zero to end up with the propagation rule

$$R^{a^p}_{l,d} = \frac{a^{p^+}_{l,d} w^+_d}{\sum_{d'} a^{p^+}_{l,d'} w^+_{d'}} R^e_l,$$

where

$$w^+ = \max(0, w_{att}),$$
$$a^{p^+} = \max(0, a^p).$$

The relevances get thus redistributed in proportion to the positive activation values of the projected annotation vectors, $a^{p^+}$, weighted by the positive

attention weights $w^+_{att}$.

In the next step, the relevances $R^{a^p}$ are redistributed to $R^a$ by applying the $z^+$-rule again

$$R^a_{l,d} = \sum_j \frac{a_{l,d} w^+_{d,j}}{\sum_{d'} a_{l,d'} w^+_{d',j}} R^{a^p}_{l,j}, \quad W^+ = \max(0, W_p).$$

The relevances $R^a$ are then propagated through the CNN by repeatedly applying the $z^+$-rule, as described in Section 8.1.

It is important to note that $R^e$ gets redistributed in proportion to the positive projected image annotations $a^{p^+}$, and not the combined context $q$, in order to ensure that no positive relevance is redistributed to nodes $a^p$ if $a^p \leq 0$. Given that the $z^+$-rule redistributes on positive inputs $a^{p^+}$ and positive weights $w^+$, the only combinations of $a^p$ and $h^p$ in the combined context $q$ that lead to non-zero relevance propagations are (i) $a^p \geq 0, h^p \geq 0$ and (ii) $a^p \geq 0, h^p < 0$. If in scenario (i) $h^p >> a^p$, then the relevance redistributed to $a^p$ is distorted, as the positive impact of the hidden state outweighed the image annotation, however the distortion's impact should be small, as $a^p$ is small and relevances are redistributed proportionally to its size. If in scenario (ii) a positive $a^p$ is counterbalanced by a negative $h^p$, then the hidden state diminishes the relevance to be redistributed. Despite these distortions, relevance backpropagation is limited to nodes $a^p \geq 0$. Section 9 discusses the disentangling of effects from hidden state and image annotations further.

### 8.2.3 DTD-C

The DTD-A and DTD-B approach both repeatedly apply the $z^+$-rule to backpropagate relevances. The $z^+$-rule redistributes relevances to lower layers proportionally to the positive weight associated to a neuron, while ignoring any effect a neuron with a negative weight had on higher layer neurons. Especially in the context of the attention weights $w_{att}$ it would be desirable to also capture the effect of negative weights.

The $\alpha\beta$-rule introduced by Binder et al. [17] also redistributes on negative weights and with that has shown to produce sparser heatmaps (see also [8]). Contrary to the DTD-framework, the $\alpha\beta$-heatmaps can contain negative relevances, which will be depicted in blue on the heatmaps.

The relevance backpropagation in DTD-C is the same as in DTD-B, only the $z^+$-rule is replaced by the $\alpha\beta$-rule at each occurrence in the CNN and attention model.

## 8.3 Mean relevances

In order to assess the difference between the relevances $R^a$ produced by the three approaches, the mean relevances $\bar{R}^a \in \mathbb{R}^L$ across all channels were calculated and visualized in Figure 15. The mean relevance for each approach and at each time step is calculated as

$$\bar{R}^a = \frac{1}{D} \sum_d R^a_{:,d}.$$

For DTD-A the relevance $R^a_{:,d}$ is the element-wise product of the attention weights $\alpha_:$ and annotations $a_{:,d}$, as defined in Section 8.2.1. The mean relevance $\bar{R}^a$ is thus a weighted average representation of channel activations $a_{:,d}$, weighted by attention weights $\alpha_:$.

The DTD-A mean relevances for images Figure 15(a) and (c) show a clear focus on the dominant object in each image over the first part of sentence, despite the soft attention weights $\alpha$ being quite diffuse. Over the latter part of the sentence, when the caption focusses on the background, the dominant objects remain present in the mean relevances, even though the attention weights focus on the background. This supports the hypothesis from Section 8.2.1 that a large number of channels are activated by these dominant objects, which dominate the effect of lower $\alpha$ weights.

The DTD-B mean relevances in Figure 15(a) and (c) show a much sparser focus on the main objects, and focus on the background in line with the attention weights towards the end of the sentences. For images in Figure 15(b) and (d) the DTD-B mean relevances are more focused on the object in focus of the attention mechanism, e.g. the woman and plate in Figure 15(b), and transition more in line with the attention weights towards objects in the background, e.g. the beach in Figure 15(d).

To gain further insights, Figure 20 in the Appendix plots the relevances $R^a_{:,d}$ from the DTD-A and DTD-B approach next to each other. The top row depicts the activation maps $a_{:,d}$ from the top ten channels, arranged in descending order of maximum activation value (as in Figure 17). The leftmost

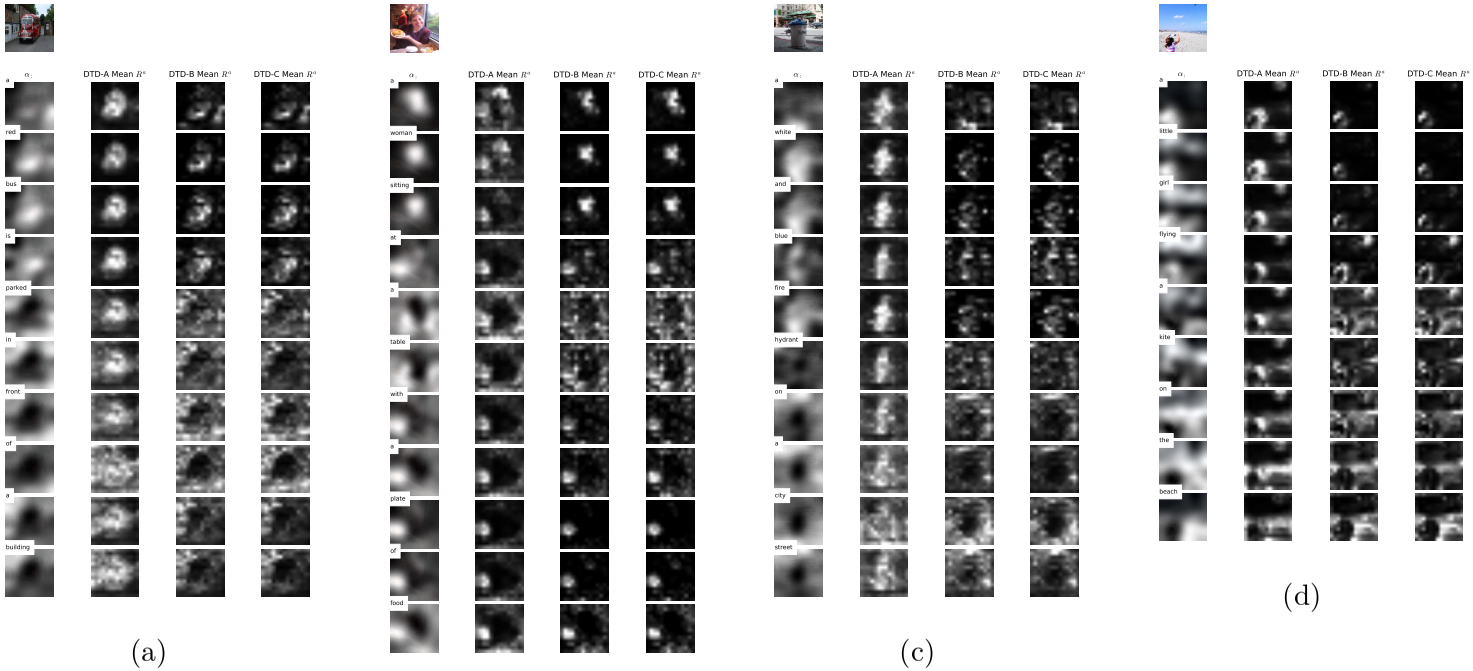



Figure 15: DTD-A, DTD-B, and DTD-C mean relevances

column depicts the alpha weights at each time step, the columns of images below each activation map depict the relevances $R^a_{:,d}$ at each time step. For the image in Figure 20(d), we can see that channels 47 and 125, that detected the kite in the image, receive higher relevances throughout the middle of the sentence, when the kite is mentioned in the description. Channels $77, 233$, and 412, that recognize the beach and waterline, receive higher relevances towards the end of the sentence, when the focus is on the beach in the background. Similarly for the image in Figure 20(c), some of the channels that detected the hydrant receive higher relevances towards the first part of the sentence, whereas e.g. channel 185 receives higher relevances towards the end.

## 8.4 Heatmaps

### 8.4.1 DTD-A

Figure 16 depicts for each input image two columns of images - the left column shows the attention weights $\alpha_:$ at each time step, and the right columns list the corresponding heatmaps from the DTD-A approach. The heatmaps for DTD-A show detailed contours of the detected objects in the image, with higher importance given to objects in the soft attention focus, as expected.

The images in the dataset generally feature a dominant object, e.g. a bus, hydrant, or person, and the NIC focuses on these objects at the beginning, and then shifts its attention to the background. The heatmaps in Figure 16 show that DTD -A identifies these objects well, e.g. the bus in Figure 16 a), but when the descriptions switch to the background, the importance of the dominant object persists in the heatmaps, although the soft attention weights show a clear focus away from that object. A hypothesis is that the activation values for filters that identify objects like buses, trains, or people, towards which the CNN is optimized, remain high, even when multiplied by a low $\alpha$ weight, relative to less clearly identifiable shapes in the background, e.g. a beach or street.

The activation maps in Figure 17 support this hypothesis. For each image the activation maps of ten channels are shown that had the highest ten activation values of all the 512 channels in the convolutional extraction layer. For images a) and c), most channels are associated to the dominant object. By applying the same $\alpha$ weights to each channel, even a low alpha weight applied to multiple activation maps with high activation values for a dominant object, will result in significant relevances $R^x$. In images b) and d) the top

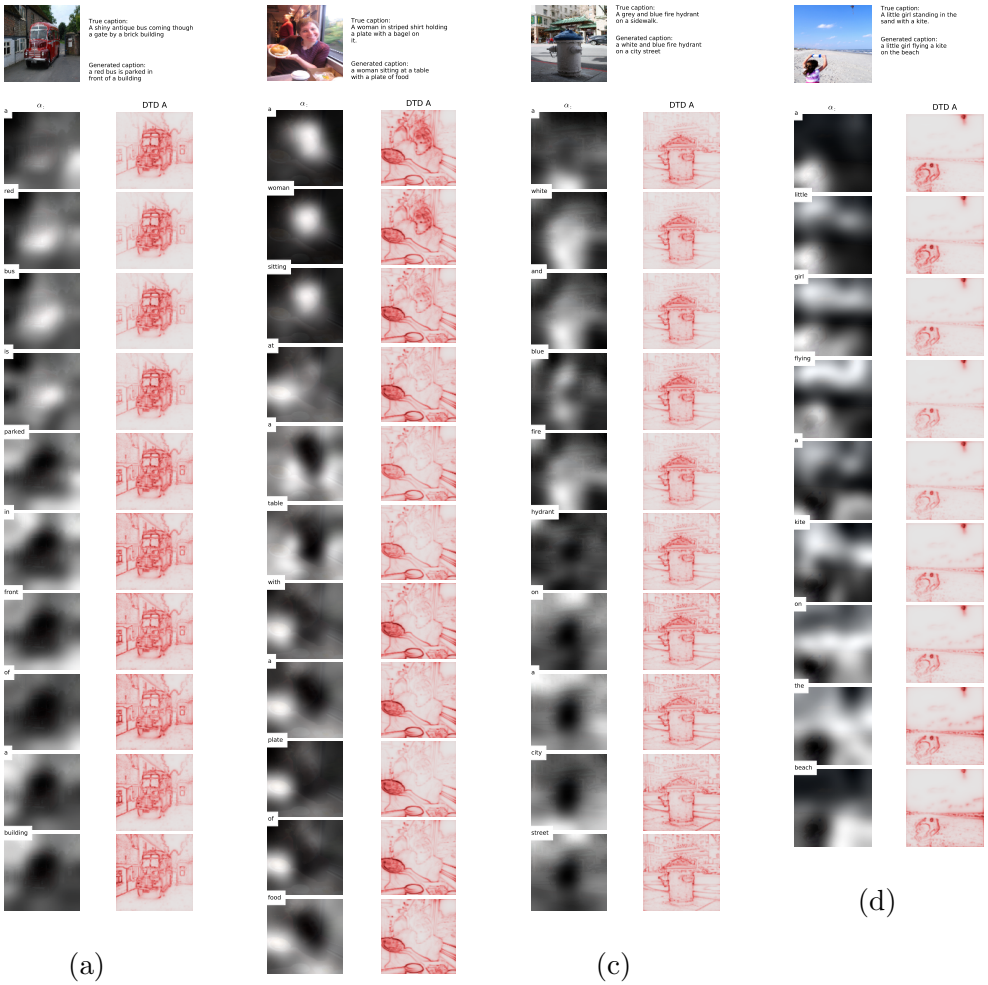



Figure 16: DTD-A heatmaps

ten channels cover a wider range of objects in the image, e.g. for image d) the kid (channels 9, 456, and 386), the kite (channels 47, 125, 386), and the beach (channels 77, 233, 144, and 412). This wider variety in activated channels is also reflected in the DTD-A heatmaps in Figure 16 d), where the heatmap intensities change from the kid to the kite, and finally to the beach.

In general the heatmaps only highlight sensible objects, i.e. corresponding to the caption's content, if the image captioning system is not overfitting and simply mapping to a previously learned caption. For example in Figure 21(c)

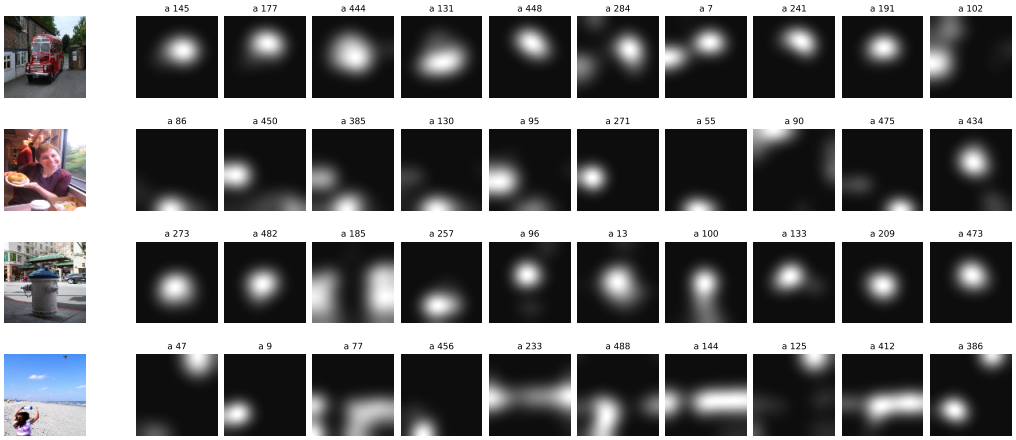



Figure 17: Activation maps of top ten channels

in the Appendix, the generated caption has a high quality, but the heatmaps, as well as the attention maps, show that no distinct object was identified at any point in time. Thus additional work would need to be done to regularize the system further.

### 8.4.2 DTD-B

The DTD-B approach selectively redistributes relevances to channels that were important for the attention mechanism, in the form of the learned attention weights, and we therefore expect the DTD-B approach to produce more focused heatmaps.

In Figure 18 the DTD-A and DTD-B heatmaps are plotted next to each other. The heatmaps for the DTD-B approach identify the objects, e.g. a bus, hydrant, or person, more precisely than DTD-A, while ignoring other objects in the background. The intensities of the dominating objects (bus and hydrant in Figure 18 (a) and (c), respectively) vanish better towards the end of the captions compared to the DTD-A approach, and are therefore more aligned to the attention weights $\alpha_{:}$.

The ability of the DTD-B approach to focus on certain objects is especially clear in Figure 18 (b) and (d). For example, the first three heatmaps in Figure 18(b) solely feature the contours of the woman while almost entirely ignoring the plate of food, the other dominant object in the image. The DTD-A approach attributes a similar importance to both the woman and the plate

of food, even though the attention weights clearly focus on the woman. When the caption transitions towards the plate of food, the DTD-B heatmaps solely feature contours of the plate of food, again a lot more precise than DTD-A.

As expected from the mean relevance results shown in Figure 15, the heatmaps in DTD-B are sparser than in DTD-A. They focus better on attributes that seem important for the current word in the caption. For example in image Figure 18(b) the contours in the first heatmaps are very focused on the facial features of the woman, while the caption begins with "A woman ...".

### 8.4.3 DTD-C

Contrary to the DTD-framework, the $\alpha\beta$-heatmaps can contain negative relevances, which are depicted in blue in the heatmaps in Figure 18, which were produced with $\alpha = 2$ and $\beta = 1$. Other parameter combinations were tested but yielded inferior results (similar to findings in [8]).

The DTD-C heatmaps are sparser than the DTD-A and DTD-B heatmaps, especially for images (a) and (c) in Figure 18. The importance of the dominant object in these images vanishes better towards the end of the caption, more in line with the $\alpha$ weights and the caption's content. A hypothesis is that by redistributing relevances on both positive and negative weights with the $\alpha\beta$-rule, the linear combination of channels, weighted by the positive and negative weights of the attention weights $w_{att}$, is more accurately backpropagated.

Negative relevances have been shown to be attributed to input nodes that introduced uncertainty to the classification outcome, e.g. a broken loop in a handwritten digit as described in [8]. The negative relevances in the DTD-C heatmaps do not seem to allow this sort of interpretation, but rather appear to be artefacts of simplifying assumptions that were made for the relevance propagation.

# 9 Discussion

The goal of this thesis was to analyse and visualize the contribution of image features on the attention model output. In order to disentangle the image features from the hidden state, which are combined in the combined context $q$, relevances were redistributed solely on the projected image annotations $a^p$.

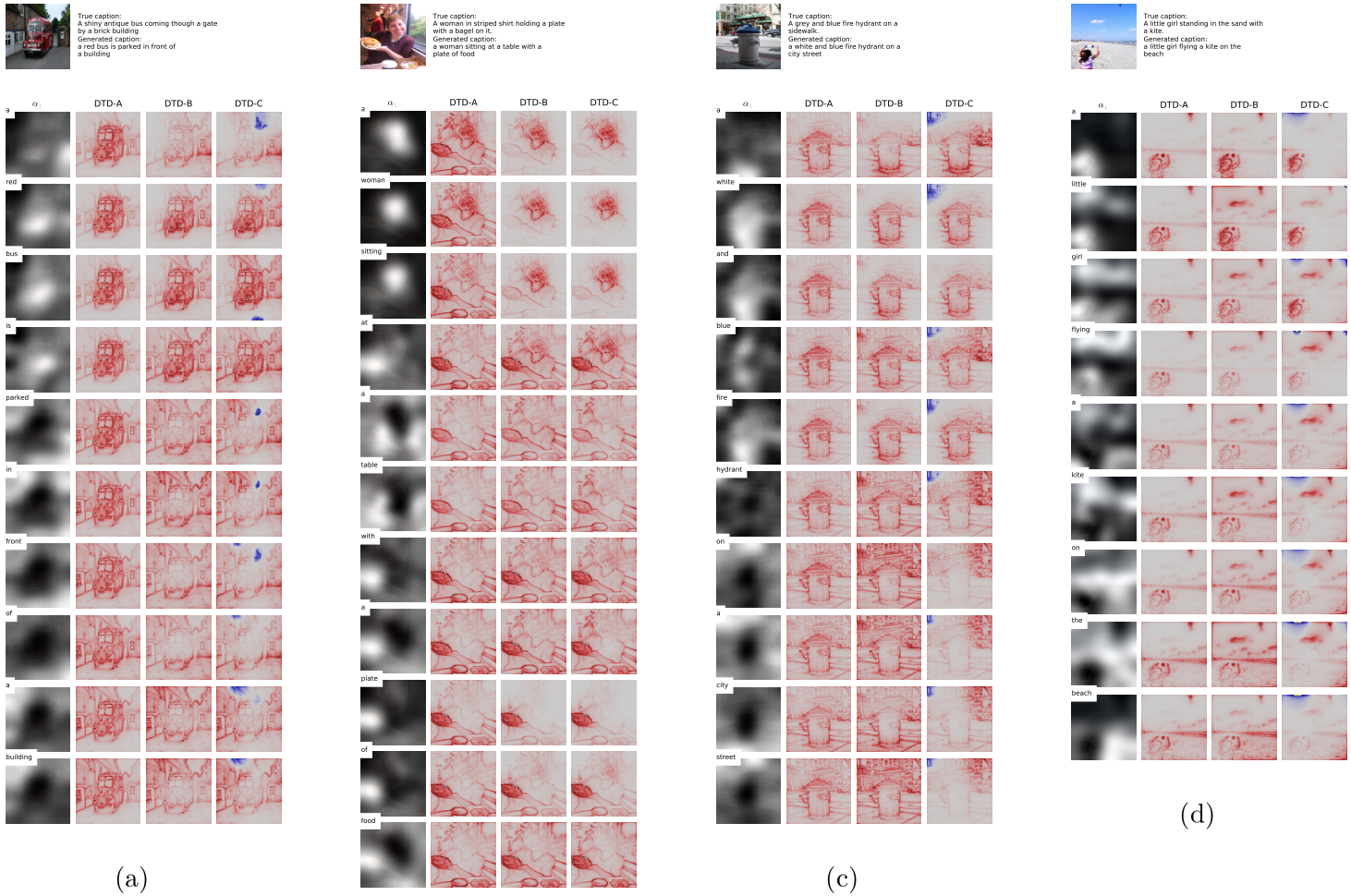



Figure 18: DTD-A, DTD-B, and DTD-C heatmaps

However, to further analyse the dynamics of the attention model, one would need to find an approach that considers the contribution of the hidden state on the attention model output. The hidden state is the sole variable that drives the transition of the attention weights over time, as the projected annotations $a^p$ are constant over time, i.e. the image gets propagated through the CNN once to produce the annotations, which then serve as input at each time period for the attention model.

An unsuccessful attempt was made to make $q$ linearly separable in $a^p$ and $h^p$ in order to fully disentangle the two variables, however the successful training of the attention model relies on the tanh nonlinearity for the combined context. If the disentangling would have been successful, it is unclear in how far DTD could be used to backpropagate the hidden state relevances through the LSTM cell, given its recurrent nature and various violations of the DTD framework.

A further extension of this thesis would be to propagate relevances from the LSTM output, i.e. word prediction, back to the input image at each time step, in order to derive which pixels were important for the predicted word, and not only the attention weights. Again, one would face the challenge of backpropagating relevances through the LSTM cell.

The architecture of the NIC violated several conditions of the DTD framework, e.g. the nodes $a^p, q$, and $e$ in the attention model were not non-negative, and all affine transformations included bias terms. Attempts were made to train the system with an architecture that complied with the DTD framework, however the performance of the system is very sensitive towards architectural changes, and no other setup could be found that achieves similar performance to the original model.

# 10 Conclusion

In order to improve the understanding of an image captioning system, and neural networks in general, this thesis aimed at producing more detailed and informative visualizations of the attention maps produced by the system. To this end a system was built and trained in TensorFlow, and the relevance propagtion method DTD was applied to backpropagte the outcome of the attention model to the images' input pixel.

Three relevance propagation approaches with an increasing level of complexity were explored, DTD-A, DTD-B, and DTD-C. The DTD-A approach uses the attention weights to mask the activations of the CNN, and backpropagates these to visualize the importance of input pixels for the masked activation values. The DTD-A heatmaps showed detailed contours of the objects in the focus of the attention model at each time step. The heatmaps were however cluttered and the transition in relevances corresponded less well with the alpha weights over the latter part of the captions. The DTD-B approach produced less cluttered heatmaps and also showed more accurate transitions across the caption's length. Motivated by the workings of the attention mechanisms, a relevance redistribution rule that captures both negative and positive weights was chosen in the DTD-C approach, which yielded even sparser heatmaps.

The heatmaps provide more insight into the importance of objects for generating the next word in a caption than the soft attention weights. The heatmaps show detailed contours of objects and showed a greater focus on distinct objects even when the alpha weights were blurry. The heatmaps thus aid the analysis of image captioning systems and contribute to the further understanding of complex neural networks.

The application of DTD to the image captioning system required a profound analysis of the attention mechanism. The heavy reliance of the attention mechanism on the nonlinear combined context of hidden state and image annotations was explored, and as no other linearly separable architectures could be successully trained, led to a simplified relevance propagation approach that ignores the hidden state. Further research needs to be done here to disentangle the effect of the hidden state and the image features on the attention mechanism.

# A Appendix

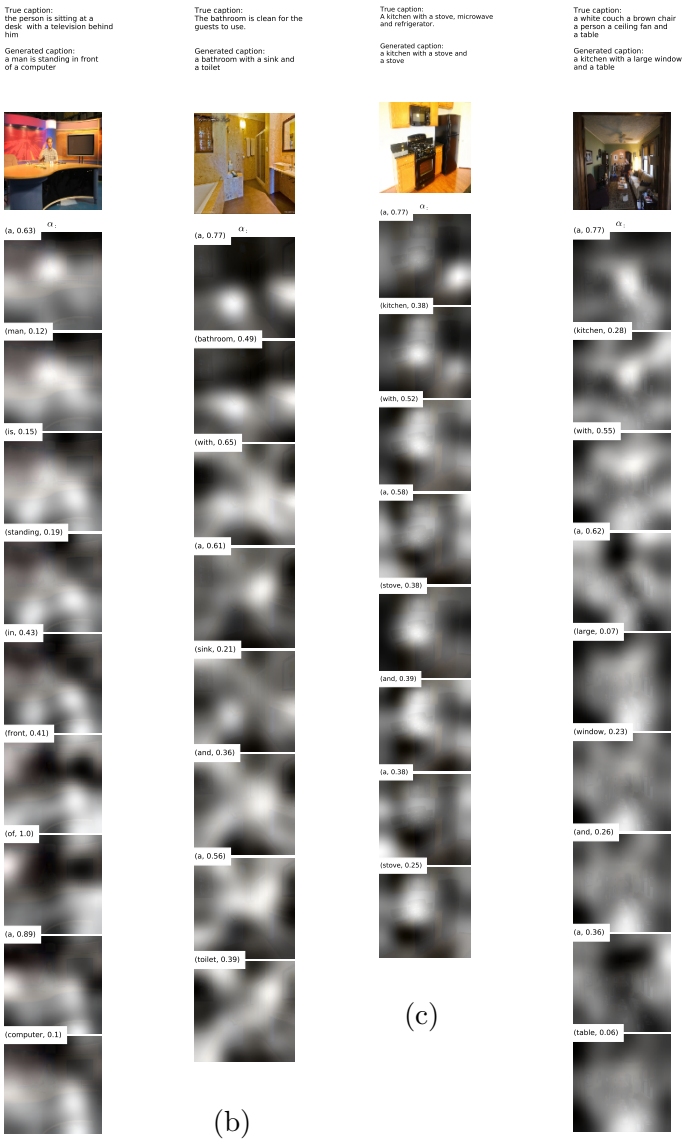
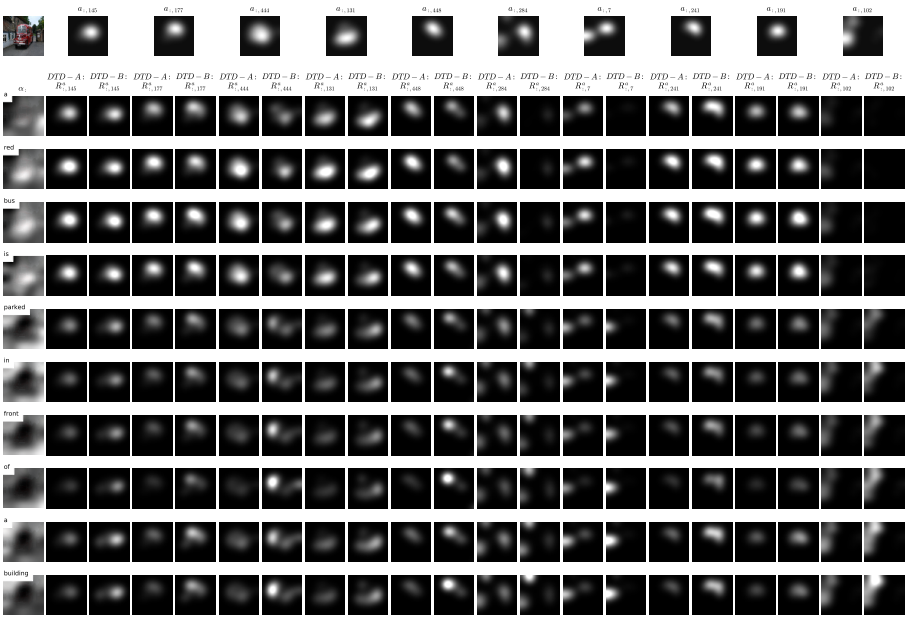
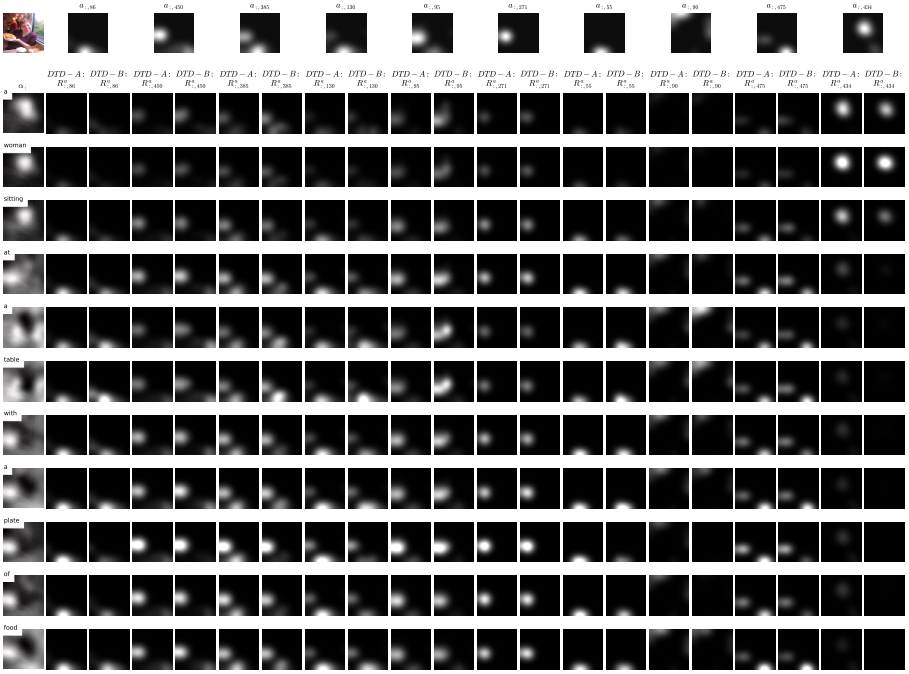


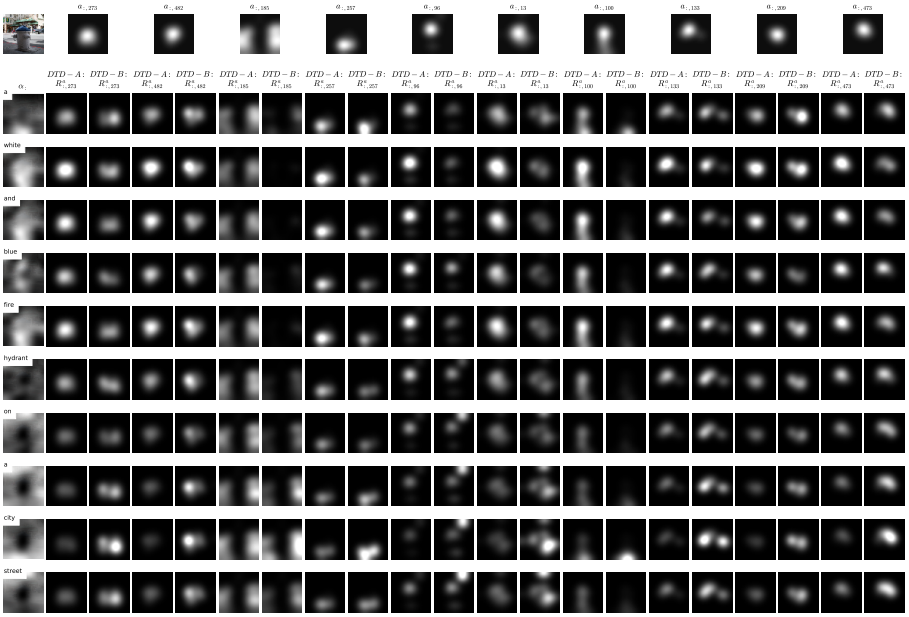

Figure 19: Attention maps generated by Thesis NIC (poor examples)


a
red
bus
is
parked
in
front
of
a
building

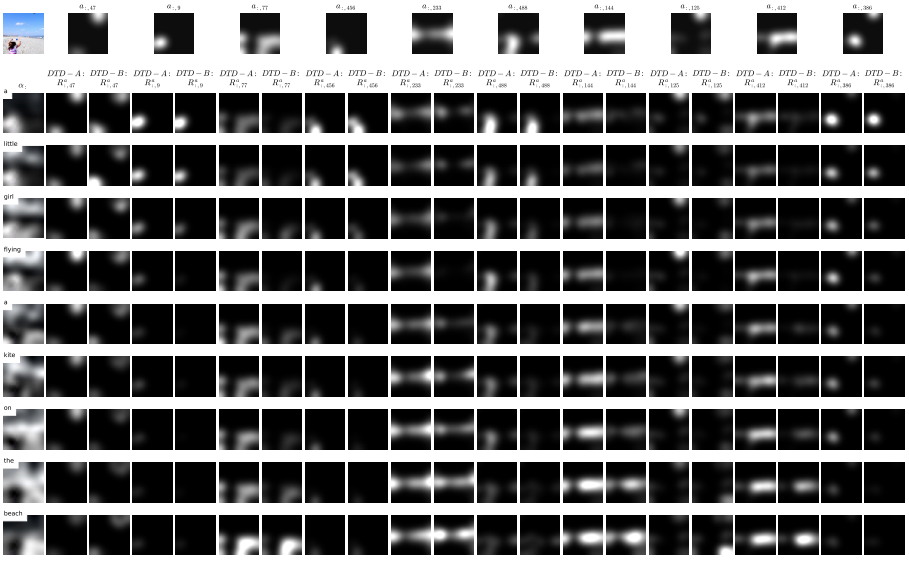
(a)


a
woman
sitting
at
a
table
with
a
plate
of
food

(b)




(c)




(d)

Figure 20: Top ten channel relevances

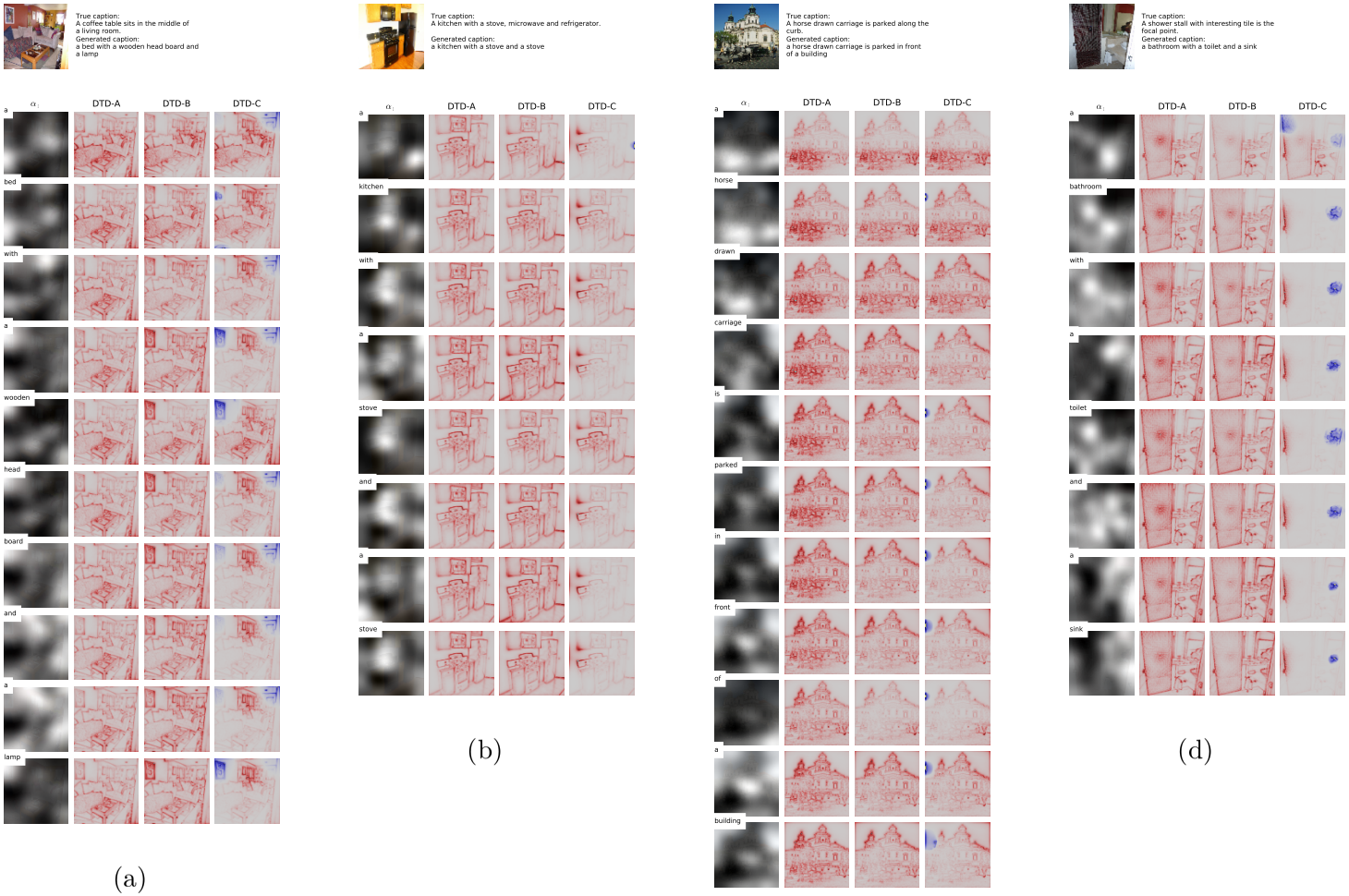



Figure 21: DTD-A, DTD-B, and DTD-C heatmaps (poor examples)

# References


[1] ILSVRC results, 2010. URL http://image-net.org/challenges/LSVRC/2010/results. (visited on 30 May 2017).

[2] ILSVRC results, 2011. URL http://image-net.org/challenges/LSVRC/2011/results. (visited on 30 May 2017).

[3] ILSVRC results, 2012. URL http://image-net.org/challenges/LSVRC/2012/results. (visited on 30 May 2017).

[4] ILSVRC results, 2013. URL http://image-net.org/challenges/LSVRC/2013/results. (visited on 30 May 2017).

[5] ILSVRC results, 2014. URL http://image-net.org/challenges/LSVRC/2014/results. (visited on 30 May 2017).

[6] ILSVRC results, 2015. URL http://image-net.org/challenges/LSVRC/2015/results. (visited on 30 May 2017).

[7] ILSVRC results, 2016. URL http://image-net.org/challenges/LSVRC/2016/results. (visited on 30 May 2017).

[8] Tutorial: Implementing layer-wise relevance propagation, 2016. URL http://heatmapping.org/tutorial/. (visited on 20 June 2017).

[9] Cs231n convolutional neural networks for visual recognition, 2017. URL http://cs231n.github.io/convolutional-networks/. (visited on 12 June 2017).

[10] WILDML: Attention and memory in deep learning and nlp, 2017. URL http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/. (visited on 10 June 2017).

[11] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay

Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

[12] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE*, 10(7):1–46, 2015. ISSN 19326203. doi: 10.1371/journal.pone.0130140.

[13] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation By Jointly Learning To Align and Translate. *Iclr 2015*, pages 1–15, 2014. ISSN 0147-006X. doi: 10.1146/annurev.neuro.26.041002.131047. URL `http://arxiv.org/abs/1409.0473v3`.

[14] Kobus Barnard, Pinar Duygulu, David Forsyth, Nando de Freitas, David M Blei, and Michael I Jordan. Matching Words and Pictures. *Journal of Machine Learning Research*, 3:1107–1135, 2003. ISSN 1532-4435. doi: 10.1162/153244303322533214.

[15] Stephen Bazen and Xavier Joutard. The Taylor Decomposition: A Unified Generalization of the Oaxaca Method to Nonlinear Models. working paper or preprint, May 2013. URL `https://halshs.archives-ouvertes.fr/halshs-00828790`.

[16] B.C. Russell, A Torralba, K.P. Murphy, and W.T. Freeman. LabelMe: a database and web-based tootl for image annotation. *Ijcv*, 77(1-3): 157–173, 2008.

[17] Alexander Binder, Sebastian Bach, Gregoire Montavon, Klaus Robert Müller, and Wojciech Samek. Layer-wise relevance propagation for deep neural network architectures. *Lecture Notes in Electrical Engineering*, 376:913–922, 2016. ISSN 18761119. doi: 10.1007/978-981-10-0557-2_87.

[18] Y-Lan Boureau, Jean Ponce, and Yann LeCun. A Theoretical Analysis of Feature Pooling in Visual Recognition. *Icml*, pages 111–118, 2010. doi: citeulike-article-id:8496352. URL `http://www.ece.duke.edu/{~}lcarin/icml2010b.pdf`.

[19] Po-Hsuan Chen, Xia Zhu, Hejia Zhang, Javier S. Turek, Janice Chen, Theodore L. Willke, Uri Hasson, and Peter J. Ramadge. A Convolutional

Autoencoder for Multi-Subject fMRI Data Aggregation. pages 1–9, aug 2016. URL http://arxiv.org/abs/1608.04846.

[20] Xinlei Chen, Hao Fang, Tsung-Yi Lin, and Ramakrishna Vedantam. coco-caption. https://github.com/tylin/coco-caption, 2015.

[21] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollar, and C. Lawrence Zitnick. Microsoft COCO Captions: Data Collection and Evaluation Server. pages 1–7, 2015. ISSN 0035-8711. doi: 10.1093/mnras/stv1365. URL http://arxiv.org/abs/1504.00325.

[22] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. 2014. doi: 10.3115/v1/W14-4012. URL http://arxiv.org/abs/1409.1259.

[23] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, 2014. ISSN 09205691. doi: 10.3115/v1/D14-1179. URL http://arxiv.org/abs/1406.1078.

[24] Francois Chollet. How convolutional neural networks see the world, 2016. URL https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html. (visited on 26 June 2017).

[25] Dan Cirean, Ueli Meier, and Juergen Schmidhuber. Multi-column Deep Neural Networks for Image Classification. *International Conference of Pattern Recognition*, (February):3642–3649, 2012. ISSN 1063-6919. doi: 10.1109/CVPR.2012.6248110.

[26] Dan C. Cirean, Ueli Meier, Jonathan Masci, Luca M. Gambardella, and J??rgen Schmidhuber. Flexible, high performance convolutional neural networks for image classification. *IJCAI International Joint Conference on Artificial Intelligence*, pages 1237–1242, 2011. ISSN 10450823. doi: 10.5591/978-1-57735-516-8/IJCAI11-210.

[27] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[28] Misha Denil, Loris Bazzani, Hugo Larochelle, and Nando de Freitas. Learning where to Attend with Deep Architectures for Image Tracking. pages 1–30, 2011. ISSN 0899-7667. doi: 10.1162/NECO_a_00312. URL http://arxiv.org/abs/1109.3737.

[29] Michael Denkowski and Alon Lavie. Meteor Universal: Language Specific Translation Evaluation for Any Target Language. *Wmt*, pages 376–380, 2014. doi: 10.1.1.675.6117. URL http://www.aclweb.org/anthology/W/W14/W14-3348.

[30] Jacob Devlin, Hao Cheng, Hao Fang, Saurabh Gupta, Li Deng, Xiaodong He, Geoffrey Zweig, and Margaret Mitchell. Language Models for Image Captioning: The Quirks and What Works. (Me Lm), 2015. ISSN 1539-3755. doi: 10.1103/PhysRevE.92.022112. URL http://arxiv.org/abs/1505.01809.

[31] P. Duygulu, Kobus Barnard, J. F. G. de Freitas, and David A. Forsyth. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. *Computer Vision ECCV 2002*, 2353:113–127, 2002. ISSN 0302-9743 (Print) 1611-3349 (Online). doi: 10.1007/3-540-47979-1. URL http://ci.nii.ac.jp/naid/10020185192/{%}5Cnhttp://www.springerlink.com/index/DMX4GC895QRRJKT4.pdf.

[32] J Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990. URL papers://d0a7bddd-a831-4436-91c1-80941a5b1e09/Paper/p522.

[33] S E Fahlman. The recurrent cascade-correlation architecture. (CMU-CS-91-100), 1991. URL .{%}5Cpdfncastin{%}5CFalhman91b.pdf.

[34] Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. Every picture tells a story: Generating sentences from images. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6314 LNCS(PART 4):15–29, 2010. ISSN 03029743. doi: 10.1007/978-3-642-15561-1_2.

[35] Meire Fortunato, Charles Blundell, and Oriol Vinyals. Bayesian Recurrent Neural Networks. 2017. URL http://arxiv.org/abs/1704.02798.

[36] Andrea Frome, Gs Corrado, and Jonathon Shlens. Devise: A deep visual-semantic embedding model. *Advances in Neural ...*, pages 1–

11, 2013. ISSN 10495258. URL http://papers.nips.cc/paper/5204-devise-a-deep-visual-semantic-embedding-model.

[37] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, 1980. ISSN 03401200. doi: 10.1007/BF00344251.

[38] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[39] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, pages 1–43, 2013. ISSN 18792782. doi: 10.1145/2661829.2661935. URL http://arxiv.org/abs/1308.0850.

[40] Josef Hochreiter. Untersuchungen zu dynamischen neuronalen Netzen. *Master's thesis, Institut fur Informatik, Technische Universitat, Munchen*, pages 1–71, 1991. URL http://scholar.google.com/scholar?hl=en{&}btnG=Search{&}q=intitle:Untersuchungen+zu+dynamischen+neuronalen+Netzen{#}0.

[41] Sepp Hochreiter and Paolo Frasconi. Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies. *A Field Guide to Dynamical Recurrent Networks*, 2009. ISSN 1098-6596. doi: 10.1109/9780470544037.ch14. URL http://ieeexplore.ieee.org/search/srchabstract.jsp?arnumber=5264952.

[42] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL http://www.mitpressjournals.org/doi/10.1162/neco.1997.9.8.1735.

[43] Yangqing Jia, Mathieu Salzmann, and Trevor Darrell. Learning cross-modality similarity for multinomial data. *Proceedings of the IEEE International Conference on Computer Vision*, pages 2407–2414, 2011. ISSN 1550-5499. doi: 10.1109/ICCV.2011.6126524.

[44] Nal Kalchbrenner and Phil Blunsom. Recurrent Continuous Translation Models. *Emnlp*, (October):1700–1709, 2013. ISSN 0147-006X. doi: 10.1146/annurev.neuro.26.041002.131047.

[45] Andrej Karpathy and Li Fei-Fei. Deep Visual-Semantic Alignments for Generating Image Descriptions. *IEEE Transactions on Pattern Analysis*

*and Machine Intelligence*, 39(4):664–676, 2015. ISSN 01628828. doi: 10.1109/TPAMI.2016.2598339.

[46] Andrej Karpathy, Armand Joulin, and Li Fei-Fei. Deep Fragment Embeddings for Bidirectional Image Sentence Mapping. pages 1–9, 2014. ISSN 10495258. URL http://arxiv.org/abs/1406.5679.

[47] Mert Kilickaya, Aykut Erdem, Nazli Ikizler-Cinbis, and Erkut Erdem. Re-evaluating Automatic Metrics for Image Captioning. 1:199–209, 2016. URL http://arxiv.org/abs/1612.07600.

[48] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL http://arxiv.org/abs/1412.6980.

[49] Ryan Kiros, Ruslan Salakhutdinov, and Richard Zemel. Multimodal Neural Language Models. *Icml*, pages 595–603, 2014. URL http://www.jmlr.org/proceedings/papers/v32/kiros14.pdf.

[50] Ryan Kiros, Ruslan Salakhutdinov, and Richard S. Zemel. Unifying Visual-Semantic Embeddings with Multimodal Neural Language Models. pages 1–13, 2014. URL http://arxiv.org/abs/1411.2539.

[51] Alex Krizhevsky, Geoffrey E Hinton, and Ilya Sutskever. ImageNet Classification with Deep Convolutional Neural Networks. *Adv. Neural Inf. Process. Syst.*, pages 1–9, 2012. ISSN 10495258. doi: http://dx.doi.org/10.1016/j.protcy.2014.09.007.

[52] G Kulkarni, V Premraj, and V Ordonez. Babytalk: Understanding and generating simple image descriptions. *on Pattern Analysis . . .*, 35(12):2891–2903, 2013. URL http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=6522402.

[53] Hugo Larochelle and Geoffrey Hinton. Learning to combine foveal glimpses with a third-order Boltzmann machine. *Nips-2010*, pages 1243–1251, 2010. ISSN 1932-6203. doi: 10.1371/journal.pone.0003290. URL papers2://publication/uuid/4E7CE0E0-C8F9-49B9-9B45-051B57B8DDF5{%}5Cnpapers2://publication/uuid/6610B2C4-A06C-4AD7-9041-5557B73F415D.

[54] Y LeCun, L Bottou, Y Bengio, and P Haffner. Gradient Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. ISSN 00189219. doi: 10.1109/5.726791.

[55] Christopher Lennan. image-captioning. https://github.com/clennan/image-captioning, 2017.

[56] Tsung-yi Lin, C Lawrence Zitnick, and Piotr Doll. Microsoft COCO : Common Objects in Context. *Arixiv*, (1405.0312v3):1–15, 2015.

[57] Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan Yuille. Deep Captioning with Multimodal Recurrent Neural Networks (m-RNN). *ICLR 2015*, 2015. ISSN 10477349. doi: 10.1.1.17.4650. URL http://arxiv.org/abs/1412.6632.

[58] Susana Martinez-Conde, Stephen L. Macknik, and David H. Hubel. The role of fixational eye movements in visual perception. *Nature Reviews Neuroscience*, 5(3):229–240, 2004. ISSN 1471-003X. doi: 10.1038/nrn1348. URL http://www.nature.com/doifinder/10.1038/nrn1348.

[59] T Mikolov, M Karafiat, L Burget, J Cernocky, and S Khudanpur. Recurrent Neural Network based Language Model. *Interspeech*, (September): 1045–1048, 2010.

[60] Margaret Mitchell, Jesse Dodge, Amit Goyal, Kota Yamaguchi, Karl Stratos, Alyssa Mensch, Alex Berg, Xufeng Han, Tamara Berg, and Oregon Health. Midge: Generating Image Descriptions From Computer Vision Detections. *Eacl*, pages 747–756, 2012.

[61] Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. Recurrent Models of Visual Attention. *Advances in Neural Information Processing Systems*, 27:1–9, 2014. ISSN 0157244X. doi: ng. URL http://arxiv.org/abs/1406.6247{%}5Cnhttp://papers.nips.cc/paper/5542-recurrent-models-of-visual-attention{%}5Cnhttp://papers.nips.cc/paper/5542-recurrent-models-of-visual-attention.pdf.

[62] Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-robert Müller. Explaining nonlinear classi fi cation decisions with deep Taylor decomposition. *Pattern Recognition*, 65 (November 2016):211–222, 2017. ISSN 0031-3203. doi: 10.1016/j.patcog.2016.11.008. URL http://dx.doi.org/10.1016/j.patcog.2016.11.008.

[63] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Methods for interpreting and understanding deep neural networks. *CoRR*, abs/1706.07979, 2017. URL http://arxiv.org/abs/1706.07979.

[64] Vicente Ordonez, Girish Kulkarni, and Tamara L Berg. Im2Text: Describing Images Using 1 Million Captioned Photographs. *Nips*, pages 1143–1151, 2011. ISSN 9781618395993.

[65] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-jing Zhu. B LEU : a Method for Automatic Evaluation of Machine Translation. (July): 311–318, 2002.

[66] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL `http://www.aclweb.org/anthology/D14-1162`.

[67] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[68] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. ISSN 15731405. doi: 10.1007/s11263-015-0816-y.

[69] Jorge Sanchez and Florent Perronnin. High-dimensional signature compression for large-scale image classification. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1665–1672, 2011. ISSN 10636919. doi: 10.1109/CVPR.2011.5995504.

[70] Jürgen Schmidhuber. Deep Learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015. ISSN 18792782. doi: 10.1016/j.neunet.2014.09.003.

[71] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. 2013. URL `http://arxiv.org/abs/1312.6229`.

[72] P Y Simard, D Steinkraus, and John C Platt. Best practices for convolutional neural networks applied to visual document analysis. *Document*

*Analysis and Recognition, 2003. Proceedings. Seventh International Conference on*, pages 958–963, 2003. doi: 10.1109/ICDAR.2003.1227801.

[73] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale visual recognition. `http://www.robots.ox.ac.uk/~vgg/research/very_deep/`, 2014.

[74] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *International Conference on Learning Representations*, pages 1–14, 2015. ISSN 09505849. doi: 10.1016/j.infsof.2008.09.005. URL `http://arxiv.org/abs/1409.1556`.

[75] Richard Socher, Andrej Karpathy, Quoc Le, Christopher Manning, and Andrew Ng. Grounded Compositional Semantics for Finding and Describing Images with Sentences. *Transactions of the Association for Computational Linguistics*, 2(0):207–218, 2014. ISSN 2307-387X. URL `https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/325`.

[76] Moses Soh. Learning cnn-lstm architectures for image caption generation. 2016. URL `https://cs224d.stanford.edu/reports/msoh.pdf`.

[77] Nitish Srivastava and Ruslan Salakhutdinov. Multimodal Learning with Deep Boltzmann Machines. *Advances in neural information processing systems (NIPS)*, 15:2222–2230, 2012. ISSN 10495258. doi: 10.1109/CVPR.2013.49.

[78] Ilya Sutskever, Oriol Vinyals, and Quoc Le. Sequence to Sequence Learning with Neural Networks. *Advances in Neural Information Processing Systems*, pages 1–9, 2014. ISSN 09205691. doi: 10.1007/s10107-014-0839-0.

[79] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 07-12-June:1–9, 2015. ISSN 10636919. doi: 10.1109/CVPR.2015.7298594.

[80] Virginia Tech, C Lawrence Zitnick, and Devi Parikh. CIDEr: Consensus-based Image Description Evaluation. 2015.

[81] Oriol Vinyals, Jonathan Long, Evan Shelhamer, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and Tell: A Neural Image Caption Generator. *arXiv*, 32(1):1–10, 2015. ISSN 9781467369640. doi: 10.1109/CVPR.2015.7298935. URL `http://arxiv.org/abs/1411.5908v1`.

[82] Songtao Wu, Shenghua Zhong, and Yan Liu. Deep residual learning for image recognition. *Multimedia Tools and Applications*, pages 1–17, 2017. ISSN 15737721. doi: 10.1007/s11042-017-4440-4.

[83] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. arctic-captions. `https://github.com/kelvinxu/arctic-captions`, 2015.

[84] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. *CoRR*, abs/1502.03044, 2015. URL `http://arxiv.org/abs/1502.03044`.

[85] Yezhou Yang, Ching Lik Teo, Hal Daume, and Yiannis Aloimonos. Corpus-Guided Sentence Generation of Natural Images. *Proceedings of EMNLP*, pages 444–454, 2011.

[86] Benjamin Z. Yao, Xiong Yang, Liang Lin, Mun Wai Lee, and Song Chun Zhu. I2T: Image parsing to text description. *Proceedings of the IEEE*, 98 (8):1485–1508, 2010. ISSN 00189219. doi: 10.1109/JPROC.2010.2050411.

[87] Ting Yao, Yingwei Pan, Yehao Li, Zhaofan Qiu, and Tao Mei. Boosting Image Captioning with Attributes. pages 1–11, 2016. URL `http://arxiv.org/abs/1611.01646`.

[88] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent Neural Network Regularization. *arXiv:1409.2329 [cs]*, (2013):1–8, 2014. URL `http://arxiv.org/abs/1409.2329{%}5Cnhttp://www.arxiv.org/pdf/1409.2329.pdf`.

[89] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8689 LNCS(PART 1):818–833, 2014. ISSN 16113349. doi: 10.1007/978-3-319-10590-1_53.