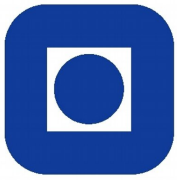
TDT4501 Project Report

Multimodal Deep Learning

Submitted by

Markus Lund
Dag Inge Helgøy

Under the guidance of
Massimiliano Ruocco



Department of Computer and Information Science
Norwegian University of Science and Technology
Trondheim, Norway

Fall Semester 2016

Abstract

Retrieving images given a textual query or getting a description of a specific image, are problems which have a wide area of applications. In this work we focus on how images and text can be compared in a common visual feature space and how this could be used for text-to-image retrieval. Inception and VGG-19 provide state-of-the-art visual feature extraction and word embedding methods like Word2Vec, Global Vectors for Word Representation (GloVe) and sequence embedding using Long short-term memory (LSTM) are used to create dense representations of text. We show that these methods can be applied using neural networks to create a common visual search space consisting of both text and images.

Contents

List of Figures

List of Tables

Acronyms

CBOW continous bag of words. 5, 7

CNN Convolutional Neural Network. iii, 1, 4, 5, 10, 30

GloVe Global Vectors for Word Representation. iii, 1, 6, 8, 12–15, 19, 22, 24, 25, 28

GRU Gated Recurrent Neural Networks. 4, 12

HierSE Hierarchical Semantic Embedding. 9

LSTM Long short-term memory. 1, 4, 14, 15, 19, 25, 27, 30

NIC Neural Image Caption. 30

R@K recall at k. 19, 24

ReLU Rectified Linear Unit. iii, 14, 15

RNN Recurrent Neural Network. 4, 13, 30

VGG Visual Geometry Group. 5, 24

Chapter 1

Introduction

Throughout the project we have encountered and worked with multiple subject related to machine learning and information retrieval. Accurately describing the objects, contexts and events in images using natural language is a one of the more challenging tasks in the area of machine learning.

Images are notoriously hard for computers to accurately describe using natural language because of the subjective nature of their descriptions. The difference in spatial and temporal properties for text and images makes the task of comparing them non-trivial. A challenge in the spatial context could be that a computer is focusing on a rock in the background while the important part of the picture is a person in the foreground. In addition, whether a person crouching should be described as "a person landing from a high jump" or "a person tired from standing" would be difficult from an computer, as well as humans, to determine. Various challenges like abstraction level, area of focus and the level of detail all add a level of uncertainty to the task of multi-modal learning.

1.1 Motivation

Over the years, tag based classification has received a lot of attention, while only recently advancements in multi-modal learning have opened the door for new approaches focusing on natural language and complete sentences. Multi-modal learning address the task of automatically captioning images and retrieving images from a caption query. Our work, which can be found on online[1], is focused on introducing a solution to such an end-to-end retrieval system. Inspiration for our model is based on an existing approach called Word2VisualVec[2], which introduces the concept of comparing both text

[1] `https://github.com/ruoccoma/master_works/tree/master/helgoy-lund`

and images in a common visual feature space. We introduce three easily exchangeable parts that allow comparison using various methods for image and text representation. The three parts consists of word embedding, image embedding and a model to create a mapping between the two. We have also compared our model to another baseline solution to measure the effectiveness of the visual embedding approach.

1.2 Objective of project

While existing approaches to multi-modal learning in the visual feature space have shown state-of-the-art results [2], we want to learn how to implement such a model using several different methods for pre-processing text and images. In this report, we present and discuss results that compare various text and image representation used in the task of text-to-image retrieval.

1.3 Organisation of report

The rest of the report is organised as follows. Chapter 2 introduces concepts and models central for understanding the background for our proposed models. In chapter 3 state-of-the-art implementations and related work in the area of multi-modal learning is presented. Here, the main inspiration for our proposed model, Word2VisualVec, is presented and described. In the following chapter, chapter 4, a baseline for comparison and the architecture of our models is described in detail. This chapter also introduces multiple ways to represent the text and images to be compared. Chapter 5 explain the dataset and metric used in the evaluation of our models. Examples of the available retrieval methods and a discussion of the results is also provided here. In chapter 6, a summary of our work and a description of the challenges faced during the project lifetime is provided. Chapter 7 sums up ideas for improvement and possible further extensions of our system.

Chapter 2

Background theory

In the following chapter we introduce multiple subjects related to machine learning and information retrieval which are encountered during our project.

2.1 Deep neural network

In its simplest form a neural networks is a model which takes input values and returns output values based on this input. The input values is manipulated by a set of weights which is adjusted to receive satisfactory output values. In deep neural networks there are multiple layers of weights manipulating the input values.

2.1.1 Feedforward network

Feedforward networks is a category of neural networks. An example of a simple feedforward network can be seen in Figure 2.1. The characteristics of these networks are the sequential propagation of values through the network as indicated by the arrows in Figure 2.1.

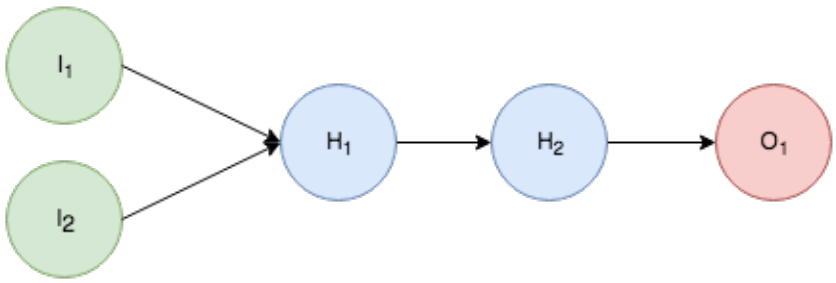


Figure 2.1: A feedforward neural network with two hidden layers.

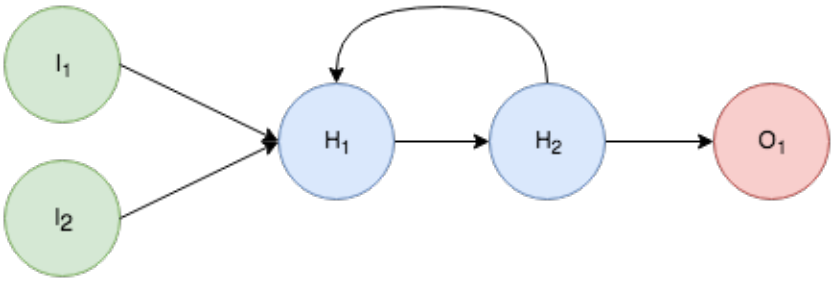


Figure 2.2: A recurrent neural network.

2.1.2 Recurrent Neural Network

Recurrent Neural Network (RNN) is an extension of feedforward networks which is able to handle input sequences with variable length. RNNs contains nodes which generates its output value based on values from nodes both before and after the node in the structure of the network. In Figure 2.2 such a network is illustrated. RNNs can be used for sequence modelling such as modelling a sentence.

Long short-term memory (LSTM) is a RNN architecture which performs well when there are long or unevenly distributed time difference between important input values [3]. **Gated Recurrent Neural Networks (GRU)** is similar to LSTM but is more computationally efficient and has shown similar results that of LSTM [4].

2.1.3 Convolutional neural network

Convolutional Neural Networks (CNNs) are feedforward neural networks specialised on recognising patterns and are very often used in problems regarding images. CNNs have been successful in tasks like image classification and face recognition. CNNs contains operations which mimics how animals visually perceive images and patterns. In Figure 2.3 a typical structure of a CNN is illustrated containing convolution layers which extract features from an image. These features are subsequently processed using sub-sampling, which reduces the dimensionality of the input. In the end there is a fully connected layer which is a high dimensional vector representing the visual features of the image.

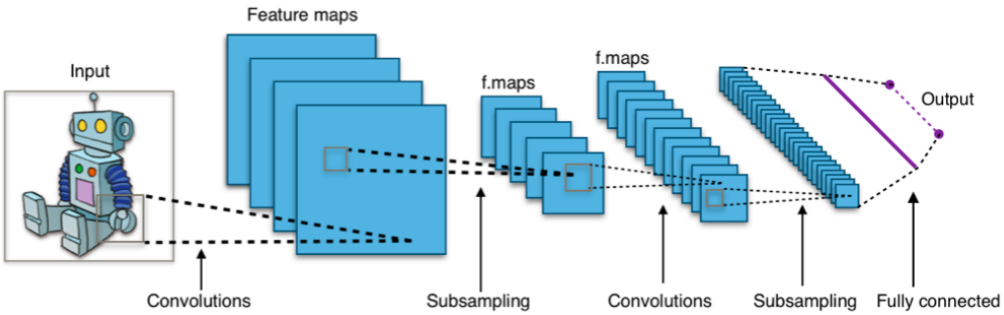


Figure 2.3: Structure of typical CNN [1]

2.2 Image embedding

We define image embedding as the method of mapping an image to a vector of real values representing its visual features. One way of achieving such embeddings are training a CNN on images with labels, letting the network learn mappings from structures in images to different classes. Instead of designing and training such a network, it is possible to use existing pre-trained image classifiers, such as Incpetion and the Visual Geometry Group (VGG) image classifier **VGG-19**. **Inception** refers to the inception v3 architecture introduced by Szegedy et al. [5].

2.3 Word embedding

Word embedding is a method used to map a word represented as a string to a vector of real values. It has the potential to describe relationships otherwise hidden from representations like one hot or hashing. Word embeddings provides a dense and rich word representation which can be used in pre-processing for deep learning to deal with the curse of dimensionality. It spreads words in a multi-dimensional embedding space allowing concepts to be calculated using algebraic vector calculations like in Figure 2.4.

2.3.1 Word2Vec

Methods that utilise the strength of neural networks to capture the semantic relationship between words like Word2Vec[6] has gained a lot of interest over the past few years because of its speed of creating semantically accurate embeddings. Word2Vec uses a neural network to predict words based on the context of the words around it. It can be structured in two ways; continous bag of words (CBOW) or Skip-gram. The former uses the predicted word

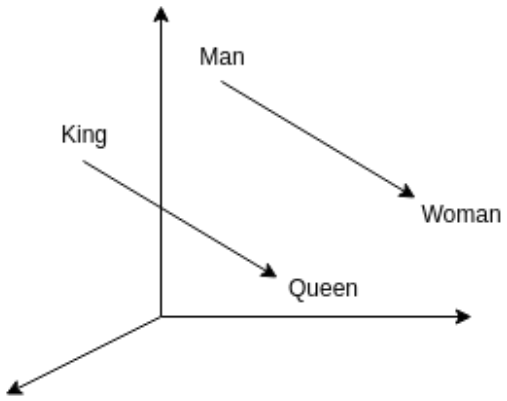
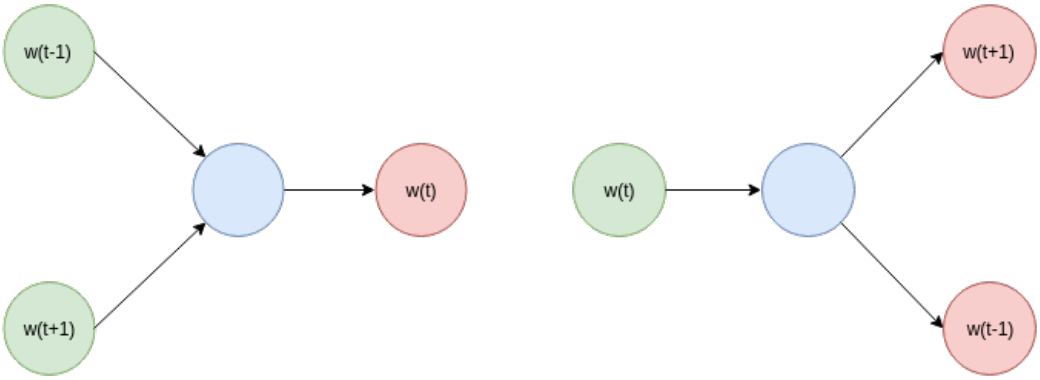


Figure 2.4: The relationship between the vector representations of man and Woman is the same as between King and Queen. This makes it possible to calculate the following: $King - Man + Woman = Queen$.

as input and the context as output as seen in Figure 2.5a, while the latter reverse these positions as seen in Figure 2.5b. By training these models on a corpus of text, the error calculated between the context and the predicted word is used to adjust the weights in the network.

2.3.2 Glove

GloVe [7] also bases its vector representation in the co-occurrence of words, but differ in the way it uses this information. GloVe is a count-based model and starts by creating a large frequency matrix for the occurrences of words in different contexts. This high dimensional matrix is then factorised to lower the dimensionality and produce a matrix which has a dense vector representation for each word.



(a) CBOW (b) Skip-gram

Figure 2.5: Word2Vec architectures. w(t) denotes the predicted word.

Chapter 3

State of the art

Comparing text and images is not a new field of study, and a lot of work has been done in the recent years. In this chapter we look at some of the solutions that have proven state of the art results in both text embedding and models doing images-to-text and text-to-images retrieval. We also take a closer look at the model proposed by Dong et al. [2], which we use as inspiration for this project.

3.1 Word embedding

While Word2Vec and GloVe only use the context of words, it has been shown that expanding with the visual context of words outperform Word2Vec on some tasks. Visual Word2Vec [8] is one such an embedding that uses deep learning to cluster visual, rather than textual, semantics. For example, a relation between a girl eating and staring at an ice cream might be challenging to represent using only a textual embedding, but the visual relation is apparent. Visual Word2Vec aims to bring such visual relations closer together and words not visually related further apart. For the tasks of common-sense assertion classification, visual paraphrasing and text-based image retrieval this model surpass the Word2Vec model.

3.2 Multi-modal comparison

The Deep Visual Semantic Embedding model (DeViSE) [9] combines two pre-processing neural networks for text and image into a neural network, called the deep visual semantic model. It trains on the similarity of the word embeddings and a transformed visual model. The pre-processing of words is done by learning a Skip-gram model to represent words as embedding

vectors using Word2Vec [6]. This model is trained on a corpus of 5.7 million documents from Wikipedia [10] and transforms words into vectors of 500 or 1000 dimensions. To create vector representations of images a model pre-trained on the ImageNet[11] dataset is used. In the combined neural network the images are transformed into dimensions equal the size of word embeddings to allow comparison. DeViSE is creating a common textual feature space for both text and images to be able to compare both in a mutual space. An illustation of how DeViSE creates a common textual space can be seen in Figure 3.1b.

Follow-up work on the DeViSE model done by Norouzi et. al [12] use Word2Vec to embed both image labels and text. The image embedding is done by finding and using a convex combination of the most relevant labels to the image.

Hierarchical Semantic Embedding (HierSE) builds on work by Norouzi et. al [12]. This model also uses Word2Vec [6], however it trains its Skip-gram model on Flickr [13] tags instead of web documents like DeViSE. The argument for this is that tags better represent visual features as the co-occurences of tags. Since HierSE uses a word embedding made from feature it can match using a single word instead of an averaging over a sentence used in prior works [9; 12]. In addition, the proposed method utilizes a hierarchical structuring of labels. This enables the model to embed the labels hierarchically and improves similarity measures for rare labels.

Images can be used to combine deep visual semantic features with traditional text features to improve webstore search. The authors of the paper "Images Don't Lie" [14] approach this by concatenating a vector representing the text features with a vector representing the visual features from the corresponding image. By using a pre-trained convolutional neural network the authors are able to convert an image into a vector representing its visual features. By creating this model, they were able to improve search results by 1.7% gain in average ranking quality.

Recent work by [15] Liu, Mei et. al improves ordinary video thumbnail selection by introducing a multi-task visual-semantic embedding model. This model is able to automatically choose query-dependent thumbnails from a video using both visual information and textual information related to the video such as title and tags. The visual semantic embedding model consists of a one word embedding model to process the textual query input and one convolutional neural network to process the thumbnail input. This is then combined into a latent semantic space.

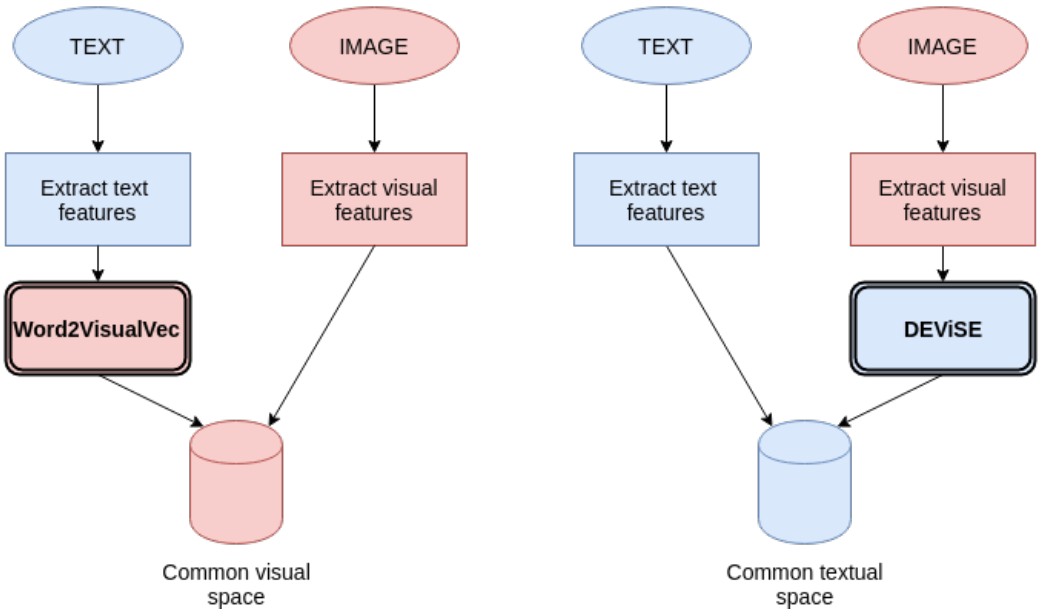
3.2.1 Word2VisualVec

The motivation behind the Word2VisualVec paper is to embed text into a visual feature space making it is possible to capture both the semantic meaning from a text and the visual features from an image into a common space. [2]

Dong et al. manages to create a new deep neural network architecture that learns to predict a visual feature encoding from text. They also show that this model outperforms state of the art models on both text-to-image retrieval and image-to-text retrieval. The goal is therefore to create a model which learns a visual representation of text and then use this text representation to say how similar it is to an unlabeled image.

Dong et al. choose to use existing pre-trained deep CNNs to extract feature vectors from images to represent images. This is done by extracting the values from the last fully-connected layer in the pre-trained models. Representing text is done by using Word2Vec, see subsection 2.3.1. The Word2Vec model used is trained on a large corpus of 30 million Flickr tags, and each word is represented as a 500 dimensional vector. Since one image can be described with more than one tag, they use mean pooling over the tags, creating one vector which represents all the tags of a single image.

The cross-media model simply consists of a feedforward network learning the visual features from a vector representing textual features. A usage of this model is illustrated in figure 3.1a. This figure show that features from text is extracted before fed into the Word2VisualVec model which generates a visual encoding of the text. This encoding can then directly be compared to other visual features generated from for instance a CNN.



(a) Word2VisualVec (b) DeViSE

Figure 3.1: Illustrations of the systems in Word2VisualVec and DeViSE

Chapter 4

Architecture

We propose an implementation that splits into three different modules that could easily be replaced; word embedding, image embedding and a module which allows for both text and images to be compared using their visual features in the form of a feedforward neural network. Such a system would allow retrieval of images and captions between both of these modalities. For comparison, we introduce an existing baseline solution. Table 4.1 show the main differences between our system and existing solutions.

4.1 Baseline

Instead of comparing text and images in a common visual feature space, our baseline model creates a new learned subspace for text and images, as seen in Figure 4.1. The model maps both text and images into a vector of a fixed dimension to allow for comparison in this mutual space [16]. The structure and topology of the baseline model can be seen in Figure 4.3. This model does not use any pre-trained model for word embedding such as GloVe, but instead learns the embedding of sentences during the training of the model. This is done using an embedding layer which generates a dense representation from a sparse input and a GRU[4] layer. The text vectors are then sent through a normalization and a element-wise absolute value lambda layer before it is compared to the vectors propagated through the image part of the network. The images are embedded as a 4096-dimensional vector and sent through the image part of the network. The image part consist of a single feedforward layer, followed by a normalization and a element-wise absolute value lambda layer. The last layer of the model merge these images together with the text vectors.

Model	Common space	Word dim	Image dim
Baseline	learned subspace	300	4096
DeViSE	textual feature space	500/1000	4096
Word2VisualVec	visual feature space	500	1024/4096
Our models	visual feature space	300	2048/4096

Table 4.1: Comparison of models

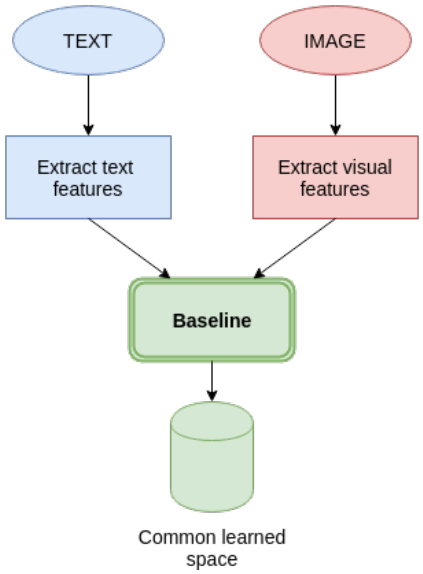


Figure 4.1: Illustration of the baseline model

4.2 Word embedding

We define three different methods for processing and representing words, a Word2Vec model trained on our dataset, a pre-trained GloVe dataset and a RNN-based method which does not require a learning based processing of words.

4.2.1 Word2Vec & GloVE

Our Word2Vec implementation contain words represented by vectors with a dimensionality of 300. These high dimensional vectors are created while training on our dataset using the method described by Mikolov et al. [6]. This generates a single vector for each distinct word in the dataset, allowing us to represent a word using this embedding. The idea behind training our own Word2Vec dataset is to create word embeddings which knows the scope of the problem well, in the sense that every word has been seen at least once by the system.

In addition to training our own Word2Vec embeddings it is interesting to look at a pre-trained dataset with word embeddings. The GloVe dataset [7] is pre-trained on 6 billion tokens from Wikipedia articles and contains a vocabulary of 400 000 words. The main drawback with this method is that the complete vocabulary of our dataset does not necessarily exist in the vocabulary of the pre-trained GloVe database.

The two word embedding methods described gives vector representations of single words. To represent a sequence of words as a vector we combine the embedded words from either Word2Vec or GloVe into a single vector using element-wise averaging. These combined vectors are used as input in the feedforward network described later in this chapter.

4.2.2 LSTM

Representing words as vectors using Word2Vec and GloVe requires preprocessing of words using a trained model. This is because the two models is required to learn a relationship between different words from a corpus of documents. An LSTM based solution is implemented to allow direct translation between sentences and the image space. The LSTM based approach only requires that a sequence of words is converted to a vector with indices of words from an inverted index dictionary. It uses a fixed length vector to represent a sequence of words, allowing it to retain information of previously seen words in a sentence. This allows the model using LSTM to better understand the meaning of sentences, not just interpret it as a set of words.

Method	Dimensions	Vocabulary size	Pre-trained
Word2Vec	300	Training vocabulary	On training data
GloVe	300	400 000	6 billion wikipedia docs
LSTM	300	Unlimited	No

Table 4.2: Overview of word embedding methods

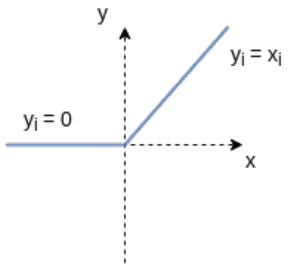


Figure 4.2: The ReLU activation function.

4.3 Image embedding

Image embeddings are created using pre-trained image classifiers. The output from the last hidden layer of Google's Inception model provides a dense embedding vector with 2048 dimensions. In addition, image embeddings as 4096-dimensional vectors using the VGG-19 [17] image classifier is added for comparison.

4.4 Feedforward network

The model translating embedded sequences of words into the image space consists of a feedforward neural network built in Keras [18] using a Tensorflow [19] backend. Our model is based on Word2VisualVec [2], where the model learns a new representation of text embeddings while the image embeddings remains unchanged. We decided to use the adam optimizer because it supports adaptive learning rate and momentum. Our proposed models, as well as the baseline, use ReLU as the activation function. As shown in Figure 4.2 ReLU is linear for x values above 0, and 0 for values below and since ReLU is a linear function, it has a low cost of computation. As the objective function for our model we use contrastive loss (4.1), since this cost function creates larger contrasts in values by moving close values closer and more

distinct values further apart [20].

$$\sum_{(c,1)} \left(\sum_{c\prime} max\{0, \alpha - S(c,i) + S(c\prime,i)\} + \sum_{i\prime} max\{0, \alpha - S(c,i) + S(c,i\prime)\} \right) \tag{4.1}$$

In Figure 4.4 the structure of our model is shown using Word2Vec for textual embeddings and Inception for image embeddings. As shown in the figure, 300-dimensional vectors created from Word2Vec is the input for the left branch, while 2048-dimensional vectors generated by Inception defines the second input. The two lambda layers in Figure 4.4 normalizes and does element-wise absolute value on each input vector. As previously mentioned we also test our model using the pre-trained GloVe embedding, but since 300 dimensions for each word vector is used for GloVe aswell, the structure of the GloVe based network would be identical as that of Figure 4.4.

Figure 4.5 shows our third word embedding method, which consists of an LSTM layer. This model does not use any pre-processed information about words or captions, but instead takes captions represented as a vector with indices from a glossary as input. These input vectors are padded with zeroes to reach a dimensionality of 82, since the largest caption in the dataset contains 82 words. The embedding layer converts these sparse vectors into a dense representation which is then used as input in a LSTM layer. The output from the LSTM layer is then mapped into a visual feature space identical of the image input.

The last layer of the models in Figure 4.3, 4.4 and 4.5 is a concatenation layer which allows us to have both the image embedding and the learned image embedding as input in the contrastive loss function.

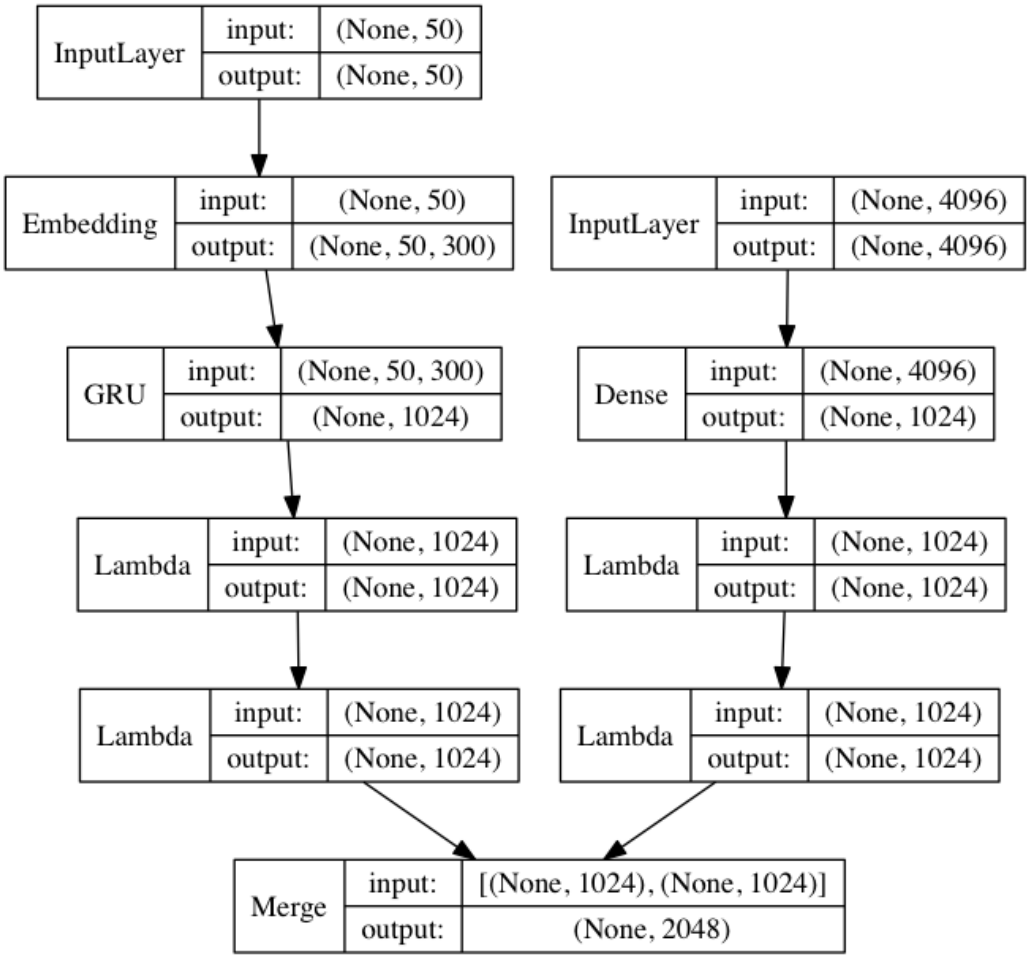


Figure 4.3: Structure of the baseline model.

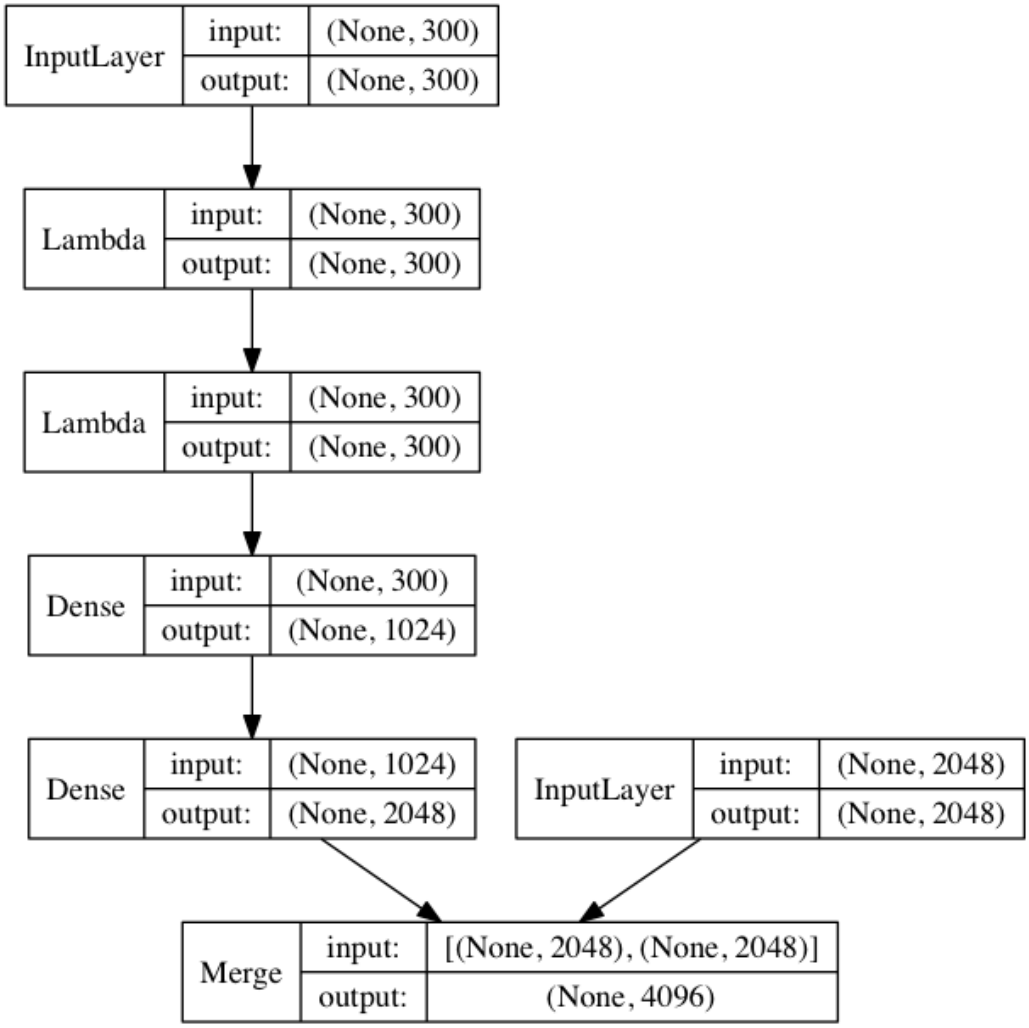


Figure 4.4: Structures of our feed forward network using Inception for image embedding and Word2Vec as word embedding

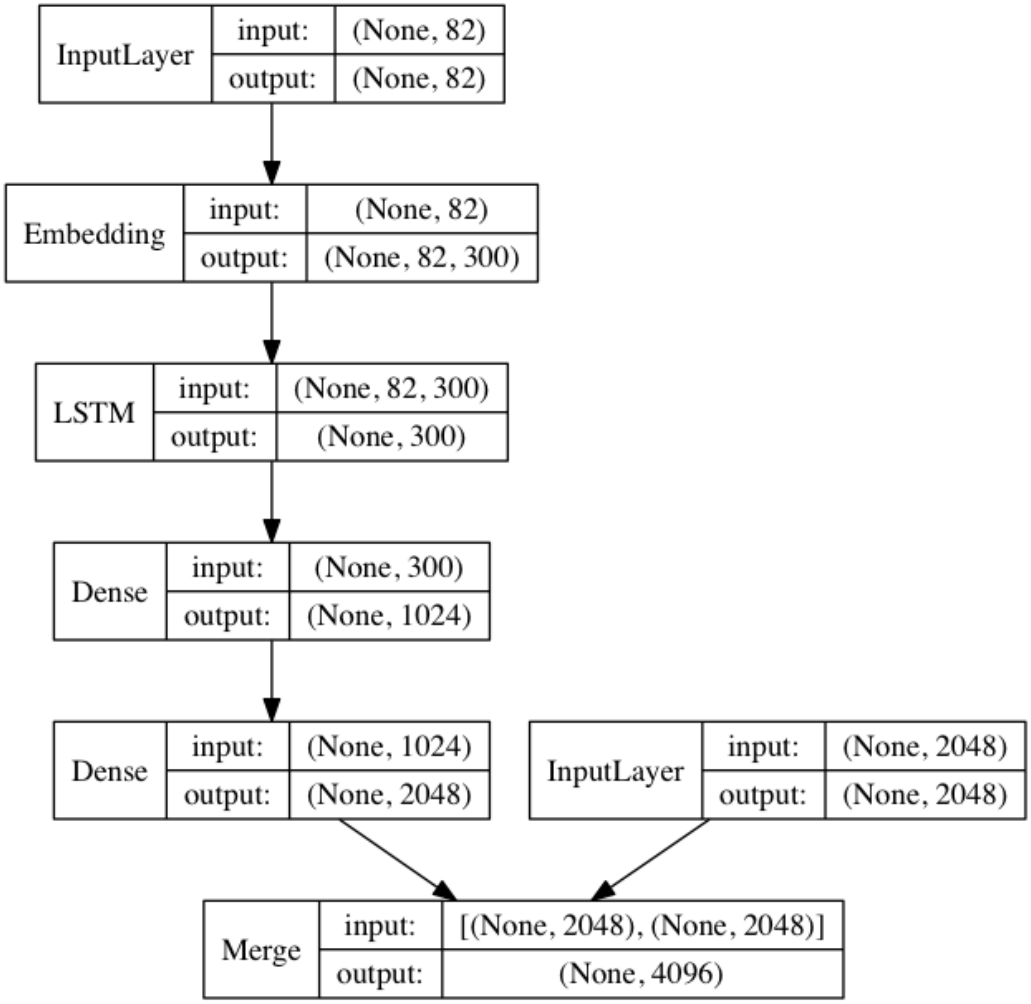


Figure 4.5: Structures of our feed forward network using Inception for image embedding and LSTM for word embedding

Chapter 5

Experiments and results

5.1 Experimental setup

Three different methods for representing the captions, LSTM, Word2Vec and GloVe, and two different pre-trained image classifiers, Inception and VGG-19, gives us six different combinations of models to evaluate. In this section we describe the evaluation methods we use for our models.

5.1.1 Data

To evaluate our model we use the Flickr30k dataset [21] which consists of around 30 000 images with 5 corresponding captions each. Each image in this high-quality dataset, D, is manually described by five individual annotators. An expample of this is shown in Figure 5.1. We use 80% of the dataset as our training set, D_{train}, and 20% as the test set, D_{test}.

5.1.2 Tasks

Evaluating the models is done by performing text-to-image retrieval. Each caption from D_{test} is used to predict a visual feature vector that is compared to the images in D. The comparison is done using the cosine similarity between the transformed caption, and all of the images in the dataset. R@K is the percentage of queries for which the model predicts the correct result within the top k retrieved results and is used as our evaluation metric.

However a caption does not necessarily describes only one image perfectly. It is interesting to also test text-to-image retrieval qualitatively by looking at the actual images retrieved.

1. A man is midair above the water on a water board as the water splashes up underneath him.
2. A man is surfboarding and is in the air with the wave splashing behind him.
3. guy on a surfboard riding the wave as he is in the air.
4. A male surfer catches some air off of a big wave.
5. Man is skim boarding and getting a lot of air.

Figure 5.1: An example image with five captions from the Flickr30k dataset

5.2 Results

In Table 5.1 three different recall intervals is shown for text-to-image retrieval for the six combinations of models and the baseline model. The results is calculated from the complete test set of captions. Figure 5.3-5.9 shows the top 3 retrieved images for the textual query "A group of people are rock climbing on a rock climbing wall". The image representing the query caption is shown in Figure 5.2.

5.3 Discussion

Our final result gave both interesting and unexpected results. We found the large difference in results between the two image embedding models especially interesting. Overall our models, except the Inception/LSTM based model, performed better than the baseline. This can be seen by comparing Figure 5.3 to Figures 5.4-5.7.

Figure 5.2: Query image with caption: *A group of people are rock climbing on a rock climbing wall*



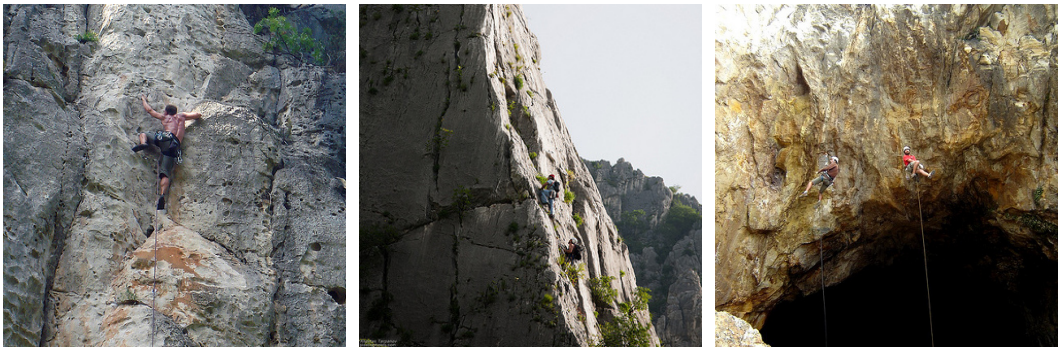
(a) Top 1 (b) Top 2 (c) Top 3

Figure 5.3: Example result for the Baseline model



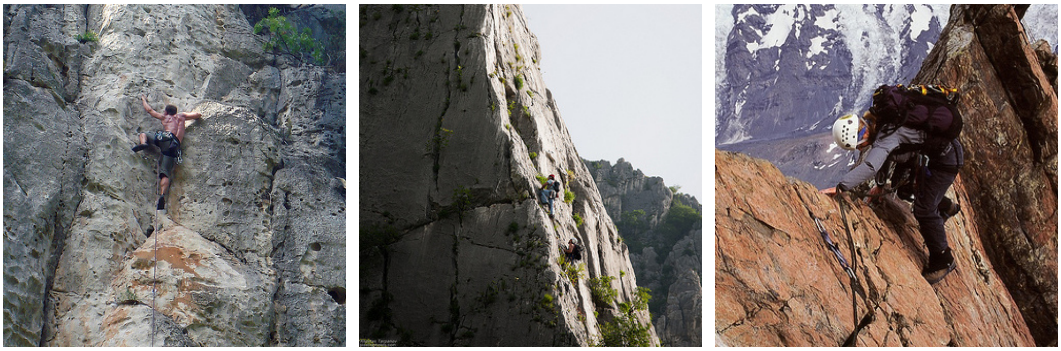
(a) Top 1 (b) Top 2 (c) Top 3

Figure 5.4: Example result for the VGG-19/LSTM model



(a) Top 1 (b) Top 2 (c) Top 3

Figure 5.5: Example result for the VGG-19/GloVe model



(a) Top 1 (b) Top 2 (c) Top 3

Figure 5.6: Example result for the VGG-19/Word2Vec model

(a) Top 1 (b) Top 2 (c) Top 3

Figure 5.7: Example result for the Inception/GloVe model

(a) Top 1 (b) Top 2 (c) Top 3

Figure 5.8: Example result for the Inception/Word2Vec model

(a) Top 1 (b) Top 2 (c) Top 3

Figure 5.9: Example result for the Inception/LSTM model

Image embedding	Word embedding	r@1	r@10	r@100
Baseline				
VGG-19	GRU	0.03	0.25	2.06
Our models				
VGG-19	LSTM	**2.39**	**13.42**	**49.42**
VGG-19	GloVE	0.60	3.66	18.74
VGG-19	Word2Vec	0.49	3.31	16.00
Inception	GloVE	0.10	0.81	5.27
Inception	Word2Vec	0.08	0.64	4.57
Inception	LSTM	0.00	0.03	0.28

Table 5.1: A comparison of models using R@K

5.3.1 Image embedding

VGG provided overall better results than inception, regardless of word embedding. Higher dimensionality of VGG could be the reason for this improvement in results. It does however require our model to learn a mapping into a higher dimension. When looking closer at the embeddings generated by Inception and VGG we can see that the former embedding contains 2048 mostly non-zero values. On the other hand the 4096 dimensional image embeddings generated from VGG contains more than 2048 non-zero values, even though it contains 4096 values in total. The embeddings from VGG contains in general around 60-70% of values equal to zero. This could perhaps allow these embeddings to better distinguish between different features, causing the increase in performance. Visually comparing the results of the VGG-19 and Inception models in Figures 5.4-5.7 show that the models differ in the way they interpret images and that they have been trained on different datasets. For example Inception seems unable to differentiate working with rocks and climbing on rocks, as can be seen in Figures 5.7a, 5.8a, 5.8c.

5.3.2 Word embedding

As can be seen in Table 5.1, using a pre-trained word embedding method like GloVe gave better results than using Word2Vec trained on our training data. The reason for this could be that in general the GloVe dataset contains a large enough vocabulary that is sufficient for both training and testing. However, it could be reasonable to think that our generated Word2Vec dataset would achieve better results on our data since it has seen all the words in the data at least once. On the other hand this is not the case when testing the data, since the testing set could contain words not existing in the training

vocabulary, causing it to perform poorer while testing. It is also possible that each word embedding itself is poorer in Word2Vec than GloVe, since it is trained on a much smaller corpus of documents. These flaws however are based on our implementation and training of Word2Vec and GloVe, rather than the methods themselves. When looking at the qualitative results from the top 3 retrieved images, it is hard to see any difference between the usage of Word2Vec and GloVe. In the VGG-19 models, there is only a difference between Word2Vec and GloVe in the third highest ranked images. However, both could be interpreted as relevant using a subjective evaluation method.

The third method for learning word embeddings using an LSTM outperformed the other word embedding methods. It has the advantage of being able to retain information about the sequence of words, causing the embedding to contain information about the meaning of a caption. Because the LSTM takes the order of words into consideration, it could ideally know the difference between the captions "dog running past man" and "man running past dog". This feature is lost when using both Word2Vec and GloVe by only averaging the word vectors in a sentence. In addition LSTM loses the main drawbacks of both the GloVe and Word2Vec models, by basing its knowledge solely on the words of the dataset and that it does not require as large an amount of data to train on as our Word2Vec model. The three combinations using VGG-19 is all performing well and the LSTM based method outperforms the other two. The reason for this could be as mentioned the ability to look at sequence of words and their placement in a sentence. The first retrieved image in Figure 5.4a is of a group of people climbing on rocks, which fits the query caption quite well. It is not a picture of a group of people where one single person is climbing, but rather multiple people climbing. A surprising result is that the combination of Inception and LSTM performs poorly. One reasoning might be that the LSTM based models are more dependent on a specific representation of images and based on the results in Figure 5.1 we can see the overall worse performance of Inception based models. However the full reasoning for these results still remains somewhat uncertain.

5.3.3 Evaluation method

This way of evaluating the model provides lacking insight to the general performance of the system, and rather evaluates the system based on how well it handles detailed captions. Since our evaluation only cares about the ranking of the image, evaluating the result of queries that have a high amount of similar images like "A dog plays in the grass", provide randomness in that a high amount of images correspond to this description. The image corresponding to the query will be ranked randomly together with the images of dogs play-

(a) A dog plays in the grass

(b) A brown dog with a collar runs in the dead grass with his tongue hanging out to the side

Figure 5.10: Two quite similar images depending on the query caption

ing in grass. However, with specific image queries like "A brown dog with a collar runs in the dead grass with his tongue hanging out to the side" have far less similar images to retrieve. In Figure 5.10 this difference is illustrated. Both pictures can be represented with the caption in Figure 5.10a, while only the more specific caption suits the picture in Figure 5.10b.

A central issue with this evaluation method is that it only considers the ranking of the single image that corresponds to the query. This is exposed in the low results showed in Table 5.1.

An improved evaluation would rather provide a score for each of the retrieved images. Instead of a binary relevance classification, each retrieved images can contain a degree of relevance. This could be done by looking at captions for the retrieved images to check the relevance or by manual annotating results.

Chapter 6

Conclusion

Our unfamiliarity with the continuous nature of the task, mapping sentence vectors to image vectors eventually led us in the direction of the regression based problem space. Initial architectures consisted of a simple 2-hidden-layer neural network using categorical cross-entropy as the loss function. However, we early found that we needed a new loss function, supporting our non-classification problem. Another challenge during our implementation phase was to learn the machine learning frameworks Tensorflow [19] and Keras [18], which we used to implement our end-to-end system. Since learning new technologies and frameworks was a large part of our project; a re-occurring challenge towards the end of the project was the lack of time to test different structures and topologies.

During this project we have created an end-to-end system which compares both images and text in a common visual feature space. While the functionality for doing text-to-image, text-to-text and image-to-text exist in the system, only experiments of the former is performed. We have implemented three different methods for mapping from a textual caption into a visual feature space, concluding that the best translation from textual features to visual features is via an LSTM based model, which learns a dense representation of an image caption. It was also apparent that the pre-trained VGG-19 network receives on average better results than the Inception model. We compared our model to a baseline model which had satisfactory results on text-to-image retrieval. Our model receives better results than the baseline model when both is trained and tested on the Flickr30k dataset. There are still many improvements that could be done to our system including both pre-processing of text and testing better topologies of the feedforward network, but also in the evaluation of the model. Some of these improvements are discussed in the final chapter of this report.

Chapter 7

Further work

There are multiple interesting improvements to look at regarding our work on comparing image and text in a mutual visual feature space. Better word embeddings, generating captions from visual features and improving performance by introducing negative examples in training data are interesting advancements that could be done to our model.

7.1 Extending word embeddings

In our work we have used two different word embedding models for preprocessing the training data, Word2Vec and GloVe. We decided to train our own Word2Vec model using our dataset, and to use a pre-trained GloVe dataset. The former would assure that word embeddings of words in the test data would exist, but could make it perform poor on test data and unseen words. GloVe on the other hand contains a large vocabulary, but words specific for our datasat or other languages would not be in this vocabulary. An interesting approach would be to combine two solutions like these. Using pre-trained word embeddings and to extend this pre-trained model with the vocabulary of our dataset. This would allow us to both exploit the good representation from a pre-trained model, but also have representations of new unseen words. An interesting use case for such a model would be to train or model on image-caption pairs containing captions with multiple languages to see if the model could become language-independent.

7.2 Negative examples

In order to improve results it would be interesting to look at the possibility to create negative examples. Our initial thought to achieve this consisted

of creating five new negative examples for every image based on the five captions of the most dissimilar image. This meant calculating the mean squared error or cosine similarity between all images and sorting the values to find the highest value, giving us $O(n^2)$ computational cost. On the Flikr30k dataset this would take too long, even with a multithread-based solution. In addition, many of the images would return the same image as the most dissimilar image, since the dataset contains images that share few features with the rest of the dataset. This introduces outliers that would be highly dissimilar to all the other data. To deal with these issues we instead chose to randomly pick 100 examples and basing our negative example on the most dissimilar image of that set. This solution provided some variation to the examples by focusing less on the outliers of the dataset. Unfortunately, we were not able to train a network with these negative examples, forcing us to leave the problem to focus on other tasks. It would however be interesting to look at this challenge in the future.

7.3 Data augmentation

To further improve results it could be interesting to look at data augmentation of both images and text. One problem with our dataset is that it is rather small and does not contain an evenly distribution of different images. The dataset does for instance contain many images of dogs, while it can contain very few images of fish. This causing our system to become good at retrieving dog images, while it would perform poorly when giving it a textual query describing a fish. Since there are so few images of fish it could be interesting to increase the amount of training data using data augmentation. We could create augmentations of specific images by rotating, flipping, adding noise or other image manipulation methods. It would also be interesting doing this on captions to do operations like stemming, removal of stop words or exchanging words for synonyms.

7.4 Sentence generation

Instead of choosing the most suitable caption from a database of captions, it could be interesting to generate captions. Generating natural language sentences from visual features require its own models and conventional feed-forward networks fall short of current state-of-the-art models. Many primitive models like rule-based or logic systems have been proposed, but have been found to be narrow in its use and does not convert to different domains.

Newer RNNs have recently had success in generating sequences in machine translation using their internal memory. An implementation of LSTM [3] networks have been shown to provide state-of-the-art performance in translation [22; 23] and sequence generation [24].

Work by Vinyals et. al [25] propose a joint-model approach to generating image captions. By using a CNN and a LSTM to maximise the likelihood of generating a sequence of words that accurately describe the image. The CNN used is the winner of ILSVRC 2014 classification problem [26]. and provides state-of-the-art image classification. The output of the last hidden layer in the CNN produces vector representations of the images fed into the network and is used as input to the LSTM. The authors call this model the Neural Image Caption (NIC). A series of LSTMs with shared parameters create a sequence of words with a special start and stop word to respectively signal the first and last word in the sentence.

7.5 Entity recognition

An interesting concept to look at in the future is entity recognition. Given an image, extract relevant entities from this image. These entities could then for instance be used to return information from an knowlegde base such as an Wikipedia article [10]. The dataset being used in this project contains five captions for each image, this allowing a model to learn at least five interpretations of each image. This entity recognition could then be used to to entity linking, what are related entities given a specific entity, which could take advantage of existing knowledge bases like Wikipedia and DBPedia [27].

References

[1] Aphex34, "CC BY-SA 4.0 (http://creativecommons.org/licenses/by-sa/4.0)."

[2] J. Dong, X. Li, and C. G. M. Snoek, "Word2VisualVec: Cross-Media Retrieval by Visual Feature Prediction," 2016.

[3] S. Hochreiter, S. Hochreiter, J. Schmidhuber, and J. Schmidhuber, "Long short-term memory." *Neural computation*, vol. 9, no. 8, pp. 1735–80, 1997.

[4] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," *arXiv*, pp. 1–9, 2014.

[5] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June, 2015, pp. 1–9.

[6] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "10.1162/Jmlr.2003.3.4-5.951," *CrossRef Listing of Deleted DOIs*, vol. 1, pp. 1–9, 2000.

[7] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global Vectors for Word Representation," *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 1532–1543, 2014.

[8] S. Kottur, R. Vedantam, J. M. F. Moura, and D. Parikh, "Visual Word2Vec (vis-w2v): Learning Visually Grounded Word Embeddings Using Abstract Scenes," *arXiv preprint*, pp. 4985–4994, 2015.

[9] A. Frome, G. Corrado, and J. Shlens, "Devise: A deep visual-semantic embedding model," *Advances in Neural ...*, pp. 1–11, 2013.

[10] "Wikipedia." [Online]. Available: https://www.wikipedia.org/

[11] "ImageNet." [Online]. Available: http://image-net.org/

[12] M. Norouzi, T. Mikolov, S. Bengio, Y. Singer, and L. G. Mar, "Zero-Shot Learning by Convex Combination of Semantic Embeddings," *Bmvc2015*, pp. 1–9, 2015.

[13] "Flickr." [Online]. Available: https://www.flickr.com/

[14] C. Lynch, K. Aryafar, and J. Attenberg, "Images Don't Lie: Transferring Deep Visual Semantic Features to Large-Scale Multimodal Learning to Rank," in *KDD*, 2015, pp. 1–9.

[15] W. Liu, T. Mei, Y. Zhang, C. Che, and J. Luo, "Multi-task deep visual-semantic embedding for video thumbnail selection," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June, 2015, pp. 3707–3715.

[16] I. Vendrov, R. Kiros, S. Fidler, and R. Urtasun, "Order-Embeddings of Images and Language," *arXiv preprint*, no. 2005, pp. 1–13, 2015.

[17] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *ImageNet Challenge*, pp. 1–10, 2014.

[18] "Keras Documentation." [Online]. Available: https://keras.io/

[19] "TensorFlow – an Open Source Software Library for Machine Intelligence." [Online]. Available: https://www.tensorflow.org/

[20] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 2006, pp. 1735–1742.

[21] P. Young, A. Lai, M. Hodosh, and J. Hockenmaier, "From Image Descriptions to Visual Denotations: New Similarity Metrics for Semantic Inference over Event Descriptions," *Transactions of the Association for Computational Linguistics (TACL)*, vol. 2, no. April, pp. 67–78, 2014.

[22] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. 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)*, pp. 1724–1734, 2014.

[23] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," *Nips*, p. 9, 2014.

[24] A. Graves, "Generating Sequences with Recurrent Neural Networks," *Technical Reports*, pp. 1–43, 2013.

[25] J. Long, E. Shelhamer, O. Vinyals, A. Toshev, S. Bengio, D. Erhan, K. Lenc, A. Vedaldi, E. Denton, S. Chintala, A. Szlam, R. Fergus, P. Fischer, H. Philip, C. Hazırbas, P. V. D. Smagt, D. Cremers, T. Brox, F. Meng, Z. Lu, Z. Tu, H. Li, Q. Liu, V. Mahadevan, and S. Member, "Show and Tell: A Neural Image Caption Generator," *arXiv*, vol. 32, no. 1, pp. 1–10, 2014.

[26] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *arXiv:1502.03167*, pp. 1–11, 2015.

[27] "DBpedia." [Online]. Available: http://wiki.dbpedia.org/