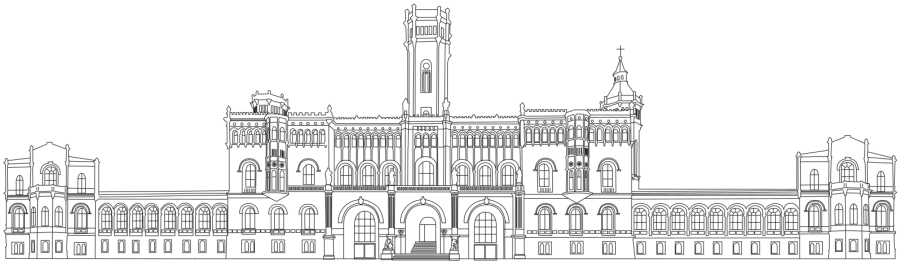
Estimating the Information Gap between Textual and Graphical Representations

Christian Henning
Institut für Verteilte Systeme – Fachgebiet Visual Analytics
Leibniz Universität Hannover



First Examiner: Prof. Dr. Ralph Ewerth
Second Examiner: Prof. Dr. Eirini Ntoutsi
Supervised By: Prof. Dr. Ralph Ewerth

A Thesis submitted for the Degree of
Master of Computer Science

6th January, 2017

Erklärung

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel und Quellen genutzt habe.

Hannover, den 06. Januar 2017

Christian Henning

Contents

1. Introduction

In recent years, research from the fields of Natural Language Processing and Computer Vision have been combined to tackle new exciting challenges, such as automatic image captioning and multimodal document retrieval. Increasing computational power allows to construct and train incredibly complex neural networks in terms of deepness and recurrency. This has led to fascinating results in many areas, especially for the task of scene comprehension in combination with language modeling.

But there are several pitfalls to current approaches, that have not been faced yet. For instance, many of these works rely on large, human-annotated data sets, which usually describe what is visually obvious and do not relate to the context of the appearing text/image. On the other hand, unsupervised approaches rely on the assumption, that co-occurring entities have a high semantic correlation, implying that they convey the exact same meaning. This master thesis aims to understand co-occurring entities drawn from multiple modalities (i.e. texts and images) and the context in which they appear in order to fully grasp their interrelation and exploit the information in which they complement one another. Moreover, our approach aims to mimic human judgment to quantify the complex hidden alignments, which are connecting elements from different modalities. Therefore, we invented a two-staged system combining an unsupervised and supervised learning regime to realize a framework that can tackle complex learning problems that only require a marginal portion of human supervision. The approach is based on conclusions drawn from observing the human learning process. Concepts are identified in massive amounts of data and generalized in order to create a compressed representation. This process happens fully unsupervised. With regard to the task of judging intermodal relations, such a dense compression learned via multiple modalities (a multimodal embedding) comprises readily usable alignments of abstract concepts. Hence, instead of aligning entities from complex input spaces (such as images), we only learn to quantify compressed concepts in the desired manner. This requires much less training data compared to supervised learning schemes that try to reveal sensible features from the redundantly encoded input modality.

Particularly, this thesis considers textual and single-framed graphical modalities, namely texts and images. We introduce a novel approach to rate the complex intermodal relation of co-occurring image-text pairs. We strongly believe that this relation can be fully described by two measurements, which we call *Mutual Information* and *Semantic Correlation*. Mutual information refers to the amount of shared knowledge by means of textually stated or visually portrayed facts, whereas semantic correlation describes the more subtle interpretation of their co-occurrence. The bridge that connects a text and an image might be vaguely defined when exclusively considering a document that comprises these two modalities. However, background knowledge might close this gap and enable our ability to capitalize their complementary information.

Motivated by recent success in deep learning, we would like to go a step further and model human-near capabilities for the task of multimodal document comprehension. Therefore, we build on recently developed models for the tasks of object recognition and language modeling. Our model maps multimodal documents onto a mutual embedding space, that

should be interpreted as a machine readable document representation. We use the popular *InceptionV3* [60] model to encode images. The overall encoding architecture is based on the *Neural Image Caption Generator* from Google [64], except that our text encoding is hierarchical to allow the comprehension of sentences as entities that appear in the context of a variable-sized text. As mentioned above, the interdependence of co-occurring modalities is usually hard or impossible to grasp without the contribution of extensive background knowledge. We are facing this problem by training an autoencoder network over a large dataset before we train a prediction network for our handcrafted measurements.

To allow the autoencoder to generalize and cluster even abstract concepts, we combined multiple datasets in order to obtain a wide variety of distinctly perceived intermodal relations. Part of the dataset is drawn from encyclopedia articles to encode properties and entities from the world (general knowledge). Therefore, articles have been retrieved via an implemented Web crawler. Samples from two other, already existing datasets have been taken to enrich the newly generated dataset, a news corpora and a human generated image-caption dataset. News articles shall contribute rather complex intermodal relations to our overall dataset. The relations of samples taken from an image-caption dataset are quite explicit and uniform, as the captions state visual obvious content. The broad range of document sources shall allow the system to properly learn how the two considered modalities are used by humans to convey information.

Having a dense feature representation of multimodal documents, a classifier is trained to predict *Mutual Information* and *Semantic Correlation*. To accomplish this task, we invented an annotation scheme for both measurements and annotated over 750 samples from two different datasets. In addition, 100 samples have been heuristically annotated to reduce label imbalance.

We accomplished results that are much better than random predictions, setting an initial baseline for future work in this field.

There is a wide variety of applications that can be directly realized from a system that is similarly well suited as humans when judging intermodal relations. For instance, sentences describing an enclosed image could be extracted from a text. Potentially leading to automatically generated data copora that are large enough to allow further improvements on image captioning tasks or to utilize these systems to generate descriptions for information graphics. Moreover, most of the developed methodologies can be easily transfered to other modality pairs (e.g. text and video).

This thesis will be structured as follows. In chapter 2 we are going to present related work. Chapter 3 gives an overview over the methods and formalism used to accomplish our goal and justifies the usage of deep learning models. It will also give a detailed introduction into neural networks with all principles necessary to understand this thesis. Subsequently, in chapter 4 we will explain our ideas and intuitions. The chapter will comprise a detailed overview of the utilized datasets and will elaborately outline the chosen labels and the annotation process followed by the presentation of the architectures we used to build our deep neural networks. We will present the conducted experiments and their outcomes in chapter 5. Afterwards, we will outline a subset of possible applications, that can be derived from our research in chapter 6 and finally close this thesis in chapter 7 with some concluding remarks and proposals for future work in this field.

2. Related Work

To the best of our knowledge, there has been no attempt yet to directly investigate the information discrepancy between textual and graphical representations in an approach similar to ours. However, there have been plenty of closely related studies, that we are going to briefly outline in this chapter.

2.1. Language Processing combined with Computer Vision

Tremendous success has been achieved in recent years when it comes to the tasks of image-sentence retrieval and image description generation. The availability of extensive annotated datasets as well as the immense progress in computational power have led to several approaches in facing those problems. Kulkarni et al. [40] are generating image descriptions by using a CRF to maximize the score of a labeled graph. These labels include scores for detected objects, object attributes and spatial relations between objects. Given a labeling that maximizes the score, they generate a description either template-based or based on a language model. Such an approach relies heavily on the underlying systems for object, object-attribute and spatial-relation detection. Yet, there have been substantial improvements over recent years in this fields, even meeting near-human capabilities on certain datasets [18, 39]. Despite their promising results, we believe that their architecture has a structural drawback that will prevent such systems from reaching human capabilities. Due to the complete decoupling of generated text and original image, that is because of the intermediate stage of generating a labeled graph with human-picked labels, there is no way to express semantics in the final description, which are shown in the image but cannot be encoded in the graph labels.

Vinyals et al. [64] generate image captions by transforming an image in a compact representation via Convolutional Neural Networks (CNN) and then using a Recurrent Neural Network (RNN), conditioned on the image and previous predicted words, to produce sentences. Their system is trained in an end-to-end fashion, such that any detail resp. context could be revealed by the hidden structure. We think, that deep learning methods are the key to success in those learning problems on the edge between CV and NLP, which we try to convey on section 3.3.

One interesting idea is to think of the syntactics and semantics of images resp. texts as laying in a hidden latent space, where both representations can be projected onto [21, 23, 33, 34, 45, 48, 64, 67]. There are several elaborated methods that do not directly incorporate neural networks when defining the projection. Gong et al. [23] lists some of them. While they gain their feature vectors through neural nets, they still provide the transformation rule, whose parameters have to be learned. The same applies to the nonlinear version of Latent Semantic Analysis (LSA) introduced by Liu and Tang [45] (though, their results arise from handcrafted features). Again, we do not think it is constructive to hold on to a specific kind of transformation for the same reason as we dismiss hand-engineered features. The transformation rule from the graphical resp. textual representation space

onto the latent embedding space is simply unknown[1]. Therefore, it seems to be wise to make no assumptions about how the transformation might look like, since neural nets can theoretically approximate any nonlinear mapping [30].

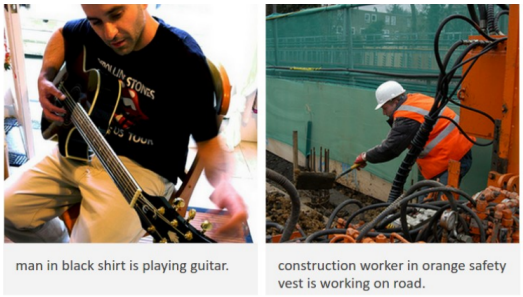
The approach which proofed to be most promising is presented in its basic form by Mao et al. [48]. They project images via a CNN and sentences via a RNN onto a multimodal embedding space. Their system is trained by predicting the probability of the next word in an image description given the image and the previous words. Frome et al. [21] make use of large text corpora by pretraining context-sensible word embeddings and projecting images onto this embedding space. This does not just allow sensible results even if the prediction does not match the ground truth, but also enables zero-shot-learning, thus identifying objects not appearing in the image-sentence training corpus. A more fine-grained approach has been recently proposed by Karpathy and Li [33] and Karpathy et al. [34]. They intentionally assimilate decompositions of their representations (in addition to the pure input) and ensure that those match up in the embedding space as well. They consider either dependency tree relations [34] or bidirectional neural networks (to better model the context of words) [33] as decompositions of sentences and object detections as decompositions for images. Their key insight is, that text snippets usually refer to a specific region of an image. Similar to this approach is the work by Yan and Mikolajczyk [67]. They extracted image resp. text features through deep neural networks and aligned those vectors through an objective. Image features were extracted using AlexNet [39] and text features were generated from TF-IDF histograms using a deep neural network with fully-connected layers. They use Canonical Correlation Analysis (CCA) to align the embeddings. To overcome shortcomings from prior work, that suffered from small feature vectors due to computational issues, they have constructed an efficient GPU implementation, that allows feature vectors that are two orders of magnitude higher than prior work (specifically, they used feature vectors of size 4096).

A general advantage of multimodal embeddings is that they can be used in a number of application, e.g. for image-sentence retrieval tasks by using ranking algorithms or for text generation by training a network above the embedding space. Ngiam et al. [52] show that multimodal embeddings learned via autoencoders can even enhance results on tasks that do not obviously incorporate more than one modality. For instance, for a task such as speech recognition, they showed that adding additional features learned over several modalities can improve the outcome compared to audio-only features.

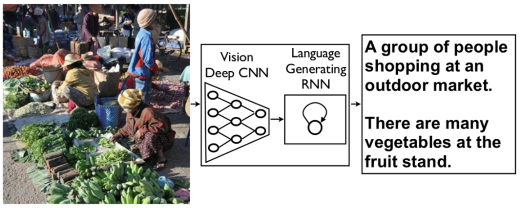
Figure 2.1 depicts the current state-of-the-art for image caption generation systems. Both systems make use of a multimodal embedding space.

Another common technique to confront problems comprising multiple modalities are probabilistic models to estimate a generative or discriminative model. While these techniques have not been able to keep up with neural networks in tasks that require to learn a complex but hidden structure, such as language models [38], they should still be considered for tasks with direct alignments, such as image tagging, possibly by incorporating neural feature extraction. Barnard et al. [3] are mainly examining the problems of tagging an image with highly related words and of finding direct correspondences of visual and textual words. Although their results are outperformed compared to recent deep learning systems (e.g. [33, 67]), they give a comprehensive and well-arranged overview over various probabilistic methods to tackle these problems. Fang et al. [17] implemented a caption-generation

[1] assuming that such a hidden latent space exists



(a) Captions generated with the tool *NeuralTalk2* developed by Karpathy and Li [33] (image source: [33])



(b) Captions generated with the *NIC (Neural Image Caption)* model developed by Vinyals et al. [64] (image source: [64])

Figure 2.1.: Examples that illustrate the capabilities of current state-of-the-art-systems for automatic image caption generation. Both systems make use of multimodal embeddings in a deep neural network architecture.

system that was state of the art by many metrics at the time of writing[2] and does utilize a probabilistic language model. However, this language model does involve detections of words with a high likelihood of being associated with the image. These detection have been computed using a deep CNN. Additionally, generated captions are ranked based on a similarity measure that is computed on a neural multimodal embedding space. Nevertheless, the work proves that there are alternatives to pure neural architectures that are worth considering.

Yet another way to convey information compactly through a graphical manner is by using tables. There is an interesting system designed by Bin Shao [5], which aims to understand tables by exploiting a huge knowledge base, i.e. Probase [29]. This knowledge base is structured into concepts (e.g. countries), entities (e.g. Germany) and attributes (e.g. area, population, ...). The first step in understanding is to detect or generate a header row and then subsequently identify the column containing the entities. They use the gained information to build a search engine, which generates statements from tables given a query. It is further used to enrich the taxonomy of Probase.

Other graphical representations should not go unmentioned, such as charts, plots or maps, as they are crucial in the transmission of knowledge, since they can express information in a succinct and compact way. There is a variety of work that focuses on the understanding of information graphics [12, 16, 50]. Regrettably, no recent work in this area is known to us, especially no studies that incorporate the use of deep learning methods. This is probably owed to the fact that there are no adequate datasets available. The works that we are aware of are heavily based on assumptions, restrictions and templates and are not comparable to the performance of current image caption generation systems. Another pitfall that has not been tackled yet is that information graphics are usually described in an accompanied text and cannot be considered as individual self-explanatory entities.

Up until this point, most of the cited work examines the generation of text given a graphical representation. On the other hand, the generation of images from a textual description is a key element of human imagination and human fantasy and therewith essential for

[2] Interesting is a comparison that has been made by 250 human raters. 34% of the generated captions have been rated as better or equally good compared to human-generated reference sentences.

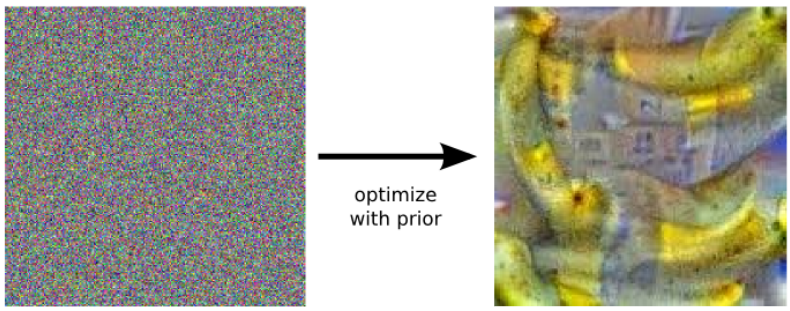


Figure 2.2.: Illustrating what a deep neural network learns by operating it in reverse mode [51].

knowledge inference and understanding. A base in this area has been laid by the work of Zitnick et al. [71]. They learn the semantic meaning of sentences in order to generate scenes from a corpus of 80 different cliparts with 58 different objects. They model the problem as a CRF with objects as nodes and relations as edges. The system still has some major restrictions. They only consider scenes of children playing outside and attributes are only modeled for humans. Nevertheless, the results are encouraging that future improvements may overcome the current limitations. Though, deep neural network architectures may be again the better choice to accomplish a system that automatically produces images from a textual description without any constraints. The reason why we think so is illustrated in an article by Mordvintsev et al. [51]. The article illustrates what deep neural networks in object detection systems learn. An example from the article is depicted in figure 2.2. It shows what happens, when an object detection system is operated in reverse mode. Instead of inputting an image, they input the keyword *banana* and observe how the network transforms an image full of random noise. The outcome shows that the network processes visual words similar as we do by having a conceptual notion of that object. Such outcomes encourage to dream of future neural networks that may generate whole virtual realities without human interference.

Being able to draw conclusions about the stated information discrepancy involves the possession of meaningful automatic evaluation methods. It appears to be inherently complicated to directly compare graphical with textual representation without incorporating human judgment. Most measurements are bounded to specific applications and withdrawn from similar tasks.

An often used method in automatically evaluating image captions is BLEU [53], which originates from machine translation. BLEU compares a generated image caption with a reference caption based on n-gram occurrences. Since there is a much broader variety of sentences to characterize an image than when translating a sentence, this method has proofed to be insufficient when evaluating image captions. There is a range of other methods being used in image description evaluation (some of them are summarized in [10]), with just a few of them specifically designed for the task of image description generation (e.g. CIDEr). But it still remains an unsolved problem to find an evaluation method that highly correlates with human judgment.

A more intuitive measurement can be applied in case of image-sentence retrieval tasks, namely *recall@k* [21, 33], which counts how often the correct sample was among the top-k

ranked results. Still, such a method cannot express the quality of the other top ranked solutions[3]. Ma et al. [47] provides a comprehensive comparison to many recently proposed state-of-the-art systems in the field of image-sentence retrieval. Their work outperforms many of the comparative systems and has an interesting way to match both modalities. A CNN (e.g. the object detector from [58]) is used to encode images into feature vectors. Multiple CNNs are then used to match these image embeddings with semantic language embeddings on several fragment levels (e.g. word, phrase or sentence level). The output scores are subsequently combined by a fully-connected layer that computes a final ranking score.

When it comes to the measurements *mutual information* and *semantic correlation*, that we mentioned in chapter 1, there is a mixture of works that addresses similar ways of looking at this problem. We will devote an extra section to this point, where we outline why we consider them as insufficient for the specific task we face.

2.2. Evaluate Shared Information and Semantics of Multimodal Documents

While we think, it is easier to quantify the relation of several modalities by two measures, other works, that we are aware of, aim to wrap the relation into a single score.

Barnard and Yanai [2] are using ratings from information theory, such as entropy and mutual information (cp. chapter 4), to measure the relation of words to images resp. image regions. The ratings are estimated using a probability model. They demonstrated the effectiveness of their model by pruning large vocabularies to those words, that are considered as *visual*. They have taken on the same issue in [68]. Here, they directly want to estimate the *visualness* of adjectives. For instance, the word *yellow* is considered as more visual than the word *religious*, simply because the attribute *yellow* is easily inferable and by far less ambitious compared to the attribute *religious*. Therefore, they measure the *image region entropy* for a concept (adjective), which is computed as the entropy of features extracted from an image region weighted by the probability that the concept belongs to that region. These probabilities themselves are estimated by using the expectation-maximization algorithm. The used images, that are tagged with a certain adjective, are gathered via Google image search. Hence, their iterative algorithm itself is carefully designed to deal with noisy images. However, aside from apparent visual words it is unlikely that a model like this can reliably process concepts, which are considered as non or barely visual, as the model does not incorporate a broader notion of comprehension. For instance, a proper detection of the concept *religious* in images requires a generalization of image features as it is infeasible to achieve due to reasonable datasets.

There have been some attempts to model semantic correlation. Zhuang et al. [70] tries to model semantic correlation in a graph-based approach, where vertices represent documents and edges represent their semantic correlation. However, these edge distances are computed based on the distance of low-level feature vectors for documents from the same modality. Documents appearing together in the dataset (such as an article with an enclosed image) are modeled by a constant distance[4]. Even if such an approach leads to improvements in

[3] Except for the case, that the semantic hierarchy of the test data is known, such as in the case of ImageNet [14].

[4] For the datasets used in this work, we can state, that co-occurrence does not imply a strong semantic correlation. Though, a positive correlation usually exists.

certain tasks, it is questionable if semantic correlation can be expressed by this model. In addition, negative correlation is not modeled at all by this approach. Another approach to model semantic correlation has been presented by Zhang et al. [69]. They map a term count vector x on a set of latent topics, via $x = Az$ (e. g. solvable with PCA). Their main incentive is, that while x and z are document dependent, A is invariant for a given language. So they define the semantic correlation of words by solely incorporating coefficients from A. This word correlation can be used to imply the semantic correlation of documents. Even though, some clues for the semantic correlation of documents are typically given by certain keywords, something as ambiguous as natural language can not be comprehended without knowledge of its syntactic structure or the context in which the document appears in[5]. Yet, their simple BoW approach has been proven quite successful when using the learned correlation to enhance a learning algorithm by background knowledge from unlabeled data. Viji [62] tried to model correlation of documents by the dot product of their feature vectors (this work used only term-based features). This approach has the same drawbacks as the previous one, due to the lack of using more expressible features (in addition, the way features are selected in this work, causes the correlation to be always positive). However, the author mentions that a proper correlation estimation might not be possible without language parsing. Xue et al. [66] try to estimate semantic correlation in an approach that aligns the semantics of visual and textual blobs (local image regions and words), which they call local-media objects (LMOs). To estimate the probability $p(b_j, d_i)$ of a LMO b_j co-occurring with a document d_i, they use probabilistic latent semantic analysis (PLSA). The major pitfall of their approach (in the context of our work) is, that they assume that the distribution of latent topics z_k is the same for both modalities ($z_k^{\mathcal{T}}$ – textual topic, $z_k^{\mathcal{I}}$ – visual topic), thus $p(z_k^{\mathcal{T}}, d_i^{\mathcal{T}}) = p(z_k^{\mathcal{I}}, d_i^{\mathcal{I}})$ for co-occurring image-text pairs. Once again, this assumption typically does not hold, as we will further discuss in chapter 4. Instead, the modalities co-occur to show distinct information and complement one another. They propose a graph-based approach to align textual and visual blobs. The depicted results are promising, but not comparable to our work, as the utilized dataset consists of images tagged with highly correlated words, such that the above criticized assumption holds.

[5]Note, that x can actually be replaced by any feature vector, which better describes the content and structure of a document than a simple bag of words.

3. Fundamentals

In the following sections we intend to convey knowledge from the field of machine learning that is necessary to thoughtfully grasp the ideas presented in the remaining chapters. After we agreed on basic terms and sketched some general but elementary concepts of artificial intelligence, we succinctly introduce the reader to the methods and algorithms of neural computations and provide references for further reading.

3.1. Natural Language Processing

In clear contrast to all other living beings, humans use sophisticated language to communicate, which enables us not just to efficiently exchange or express information, but moreover allows us to reason about the world and its entities. This at least raises the question, whether language is the key to what we intuitively conceive as *intelligence*. Even though we cannot evidentially answer this question, we can yet clearly state that no program will be considered as intelligent if it does not provide an intuitive and universal interface. An obvious choice for such an interface would be a natural language interface.

This is where Natural Language Processing (NLP) comes into play. NLP is the field of computer science that is concerned about the computer-aided processing of human (or natural) language. This does not necessarily imply that NLP is a part of the discipline of artificial intelligence. For instance, a simple template-based approach to extract calendar events from emails would be as much an NLP task as a language-capable program that would pass the Turing test. Anyhow, as human language is inherently complex, most modern NLP tasks comprise the usage of methods that originate from machine learning algorithms. Unlike formal languages, it is infeasible to list a complete grammar for something as ambiguous and as quickly developing as, for instance, the English language. An additional aggravating factor is that each individual uses tiny perturbations when forming resp. intonating words. Collectively, this emphasizes that it is practically impossible to design a program that processes natural language with human-capabilities based on a case-by-case analysis.

One major difficulty to overcome when applying machine learning techniques on natural language problems is the finding of a proper feature extraction, meaning a vectorial representation of a string. The challenge arises from the complex and unknown structure that forms the syntax of a sentence or text and from a lack of modeling methods to define the underlying semantics. Many solutions have been proposed and examined to resolve structural issues. One easy attempt is to assume that the underlying structure forms a sequence and can be satisfied with Hidden Markov Models. But there are also modeling methods that can cope with tree structures or unconstrained graphs such as Conditional Random Fields. Yet another way to deal with structural concerns is to encode structural properties into the feature vector. This might happen in the form of short phrases (such as n-grams), whose possibilities are enumerated along the feature vector, or as triple encoded dependency parse or constituency trees, which aim to appropriately reveal the syntactic structure of a sentence.

As there seems to be a significant relationship between syntax and semantics, that is inevitable to disambiguate words, some features intend to represent both. One prominent example is Semantic Role Labeling, that has the goal to identify the predicate and its arguments in a clause. All these features have in common, that humans handcrafted them and therewith decided what is necessary for a learning algorithm to grasp the meaning of a sentence, whereas the original text got lost or became hidden among tons of additional information. Another obstacle lays in the computation of these features. Most of them require schooled knowledge plus experience for their extraction. Therefore, many of them are using machine learning algorithms as well, which leads to features that are only as reliable as the performance of the underlying extraction algorithm.

Although, the above described *conventional* NLP approaches have been incredibly successful for many tasks (e.g. POS-tagging [61] or Chunking [57]), even reaching inter-annotator-agreement rates [4], the performance of other tasks seems to be limited due to the stated issues (e.g. Paragraph Detection [25]). We will present ways to overcome these limitations in section 3.3.4.

3.2. Computer Vision

Computer Vision (CV) is a field of AI that is focusing on the development of algorithms that can elicit semantic interpretations from images. Anyhow, to be as exact as in the previous section, CV is not necessarily part of AI. But as it aims to mimic the human visual system, many CV tasks are unavoidably dependent on machine learning methods. Non-neural network approaches comprise the utilization of sophisticated feature extraction to reduce the complexity of an input image and to overcome the data sparsity problem. One prominent feature extraction method for CV is Scale Invariant Feature Transform (SIFT) [46]. This method allows to extract local features, that are translation-, rotation- and scale invariant and possess a high robustness against noise, illumination and background cluttering. However, utilizing such methods leads to a discard of a bulk of information, that may be insignificant for the most part, but is prohibiting the concentration of details that may shape a relevant context, which arises from an interplay with background knowledge not inferable from the image. This consideration leads us to the conclusion, that systems with human capabilities must have at least the same informational input as humans have.

Prominent applications of CV are object detection and image matching/retrieval, to name just a few. Many interesting novel and future applications of CV originate from collaborations with other fields, such as robotics or NLP. Other promising future perspectives of CV are including combined applications with augmented reality. Capturing such a broad scope does require solid performance on basic CV tasks. In the following, we will learn the necessary skills to tackle such challenges.

3.3. Learning in Deep Neural Network Architectures

Primarily, we talked about limitations of NLP and CV so far. Therewith, we have been keeping quiet about the enormous progress that has been taking place in recent years. A major role in this progress has been held by deep learning methods. Therefore, the rest of this chapter will be devoted to deep learning, its fundamentals and application.

We start with the very basics of machine learning and give an overview on how an algorithm might look like, that can automatically learn patterns in its input data in order

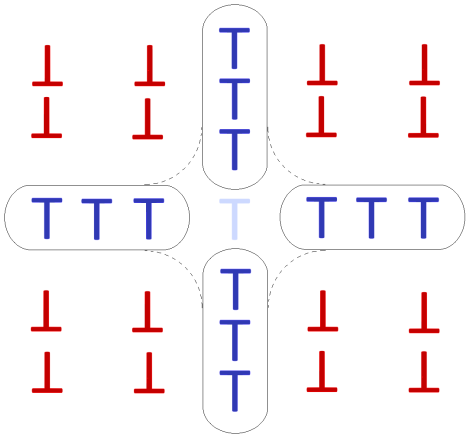


Figure 3.1.: Overfitting occurs when the learned model adapts too much to the training data set and does not generalize well. In the given example, positive training samples are blue and negative training samples are red. Test samples are transparent. The continuous boundary shows the learned model, whereas the dashed boundary marks a preferred decision boundary, that would better generalize on unseen data and not adapt too much to the training samples.

to correctly predict an output.

What holds true for all learning algorithms is, that they conduct learning on an appropriate data source. This dataset is typically divided into three parts: training, validation and test set. Each sample from the training set is processed by the learning algorithm to extract patterns from its feature representation[1] in order to optimize a predefined goal depending on its input and output. Sometimes, a smaller validation set is held back to fine-tune hyperparameters before starting the training process. One major learning goal is to generalize the training data. If the learned model adapts too much to the training data, it is at risk to perform poorly on unseen data. This problem is called overfitting and it is sketched in figure 3.1. To prevent overfitting, various regularization methods can be applied, which basically shall perturb the optimization goal to avoid a too restrictive prediction model. To assess the performance of the learned model the test set, that consists solely on samples not available in the training set, is used. This allows to evaluate whether the model can generalize on previously unseen samples. If the training data is processed as a single batch, the learning process is called batch learning. On the other hand, if the training does not terminate but waits continuously for new samples to update its internal weights, then the learning process is called online learning.

Another important distinction is to be made between supervised and unsupervised learning algorithms. Supervised learning requires the availability of an annotated dataset, meaning that each sample is labeled with the desired outcome. For instance, such a labeling might be the objects appearing in a sample image. Unsupervised learning schemes do not require labeled input data. In this case, patterns have to be inferred from the distribution of the input data or labels have to be automatically derived from the input as it is the case for an autoencoder, that predicts its own input. Sometimes, another distinction for a third case, named weak supervision, is made. In that instance, labels are generated based on heuristics

[1] Even if the raw data is directly encoded as input for the learning algorithm, without further preprocessing, we call this encoding feature vector.

[27].

Basically, there are two different types of learning algorithms when considering the shape of their outcome. Classification algorithms have a discrete set of possible outcomes, whereas regression algorithms predict an outcome from a continuous range.

Learning means, that we estimate parameters of a mathematical function. These parameters are typically called weights. If the weights shall fit a linear function, the algorithm is a linear learning algorithm, otherwise it is called nonlinear. Example 3.3.1 should convey the intuition how a simplistic learning algorithm might look like. The example will also introduce some of the fundamental notation used in this thesis.

Example 3.3.1. *This example considers a binary classification problem. The outcome is predicted using a supervised linear learning scheme. The dataset consists of annotated samples*

$$\mathbb{D} = \{(x_1^{raw}, y_1), (x_2^{raw}, y_2), \dots\} \quad , \tag{3.1}$$

where x_i^{raw} is the raw input to the algorithm (e.g. an image) and $y_i \in \{\top, \bot\}$ is the label (e.g. whether the image depicts an outdoor scene ($\top$) or not ($\bot$)). The algorithm expects a numeric vector as input, the features representing the sample. The feature extraction method φ maps a sample onto its feature representation.

$$\mathbf{x}_i \coloneqq \varphi(x_i^{raw}) \quad , \qquad \mathbf{x}_i \in \mathbb{R}^n \tag{3.2}$$

For the sake of simplicity, it is assumed that the samples are linearly separable in their feature space. Consequently, we can learn a hyperplane that separates the samples and therewith classify them according to the half space in which they lay. Such a hyperplane is fully defined by its normal vector $\mathbf{w}$ plus its distance from the origin b. To decide on which side of the plane a sample is located, the following equation can be utilized

$$d_i = \langle \mathbf{w}, \mathbf{x}_i \rangle + b \quad , \tag{3.3}$$

whereas d_i is proportional to the signed distance from the hyperplane. The output can then be interpreted as follows

$$\begin{aligned} \textit{if} \quad \langle \mathbf{w}, \mathbf{x}_i \rangle + b > 0 \quad \textit{then} \quad y^* = \top \quad , \\ \textit{else} \quad y^* = \bot \quad . \end{aligned} \tag{3.4}$$

The remaining question is, how the parameters $\mathbf{w}, b$ can be estimated, such that the defined hyperplane correctly separates the data. Therefore a training algorithm has to be constructed, that will learn appropriate weights. We denote $\mathbb{T}$ as training set and $\mathbb{E}$ as test set, such that $\mathbb{T} \cap \mathbb{E} = \emptyset$ and $\mathbb{T} \cup \mathbb{E} = \mathbb{D}$. Now, the training set can be used to sweep over all its samples and adjust the weighs until they fit $\mathbb{T}$. This is done by an error-driven update rule, which corrects the weights in case of a misclassification.

The learning scheme is sketched in algorithm 1. The $\hat{}$-operator, used in line 4 and line 5 to avoid the distinction of several cases, is defined as follows:

$$\hat{y} = \begin{cases} +1, & \textit{if } y = \top \\ -1, & \textit{otherwise} \end{cases} \tag{3.5}$$

If the condition in line 4 holds, then the dot product projects the sample onto the plane

$\begin{pmatrix}\mathbf{w}\\ b\end{pmatrix} = \begin{pmatrix}\mathbf{0}\\ 0\end{pmatrix}$; // Initialization of weights

/* Sweep *NUM_ITERATION* times over the training set */

for *epoch=0* **to** *NUM_ITERATION* **do**

  foreach $(\mathbf{x}, y)$ *in* $\mathbb{T}$ **do**

    /* If sample is misclassified */

    if $\hat{y}\langle \begin{pmatrix}\mathbf{w}\\ b\end{pmatrix}, \begin{pmatrix}\mathbf{x}\\ 1\end{pmatrix}\rangle < 0$ **then**

      $\begin{pmatrix}\mathbf{w}\\ b\end{pmatrix} = \begin{pmatrix}\mathbf{w}\\ b\end{pmatrix} + \hat{y}\begin{pmatrix}\mathbf{x}\\ 1\end{pmatrix}$; // Update rule

    end

  end

end

Algorithm 1: An example of a simplistic learning algorithm

normal in the wrong half-space. The term $\hat{y}\begin{pmatrix}\mathbf{x} & 1\end{pmatrix}^T$ *corrects this misclassification by a certain amount.*

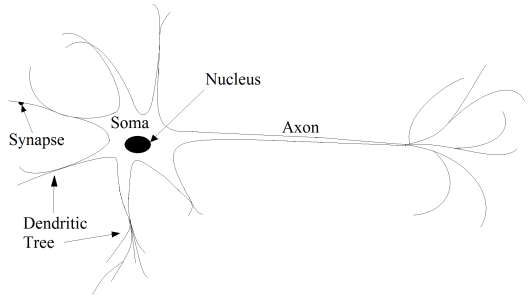
The algorithm presented in this example is called Perceptron and is one of the first machine learning algorithms that have been invented. It was published by Rosenblatt [56] over a half century ago. Perceptron is proven to eventually find a correct hyperplane, if the data is linearly separable.

The classifier presented in example 3.3.1 obviously has some pitfalls as well as in respect of dedication and efficiency of the training process and in terms of expressiveness. Further enhancements of the presented approach can resolve most of Perceptrons drawbacks. But also subtle algorithms, such as SVMs, have the disadvantage that they can only learn to linearly separate the data[2]. Therefore, non-linear approaches, especially neural networks, seem to be more promising for a great variety of tasks. Of particular interest in this context are deep neural networks. Deep neural networks consist of several neural layers (see subsection 3.3.1) and are therefore capable of learning hidden alignments or representations that are otherwise infeasible to envisage by ML engineers. LeCun et al. [44] are describing deep learning methods similarly as: "computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction". A major advantage of these models is, that their architecture perfectly suit the task of feature learning, which makes handcrafted features dispensable. This is favorably as humans can merely guess what features reveal a desired interpretation or meaning and what features are particularly beneficial to a certain learning algorithm or architecture. Their recent success, enabled by the increase of computational power, makes them inevitable in the field of machine learning and gives hope for a brighter future due to further boosts in available computing capacity.

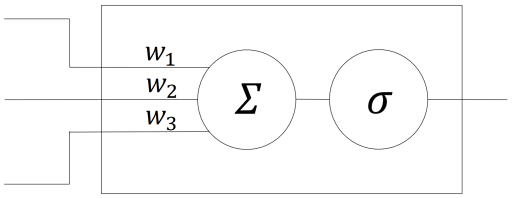
3.3.1. Fully-Connected Feed-forward Neural Networks

Artificial Neural Networks are a class of machine learning algorithms that are trying to mimic the behavior and the architecture of biological neural networks. A neuron is a special

[2] Though, the learned separation might be correct when predicting the training data, as the hyperplane can be learned in a higher dimensional space where the training data becomes linearly separable [6].



(a) Model of a biological neuron



(b) The Perceptron algorithm can be interpreted as a computational neuron

Figure 3.2.: A biological (a) and artificial (b) neuron

cell used in brains to transfer signals if certain preconditions are satisfied. In the human brain billions of neurons are interconnected via synapses to form the neural network.

A simplified model of a biological neuron is depicted in figure 3.2a. Dendrites (input) are connected with the axons (output) of other neurons. The accumulated excitement of the dendrites decides whether the axon of a neuron fires or not. There are certain analogies when considering the behavior of a single biological neuron compared to the Perceptron algorithm presented in example 3.3.1. Each element x_i of the feature vector can be seen as a stimulus and each weight w_i is a synapse that modifies x_i. Those individual excitements $w_i x_i$ are accumulate and mapped through a step function, that decides whether the neuron fires ($\top$) or not ($\bot$). Due to this analogy, Perceptron is a simple mathematical model of a biological neuron.

The most basic neural network architecture is a fully-connected feed-forward network with three layers: an input layer, a hidden layer and an output layer; as depicted in figure 3.3. The network is called fully-connected because each neuron accumulates as input the modified outputs of all neurons from the previous layer and its output is forward to all neurons in the next layer. A feed-forward network is not able to process or reflect its own output in contrast to recurrent neural networks (see subsection 3.3.3).

Each neuron performs a linear combination of its input and maps it through a non-linear function (e.g. equation 3.4). As we will see later, the learning process involves the computation of the gradients. Therefore, approximations of a step-function are typical used as non-linear activation functions. Exemplary activation functions are the sigmoid or hyperbolic tangent function. In this work, we chose the recently most popular, partial differentiable rectifier function as activation function for neurons in hidden layers:

$$f(x) = \max(0, x) \qquad . \tag{3.6}$$

To explain the mathematical background of neural networks, we are going to shortly introduce some notations. We assume a training set $\mathbb{T}$ with N annotated samples $(\mathbf{x}, \mathbf{y})$. The considered network contains L layers with m_{l-1} neurons in layer l. The aggregated output of all neurons of layer l is denoted by $\mathbf{z}_l$, which is computed by

$$\mathbf{z}_l = \boldsymbol{\sigma}_l(\mathbf{W}_l \mathbf{z}_{l-1} + \mathbf{b}_l) \qquad , \tag{3.7}$$

where $\boldsymbol{\sigma}_l$ is a function that applies the activation function σ_l element-wise on its input vector. Furthermore, it is defined that

$$\begin{aligned} \mathbf{z}_0 &= \mathbf{x} \\ \mathbf{z}_L &= \mathbf{y}^* \qquad . \end{aligned} \tag{3.8}$$

$\mathbf{W}_l$ and $\mathbf{b}_l$ are the parameters that have to be learned for each layer l. Each row in $\mathbf{W}_l \in \mathbb{R}^{m_l \times m_{l-1}}$ models the synaptic weights of one neuron in the layer, whereas $\mathbf{b}_l \in \mathbb{R}^{m_l}$ models the biases to enable the capability of handling unnormalized inputs.

Now, as the feed-forward step is fully defined, we can consider the network as computing a non-linear function (ensured by proper activation functions)

$$\mathbf{y}^* = \mathcal{F}(\mathbf{x}) \qquad . \tag{3.9}$$

Equation 3.9 does not hold for the training phase, where the parameters are not fixed yet. During training, the function computed by the neural network can be expressed by

$$\mathbf{y}^* = \mathcal{F}(\mathbf{x}, \mathbf{W}_1, \mathbf{b}_1, \ldots, \mathbf{W}_L, \mathbf{b}_L) \qquad . \tag{3.10}$$

For the sake of readability, we will omit the extra parameters of equation 3.10, if the context is clear[3].

To evaluate the computed output $\mathbf{y}^*$, a loss unit is stacked on top of the network, that compares the predicted output with the desired output $\mathbf{y}$. Popular choices for loss functions are the squared loss and the cross-entropy loss function. In the following, we will use superscripts to access elements of a vector or matrix. The mean squared error can be calculated as follows

$$L_{\text{MSE}}(\mathbf{y}, \mathbf{y}^*) = \frac{1}{2|\mathbf{y}|} \sum_{i=1}^{|\mathbf{y}|} (y^i - y^{*i})^2 \qquad . \tag{3.11}$$

To compute the cross-entropy loss, the outputs of the network have to be considered as probabilities, whereas $\mathbf{y}$ is an one-hot vector. For the case of multiple outputs, this can be reached by applying the softmax function to the network output:

$$\hat{\mathbf{y}}^* = f_{\text{softmax}}(\mathbf{y}^*) = \begin{pmatrix} \frac{e^{y^{*1}}}{\sum_i e^{y^{*i}}} \\ \vdots \end{pmatrix} \qquad . \tag{3.12}$$

For a binary classification problem ($y \in \{0, 1\}$) a bounded activation function can be used to map a single output onto the range $[0, 1]$ or two outputs can be used and normalized with the aid of equation 3.12. The binary cross-entropy loss function is defined as:

$$L_{\text{bCE}}(\hat{y}, \hat{y}^*) = -\hat{y} \log(\hat{y}^*) - (1 - \hat{y}) \log(1 - \hat{y}^*) \qquad . \tag{3.13}$$

For several outputs, the cross-entropy loss can be computed as

$$L_{\text{CE}}(\hat{\mathbf{y}}, \hat{\mathbf{y}}^*) = -\sum_{i=1}^{|\hat{\mathbf{y}}|} \hat{y}^i \log(\hat{y}^{*i}) \qquad . \tag{3.14}$$

[22] shows that the cross-entropy loss usually finds better local optima, whereas squared-

[3]Note, that $\mathbf{z}_l$ does actually depend on parameters as well: $\mathbf{z}_l(\mathbf{x}, \mathbf{W}_1, \mathbf{b}_1, \ldots, \mathbf{W}_l, \mathbf{b}_l)$.

error depends strongly on the weight initialization. Both functions are only example loss functions. One might use any loss function that states the optimization problem being faced during learning. We will denote the chosen differentiable loss function to evaluate a single sample as $L_s(\mathbf{y}, \mathbf{y}^*)$. Now, we present the back-propagation algorithm. For a more detailed explanation, please review [43]. The idea of this learning algorithm in neural networks is to back-propagate the error, computed by the loss function, through the network to update its weights. Therefore, the gradient of the loss with respect to a learnable weight is computed as it states the weights influence to the loss. The gradients can be computed by using the chain rule. If the gradient is computed for a subset of the training set $\mathbb{T}_b \subseteq \mathbb{T}$, then the overall loss function is defined as:

$$L(\mathbb{T}_b) = \frac{1}{|\mathbb{T}_b|} \sum_{(\mathbf{x}_s, \mathbf{y}_s) \in \mathbb{T}_b} L_s(\mathbf{y}_s, \mathcal{F}(\mathbf{x}_s)) \tag{3.15}$$

This setup allows us to compute the gradient of the total loss $L(\mathbb{T}_b)$ with respect to a weight w_l^{jk} resp. b_l^j, assuming the gradient of L_s with respect to $\mathbf{z}_L$ is known as well as the gradient of $\boldsymbol{\sigma}_l(\mathbf{v})$ with respect to $\mathbf{v}$ is known.

$$\begin{aligned}
\frac{\partial L(\mathbb{T}_b)}{\partial w_l^{jk}} &= \frac{1}{|\mathbb{T}_b|} \sum_{(\mathbf{x}_s, \mathbf{y}_s) \in \mathbb{T}_b} \frac{\partial}{\partial w_l^{jk}} L_s(\mathbf{y}_s, \mathbf{z}_L) \\
\frac{\partial L_s(\mathbf{y}_s, \mathbf{z}_L)}{\partial w_l^{jk}} &= \frac{\partial L_s(\mathbf{y}_s, \mathbf{z}_L)}{\partial \mathbf{z}_L} \frac{\partial \mathbf{z}_L}{\partial w_l^{jk}} \\
\frac{\partial \mathbf{z}_L}{\partial w_l^{jk}} &= \frac{\partial}{\partial w_l^{jk}} \boldsymbol{\sigma}_L(\mathbf{W}_L \mathbf{z}_{L-1} + \mathbf{b}_L) = \boldsymbol{\sigma}'_L(\mathbf{W}_L \mathbf{z}_{L-1} + \mathbf{b}_L) \mathbf{W}_L \frac{\partial}{\partial w_l^{jk}} \mathbf{z}_{L-1} \\
&\vdots \\
\frac{\partial \mathbf{z}_{l+1}}{\partial w_l^{jk}} &= \frac{\partial}{\partial w_l^{jk}} \boldsymbol{\sigma}_{l+1}(\mathbf{W}_{l+1} \mathbf{z}_l + \mathbf{b}_{l+1}) = \boldsymbol{\sigma}'_{l+1}(\mathbf{W}_{l+1} \mathbf{z}_l + \mathbf{b}_{l+1}) \mathbf{W}_{l+1} \frac{\partial}{\partial w_l^{jk}} \mathbf{z}_l \\
\frac{\partial \mathbf{z}_l}{\partial w_l^{jk}} &= \frac{\partial}{\partial w_l^{jk}} \boldsymbol{\sigma}_l(\mathbf{W}_l \mathbf{z}_{l-1} + \mathbf{b}_l) = \boldsymbol{\sigma}'_l(\mathbf{W}_l \mathbf{z}_{l-1} + \mathbf{b}_l) \frac{\partial \mathbf{W}_l}{\partial w_l^{jk}} \mathbf{z}_{l-1} \\
&= \boldsymbol{\sigma}'_l(\mathbf{W}_l \mathbf{z}_{l-1} + \mathbf{b}_l) \begin{bmatrix} 0 & \cdots & z_{l-1}^k & \cdots & 0 \end{bmatrix}^T
\end{aligned} \tag{3.16}$$

Note, that $\boldsymbol{\sigma}'_l(\mathbf{v}) = \frac{\partial \boldsymbol{\sigma}_l(\mathbf{v})}{\partial \mathbf{v}} \in \mathbb{R}^{m_l \times m_l}$ is a diagonal matrix as $\boldsymbol{\sigma}_l$ operates the activation function element-wise on $\mathbf{v}$. The calculations to compute the gradient of $L(\mathbb{T}_b)$ with respect to b_l^j are similar to those depicted in equation 3.16. It is worth noticing, that the calculations to compute the gradient in equation 3.16 with respect to another weight $w_l^{j^*k^*}$ with $j^* \neq j$ and $k^* \neq k$, are identical except for the last line. This should be exploited when implementing the algorithm. Example 3.3.2 presents the gradient computation on a sample network.

Example 3.3.2. *This example shows the gradient computation as highlighted in equation 3.16 for the fully-connected feed-forward network depicted in figure 3.3. The sigmoid function is used for σ_l in all layers:*

$$\sigma_l(v) = \frac{1}{1 + e^{-v}} \qquad \sigma'_l(v) = \sigma_l(v)(1 - \sigma_l(v)) \qquad . \tag{3.17}$$

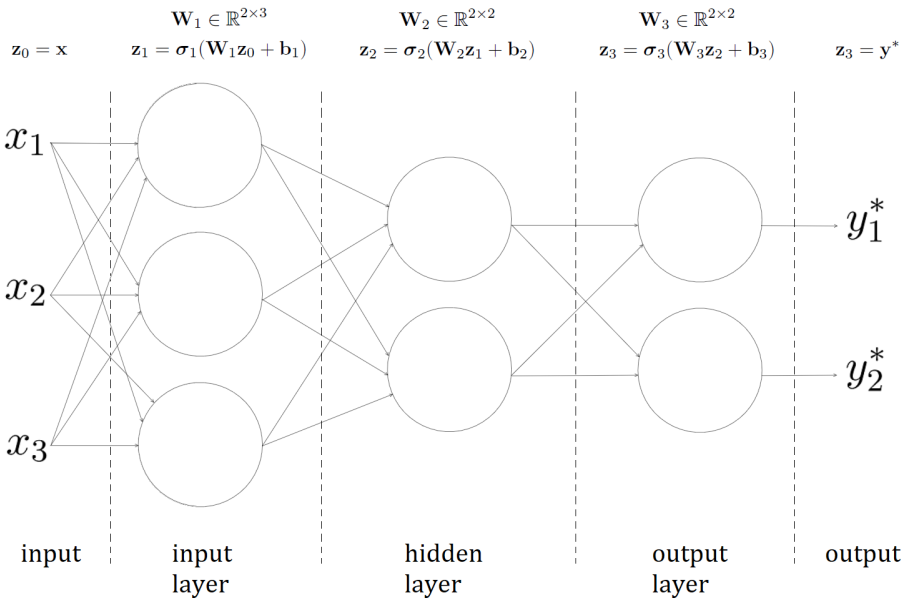


Figure 3.3.: A fully-connected feed-forward neural network with three layers. Circles are representing neurons and edges are illustrating the data flow.

This choice results in

$$\begin{aligned}\boldsymbol{\sigma}_l'(\mathbf{v}) &= diag\big(\sigma_l'(v^1),\ \sigma_l'(v^2),\ \dots\big) \\ &= diag\big(\sigma_l(v^1)(1-\sigma_l(v^1)),\ \sigma_l(v^2)(1-\sigma_l(v^2)),\ \dots\big) \quad .\end{aligned} \tag{3.18}$$

To evaluate the output of the network during training, the mean squared loss function from equation 3.11 is used

$$L_s(\mathbf{y},\mathbf{y}^*) = \frac{1}{2|\mathbf{y}|}\sum_{i=1}^{|\mathbf{y}|}(y^i - y^{*i})^2 \qquad \frac{\partial L_s(\mathbf{y},\mathbf{y}^*)}{\partial \mathbf{y}^*} = -\frac{1}{|\mathbf{y}|}(\mathbf{y}-\mathbf{y}^*)^T \quad . \tag{3.19}$$

As the complete network architecture is known, the gradient, averaged over a set of training samples $\mathbb{T}_b$, with respect to a learnable weight can be computed. As an example, w. l. o. g. we compute the gradient with respect to w_2^{00}:

$$\begin{aligned}\frac{\partial L(\mathbb{T}_b)}{\partial w_2^{00}} &= \frac{1}{|\mathbb{T}_b|}\sum_{(\mathbf{x}_s,\mathbf{y}_s)\in\mathbb{T}_b}\frac{\partial}{\partial w_2^{00}}L_s(\mathbf{y}_s,\mathbf{z}_3) \\ &= \frac{1}{|\mathbb{T}_b|}\sum_{(\mathbf{x}_s,\mathbf{y}_s)\in\mathbb{T}_b}\frac{1}{2}(\mathbf{y}_s-\mathbf{z}_3)^T\frac{\partial}{\partial w_2^{00}}\mathbf{z}_3 \\ &= \frac{1}{|\mathbb{T}_b|}\sum_{(\mathbf{x}_s,\mathbf{y}_s)\in\mathbb{T}_b}\frac{1}{2}(\mathbf{y}_s-\mathbf{z}_3)^T\boldsymbol{\sigma}_3'(\mathbf{W}_3\mathbf{z}_2+\mathbf{b}_3)\mathbf{W}_3\frac{\partial}{\partial w_2^{00}}\mathbf{z}_2 \\ &= \frac{1}{|\mathbb{T}_b|}\sum_{(\mathbf{x}_s,\mathbf{y}_s)\in\mathbb{T}_b}\frac{1}{2}(\mathbf{y}_s-\mathbf{z}_3)^T\boldsymbol{\sigma}_3'(\mathbf{W}_3\mathbf{z}_2+\mathbf{b}_3)\mathbf{W}_3\boldsymbol{\sigma}_2'(\mathbf{W}_2\mathbf{z}_1+\mathbf{b}_2)\frac{\partial\mathbf{W}_2}{\partial w_2^{00}}\mathbf{z}_1\end{aligned} \tag{3.20}$$

Hence, the overall gradient is:

$$\frac{\partial L(\mathbb{T}_b)}{\partial w_2^{00}} = \frac{1}{|\mathbb{T}_b|} \sum_{(\mathbf{x}_s,\mathbf{y}_s)\in\mathbb{T}_b} \frac{1}{2} \underbrace{(\mathbf{y}_s - \mathbf{z}_3)^T}_{\mathbb{R}^{1\times 2}} \underbrace{\boldsymbol{\sigma}_3'(\mathbf{W}_3\mathbf{z}_2 + \mathbf{b}_3)}_{\mathbb{R}^{2\times 2}} \underbrace{\mathbf{W}_3}_{\mathbb{R}^{2\times 2}} \times \underbrace{\boldsymbol{\sigma}_2'(\mathbf{W}_2\mathbf{z}_1 + \mathbf{b}_2)}_{\mathbb{R}^{2\times 2}} \underbrace{\begin{bmatrix}1 & 0\\ 0 & 0\end{bmatrix} \mathbf{z}_1}_{\begin{bmatrix}z_1^0 & 0\end{bmatrix}^T} \tag{3.21}$$

The computed gradient can now be used to update the weight w_2^{00}, meaning subtracting $\frac{\partial L(\mathbb{T}_b)}{\partial w_2^{00}}$ from w_2^{00}, as the goal is to minimize the loss.

The approach that is virtually universally used to embed back-propagation into a learning scheme is gradient descent. The idea is to move the learnable parameters in the direction opposite to the gradient to minimize the computed loss. As the true gradient depends on the unknown distribution of the sample space, commonly two approximations are distinguished to describe algorithms that utilize the estimated gradients.

The first algorithm is simply called *Gradient Descent* (GD). The algorithm computes the gradient as outlined in equation 3.16, where $\mathbb{T}_b = \mathbb{T}$. Thus, the gradient of the loss computed on the sample distribution is estimated by averaging the gradients of each sample in the training set.

$\mathbf{W}_l, \mathbf{b}_l = \text{initializeWeights}(\mathbf{W}_l, \mathbf{b}_l) \quad \forall l \in [0, L]$; `// Initialization of weights`
`/* Sweep NUM_ITERATION times over the training set */`
for *epoch=0* **to** *NUM_ITERATION* **do**
  $\nabla L_{\mathbf{W}_l} = 0, \nabla L_{\mathbf{b}_l} = 0 \quad \forall l \in [0, L]$;
  foreach $(\mathbf{x}, \mathbf{y})$ *in* $\mathbb{T}$ **do**
    $\mathbf{y}^* = \mathcal{F}(\mathbf{x}, \mathbf{W}_1, \mathbf{b}_1, \ldots, \mathbf{W}_L, \mathbf{b}_L)$; `// Feed x forward`
    `/* Compute gradients */`
    $\nabla L_{\mathbf{W}_l} = \nabla L_{\mathbf{W}_l} + \left[\frac{\partial L_s(\mathbf{x},\mathbf{y})}{\partial w_l^{jk}} \quad \forall j \in [0, m_l] \; \forall k \in [0, m_{l-1}]\right] \quad \forall l \in [0, L]$;
    $\nabla L_{\mathbf{b}_l} = \nabla L_{\mathbf{b}_l} + \left[\frac{\partial L_s(\mathbf{x},\mathbf{y})}{\partial b_l^{j}} \quad \forall j \in [0, m_l]\right] \quad \forall l \in [0, L]$;
  end
  `/* Use average gradients to update weights */`
  $\mathbf{W}_l = \mathbf{W}_l - \frac{\lambda}{|\mathbb{T}|}\nabla L_{\mathbf{W}_l} \quad \forall l \in [0, L]$;
  $\mathbf{b}_l = \mathbf{b}_l - \frac{\lambda}{|\mathbb{T}|}\nabla L_{\mathbf{b}_l} \quad \forall l \in [0, L]$;
end

Algorithm 2: The *Gradient Descent* learning scheme

The parameter λ from algorithm 2 is the learning rate, a hyperparameter that trades off weight adaptiveness and error robustness due to errors emerging from the gradient approximation. The learning rate may be a constant or a sophisticated function that depends on the current epoch or on the epoch and the considered weight (see [8, 43]).

Another way to estimate the gradient of the loss computed on the sample distribution is to use the *Stochastic Gradient Descent* (SGD) algorithm. SGD considers a single (randomly selected) sample per epoch and uses its gradient to update the weights. There are two advantages compared to GD. SGD is much faster, as it does not iterate over the whole

training set each epoch, and it implements some sort of self-regularization, as the poor gradient estimation asserts that noise is introduced to avoid that the model overfits. Additionally, this does enhance SGD its ability to jump away from local minima compared to GD.

$\mathbf{W}_l, \mathbf{b}_l = \text{initializeWeights}(\mathbf{W}_l, \mathbf{b}_l) \quad \forall l \in [0, L]$; // Initialization of weights
/* Update weights *NUM_ITERATION* times */
for *epoch=0* **to** *NUM_ITERATION* **do**
  $\mathbf{x}, \mathbf{y} = \text{getRandomSample}(\mathbb{T})$; // Pick a random sample from the training set
  $\mathbf{y}^* = \mathcal{F}(\mathbf{x}, \mathbf{W}_1, \mathbf{b}_1, \ldots, \mathbf{W}_L, \mathbf{b}_L)$; // Feed x forward
  /* Compute gradients */
  $\nabla L_{\mathbf{W}_l} = \left[\frac{\partial L_s(\mathbf{x},\mathbf{y})}{\partial w_l^{jk}} \quad \forall j \in [0, m_l] \; \forall k \in [0, m_{l-1}]\right] \quad \forall l \in [0, L]$;
  $\nabla L_{\mathbf{b}_l} = \left[\frac{\partial L_s(\mathbf{x},\mathbf{y})}{\partial b_l^{j}} \quad \forall j \in [0, m_l]\right] \quad \forall l \in [0, L]$;
  /* Use gradients to update weights */
  $\mathbf{W}_l = \mathbf{W}_l - \lambda \nabla L_{\mathbf{W}_l} \quad \forall l \in [0, L]$;
  $\mathbf{b}_l = \mathbf{b}_l - \lambda \nabla L_{\mathbf{b}_l} \quad \forall l \in [0, L]$;
end

Algorithm 3: The *Stochastic Gradient Descent* learning scheme

A trade-off between SGD and GD is to use *mini-batches*. Thus, instead of iterating over the whole training set every epoch in algorithm 2, GD with *mini-batches* would iterate over a small subset $\mathbb{T}_b \subset \mathbb{T}$ of the training set.

There are many tricks that can be applied to increase the chance of convergence and to speed up the training process. Primary advices typically incorporate a proper weight initialization and input normalization as well as input transformations/perturbations to artificially enlarge the training set. For more information, please refer to the works of Bottou [8] and LeCun et al. [43].

It is worth mentioning that there exists a variety of second-order methods that incorporate the use of the Hessian of the loss function to accelerate the learning process. However, as mentioned in [43], they are mostly impractical or infeasible on large networks and are therefore by far not as common as first order methods.

Before we move our attention to special neural net architectures, we want to highlight one more property or emergent effect of the just presented machine learning technique. Bottou [7] has demonstrated in his work, that neural networks are capable of learning general attributes of the world, such that they can be easily transferred to other tasks. He emphasizes, that this is similar to human reasoning. In general, our neurons are not trained to be used for single task (e.g. handwriting recognition). Moreover, the network body is reused for a variety of tasks. Transfer learning can be easily applied to artificial neural networks, as it only requires the output layer to be exchanged.

3.3.2. Convolutional Neural Networks

Fully-connected neural networks have a serious drawback. For each layer, the algorithm has to learn $m_l \cdot m_{l-1} + m_l$ parameters. Imagine a single layer, that should perform object recognition for 200 different objects on low resolution images with 200×200 pixels. This setup would require to learn around 8 million parameters. It is doubtful that such a complex model can be learned by a moderate-sized training set.

(a) Original image

(b) Filtered image

Figure 3.4.: A usage example of convolutions in image processing. The kernel from equation 3.23 extracts the vertical edges from the original image.

An effective alternative are Convolutional Neural Networks (CNN). They have been proven to be incredibly successful compared to fully-connected networks for tasks where the input has high local correlations, e.g. objects in an image. Their architectural design is inspired by the neuronal image processing in the visual cortex of animals [32].

A convolution is a mathematical operation that modifies each element of a matrix by incorporating its local area based on weights defined by a kernel. If $\mathbf{C}$ and $\mathbf{K}$ are matrices, whereas $\mathbf{K} \in \mathbb{R}^{k \times k}$ where k is an odd number ($k^* := \lfloor \frac{k}{2} \rfloor$), a convolution can be expressed as

$$(\mathbf{C} * \mathbf{K})_{i,j} = \sum_{a=-k^*}^{k^*} \sum_{b=-k^*}^{k^*} c_{i-a,j-b} \cdot k_{a+k^*,b+k^*} \quad . \tag{3.22}$$

Convolutions are widely used in image processing for local feature extraction. For instance, convolutions are useful to detect edges as depicted in figure 3.4. The kernel that was used to filter the image is

$$\mathbf{K} = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \quad . \tag{3.23}$$

This filter can only detect vertical edges. The intuition behind this specific filter is, that a vertical edge results in a gradient between the left and right pixel at a certain pixel position. A homogeneous area with the same pixel values in the local neighborhood will result in an annihilation of horizontal pairs when applying this filter.

Convolutional Neural Nets utilize this powerful method of feature extraction. But instead of taking hand-engineered kernels, they learn the filters that are most relevant for the task by themselves. A deep CNN architecture typically involves a series of layer modules. For instance, a convolutional layer followed by a non-linear activation followed by a down-sampling step based on heuristics (called pooling-layer). The generated representation (e.g. a list local object detections) can be passed to a fully-connected layer, that infers the global semantics of the input. Such a sample architecture is depicted in figure 3.5.

Convolutional layers are feed-forward layers just as fully-connected layers. They may compose of several *feature maps*. Each feature map learns a different kernel, that is applied to the input. There are two major differences compared to fully-connected layers. Firstly, a neuron in a convolutional layer only connects to some local outputs of the previous layer, the so called *local receptive field*. Secondly, all neurons of a feature map share the same weights (the kernel learned for this feature map).

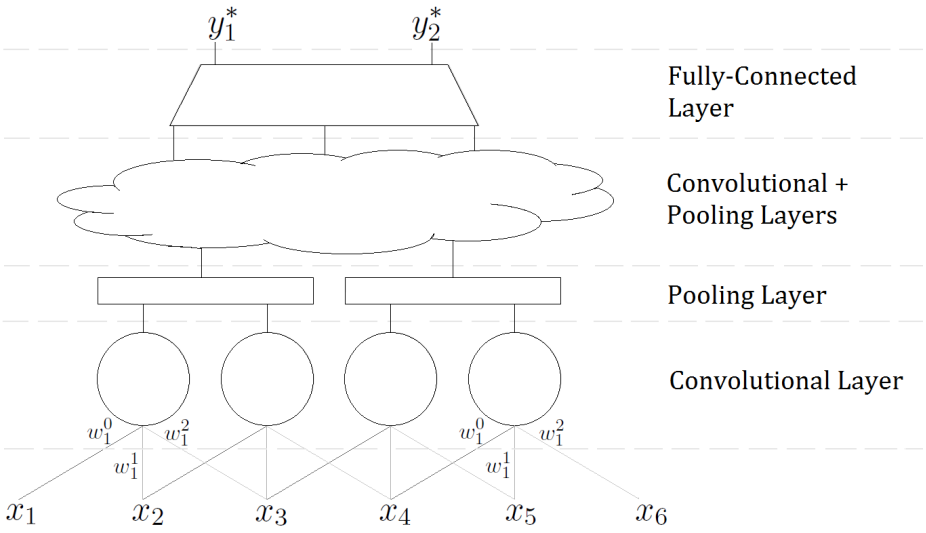


Figure 3.5.: The typical architecture of a deep convolutional neural network.

We denote the weights of a convolutional layer l with ${}^f\mathbf{W}_l \in \mathbb{R}^{k_f \times k_f}$ where f refers to the index of the considered feature map. Thus ${}^f\mathbf{W}_l$ is the kernel of the f-th feature map in layer l. When assuming that the output of the previous layer is a matrix $\mathbf{z_l}$ (or reshaped to a matrix), then the computation performed by each neuron is

$$ {}^f z_l^{ij} = \sigma_l({}^f v_l^{ij}) \quad \text{with} \quad {}^f v_l^{ij} = \sum_{a=-k_f^*}^{k_f^*} \sum_{b=-k_f^*}^{k_f^*} z_{l-1}^{i-a,j-b} \cdot {}^f w_l^{a+k_f^*,b+k_f^*} + {}^f b_l \quad . \tag{3.24} $$

If there are several input channels (e.g. the RGB channels of an input image or the feature maps of a previous convolutional layer), then the correct ${}^f v_l^{ij}$ is computed by summing over all channels. To improve readability, we assume that there is only one input channel and only one feature map, such that the pre-superscript f can be ignored.

Due to the feature extraction on local receptive fields, CNNs are robust against translations. To some extend, they are even robust against distortions and scaling [42]. However, to grasp the features detected by a convolutional layer, they are typically combined with a subsequent pooling layer, that shall heuristically compress the output. The most prominent among those layers is the max-pooling layer, which considers a local receptive field and only forwards the maximum value occurring in this field. This architecture allows to detect low-level features in early layers (such as edges) and high-level features in deeper layers (such as objects).

Learning in convolutional layers is similar to fully-connected layers. To apply back-propagation, the gradients $\partial L / \partial \mathbf{z}_{l-1}$ has to be calculated.

$$ \frac{\partial L}{\partial \mathbf{z_{l-1}}} = \frac{\partial L}{\partial \mathbf{v_l}} \frac{\partial \mathbf{v_l}}{\partial \mathbf{z_{l-1}}} \tag{3.25} $$

We assume, that $\partial L / \partial \mathbf{v_l}$ is known (see subsection 3.3.1 for computation details). Note, if layer $l+1$ assumes a vector as input and not a matrix, v_l can be reshaped to a vector or preferably, all matrices are vectorized for internal computations. The derivative of equation

3.24 with respect to a single input activation is

$$\begin{aligned}\frac{\partial v_l^{ij}}{\partial z_{l-1}^{i'j'}} &= \frac{\partial}{\partial z_{l-1}^{i'j'}}\Big(\sum_{a=-k^*}^{k^*}\sum_{b=-k^*}^{k^*} z_{l-1}^{i-a,j-b}\cdot w_l^{a+k^*,b+k^*} + b_l\Big) \\ &= \begin{cases} w_l^{i-i'+k^*,j-j'+k^*}, & \text{if } -k^* \le i-i' \le k^* \text{ and } -k^* \le j-j' \le k^* \\ 0, & \text{otherwise.}\end{cases}\end{aligned} \tag{3.26}$$

Equation 3.26 utilizes, that we only consider the $w_l^{a+k^*,b+k^*}$, that satisfies the conditions $i' = i - a$ and $j' = j - a$. We will denote the case-distinction from equation 3.26 by the $\hat{}$-operator, thus $\hat{w}_l^{i-i'+k^*,j-j'+k^*}$ is zero if one of the conditions does not hold. Hence, we can state that

$$\frac{\partial \mathbf{v}_l}{\partial z_{l-1}^{ij}} = \begin{pmatrix} \hat{w}_l^{0-i+k^*,0-j+k^*} & \hat{w}_l^{0-i+k^*,1-j+k^*} & \cdots \\ \hat{w}_l^{1-i+k^*,0-j+k^*} & \ddots & \\ \vdots & & \end{pmatrix} \quad . \tag{3.27}$$

To avoid working with tensors, we vectorize $\partial L/\partial \mathbf{v_l}$ and the matrix from equation 3.27. Now, the targeted derivative can be computed as dot product of both vectors. But by taking into consideration that only k elements of the above matrix are non-zero per column, we can simplify the derivative of the loss with respect to an input activation of the convolutional layer l as

$$\begin{aligned}\frac{\partial L}{\partial \mathbf{v_l}}\frac{\partial \mathbf{v_l}}{\partial z_{l-1}^{ij}} &= \sum_{a=-k^*}^{k^*}\sum_{b=-k^*}^{k^*}\Big(\frac{\partial L}{\partial \mathbf{v_l}}\Big)^{i+a,j+b} w_l^{a+k^*,b+k^*} \\ &= \sum_{a=-k^*}^{k^*}\sum_{b=-k^*}^{k^*}\Big(\frac{\partial L}{\partial \mathbf{v_l}}\Big)^{i-a,j-b} w_l^{-a+k^*,-b+k^*} \\ &= \Big(\frac{\partial L}{\partial \mathbf{v_l}}\Big)^{ij} * \mathrm{rot180}(\mathbf{W_l}) \quad ,\end{aligned} \tag{3.28}$$

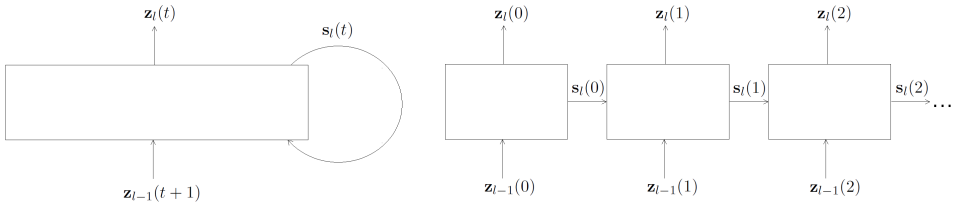
where the operation *rot180* rotates the kernel $\mathbf{W_l}$ by 180 degrees.
Similarly, the update rule can be derived

$$\frac{\partial v_l^{ij}}{\partial w_l^{ab}} = z_{l-1}^{i-a,j-b} \tag{3.29}$$

The resulting matrix $\partial \mathbf{v}_l/\partial w_l^{ab}$ can be vectorized and utilized to compute $\partial L/\partial w_l^{ab}$. A more detailed description of the back-propagation process in convolutional neural networks can be found in [9].

3.3.3. Recurrent Neural Networks

The fact that complex input, that humans can perceive, typically decomposes into smaller parts that can be individually categorized, makes CNNs to a powerful machine learning tool, that can solve complex tasks with reasonable-sized datasets and with the limited



(a) A recurrent neural layer (b) Recurrent layer unfolded over time

Figure 3.6.: A recurrent layer does backproject its state to allow that past inputs modify the next output.

computational power currently available. But they are not designed to manage temporal or variable-sized input. Additionally, they cannot reflect on what they have processed just like all feed-forward neural networks. To overcome this shortcoming, Recurrent Neural Networks (RNN) have been designed. They are inspired by the recurrent structure of biological neural networks. Humans can perceive four dimensions. This makes it unlikely that neural networks can mimic human intelligence if they only have spatial and no temporal awareness.

A recurrent neuron possesses a state that encodes the history of input activation. Every time step, the recurrent neuron morphs its own state based on the new input and generates an output activation from the newly generated state. We denote the internal state of layer l at time step t by $\mathbf{s}_l(t) \in \mathbb{R}^{n_l}$. The state modification of a simple recurrent neural network can now be described as

$$\mathbf{s}_l(t) = \boldsymbol{\sigma}\big(\underbrace{\mathbf{W}_{l,i\to s}\mathbf{z}_{l-1}(t) + \mathbf{W}_{l,s\to s}\mathbf{s}_l(t-1) + \mathbf{b}_{l,s}}_{\mathbf{v}_l(t)}\big) = \boldsymbol{\sigma}_{l,s}\big(\mathbf{W}_{l,s}\begin{pmatrix}\mathbf{z}_{l-1}(t)\\ \mathbf{s}_l(t-1)\end{pmatrix} + \mathbf{b}_{l,s}\big) \quad . \tag{3.30}$$

The output may then be computed as

$$\mathbf{z}_l(t) = \boldsymbol{\sigma}_{l,o}\big(\mathbf{W}_{l,o}\mathbf{s}_l(t) + \mathbf{b}_{l,o}\big) \tag{3.31}$$

The model expressed by above equations is depicted in figure 3.6a. It shows that the last state is fed back as input into the recurrent layer to incorporate the layer its history (the previous time steps) in the output computation. A common approach to train recurrent neural networks is to back-propagate the error through time [65]. Therefore, the recurrent layers are unfolded over time (see figure 3.6b), such that they can be considered as usual feed-forward networks. The gradients with respect to the weights are computed for each time step and accumulated before the update occurs.

$$\frac{\partial L}{\partial w_l^{ij}} = \sum_{t=0}^{T} \frac{\partial L(z)}{\partial w_l^{ij}} \tag{3.32}$$

One major difference to earlier gradient computations is the recurrence in equation 3.30. The derivative of equation 3.30 with respect to an element of $\mathbf{W}_{l,s\to s}$ requires to consider the derivative of $\mathbf{s}_l(t-1)$ too.

$$\frac{\partial L(t)}{\partial w_{l,s\to s}^{ij}} = \frac{\partial L(t)}{\partial \mathbf{v}_l(t)} \frac{\partial \mathbf{v}_l(t)}{\partial w_{l,s\to s}^{ij}} = \frac{\partial L(t)}{\partial \mathbf{v}_l(t)} \Big(\frac{\partial \mathbf{W}_{l,s\to s}}{\partial w_{l,s\to s}^{ij}} \mathbf{s}_l(t-1) + \sum_{t'=0}^{t-1} \frac{\partial \mathbf{s}_l(t-1)}{\partial \mathbf{s}_l(t')} \frac{\partial \mathbf{s}_l(t')}{\partial w_{l,s\to s}^{ij}} \Big) \quad (3.33)$$

We can rewrite the last term in equation 3.33 for $t' = 0$ as

$$\frac{\partial \mathbf{s}_l(t-1)}{\partial \mathbf{s}_l(0)} \frac{\partial \mathbf{s}_l(0)}{\partial w_{l,s\to s}^{ij}} = \frac{\partial \mathbf{s}_l(t-1)}{\partial \mathbf{s}_l(t-2)} \frac{\partial \mathbf{s}_l(t-2)}{\partial \mathbf{s}_l(t-3)} \cdots \frac{\partial \mathbf{s}_l(1)}{\partial \mathbf{s}_l(0)} \frac{\partial \mathbf{s}_l(0)}{\partial w_{l,s\to s}^{ij}} \quad . \quad (3.34)$$

Equation 3.34 reveals one of the major drawbacks of the simple RNN formalism described above. For large values of t the computed gradient might explode or vanish, as the product promotes an exponential amplification. It has been shown, that the exploding gradient problem can be managed by clipping large gradients [54]. The vanishing gradient problem has not been solved for this RNN architecture. In contrast to exploding gradients, vanishing gradients do not stop the network from learning temporal relations, since the gradient is unlikely to vanish if we only look at the recent history. On the other hand, long-term dependencies cannot be grasp if the gradient vanishes. However, the major incentive for the development of RNNs is the processing of variable-sized sequences, as local temporal behavior can be simulated by feed-forward neural networks as well (e.g. by considering n-grams). One architectural enhancement that sufficiently overcomes the vanishing gradient problem was initially proposed by Hochreiter and Schmidhuber [26]. The recurrent model they invented is called Long-Short-Term Memory (LSTM) and allows to hold past information over a long timespan by incorporating the model of a memory cell. This memory cell is accessed by gates. The input gate $\mathbf{i}_l$ controls what information, that can be perceived from the outside world, should be considered to accomplish the task assigned to this cell $\mathbf{c}_l$. The output gate $\mathbf{o}_l$ decides what part of the maintained knowledge should be shown to the next layer, as the whole internal state might flood the outer world with information and leads to distraction. Finally, the forget gate $\mathbf{f}_l$ allows the cell to forget unnecessary information and keep only the essential part. The vector $\mathbf{g}_l$ is a helper to describe the actual input to the layer. An LSTM cell as described by the following equations is shown in figure 3.7 (the equations are based on the LSTM defined in [36]).

$$\mathbf{i}_l = \boldsymbol{\sigma}_{l,i}\big(\mathbf{W}_{l,i} \begin{pmatrix} \mathbf{z}_{l-1}(t) \\ \mathbf{z}_l(t-1) \end{pmatrix} + \mathbf{b}_{l,i}\big) \quad (3.35)$$

$$\mathbf{o}_l = \boldsymbol{\sigma}_{l,o}\big(\mathbf{W}_{l,o} \begin{pmatrix} \mathbf{z}_{l-1}(t) \\ \mathbf{z}_l(t-1) \end{pmatrix} + \mathbf{b}_{l,o}\big) \quad (3.36)$$

$$\mathbf{f}_l = \boldsymbol{\sigma}_{l,f}\big(\mathbf{W}_{l,f} \begin{pmatrix} \mathbf{z}_{l-1}(t) \\ \mathbf{z}_l(t-1) \end{pmatrix} + \mathbf{b}_{l,f}\big) \quad (3.37)$$

$$\mathbf{g}_l = \boldsymbol{\sigma}_{l,g}\big(\mathbf{W}_{l,g} \begin{pmatrix} \mathbf{z}_{l-1}(t) \\ \mathbf{z}_l(t-1) \end{pmatrix} + \mathbf{b}_{l,g}\big) \quad (3.38)$$

$$\mathbf{c}_l(t) = \mathbf{f}_l \odot \mathbf{c}_l(t-1) + \mathbf{i}_l \odot \mathbf{g}_l \quad (3.39)$$

$$\mathbf{z}_l(t) = \mathbf{o}_l \odot \mathbf{tanh}(\mathbf{c}_l(t)) \quad (3.40)$$

This RNN architecture can better deal with vanishing gradients because of the addition

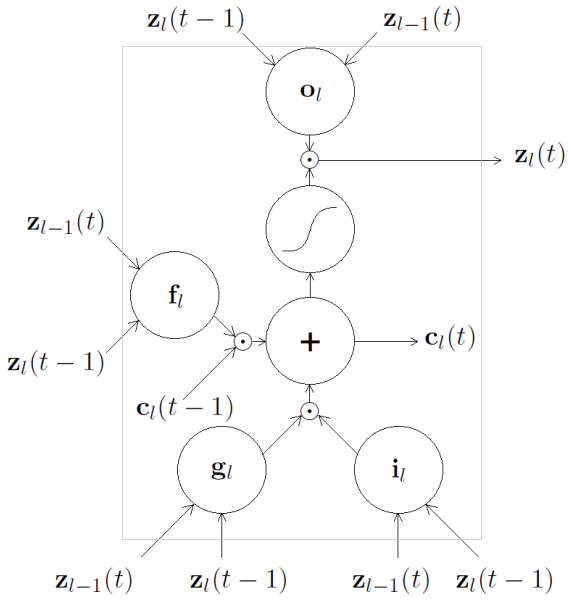


Figure 3.7.: The model of an LSTM cell as described by equations 3.35 to 3.40.

in the recurrence equation 3.39. If back-propagation through time is applied on the cell, the recurrence equation leads to

$$\frac{\partial \mathbf{c}_l(t)}{\partial \mathbf{c}_l(t-1)} = \mathbf{f}_l \qquad . \tag{3.41}$$

The original recurrence equation from Hochreiter and Schmidhuber [26] had no forget gate, so ${}^{\partial \mathbf{c}_l(t)}/_{\partial \mathbf{c}_l(t-1)}$ would have been 1. The forget gate was later introduced as it had been proven to induce more successful LSTM networks. But the general idea of a constant error flow makes LSTMs capable of learning long-term dependencies. Karpathy et al. [36] showed on character sequences that LSTMs can learn over long distances (> 200). Additionally, they observed that for the considered task some of the neurons learned an interpretable representation (e.g. whether the current input is inside a quote). This is remarkably, as it proofs the general ability of RNNs to infer properties of the global structure by considering one entity at a time.

3.3.4. Deep Learning in NLP

Of particular interest for this section is the paper from Collobert et al. [11], because it highlights the benefits of deep neural networks. In section 3.1 we mentioned, that one problem of conventional NLP feature extraction is, that these features themselves are not easy to formalize and are therefore learned by ML tools. More sophisticated tasks are therefore forced to be trained with noisy feature vectors. Collobert et al. [11] have shown that one can reach state-of-the art results in a variety of basic NLP tasks (e.g. POS-tagging, NER, SRL) by using deep neural networks that automatically learn the features required to solve the task. They implemented a tool *SENNA*, that is in terms of runtime as well as in terms of memory-usage superior compared to conventional tools.

That CNNs can not only successfully be used to accomplish tasks in CV is proven by the work of [37]. They also have been advantageous for paraphrase detection [25, 31].

Paraphrase detection aims to identify pairs of sentences that have the same meaning. This is particularly useful, as human language provides many different ways to express the same thought, whereas a tiny modification in a sentence can change its overall meaning. Another problem is, that semantic alignments are not knowingly expressible as features, since they are encoded in a hidden structure. It is just another prominent example, where deep neural networks can overcome the problems arising from a lack of proper handcrafted features.

The most promising architecture (if not a combined one) for NLP tasks are RNNs, as language is perceived as a variable-sized sequence. Graves et al. [24] show that state-of-the art results in speech recognition can be achieved with deep RNNs. In contrast to the RNNs presented in subsection 3.3.3, they use bidirectional RNNs. These RNNs can utilize knowledge from past and future input, so they have the additional ability to modify their output based on what comes next.

Lai et al. [41] combine a recurrent structure in a CNN to classify texts. Their architecture is especially designed to capture the contextual usage of words. This is interesting as it allows to translate raw tokens into a representation that encodes the meaning of those words. In general, this approach is called *word embeddings*. They tackle another problem mentioned in section 3.1: How to input raw text into a learning algorithm?

Word Embeddings

Word embeddings are representations of words as numerical vectors. A simple illustration is to imagine a lookup table that translates each token in a vocabulary $w \in \mathbb{V}$ into a d-dimensional vector $\mathbf{w} \in \mathbb{R}^d$. We specifically want to present an approach by Mikolov et al. [49], that is utilized by the implementation used in our system. The word vectors should be created such that similar words (words appearing in similar contexts) have vector representations that are close to each other. To reach this goal, they trained a neural network that shall predict words in the context of a given word (so called *skip-gram model*). More precisely, their optimization goal is

$$\frac{1}{N}\sum_{n=1}^{N}\sum_{\substack{-c\leq j\leq c\\ j\neq 0}} \log p(w_{n+j}|w_n) \qquad , \tag{3.42}$$

where $w_1, w_2, \ldots, w_N$ is a token sequence and c is the size of the context window. The probability of a context word given the current word can be estimated as[4]

$$p(w_c|w_n) = \frac{e^{\langle \mathbf{w}_c, \mathbf{w}_n\rangle}}{\sum_{v\in\mathbb{V}} e^{\langle \mathbf{v}, \mathbf{w}_n\rangle}} \qquad . \tag{3.43}$$

As the normalization term in equation 3.43 is computationally too expensive to calculate, they propose a few methods to overcome the problem. For example, instead of considering all words, they only consider a fixed number of tokens drawn from a distribution of negative samples.

The resulting vectors have an astonishing quality. An often cited example is that $\mathbf{w}_{\text{“King”}} - \mathbf{w}_{\text{“Man”}} + \mathbf{w}_{\text{“Woman”}}$ results in $\mathbf{w}_{\text{“Queen”}}$. This distinctly highlights how simple vector operations in the constructed space can translate meaning. Such word vectors are much more expressive than raw tokens when used as input to an learning algorithm. For instance, they

[4] Strictly speaking, they distinguished between ordinary word vectors and those of contextual words taken from different weight matrices in their network architecture.

allow that a word can be understood even if it is not in the training set as long as a synonym has been seen in the learning phase.

3.3.5. Deep Learning in CV

Aside from NLP, there is a variety of other research fields that have benefited from deep neural network architectures. An interesting architecture has been recently proposed by Badrinarayanan et al. [1]. They use a deep CNN with an encoder-decoder structure to segment an input image. So, instead of directly converting the input image into a segmented image, they compress the input and up-sample this representation and yet reach competitive results in this first architecture appliance.

Another emerging field of CV is video recognition. Karpathy et al. [35] proposed an architecture that uses CNNs with a fixed-sized temporal window and showed that this architecture can beat single-frame considerations. Donahue et al. [15] suggested a more promising architecture for video recognition, that encodes frames through a CNN and learns their interdependencies using an RNN.

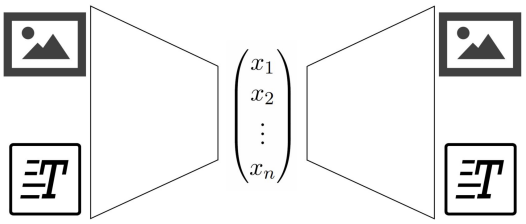
The CV task that has gotten the most attention due to its success with deep learning models is object recognition. The first incredibly successful deep learning network was *AlexNet* [39]. There had been a lot of improvements since *AlexNet* was published [58, 59]. Currently, the *Inception* model by Szegedy et al. [60] is considered as the best network for object-recognition and it has been proven successful when transferring the model to other tasks [64]. The model has a depth of 42 layers while minimizing the number of parameters to learn and the computational cost of a forward-sweep compared to competitive models. This is achieved by carefully engineered layers and state-of-the art regularization techniques. On account of this, we decided to use the *InceptionV3* model to encode images in our architecture.

4. A Deep Neural Network Architecture for Multimodal Document Comprehension

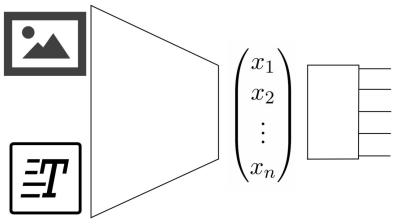
The work that has been done to accomplish this thesis will be summarized in this chapter. The overall task is to implement a novel system that has human intuition when judging the interrelation of co-occurring images and texts and therewith being able to fully comprehend individual modalities and to conclude whether missing or additional information can be taken from the other modality to enrich and augment the perceived information. Our main incentive is, that humans use several modalities to convey information that complements one another. This insight has been already stated by others (e.g. [3]), but it has been poorly addressed by the related work as outlined in chapter 2, which is mainly due to difficulties in modeling human intuition and because of the immense computational power necessary to process enough data to learn an appropriate understanding of the world, which is necessary when considering multimodal documents from unconstrained domains. Most multimodal document sources use a specific modality to convey information that is easier to perceive by the intended receiver as if another modality would be used. In the specific case of co-occurring images and texts, the document text normally does not explain what is visual obvious (which is the primary goal of image-captioning systems). Instead, the text encodes information that is hard or infeasible to visualize in a single image, such as sequential information or specific facts (e.g. a calendar date or name). On the other hand, images are easier to perceive and can depict some information far more succinctly (e.g. shapes and textures).

To meet this claims, a diverse database, that sufficiently encodes knowledge about the world, represents natural co-occurrences and allows to understand the semantics of images and texts is needed. To achieve this, we combine three different datasets to meet each of the previously stated goals. The specifics of each dataset as well as their importance will be clarified in section 4.1. The first dataset (subsection 4.1.1) shall enable the system to learn a translation of salient information from one modality to another. Therefore, an image captioning dataset is used as it uniquely represents how the semantics of both modalities are captured by humans, if they are considered separately. A news article dataset (subsection 4.1.2) is used as an example corpus of particularly complicated image-text relation, as their content is typically loosely correlated and the meaning of their co-occurrence mostly hard to interfere. Supplementary, a dataset of encyclopedia articles (subsection 4.1.3) is included, to incorporate knowledge about the world. This is necessary, as the understanding of relations between different modalities often requires background knowledge. The encyclopedia dataset, that we call *SimpleWiki* dataset, is a new dataset, that has been collected by us to meet the requirements that we demand from a representative dataset.

The human learning process is twofolded: supervised and unsupervised. We are observing the world and draw our own conclusions from it, but we get also directed and corrected by the people surrounding us. For instance, if we observe *elephants*, we are capable of extracting prominent features (shape, trunk, skin color, etc.) and generalize all elephants into a single concept without supervision. However, someone has to tell us, that these mammals are



(a) The autoencoder encodes an image-text pair into a compact representation. Subsequently, a decoder network tries restore the original image-text pair.



(b) The classifier uses the encoding architecture from the autoencoder to map an input pair onto a multimodal embedding. A classifier on top of that embedding quantifies the image-text relation.

Figure 4.1.: The simplified architectures of the autoencoder and classifier.

called *"elephants"*. Hence, the overwhelming majority and the apparently more complicated part of the work is done fully unsupervised. This approach is encouraging, as it may allow to learn complex system with just a small portion of supervised interference. Still, annotated training data is necessary to direct the learning process such that the semantic outcome aligns with our understanding of the world.

Maybe, it is even possible to invent a system that learns without any supervision, just as humans can learn new concepts fully unsupervised (e.g. from books). However, the semantics of language must be understood prior to learning in such a system. That those semantics can be inferred without any supervision is questionable. The meaning of many words might be trainable only from their context. However, to gain real world knowledge, some words have to be mapped to real world entities to grasp our semantic meaning. Such a mapping would be a supervised or weakly supervised (cp. section 3.3) act through suitable datasets. On the other hand, it is likely that all necessary links are already available in datasets from the Internet. Nevertheless, this approach would be less directed and more complicated[1] and is therefore neglected in all further explanations.

To sum up, the general idea of our system is to unsupervised generalize concepts observed in each modality as well as to unsupervised learn the translation of a concept from one modality to another. Subsequently, human annotations shall guide the system to judge about the relation of image-text pairs.

The annotations shall quantify the relations in terms of shared information and shared meaning. Therefore, we invented two measurements *Mutual Information* and *Semantic Correlation*, as we think it is much easier to quantify the complex relation by means of two rather than one score. Both measurements, the labels assigned to them and the annotation process will be explained in section 4.2.

The ability to generalize concepts and to capitalize background knowledge in further comprehension tasks should be achieved by an autoencoder, whose specifics are explained in section 4.3. The overall system architecture of the autoencoder is depicted in figure 4.1a. An encoding and a decoding network are trained to learn a comprehensive and meaningful feature representation. Finally, the part of the dataset we annotated is used to teach the system human intuition when judging the co-occurrence of a text with an enclosed image. This human judgment is mimicked by a multiclass classifier, that is trained above

[1] Humans do not expect their children to start learning from books. Instead, they choose the approach explained beforehand.

a multimodal embedding space, which has been learned by the autoencoder. This classifier is described in section 4.4. An architectural overview of the classifier is depicted in figure 4.1b.

4.1. Datasets

To ensure a vast variety of the amount of complementary information in image-text pairs, we gathered documents from a broad scope of sources. In the following, we explain the individual datasets that have been selected and combined to train our system. How these datasets have been utilized for our task is explained in following sections. Furthermore, in section 4.1.3 we will explain how we retrieved the newly generated *SimpleWiki* dataset.

4.1.1. Microsoft COCO Captions

The *MS COCO* dataset has been generated for image-captioning tasks by Chen et al. [10]. Therefore, they collected images from Flickr[2], depicting various scenes and objects. The dataset contains nearly $165,000$ images with a split of $\approx 50\%$ for the training set and a validation and test set with the remainder of $\approx 25\%$ each. They crowdsourced multiple human-generated captions for each image, where they were asking for captions that briefly describe what is visually obvious and prominent in the image and does not require an interpretation. Images were labeled with 5 captions from different annotators, except from a fraction of $5,000$ images belonging to the test set, that received 40 captions each. The training and validation sets are freely available. The test set is not directly accessible. But they provide an evaluation server, that computes typical metrics used to score auto-generated captions. Hence, one might test its image-captioning algorithm by sending generated captions on test images to the server, that computes a set of metrics incorporating human-annotated reference captions. This procedure shall ensure, that test samples have not been accidentally used during the training phase and that implemented systems are comparable.

Figure A.1a shows that the sentence structure is relatively uniform across the validation set, which is the part of the dataset that we are using. Sentences tend to be short, which is a sign for easy structured and easy understandable content.

We consider this dataset as it provides pairs with a strong relationship, what typically does not appear in a natural environment. Having samples that state what is visually obvious is essential to learn the semantics of the image content. A deeper, complementary relationship can only be understood, if the two entities themselves have been comprehended.

4.1.2. BBC News Corpora

News articles have a particularly interesting relation to their co-occurring images. Mostly, there are neither direct references in the text to the image content nor is their semantic correlation easily inferable. Even humans do often need the enclosed caption to see why the image fits to its article. For instance, imagine a news article about a new canteen at a certain university, that is published with a photograph of the university its president. However, the article might not mention the president and so it is up to the image caption to allow the understanding of this co-occurrence. To avoid the immense reliance on captions,

[2] `https://www.flickr.com/`

extensive background knowledge might close the gap as well. In general, humans are not capable of memorizing such minor details (e.g. the president of some arbitrary university is considered non-relevant). On the other hand, as computers can access arbitrarily detailed knowledge bases, they can theoretically omit the caption and yet understand a complex, but meaningful, image-text co-occurrence.

Those complex relations should be represented by news articles in our dataset. There are several news datasets freely available. The *ION* dataset by Hollink et al. [28] contains over 300,000 news articles collected from several newspapers over a timespan of one year. The dataset contains features for the article texts and images, but not the raw data itself. However, the URLs are included and they provide a software to re-extract the articles.

Ramisa et al. [55] recently published a new dataset with 100,000 articles, called *BreakingNews*. The articles have been collected from various sources in 2014. Again, the published data contains mostly features, metadata and URLs to the original articles. Additionally, they performed various tasks on this dataset including caption generation. Therefore, they encoded a word embedding matrix as well as the image with a CNN and trained an LSTM language model on top to produce captions. Their captioning results are poor, which may come from the loose relation between an article text and its image as mentioned above.

Another dataset, that is known to us, is the *BBC News Database* from Feng and Lapata [19], which contains over 3,300 articles. Each article is represented by three files, one containing the article text, one containing the image caption and one is the actual image file. The article text still comprises some basic HTML formatting, such that the original text structure (e.g. sections, subsections) is preserved. Furthermore, they provide a data split into training and test ($\approx$7%) set.

They reused the dataset in [20] for caption generation. However, they faced the same problems as [55][3] and achieved poor results. This is one of the reasons, why we believe that captions, as they appear in a natural environment, cannot be generated by a system that does not aim to explicitly encode knowledge about the world, such that the relation of two seemingly unrelated entities can be inferred.

We decided to incorporate the *BBC News Database* into our work, as it is ready to use and assumed to be sufficiently sized to meet our needs. News articles generally express a voluminous content with $\approx 51\%$ of the articles having more than 20 sentences and $\approx 60\%$ of the sentences have more than 20 tokens, when excluding headings (cp. figures A.2a and A.1b). Beside of this inherent complexity, a large vocabulary and a wide range of topics, that incorporates many minor local events, introduce a particular hard comprehension task, that is even for humans challenging.

4.1.3. SimpleWiki Dataset

An online-encyclopedia like Wikipedia is a massive knowledge base, that is at the same time structured and sufficiently trustworthy. In the past, there have been several attempts to embrace the enormous amount of information available in such sources (e.g. [27]). Wikipedia contains general knowledge about the world, but also specialized knowledge about individuals, historic as well as recent events or even proprietary products. However, many articles are not or at least difficult to understand for someone who is outside the subject area.

[3] Beside of the caption generation, they also proposed a method to extract a sentence from the article text that can serve as a legitimate image caption. But as outlined above, such sentences typically do not appear in a news text.

Another aggravating factor is, that many articles do not have a consistent textual description in their section, as it comprises a lot of special characters and formula. Therefore, we decided to use Simple English Wikipedia[4] (SimpleWiki) instead of the more extensive but also more complex English Wikipedia.

SimpleWiki is the same as the normal Wikipedia, except that it aims to convey complex matters with simple textual description, such that everyone can understand the article. This leads to texts that use simple words and grammar, are more coherent and do only require basic mathematical knowledge.

We have implemented a Web crawler, that takes a list of all available article IDs and shuffles them to download articles in a random order. The crawler can be interrupted and relaunched at any time. The articles are downloaded such that their hierarchical section structure is preserved. For each section (or subsection) the text is stored (formula are ignored). Additionally, lists and keywords are extracted as well as enclosed images. All graphics[5] are converted to the JPEG format. Alpha blending is used to resolve transparency by putting the graphic on top of a white background. SVG graphics are rasterized and for GIF graphics only the first frame is considered. To ensure memory efficiency, all images are down-sampled resp. up-sampled using bicubic interpolation. The original aspect ratio will be kept, but the smaller dimension is set to 346 pixels. The gathered images are stored in its original server folder hierarchy, whereas the top folder has as name the first letter in the MD5 hash of the image filename and the image-containing subfolder has as name the first two letter of the same hash.

The articles themselves are stored in JSON format, which are all stored in a JSONL file. The exact article format can be viewed in appendix A. Currently, the dataset consists of 3764 articles containing 2999 images. As mentioned before, the Web crawler can be restarted at any point to gather additional articles.

Even though the text structure can be assumed to be much simpler as in the normal Wikipedia, figure A.1c shows that the structural sentence complexity is more similar to the one of the *BBC News Database* than the one of the *MS COCO* dataset. Figure A.2b shows, that many sections convey their content in a very short form, which is favorable, as the comprehension of short texts is easier.

Before we close the section, we want to mention an alternative dataset that might replace or enhance a collection of datasets as we use. Villegas and Paredes [63] published an automatically collected image-text dataset from crawling webpages on the Internet. The dataset consists of $250,000$ images together with their captions and surrounding text. Just like many others, they distributed feature vectors instead of the raw data to avoid copyright issues. Even though working with such samples is the overall goal of this thesis, there are several reasons why we consider such collections as unsuitable for a first system. First of all, the data is extremely noisy, as they are from arbitrary websites and the text extraction is heuristically. On the other hand, the intentions that we associated with each of the above datasets can not be met. For instance, credential sources to learn the desired background knowledge are missing. Furthermore, many samples might use colloquial language. Note, all the datasets we utilize do not incorporate colloquial language (e.g. from social media sources), as it was thought it would induce a too high level of complexity for the time and computational power available in this thesis.

[4] `https://simple.wikipedia.org`

[5] As almost all graphics are in JPEG, PNG, SVG or GIF format, graphics of other formats are ignored.

4.2. Quantifying Human Intuition

In order to guide a learning process such that it perceives co-occurring image-text pairs in the same way as humans do, annotated training samples are required. Before we state the labels that have been hand-crafted to serve the training supervision, we aim to convey the intuition behind the two annotated characteristics, namely *Mutual Information* and *Semantic Correlation*. In the following, we will try to clarify what we are trying to achieve with our annotations and we will justify them with methodologies borrowed from information theory.

Our first attempts to find a proper annotations have been similar to many of the works presented in section 2.2, as we tried to quantify an intuitively estimated *Semantic Correlation*. Consequently, the initial goal was to label the *correlation* of documents[6] by a correlation term that ranges from a negative number to a positive (e.g. [-1, 1]). Well, such a correlation coefficient is not applicable in our case, since we do not measure the correlation of two random variables which can be viewed as self-contained entities. Instead, each random variable (each modality) captures an abundance of information, where some of this information might be intuitively positively or negatively correlated.

To further investigate this problem, a quick recap of basic definitions taken from information theory is required. In information theory, the behavior of a communication channel is described with mathematical tools (e.g. statistics). The most basic model is to think of a sender that encodes information. This encoded information is subject to noise during the transmission on a channel. It does depend on the occurred perturbations and the encoding, whether the receiver is able to decode the original information. The information, that is encoded in a transmitted message, shall reduce the uncertainty on the receiver side. If the received message can be fully predicted by the receiver, then the message does not contain any information. In other words, the more unlikely an information is, the higher is its *information content*.

Entropy measures the information emitted by a sender. It describes the average amount of information, that is revealed by a received message X, where X is an unknown message (random variable). Hence, entropy can be computed by summing over all possible messages

$$H(X) = -\sum_{x} p(X = x) \log p(X = x) \qquad . \tag{4.1}$$

We can think of X as an image or a text. Recall, that we intuitively attempted to estimate a correlation. In information theory, a correlation is mostly expressed by measures such as Mutual Information or the Kullback-Leibler divergence. We stick to Mutual Information, since it is more intuitive, as we will see later.

Mutual Information describes the amount of information, that is shared by two received messages X and Y on average

$$\begin{aligned} MI(X, Y) &= H(X) - H(X|Y) \\ &= H(X) + H(Y) - H(X, Y) \qquad . \end{aligned} \tag{4.2}$$

Equation 4.2 can be interpreted as the average information that can be taken from X

[6] The term *document* refers to a multimedia document (e.g. a document containing textual and visual objects). Given the context of this work, such a multimedia document usually comprises a text enclosed by an image.

A yellow circle with a diameter of 2cm.

Figure 4.2.: Example of a text and an image, that convey the same information

minus the average amount of information that Y does not reveal about X. So, if Y fully describes X, then $MI(X,Y) = H(X)$. Contrarily, if Y does not reveal any information about X (meaning $H(X|Y) = H(X)$), then the Mutual Information is zero. However, this measurement is not interesting to accomplish our goal, since we only consider a certain text resp. image. This can be achieved by the so called Pointwise Mutual Information. Let $X = x$ and $Y = y$, then the Pointwise Mutual Information is defined as

$$pmi(x,y) = \log \frac{p(x,y)}{p(x)p(y)} \quad . \tag{4.3}$$

First, it has to be clarified what the desired intuition is in order to find out whether equation 4.3 matches it. While having the information theoretical background in mind, a few examples shall be considered.

As highlighted above, all that has to be known to measure information is the probability distribution that describes how the sender ejects messages. In the context of this work two senders have to be considered, one that emits images and one that emits texts. Each of the following examples should emphasize properties of these probability spaces as well as their differences.

Example 4.2.1. *Figure 4.2 shows the rare case, that a text and an image actually depict the same information. However, it is easier for humans to perceive attributes such as shape and color from the image, whereas an attribute such as the exact size is easier to read from a text rather than from an image. There are two major takeaways from this example. Firstly, some information can be expressed in either of both modalities. Though, it is not clear yet, whether this accounts for all kinds of information. This means, that images and texts can actually share information. Secondly, different modalities can make the same information differently difficult to perceive. This leads to a natural usage where the depicted information shall complement one another such that each modality depicts partially unique information, that is easier to read from it than from other modalities.*

Example 4.2.2. *Consider the following simple sentence.*

Leibniz was born in 1646.

It is not possible to display the same information as a photograph without utilizing natural language. Every attempt to display this fact would result in an image that contains a massive amount of additional information such as facial details of Leibniz and historic events that are unique to that year. From this it follows that equal information content in different modalities is not always possible. For this example it can be concluded, that there are image-text pairs, where the information encoded in the text is always a subset of the information depicted in the image. Leading to the assumption, that the sought probability spaces are different. Hence, probabilities estimated for one modality cannot be transferred to another

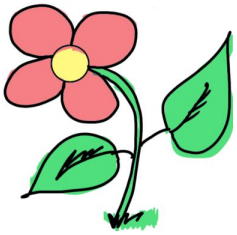


Figure 4.3.: The exact shape of the depicted leaves can not be described in a finite text without the usage of math.

one. I.e. the assumption from [66] stated in section 2.2, that the distribution of latent hidden topics might be equal, does not hold in general.

Example 4.2.3. *Figure 4.3 depicts a cartoonish flower. Hence, the graphic is not as complicated and detailed as a photograph (e.g. missing textures). However, describing the exact shape of the leaves in a text would be an inherently complex task if mathematical tools are not available. Once again it can be stated, that the probability spaces to emit information are obviously not completely the same. This observation excludes any model that would reduce the problem to a single modality. For instance, a model that translates images into text (e.g. an image description generation algorithm) and then compares the semantics of two entities from the same modality (e.g. paragraph detection) would not work.*

It should be acknowledged, that different modalities are intentionally not used to convey the same information. Moreover, a modality shall contain the information that is not or not as easily expressible in another one.

Example 4.2.4. *In this example it should be illuminated, how tiny perturbations can change the overall semantics. Therefore, we consider figure 4.4. All of these images share a massive amount of information (i.e. there is a lot of mutual information). If the graphics would be photographs, there would be even more shared information due to complex textures, shapes and backgrounds. Yet, all of these figures differ in a certain detail.*

If a group of people would have to decide which two of these figures are more similar, most of them would decide on figures 4.4a and 4.4c. This is due to the fact, that some details are more important for the grasped semantics than others. For instance, a human generated caption for figure 4.4a would probably contain the term "happy family", but presumably not the term "blue plate".

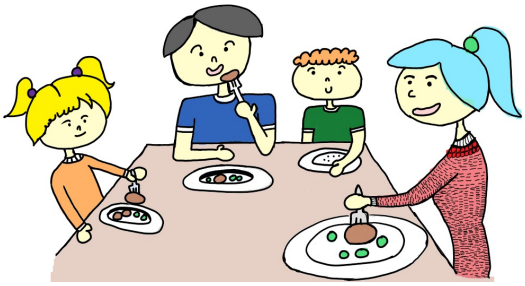
The following text could be associated with one of the pictures.

> *A family of four is sitting at a table to have a warm meal. They are all talking elaborately about their day.*

As usual, the text and image complement one another, as the text merely states what is visual obvious but provides additional insights in the temporal occurrences of the depicted situation. Though, figure 4.4b does intuitively not fit to the text, as experience teaches that a sad mood comes along with less talking. Note, that the text contains no direct hints about the facial expressions or the mood of the people. On the other hand, figure 4.4a and 4.4c are equally good choices to enhance the text.

(a) Happy facial expressions.

(b) Sad facial expressions.



(c) Color of plates has changed.

Figure 4.4.: All pictures show a dining family, but they differ in a certain detail. In figure (a) and (b) the attribute *happy* has been changed to *sad*, whereas in figure (a) and (c) the attribute *blue plate* has been changed to *white plate*. The change occurring in (a) and (b) modifies the intuitive semantics, while the change from (a) to (c) is negligible.

If the information conveyed by each entity of a pair contradicts each other, as the text from example 4.2 combined with figure 4.4b, a negative *correlation* would be desirable to express the situation, even if they have a high amount of mutual information. Such an outcome is expressible by equation 4.3. Experience would cause $p(x, y)$ to converge to zero, where x could be the text and y the image. But as x and y are valid representations of their modalities if considered separately, it is clear that the marginal probabilities are substantially greater than the joint probability, i.e. $p(x)p(y) > p(x, y)$.

However, to compute the Pointwise Mutual Information proper marginal and joint probability estimations have to be known. Assume an unsupervised scheme that has access to a large collection of documents with a massive amount of distinct sources. Could such a scheme estimate a meaningful probability for a text x?

$$x = \text{“}\underbrace{\text{I}}_{w_1}\ \underbrace{\text{am}}_{w_2}\ \underbrace{\text{studying}}_{w_3}\ \underbrace{\text{for}}_{w_4}\ \underbrace{\text{the}}_{w_5}\ \underbrace{\text{next}}_{w_6}\ \underbrace{\text{exam.}}_{w_7}\text{”}$$

$$p(x = w_1 w_2 w_3 w_4 w_5 w_6 w_7)?$$

An effective approach to build language models has been Naïve Bayes. But one should keep in mind, that subtle difference can change the overall meaning of an entity. Therefore, any kind of independence assumption would destroy the estimations. On account of this and the prior explanations, we claim that it is impossible with any statistical model to estimate probabilities from such diverse sources such that a computed score based on equation 4.3 can match our intuition.

Moreover, exact correlation scores are not desirable, since they cannot be verified and do not mirror human capabilities. In addition, we argue that human intuition is not interpretable as a single score. For instance, a poem about peace combined with an image that depicts an area of conflict should be distinguishable from the above example for negative correlation. This is due to the fact that both entities do not share information, whereas the previous example possesses a huge amount of mutual information.

Therefore, we propose two categorical resp. ordinal measures: *Mutual Information* (MI) and *Semantic Correlation* (SC). Note, that the measure MI that we defined is not the same as the information theoretical measure defined in equation 4.2. There are a few things to acknowledge in order to understand the meaning of both measurements. Most importantly, negative correlation does only exist on a semantic level. Consider two sources that would emit figures such as figure 4.4a and 4.4b, that are equal except for a few details. Or to be more precise, the joint entropy of both sources is nearly as high as the marginal entropy of each individual source, since there is only a small amount of information that has to be conveyed to describe one of the images given that the other image is known. Obviously, it is easier to decide whether two entities share a lot of information than to decide whether their differences can cause a major shift in their semantic interpretation. Supplementary, the amount of shared information is not particularly important to judge semantic correlation. For instance, the concept *spring* can be equally well represented by an image of a single flower as by an image depicting the annual spring festival. Both of these images do not share information (i.e. SC is high and MI is low). Otherwise, if one of the pictures is perturbed such that snow is visible, then SC should be low, whereas MI can be still estimated as high compared to the original image. From this it follows, that both measurements can be separately determined. There is no mutual dependence among them.

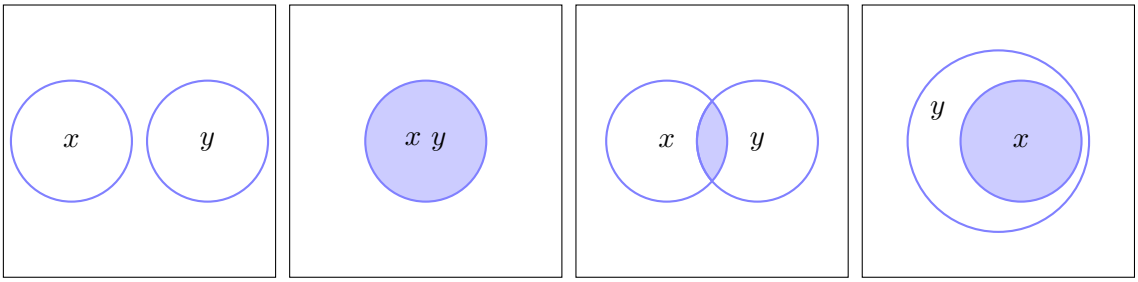
Nonetheless, SC as well as MI are crucial for our task and further applications. Whereas SC describes how well two entities fit to each other, MI predicts how much complementary information can be taken from another entity. For instance, consider the task of automatically retrieving a few documents that contain as much information about a concept as possible. This could be achieved by querying documents with high SC and low MI.

In the following two subsections, we will further outline the measurement separately and explain the annotations we have chosen. Subsequently, we will explain the annotation process itself.

4.2.1. Annotations to adequately express Mutual Information

Here we want to further elaborate the meaning of *Mutual Information* (MI) in the context of our annotations. There are 5 basic cases to consider (depicted in figure 4.5). The by far most prominent case is depicted in figure 4.5c as $x \cap y$. This case sketches how co-occurring image-text pairs are naturally used, i.e. they share information to some extend, but the overall intention is to provide additional information through each modality. Therefore, we decided to pay extra attention to this case.

Table 4.1 presents the labels we have chosen together with their description. To simplify the annotation process, labels are typically associated with two descriptions. The first one describes the properties an image should have under consideration of the text, whereas the other one describes the constraints a text has to satisfy under consideration of the image.



(a) $x \cap y = \emptyset$ (b) $x = y$ (c) $x \cap y$ (d) $x \subset y$ (or $y \subset x$)

Figure 4.5.: Possible MI states of a text x and image y.

Label	**Description**
0 – $T \cap I = \emptyset$	The content of the text and the image do not overlap. Not even a contextual relationship can be built.
1 – $T = I$	The image depicts solely situations/actions/characteristics that are mentioned in the text. Nothing is depicted which can not be concluded from the text.
	The text describes exclusively the image. The content description is complete and rich in detail and all outstanding characteristics are mentioned. (The text alone shall be sufficient to be able to imagine the depicted scenery without prior knowledge of the image.)
2 – $T \cap I$	The image depicts, subjectively seen, more important characteristics than the text describes. Though, part of the image is described exhaustively in the text.
	The text describes a process/story, where an excerpt is depicted in the image. Particularly, the text references to details in the image.
3 – $T \cap I$	The image can be viewed as a summary of the text content, but only by neglecting many of the details. (e.g. photo of the conflict area, whereof the current news story is reporting about)
	The text sketches the image but on an abstract level such that many different pictures would be similarly well described by the text.
4 – $T \cap I$	The image concentrates on a detail of the text. **or** The content of the image is not explicitly mentioned in the text. However, due to background knowledge the image can be brought into context.
	The text only describes a part of the image. **or** The text describes a storyline which is usually associated with the image due to background knowledge.

Continued on next page

Continued from previous page

Label	Description
5 – $T \cap I$	The image depicts one of the aspects of the text. (e.g. a typical representation of a keyword is shown)
	The text is mainly concerned about a minor detail of the image.
6 – $T \subset I$	The image depicts, subjectively seen, more important characteristics than the text describes.
	Everything mentioned in the text can be related to the image.
7 – $T \supset I$	The image depicts solely situations/actions/characteristics that are mentioned in the text. Nothing is depicted which can not be concluded from the text.
	The text describes a process/story, where an excerpt is depicted in the image.

Table 4.1.: Annotation guidelines for *Mutual Information*

Note, that label 4 and 5 incorporate background knowledge. This is a tricky part that is caused by the varying visualness of concepts [68]. For instance, the word *spring* could be represented by an image of a flower meadow. It is hard to objectively quantify mutual information of such concepts. Therefore, we decided to rate them with low MI rather than none. In other words, if a concept does not have an obvious and unambiguous representation within the other modality, then we quantify it with low MI, if the semantic correlation between them is considered as high. Note, that this does not mean that the semantic correlation between the considered entities must be high, as each of them can contain an arbitrary number of concepts. Example 4.2.5 presents such a case.

Example 4.2.5. *Figure 4.6 depicts a short news article and an image that might be enclosed to it. The scenery described by the text is conflictive with the one shown in the image. Hence, SC should be negative. Interestingly, the concept "spring" appears in the news article. This concept has no obvious visual representation. However, figure 4.6b has intuitively a high semantic correlation with that concept. Therefore, it can be assumed that the shown co-occurrence it is not completely meaningless. Probably, the author of the article wanted to depict how it usually should look like. This can be captured by a low mutual information (e.g. 5).*

This example illuminates a complex relation between an image and a text, that is far easier to quantify with two measurements rather than one.

4.2.2. Annotations to adequately express Semantic Correlation

This subsection presents the annotations used to quantify *Semantic Correlation* (SC). An extensive amount of background knowledge is required in order to properly grasp semantic correlation. The ability to infer the central theme of an entity and to omit insignificant details is inevitable. Additionally, the central themes of two different modalities have to be aligned in order to quantify their relation. As this is also a challenging task for humans,

Nature still doesn't show that spring has already arrived. The weather remains wet and cold, partly frosty.

(a) News article, that implies muddy forests without blooming flowers.

(b) Blooming flowers in spring.

Figure 4.6.: Example of an image-text pair with contradicting semantics (negative SC). Nonetheless, the *low-visual* concept spring does appear in both, causing low MI.

we decided to annotate on a small scale that requires only approximate decisions. The annotation guidelines are presented in table 4.2.

Label	**Description**
1.0	The meaning expressed by the image and the one expressed by the text are almost identical.
0.5	There are slight variances in the expressed semantics. Although, the impressions made by each of them are quite similar.
0.0	The semantics of image and text are not correlated to each other. Note, even if both share mutual information, the meaning which is expressed by them might yet be unrelated.
-0.5	The image depicts a scenery that fits to the text, but was incorrectly interpreted. **or** A critical detail of the image leads to an opposite interpretation of text and image. (e.g. a campaign event of the Republic party is shown, whereas the text reports about a campaign event of the Democratic party) **or** Keywords from the text can be identified as actions or objects in the image, but they appear in a whole different context. **or** One of the main story characteristics of the text is incorrectly depicted by the image. (e.g. a male protagonist is depicted by a female person) **or**

Continued on next page

Continued from previous page

Label	Description
	The actual content of the text has no relation to the image, but the text refers to some details which are either depicted as described or at least depicted similarly in the image. (e.g. a text about nutrient-rich soil in forests has an image that depicts an indoor plant)
-1.0	The text describes the opposite of what is depicted in the image.

Table 4.2.: Annotation guidelines for *Semantic Correlation*

4.2.3. Annotation Process

Annotations have been gathered for subsets of all three datasets described in section 4.1. While the annotations for the *MS COCO* dataset have been heuristically generated, the other two datasets have been manually annotated by the author of this work. In order to do so, two tools have been developed to assist the annotation process: the *bbcAnnotator* (figure 4.7) and the *wikiAnnotator* (figure 4.8).

Both annotation tools have the ability to read the dataset and to store the made annotations into a JSONL file. Before we mention features that are unique to each tool, we want to shortly summarize some shared properties. The tools provide cross-platform support for Linux, Windows and Mac OS (only tested on Linux and Windows) and are implemented in Java 8. To prevent data loss, the annotation tools instantly backup changed annotations using HSQLDB. The backup is locally stored and reread on program start to allow that the annotation process can be continued from the last modified annotation state. Each tool provides a variable wide window to display an image text pair. The text window highlights headings and allows to change the font size (Ctrl + Mouse Scrolling). Images are displayed together with their caption, as the caption might provide the necessary background knowledge to annotators, that is necessary to map the semantics. Pagination allows to navigate between image-text pairs, either by selecting neighboring pairs or by jumping to a predefined position. The user might annotate the following properties for the currently depicted image-text pair:

- **Mutual Information:** The user should select a label to rate MI according to the annotation guidelines given by table 4.1.
- **Semantic Correlation:** The user should select a label to rate SC according to the annotation guidelines given by table 4.2.
- **Relevant Text Snippets:** The user can mark and add text snippets from the text window that directly refer to the image or describe content visible in the image (does not have to be comprehensive; may describe only a part of the image).
- **Image Type:** The user should further define the type of graphical representation that is visible. Therefore, he/she can choose one of the following image types: *"Chart – Bar Chart", "Chart – Flow Chart", "Chart – Histogram", "Chart – Line Chart", "Chart – Pie Chart", "Chart – Tree Chart", "Chart – Other", "Drawing", "Graph", "Map", "Photograph", "Table".*

Figure 4.7.: Screenshot of the *bbcAnnotator*. The content had to be blurred due to copyright reasons. The sample where the content had been taken from is called "Toll_rises_after_Chinese_floods_271" and is located in the training set of the *BBC News Database* [19].

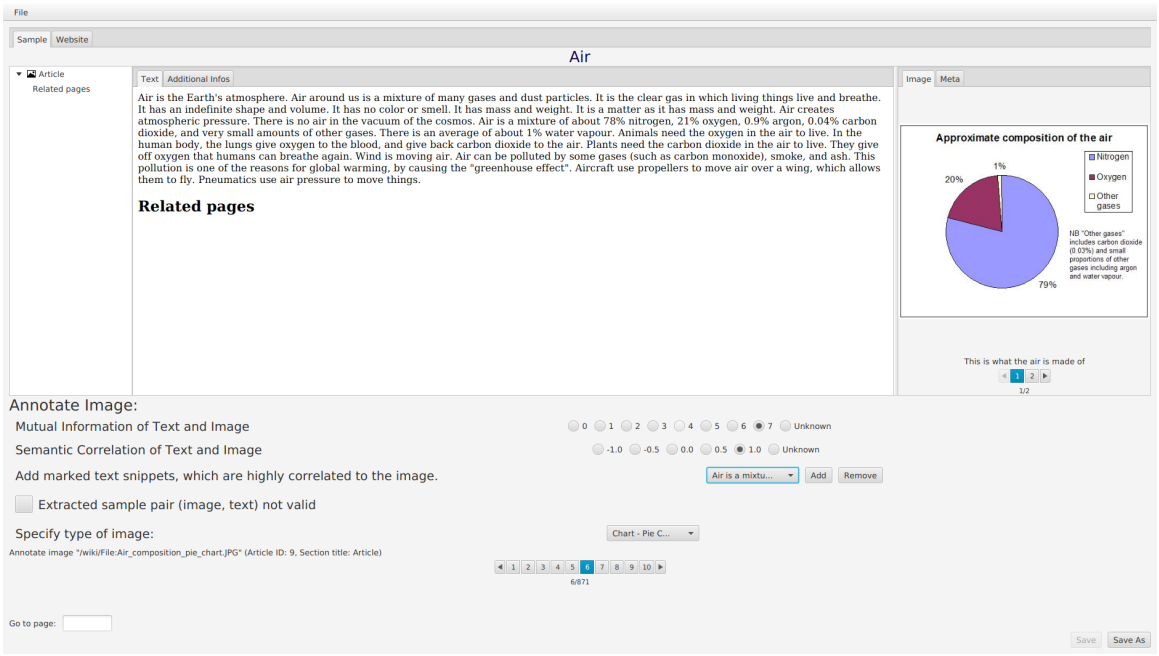


Figure 4.8.: Screenshot of the *wikiAnnotator*. The content originated from the article `https://simple.wikipedia.org/wiki/Air`. Textual and graphical content are released under CC BY-SA (`https://creativecommons.org/licenses/by-sa/3.0/`).

```
{
  "sc":0.0,
  "mi":0,
  "snippets":[ ... ],
  "type":"",
  "name":"",
  "train":true
}
```

Fragment 4.1: JSON object that defines the annotations for a sample made with the *bbcAnnotator*

To export the annotations, the user may save them as a JSONL file. Samples annotated with the *bbcAnnotator* will have a format as defined by the JSON object in fragment 4.1. The following listing will describe each property of the depicted object.

- **sc:** The SC label that has been assigned to the sample according to table 4.2.
- **mi:** The MI label that has been assigned to the sample according to table 4.1.
- **snippets:** A list of strings, that have been considered by the user as direct reference to the image content.
- **type:** A string identifying the type of graphic (the chosen *Image Type*).
- **name:** The name of the sample.
- **train:** A boolean flag, stating whether the sample has been originally taken from the training or test set.

The *wikiAnnotator* has some extra features due to the varying article structure. In addition, it provides mechanisms to validate a depicted sample, as the dataset has been newly generated. Every page depicts another article, that may contain several images. However, for each image and its associated text a separate annotation has to be created. As mentioned in section 4.1.3, images that co-occur with an article summary are associated with the text of the whole article. Other images are only associated with the text of the specific section in which they occur. The *wikiAnnotator* has an additional annotation option, which allows to mark an extracted sample as invalid. This might be due to several reasons, that could not be handled during the automatic retrieval. For instance, lists are omitted from the section text. However, the remaining text in a section might be meaningless without the filtered content. Hence, the user possesses several features to evaluate the validity of a sample in case the text appears strange or inconsistent. First, one may see the original website for a direct comparison. In addition, one may review the filtered content or the identified keywords under the tab *Additional Infos*. In the image view, the tab *Meta* allows one to see the meta data (e.g. license information), that has been retrieved together with the image. Furthermore, one can inspect the keywords, that have been identified in the caption.

```
{
  "sc":0.0,
  "mi":0,
```

```
  "snippets":[ ... ],
  "type":"",
  "id":1,
  "name":"/wiki/File:filename.jpg",
  "section":"",
  "valid":true
}
```

Fragment 4.2: JSON object that defines the annotations for a sample made with the *wikiAnnotator*

Similar to the *bbcAnnotator*, the annotations can be exported as a JSONL file. The annotations of a single sentence are defined in fragment 4.2. The following listing defines the properties that differ from fragment 4.1.

- **id:** The ID of the article, where the sample appears in.
- **name:** The name of the image file.
- **section:** The name of the section, that the sample has been associated with. If associated with the whole article, then this field is assigned with the value *"Article"*.
- **valid:** A boolean property that states, whether the sample is a valid sample or not. If not, then it should be discarded from further processing.

Now, as the annotation tools have been clarified, we want to further elaborate how the annotations have been retrieved. In total **761** samples have been manually annotated using the above explained annotation tools. The exact distribution among the considered datasets and image types is depicted in table 4.3.

	BBC News Database	**SimpleWiki dataset**
Total	205	556
Photograph	200	397
Drawing	5	90
Map	–	29
Charts	–	7
Graph	–	6
Table	–	–
Invalid	–	26

Table 4.3.: Number of samples that have been annotated using the *bbcAnnotator* resp. *wikiAnnotator* plus their distribution among different image types.

All the annotations have been retrieved by the author of this thesis using the guidelines from table 4.1 and table 4.2.

Tables 4.4 and 4.5 show the label distributions for MI and SC labels on both datasets. It accounts for both datasets, that the overwhelming majority of samples are image-text pairs that share information, but mostly in terms of abstract concepts, that typically require background knowledge to get detected. However, as expected, there is a substantial difference in the semantic correlation for co-occurring pairs between these datasets. Pairs

Label	0	1	2	3	4	5	6	7
Total	16	–	1	6	91	91	–	–
Percentage	7.8	–	0.49	2.93	44.39	44.39	–	–

(a) Distribution of MI labels.

Label	-1.0	-0.5	0.0	0.5	1.0
Total	7	28	84	69	17
Percentage	3.41	13.66	40.98	33.66	8.29

(b) Distribution of SC labels.

Table 4.4.: Label distribution of the annotated samples from the *BBC News Database*.

Label	0	1	2	3	4	5	6	7
Total	28	–	6	46	376	66	–	8
Percentage	5.28	–	1.13	8.68	70.94	12.45	–	1.51

(a) Distribution of MI labels.

Label	-1.0	-0.5	0.0	0.5	1.0
Total	–	3	26	70	431
Percentage	–	0.57	4.91	13.21	81.32

(b) Distribution of SC labels.

Table 4.5.: Label distribution of the annotated samples from the *SimpleWiki* dataset.

in the *SimpleWiki* dataset are generally highly correlated with an average semantic correlation of **0.88** (and a mean value of 1.0). On the other hand, the semantic correlation for the *BBC News Database* is rather low, having an average value of **0.15** (and a mean value of 0.0). This observation highlights the difficulties of a news dataset and shows the inherent complexity that has to be dealt with when interpreting the relation of image-text pairs occurring in a natural setting.

This can also be seen when considering the number of texts that contain direct references to the associated graphic or sentences that describe high-visual content of the graphic. 166 (≈31%) *SimpleWiki* samples contain such relevant text snippets compared to only 9 (≈4%) of the news samples.

Appendix B compiles the presented annotation statistics for separate image types.

In order to reduce the effects caused by the strong label imbalance, we heuristically generated samples with high SC and high MI from captioning datasets, namely the *MS COCO* dataset. Therefore, we randomly picked a 100 images and their reference sentence from the *MS COCO* validation set. All of these image-text pairs are annotated with maximum semantic correlation (SC is 1.0). The texts are assumed to be fully concerned with their assigned image. Although, they do not exhaustively describe the image, as they have been generated with the expectation to describe the most prominent content. Therefore, we assume that a reference sentence describes the content only partly (MI is 6). As all *MS COCO* samples are annotated with the same labels (SC – 1.0, MI – 6), some precautions had to be taken to avoid that the classifier learns the text length when seeing MI label 6. Therefore, we have chosen a variable-sized subset of all 5 reference captions associated with an image and concatenated them to a text.

In summary, we used **834** image-text pairs for classifier training.

4.2.3.1. A Critical Look at the Annotation Process

The are several steps in the annotation process that can be criticized. First of all, it is not possible to make any statements about the adequateness of the made annotations as they all have been collected by a single person. The label distribution might be extremely biased resp. distorted as our assumption of a uniform intuition among humans can not be verified without substantial measurements on inter-annotation agreement rates.

Furthermore, it might be that the defined labels to not properly represent the designed measures MI and SC. Especially the labels 6 and 7 might be unrealistic, because each entity contains usually information that is not depicted in an entity of another modality (e.g. shapes and textures in an image). Without these two labels, the annotation process would be simplified. Additionally, the labeling would represent an ordinal scale and the classification problem could be phrased as a regression problem, similar to the task of predicting SC. This can be beneficial, as label differences are naturally encoded in a regression learning scheme, i.e. misclassifications among close labels are weaker punished than those among distant labels.

We want to state a few more drawbacks, that we have observed. Firstly, the annotation tools should allow to distinguish between artistic and technical drawings, as these are extremely different graphic types, that are used for significant distinct purposes. Another possible drawback is the label assignment we made for heuristically generated annotations. It might be advantageous to label captioning samples with high MI (e.g. label 1 or 2) instead of considering the information as inclusive. This is for the same reasons as mentioned above, a simpler labeling might be more expressive. Furthermore, captions usually contain direct references and share concepts with the image, that have a high visualness. These are exactly the concepts that should be assigned with high MI values. Supplementary, the concatenated sentences, which form the text associated with an image, do not represent a fluent and consistent text. Moreover, they repeat the same information over and over again, since most human annotators recognize the same prominent objects/actions, which they summarize in a caption. In other words, most captions that are assigned with an image can be considered as paraphrases. However, using a single caption as text would probably lead to overfitting when assigning those samples with MI label 6, as this label does not occur in the other two datasets.

Lastly, we want to point out, that it is much more complicated to assign negative semantic correlation. The reason for that is, that an image, which is completely unrelated to its associated text, is intuitively perceived as negatively correlated. It does require extra effort to disambiguate those cases.

4.3. Automatic Generalization of Concepts drawn from the World

As it has been highlighted during the description of our measurements (MI and SC), extensive knowledge about the world is required to quantify the co-occurrence of images and texts. More precisely, concepts have to be generalized within and across modalities. For instance, synonyms and paraphrases in sentences resp. texts have to be identified as well as objects/actions in images. These generalized concepts within modalities have to be mapped in between modalities. For instance, the *high visual* word “elephant” should be mapped to the object elephant resp. the *low visual* word “spring” should be mapped to scenes that are unique to this time of the year (e.g. a blooming flower meadow).

A supervised scheme would require an infeasible amount of annotated data to accomplish

this goal. Therefore, we aim to learn this ability through an unsupervised scheme by using an autoencoder. An autoencoder is a deep learning architecture that attempts to output its own input. For instance, this could be done by leaning the identity function. However, this is only possible if the hidden representations are expressive enough to hold the same input encoding. For this reason, autoencoders are typically used to learn a compression. Hence, an encoder network compresses the input onto a lower dimensional representation, that contains less redundant information. Subsequently, a decoder network decompresses this intermediate representation back to the original input encoding. The compression and decompression algorithm (encoder and decoder network) have to be learned. The intermediate representation (the compressed input) can be considered as a feature vector, describing the complete input in a lower-dimensional space. To achieve this, the encoder has to generalize concepts (e.g. objects, shapes, poses in images) that are available in the input distribution.

The same behavior shall be accomplished with the autoencoder architecture we designed. The autoencoder and classifier (cp. section 4.4) are implemented in *Python 3* using the deep learning framework *Tensorflow*[7]. We used the implementation of Vinyals et al. [64] as a starting point to implement our own model, as we intended to use the same image encoding network *InceptionV3* [60]. The implementation of Vinyals et al. [64] is currently considered as the best system for generating image captions. The simplified architecture of the system is depicted in figure 2.1b. It can be summarized as follows. An image embedding $\mathbf{i}_e$ is generated via a fully connected layer from the output of a deep CNN, namely *InceptionV3*. Words are mapped onto word embeddings $\mathbf{w}_e$. These embeddings are learned during training. A $LSTM(\mathbf{z}_{l-1}, \mathbf{c})$ is used to predict a future token, where $\mathbf{z}_{l-1}$ denotes the input to the LSTM layer and $\mathbf{c}$ its memory cell (initialized as $\mathbf{c} = \mathbf{0}$). A fully-connected layer FC is used to determine the predicted token. Thus, the layer has as many outputs as the number of words in the vocabulary. To ensure valid start and end conditions[8], each caption is augmented with a specific start and end token. Algorithm 4 illustrates how the system would work during training (the argmax function is actually replaced by a softmax layer, whose output is compared with the ground truth via a cross-entropy loss function).

Input: $\mathbf{i}_e$, $\mathbf{w}_e(1)$, …, $\mathbf{w}_e(T-1)$
Output: $w^*(2)$, …, $w^*(T)$

$\mathbf{c}(0) = \mathbf{0}$; // Initialization of LSTM cell state

$\_, \mathbf{c}(1) = LSTM\big(\mathbf{i}_e, \mathbf{c}(0)\big)$; // Compute beginning cell state, that encodes image

for *t=1* **to** *T-1* **do**

  /* Predict next token given the image and all tokens up to time step t */

  $\mathbf{w}_e^*(t+1), \mathbf{c}(t+1) = LSTM\big(\mathbf{w}_e(t), \mathbf{c}(t)\big)$;

  /* Map predicted token embedding to actual vocabulary index */

  $w^*(t+1) = \text{argmax}\Big(FC\big(\mathbf{w}_e^*(t+1)\big)\Big)$;

end

/* The predicted tokens can subsequently be used to compare the predicted caption to the one associated with the current training sample (compute a loss). */

Algorithm 4: Simplified presentation of the caption generation algorithm from [64] during training.

[7] `https://www.tensorflow.org/`

[8] If the start token is read, the LSTM should predict the first word in the caption. The caption is complete as soon as the end token is predicted.

We have kept from this network architecture the generation of image embeddings via *InceptionV3* and word embeddings[9]. The rest has been exchanged by our autoencoder model as described in subsection 4.3.2. Before we will explain the autoencoder model, we will explain how validation and training samples have been prepared for it.

4.3.1. Input Preprocessing for the Autoencoder

Image-text pairs from the *MS COCO*, *BBC News Database* and *SimpleWiki* dataset have been combined to form a multifaceted dataset. Therefore, all $40,504$ images together with their 5 reference sentences have been taken from the *MS COCO* validation set to generate a total of $202,654$ image-text pairs. These have been merged with $3,361$ pairs from the *BBC News Database* and $2,999$ pairs from the *SimpleWiki* dataset.

This leads to a total of $\mathbf{209,014}$ samples, that are randomly distributed among training $(190,202)$, validation $(6,270)$ and test $(12,542)$ set.

All texts have been split into sentences using the *Punkt Sentence Tokenizer* from the *NLTK*[10] package. Subsequently, sentences have been split into tokens using NLTK its standard word tokenizer.

When considering this raw tokens, the vocabulary is extremely large due to the complex language used in encyclopedia and news articles. Table 4.6 shows sample vocabulary sizes for different constraints on word occurrences. Such a constraint is typically made to reduce the vocabulary size by omitting rare words. Hence, all words that have less than o occurrences in the training set are replaced by a special *UNKNOWN* token. For small values of o, this typically does not result in a significant loss of information, since it is not possible to infer meaningful embeddings for those tokens. For comparison, Vinyals et al. [64] had a vocabulary size of $\approx 12,000$ for $o = 4$, generated from the captions in the *MS COCO* training set plus nearly 90% of the *MS COCO* validation set.

Min. #occurrences	4	5	6	7	8	9	10
Vocabulary size	$25,885$	$22,427$	$20,156$	$18,244$	$16,866$	$15,721$	$14,767$

Table 4.6.: Evolution of vocabulary size when increasing the minimum number of occurrences required for a word in the training set to be included in the vocabulary.

We have set o to 10. To allow the learning of semantically meaningful word vectors, the vocabulary size has been further reduced with methods that avoid severe distortions of the text content. To achieve this, all tokens have been lemmatized using the WordNet Lemmatizer from NLTK. Furthermore, a dictionary has been used to translate words from British English to American English for all samples taken from the *BBC News Database*. Due to this measurements, the vocabulary could be reduced from its original size of $59,349$ tokens to a final size of $\mathbf{12,591}$.

To generate a proper initialization of word embeddings, when training the autoencoder, a *Word2Vec* model (cp. subsection 3.3.4, [49]) has been implemented[11].

Image preprocessing has been done as in [64]. Note, that the image preprocessing step is actually part of the autoencoder implementation and applied to each incoming image-text pair (irrespective whether the pair has been seen before). This technique is widely

[9] Except, that we initialize these word embeddings with pretrained word vectors.

[10] `http://www.nltk.org/`

[11] The *Word2Vec* model implementation is based on a tutorial available under `https://www.tensorflow.org/tutorials/word2vec/`; accessed 2016-12-22.

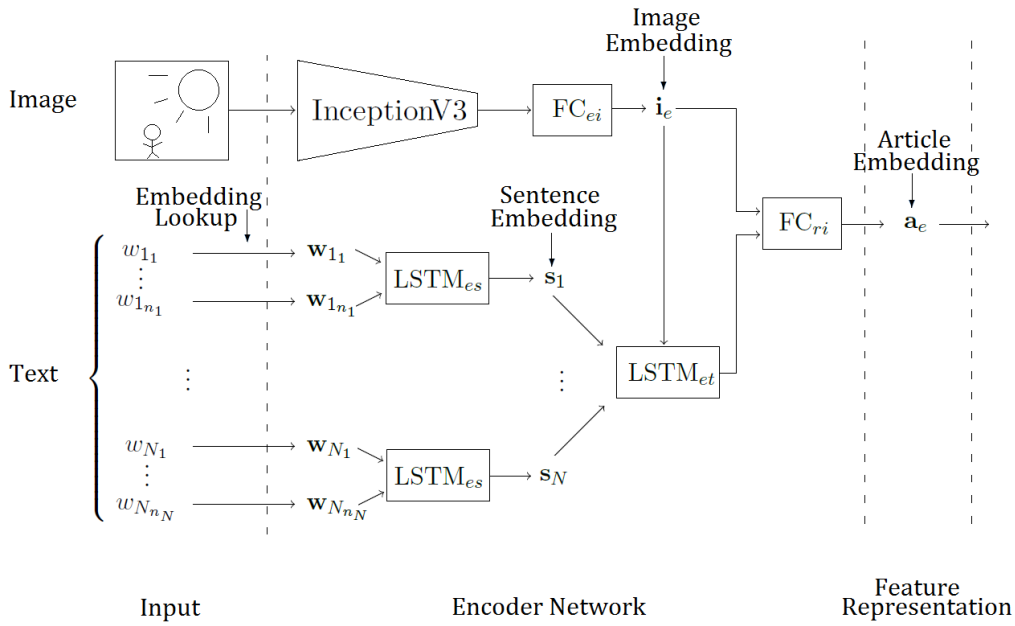


Figure 4.9.: The encoder network architecture of the implemented autoencoder. An image and variable-sized text are mapped onto an article embedding $\mathbf{a}_e$.

used to artificially increase the number of training samples and therewith as a method of self-regularization. All images are resized via bilinear interpolation to $h_R \times w_R$ dimensional images before random windows of size $h \times w$ are cropped from them[12]. Supplementary, random distortions are applied to the images to further increase the input variance. These distortions include perturbations of brightness, saturation, hue and contrast as well as random horizontal flips.

4.3.2. The Autoencoder Network Model

For efficient input processing, the input samples for the autoencoder are distributed among several binary files (shards). During training at least two of these shards ($\approx$ 800 samples) are hold in memory. Random samples are selected out of this pool to generate batches. Let the batch size be n_{bs}. In addition, let n_e denote the embedding size and let $\mathbf{V}$ be the vocabulary.

Figure 4.9 depicts the encoder network, that maps the input onto a feature representation. We call this feature representation *article embedding* $\mathbf{a}_e \in \mathbb{R}^{n_e}$, since the samples that shall be classified in a subsequent step originate from articles (news resp. encyclopedia articles). All weights are initialized according to an uniform distribution if not mentioned otherwise. The word embedding matrix $\mathbf{W}_{we} \in \mathbb{R}^{|\mathbf{V}| \times n_e}$ can be initialized with weights learned by a pretrained *Word2Vec* model. The part of the network that represents the *InceptionV3* model is initialized with weights from a pretrained model, to start training with meaningful image embeddings. The output $\mathbf{i}'_e$ of the InceptionV3 model is processed by a fully-connected layer FC_{ei} to produce an image embedding $\mathbf{i}_e \in \mathbb{R}^{n_e}$, that satisfies the dimensionality requirements of the autoencoder.

[12] We have set $h_R = w_R = 346$ and $h = w = 300$.

The text is processed using a hierarchical LSTM structure, that embeds individual sentences before it infers the semantics of their interplay. The network can adjust to dynamically sized sentences resp. text (though, sentences are constrained by a maximum length). Such a structure allows a more natural way of text processing, as it allows to consider sentences as self-contained entities. Furthermore, it is doubtable that a single LSTM layer can process a text as a sequence of all tokens. As we have mentioned in section 3.3.3, even LSTM cells have difficulties to maintain long-term dependencies if the input sequence goes beyond a few hundred items.

Let N be the number of sentences in the current sample and n_i be the number of tokens in sentence i. All sentences/texts in a batch are zero-padded to gain the size of the largest sample in that batch. Hence, a text T is a sequence of sentences which themselves are sequences of tokens:

$$T = \begin{bmatrix} & [w_{1_1}, \ldots, w_{1_{n_1}}], \ldots, [w_{N_1}, \ldots, w_{N_{n_N}}] & \end{bmatrix} \quad .$$

Each token w_i is stored as the index of the represented word in the vocabulary. Tokens are translated to word embeddings by a lookup operation. For the sake of simplicity, this operation can be considered as a matrix-vector product

$$\mathbf{w}_i = \mathbf{W}_{we}^T \hat{w}_i \quad , \tag{4.4}$$

where $\hat{w}_i$ denotes the one-hot encoding of the index w_i. Each sentence is subsequently encoded as a single embedding via an LSTM layer called LSTM_{es}. The sentence embedding $\mathbf{s}_i$ is the output of LSTM_{es} when processing the last non-padded token in the i-th sentence.

All LSTM layers in our networks use dropout for regularization. Dropout is an effective and efficient technique to prevent layers from overfitting. Dropout shall prevent that learning in a layer depends too strongly on single neurons. Therefore, each training iteration a subset of randomly selected neurons is gated to have a zero output, such that back-propagation (and therewith learning) is stopped through those neurons.

The sequence of sentence embeddings, that shall comprise the complete semantic meaning of a sentence, is processed to a single article embedding by an LSTM layer called LSTM_{et}. LSTM_{et} computes its initial cell state by looking at $\mathbf{i}_e$. Again, as article embedding $\mathbf{a}_e$ is considered the output of LSTM_{et} for the last non-padded word. Afterwards, an optional fully-connected layer FC_{ri} can be used to reconsider the image, as the image information might have gone lost in long texts.

The embedding vector $\mathbf{a}_e$ is now expected to be a compression of the whole sample. Ideally, $\mathbf{a}_e$ can be decompressed by the decoder network without loss of information compared to the original input. The basic architecture of the decoder network is depicted in figure 4.10. The decoder is split into two networks, that receive $\mathbf{a}_e$ as input.

The upper part of figure 4.10, the image decoding, will be considered first. The user can actually choose between three different image decoding architectures, where only option 3 is depicted in figure 4.10.

Option 1 is a very basic approach. The embedding $\mathbf{a}_e$ is mapped onto a large image through a fully-connected layer, which is then refined by a series of two convolutional layers. The major drawback of option 1 is the fully connected layer. Assume that the input image has size $h \times w$. If the FC layer maps $\mathbf{a}_e$ onto a $2h \times 2w$ image, then $n_e \cdot 4 \cdot h \cdot w$ weights have to be estimated only for this step. This is infeasible, for the amount of data we are using.

Option 2 uses a fully-connected layer to map $\mathbf{a}_e$ onto a small thumbnail. This thumbnail is horizontally and vertically replicated to reach an initial decoding of size $2h \times 2w$ as

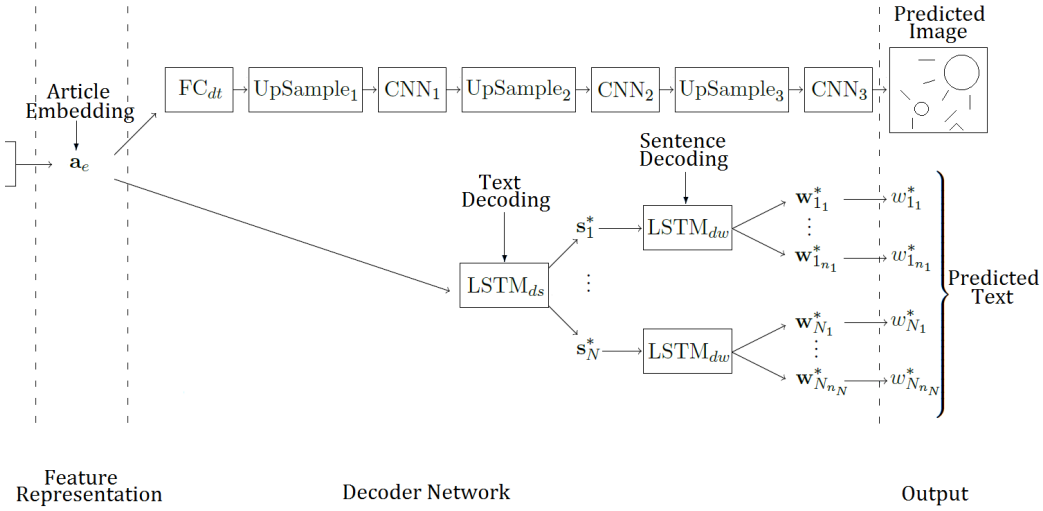


Figure 4.10.: The decoder network architecture of the implemented autoencoder. The network aims to predict the inputted image-text pair from a given article embedding $\mathbf{a}_e$.

in option 1, followed by the same convolutional layers. However, this initial decoding would contain the complete image information in each image region. A CNN, as it only operates locally, cannot distinguish which part of the global information is significant for the currently considered region. Thus, the approach is doomed to fail as a CNN layer has no global awareness.

A trade-off between option 1 and 2 might be a stack of two consecutive fully-connected layers, where the first layer spreads the visual information from $\mathbf{a}_e$ vertically and the second spreads the information encoded in each row (or a local region of rows) from the output of the prior layer horizontally to generate an initial decoding, which can be locally refined by a series of CNN layers. For instance, the first layer computes an output of size $2h \times c$, where c should be large enough to encode the information of a complete row. This layer would require to learn $n_e \cdot 2 \cdot h \cdot c$ parameters. The subsequent layer would take a single row (or a local cluster of rows) of size $1 \times c$ as input. It can then compute a row in the initial decoding of size $2w$. This layer requires to learn only $c \cdot 2 \cdot w$ weights. All these rows can be combined to retrieve an estimate of the input image.

Option 3 generates a thumbnail through a fully-connected layer to extract visual information from $\mathbf{a}_e$ similarly to option 2. This thumbnail is then gradually up-sampled and refined through CNN layers until the size of the input image is reached. More precisely, the network consists of three up-sample layers, each followed by a convolutional layer. The up-sample layers use nearest neighbor interpolation to increase the input size. The first two convolutional layers use the rectifier function (cp. equation 3.6) to introduce non-linearity. Pooling is not applied, since no information should get lost. CNN_1 uses 32 feature maps. CNN_2 uses 8 and CNN_3 uses 3, respectively. Many feature maps are considered at the beginning to allow to look at the task of information extraction from the denser image representation from many viewpoints.

The predicted output image is compared with the input image by using the squared error loss function (cp. equation 3.11).

The lower part of figure 4.10 depicts the text decoder. The text decoding architecture is reverse to the text encoding architecture. LSTM_{ds} generates a sequence of predicted sentence embeddings $\mathbf{s}_i^*$. Therefore, at each time step it takes the article embedding $\mathbf{a}_e$ as input in addition to its previous state. Analogously, predicted sentence embeddings are decoded into tokens $\mathbf{w}_i^*$ via LSTM_{dw}. The text decoding network does not allow dynamically sized predictions, i.e. if N sentences are inputted, then N sentence embeddings will be predicted from $\mathbf{a}_e$. The same restriction accounts for the length of predicted sentences.

To compare the predicted token embeddings $\mathbf{w}_i^*$ with the input text, they have to be translated back to words from the vocabulary. This is done by computing the cosine similarity of $\mathbf{w}_i^*$ with each embedding defined by $\mathbf{W}_{we}$. The maximum cosine similarity would identify the most probable token. However, as the argmax is not differentiable, a softmax layer followed by a cross-entropy loss is used to compare the predictions with the input text.

If optimal pretrained word embeddings could be utilized to initialize $\mathbf{W}_{we}$, such that it does not has to be trained, then a direct comparison of predicted token embeddings with input embeddings would be possible. Such a loss function may be favorable, since it would not punish the prediction similar words as hard as the prediction distant words.

4.4. Learning to mimic Human Intuition

The task of this thesis is to estimate the informational gap between two modalities, namely texts and images. The informational gap between two entities (i.e. a text, an image, etc.) is small, if each of the entities is nearly fully described by the other one. On the other hand, the informational gap is large, if the content of one entity is completely uncertain given that the other one is known. In section 4.2 we have already shown, that there are generally certain differences between the modalities considered in this thesis. Thus, each modality is more suitable to present specific types of information (e.g. calendar dates in texts or shapes in images). We concluded, that this leads to an inherently more complicated task as, for instance, the finding of semantical alignments between two texts. Subsequently, we presented a novel quantification scheme for rating image-text co-occurrences based on human intuition. As we proceeded with our explanations, we recognized that the mapping and rating of non-visual concepts requires extensive background knowledge. However, it can be assumed that concept learning is basically a generalization task, that can be learned unsupervised given a sufficiently sized database. In section 4.3, we presented a network architecture that may achieve this goal due to a lower-dimensional hidden representation, that hopefully requires the ability to generalize concepts. For instance, such a hidden representation may describe an object by a set of features, e.g. type of object, pose, position, shape, etc., rather than encoding a complete pixel matrix.

In this section, we want to combine the already accomplished achievements to complete the description of an algorithm that may potentially allow a computer to mimic human intuition when judging co-occurring image-text pairs. Therefore, annotated samples (as described in subsection 4.2.3) are mapped onto a feature representation via the encoder network depicted in figure 4.9, which has been initialized with the weights learned by the autoencoder. Subsequently, a classifier network tries to deduce MI and SC labels for the sample. Recall, that the feature representation ideally contains all the information comprised by the inputted entities, as the features were trained with the ability to restore this input from them. Hence, the feature representation can be viewed as machine readable representation of the sample, that hopefully allows an easy concept matching, even for non-

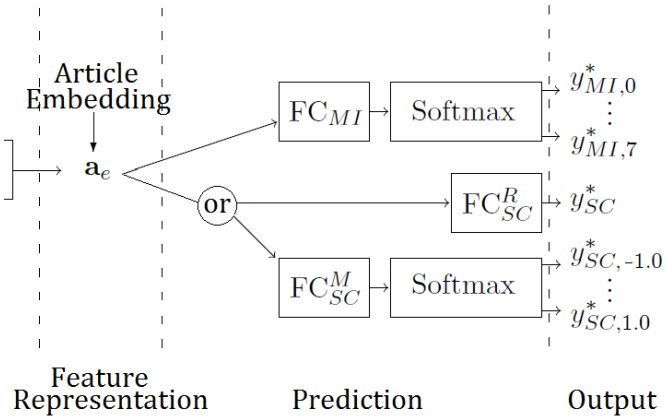


Figure 4.11.: The classifier, which is trained on top of the encoder network depicted in figure 4.9. The network aims to predict the MI resp. SC label of an inputted sample by reading its feature representation $\mathbf{a}_e$.

visual concepts. A desired analogy might be the accumulated neural activity in a human brain when processing the sample. This is due to the fact, that after processing the whole sample humans are able to judge the image-text relation based on their current internal representation of the sample.

The architectural idea behind the classifier is depicted in figure 4.1b. Sample preprocessing has been handled similarly as for the autoencoder. Except, that samples have been distributed among training and test set in a stratified fashion, so the same imbalanced label distribution should account for both sets. This has been achieved by grouping samples according to their MI label and further subgrouping those groups according to the SC label. These subgroups have been distributed into a training (**741** samples) and test (**93** samples) set.

The encoding network is initialized via a pretrained autoencoder model. In the best case, the encoder network itself does not have to be trained. Remember, that we justified our small annotated dataset with the automatic learning of strong feature representations. So, the supervised process is only needed to learn the relatively small classifier network on top of a complex encoder network. The encoded sample $\mathbf{a}_e$ is processed by two different units, one generating an MI prediction and the other an SC prediction, respectively. Figure 4.11 depicts the classification network. The MI label is predicted through a fully connected layer FC_{MI}, that has 8 output neurons such that the neuron with the highest activation denotes the predicted label. Recall, that MI is classified via 8 different labels according to table 4.1. A softmax layer is stacked on top of that output map to normalize the predictions.

Semantic Correlation can either be stated as a multiclass or a regression problem (denoted by an "*or*"-node in figure 4.11). If considered as a multiclass problem, the prediction network would be the same as for MI labels using a fully-connected layer FC^M_{SC} that has 5 output neurons, as SC is quantified using 5 labels according to table 4.2. If SC prediction is stated as a regression problem, a fully-connected layer FC^R_{SC} with a single output neuron predicts the outcome. The output neuron is restricted to output values of y^*_{SC} between -1 and 1 via a sigmoid function.

Something as vague or subjective as intuition should not be learned in a too strict regime, that enforces the correct label disregarding the value of an incorrect prediction. In case of

a regression problem, this can be naturally achieved by a loss function such as the squared error loss. This loss penalizes a close prediction less severe than predictions of distant labels. This property is preferable, since the utilized labeling expects a subjective judgment, that shall be similar among annotators, but not necessarily equal.

In case of a multiclass problem, the choice of a loss function is not that clear. There is no general distance measure for multiclass labels. For instance, the annotations in handwritten digit recognition are explicit and any misclassification should be punished equally. In such a case, a common approach to direct learning is by using the cross entropy loss, which cannot rate the severeness of misclassifications.

For that reason, we propose another loss function, that we call *Distance Aware Loss*.

4.4.1. Incorporating Label Distances in Multiclass Problems

In this subsection we want to describe a loss function that can rank misclassifications to incorporate their severeness. Therefore, a distance metric among labels has to be manually generated in advance, as multiclass labels do not comprise information about their natural distance. Thus, a distance metric $d(y_i, y_j) \in \mathbb{R}^{\geq 0}$ is required, that maps two labels $y_i, y_j \in \mathbb{L}$ onto a real number that quantifies their dissimilarity. $\mathbb{L}$ is a set of multiclass labels (e.g. MI or SC labels). The distance metric we have used to quantify dissimilarity among MI and SC labels is defined in appendix C.

Having the metric d, it would be desirable to rate a misclassification $y \neq y^*$ by a loss

$$L(y, y^*) = d(y, y^*) \qquad . \tag{4.5}$$

However, if the distance metric is based on hard assignments (as the ones presented in appendix C), then the loss function in equation 4.5 is not differentiable. Hence, it cannot serve as a loss function that guides learning via back-propagation in neural networks.

An identical description of the loss function from equation 4.5 would be

$$L(y, y^*) = \sum_{\hat{y} \in \mathbb{L}} s(y, \hat{y}) \sum_{\hat{y}^* \in \mathbb{L}} g(y^*, \hat{y}^*) d(\hat{y}, \hat{y}^*) \qquad , \tag{4.6}$$

with s and g being gate functions

$$s(y, \hat{y}) = \begin{cases} 1, & \text{if } \hat{y} = y \\ 0, & \text{otherwise} \end{cases} \qquad \text{and} \qquad g(y^*, \hat{y}^*) = \begin{cases} 1, & \text{if } y^* = \hat{y}^* \\ 0, & \text{otherwise} \end{cases} \qquad . \tag{4.7}$$

One way to easily achieve such step functions, which have to be differentiable, is by finding proper parameters for the sigmoid function σ. Note, that it holds that

$$(y_1 - y_2)^2 = \begin{cases} \geq 1, & \text{if } y_1 \neq y_2 \\ 0, & \text{otherwise} \end{cases} \qquad \forall y_1, y_2 \in \mathbb{L} \qquad , \tag{4.8}$$

if labels in $\mathbb{L}$ are encoded as natural numbers. Thus, a sigmoid function

$$\sigma(t) = 2 \cdot \frac{1}{1 + e^{-\alpha t}} - 1 \tag{4.9}$$

is restricted to a range of $[0, 1]$ for positive values of t (e.g. $t = (y_1 - y_2)^2$). The parameter α has to be properly chosen such that

$$\sigma\big((y_1 - y_2)^2\big) = \begin{cases} 0, & \text{if } y_1 = y_2 \\ \geq 1 - \epsilon, & \text{otherwise} \end{cases} \quad . \tag{4.10}$$

Therefore, α can be estimated by solving the following equation for α:

$$1 - \epsilon = 2\frac{1}{1 + e^{-\alpha \cdot 1}} - 1 \quad . \tag{4.11}$$

Subsequently, equation 4.6 can be written as a differential equation as follows

$$L(y, y^*) = \sum_{\hat{y} \in \mathbb{L}} \underbrace{\Big(1 - \sigma\big((y - \hat{y})^2\big)\Big)}_{s(y,\hat{y})} \sum_{\hat{y}^* \in \mathbb{L}} \underbrace{\Big(1 - \sigma\big((y^* - \hat{y}^*)^2\big)\Big)}_{g(y^*,\hat{y}^*)} d(\hat{y}, \hat{y}^*) \quad . \tag{4.12}$$

Alternatively, s can be omitted and g could be replaced by a step function that incorporates the correct label y. For instance, consider the function[13]

$$f(y, y^*, \hat{y}^*)) = \Big(y - \frac{y}{y^*}\hat{y}^*\Big)^2 \quad . \tag{4.13}$$

f will only be zero, if $y^* = \hat{y}^*$. The minimum value of f[14], subject to $y^* \neq \hat{y}^*$, can be used to approximate an α value with the same procedure described above. This leads to a differentiable step function that can be used as an alternative to s and g.

However, as only $\partial L / \partial y^*$ has to be computed, s can also be replaced by a hard step function.

The above considerations require that the predicted class y^* is known during loss computation. This is usually not the case, as it would require a non-differentiable operation (e.g. argmax function). Typically, evidence scores for each class are computed by an output layer. These scores can be mapped to probabilities $p(\hat{y}^*)$ due to a softmax layer. This final clue leads to a loss function, that we call *Distance Aware Loss*[15]

$$L_{\mathrm{DWL}}(y, \mathbf{y}^*) = \sum_{\hat{y} \in \mathbb{L}} s(y, \hat{y}) \sum_{\hat{y}^* \in \mathbb{L}} p(\hat{y}^*) d(\hat{y}, \hat{y}^*) \quad , \tag{4.14}$$

where $p(\hat{y}^*)$ is the $\hat{y}^*$-th element of $\mathbf{y}^*$. The inner sum of equation 4.14 can be replaced by a dot product of the softmax output $\mathbf{y}^*$ and the $\hat{y}$-th column of a symmetric distance matrix (as the ones defined in appendix C).

[13] Assuming all labels are represented by consecutive natural numbers starting at 1.

[14] Let l be 1 and m be the highest natural number in $\mathbb{L}$. The minimum value of f for $y^* \neq \hat{y}^*$ is $f(l, m, m-1)$. This is because both terms y and $\frac{y}{y^*}\hat{y}^*$ of f increase with increasing y, whereas the second term is scaled by $\frac{\hat{y}^*}{y^*}$. Hence, y has to be as close as possible to 0 while $\frac{\hat{y}^*}{y^*}$ has to be as close as possible to 1.

[15] Again, the outer sum can be omitted, since $\partial L / \partial y$ does not has to be computed and s is only for $y = y^*$ non-zero.

5. Experiments and Evaluation

This chapter presents experimental results on a first attempt to implement the novel intuition model from section 4.2 as a deep learning network. All experiments have been conducted on the system explained in section 4.3 and 4.4 using the dataset described in section 4.1 resp. 4.2.

Due to memory restrictions, we had to face several limitations. The maximum batch size that could be processed is 16. Note, that all samples within a batch are padded to have the same size as the largest sample. Hence, the largest sample among the complete dataset will generate a batch full of samples, that will all have this maximum size. To further reduce this maximum size, texts have been truncated during preprocessing. We have found out that a maximum text size of 50 sentences and a maximum sentence length of 40 tokens will result into a manageable memory utilization per batch. This restriction does not severely distort the sample texts, as 91.15% of the articles (or 99.87% of the sections) in the *SimpleWiki* dataset resp. 99.12% of the texts in the *BBC News Database* contain less than or equal to 50 sentences. Furthermore, 97.97% of the sentences from the *SimpleWiki* dataset resp. 96.45% of the sentences from the *BBC News Database* contain less than or equal to 40 tokens (cp. appendix A). Hence, only a small portion of texts resp. sentences are affected by this measurements. Yet, outliers are effectively filtered.

The most severe restriction is probably the size of feature vectors. We had to limit their dimensionality to 300, which does require the learning of a very strong compression. Recall, that the whole image and text must be encoded into such a dense representation. Yan and Mikolajczyk [67] already mentioned that larger feature vectors are preferable for representing multimodal embeddings.

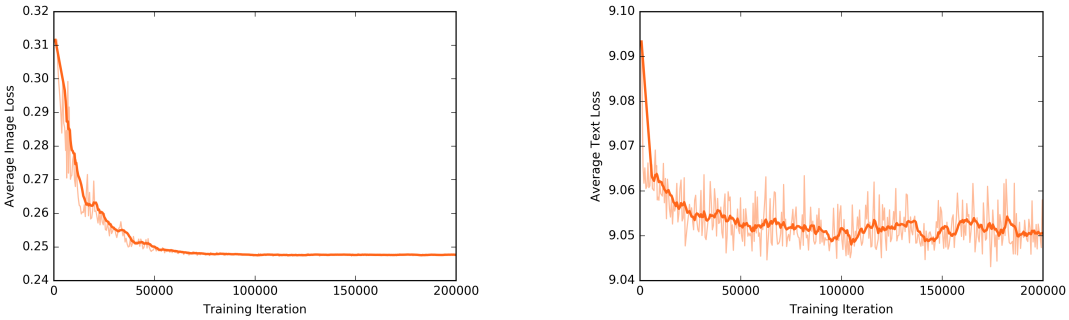
The experiments have been conducted on a machine with an Intel(R) Core(TM) i7-5930K CPU, 64 GB RAM and four NVIDIA GeForce GTX TITAN X graphic cards (having 12GB GDDR5 each). Training experiments for the autoencoder and classifier have ran on those GPUs using the allocated training set. Training phases have been monitored by an evaluation script running concurrently on the CPU for a constant analysis of the training progress. For the autoencoder, evaluation has been performed on the designated validation set. As the classifier dataset has no validation set, due to its small size, evaluation has been performed on the test set.

All graphics in this chapter show smoothed curves[1] to mitigate the effects of a poor gradient estimate due to the small batch size that had to be used.

5.1. Evaluating the Autoencoder

The experiments reported in this section are performed on the system described in section 4.3. Before the results are presented, the experimental setup is outlined.

[1] Curves are smoothed using a moving average approach. Although, graphics with single curves depict the original data transparently in the background.



(a) Evolution of the image loss.

(b) Evolution of the text loss.

Figure 5.1.: The graphics show how the loss functions that rate the image resp. text prediction have decreased during the autoencoder training.

5.1.1. Experimental Setup

The complete dataset decomposes into three parts. 202,654 samples have been generated from the *MS COCO* validation set. Additionally, all 3,361 resp. 2,999 image-text pairs from the *BBC News Corpora* resp. *SimpleWiki* dataset have been included. From this randomly shuffled corpus, samples have been selected to generate a disjoint split of 190,202 training and 6,270 validation samples[2].

Stochastic Gradient Descent (SGD) with mini-batches has been used as training algorithm with an initial learning rate of 1.0. The learning rate is divided by 2 after a complete sweep through the training set has been accomplished.

Option 3 has been used as image decoder architecture. Additionally, image reconsideration subsequently to text processing was enabled (cp. section 4.3). The image encoding network was initialized with weights of a pretrained *InceptionV3* model. Initial word embedding estimates have been taken from a *Word2Vec* implementation that was trained among the whole text contained in training, validation and test set.

5.1.2. System Performance

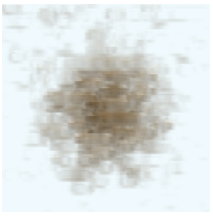
As a result of the high memory requirements of the implemented architecture and the resulting low dimensional internal feature representations, the learned compression was not as applicable as desired. The loss evolution during training is depicted in figure 5.1. Note, that all graphics and metrics presented in this chapter have been computed on a validation/test set. As one can see, the autoencoder architecture is capable of learning properties from the input distribution. Recall, that the output is completely decoupled from the input except of a dense 300-dimensional feature representation.

The average image loss (averaged over the whole validation set) has decreased by 25.82%, whereas the average text loss only decreased by 0.45%. However, as the absolute loss values are not comparable, we additionally provide perplexity measures. Intuitively, perplexity measures the complexity of a model (lower is better). It is computed as the exponentiation of the model its entropy.

The image perplexity has decreased by 6.61% and the text perplexity by 7.42%, respectively. Hence, it can be concluded that both models (image and text decoding) are similarly

[2]The remaining samples are allocated in an unused test set.

(a) Input image of the trained autoencoder.



(b) Predicted output image.

Figure 5.2.: This figure illuminates what the autoencoder was able to learn in spite of an extremely dense hidden feature representation. Note, the input image had to be blurred due to its unknown copyright status. The original input image depicts a skier in the midst of a snowscape.

well suited to accomplish their task. That learning actually has been taken place can be seen in figure 5.2. The figure depicts the same image before and after it has been processed by the autoencoder. The system correctly detects that the bulk of content is gathered in the center of the graphic. A similar outcome could be observed for most images, i.e. prominent contours could be recognized or global color distributions were reflected in the output. Therefore, it can be assumed that the autoencoder architecture is suited for feature learning and especially for conceptualization. Yet, the experimental results are by far not satisfactory and call for further architectural improvements in order to improve computational efficiency.

5.2. Evaluating the Classifier

The experiments reported in this section are performed on the system described in section 4.4.

Based on the poorly encoded samples delivered by the just examined autoencoder model and the strong coupling of classifier and autoencoder, the intuition estimation procedure derived in section 4.2 has to be slightly modified with respect to feature learning. The original approach relied on an encoding that extensively represents the original input but comprises a simplified characterization of depicted concepts and their alignment, such that the classifier only has to learn a rating of those alignments in a supervised process. This goal cannot be fully achieved with the given compression model learned by the autoencoder. Therefore, we decided to finetune the encoder model during training of the classifier, such that the classifier can identify and extract the properties from the input distribution necessary to judge inter-modal relations. Thus, during classifier training, the learnable weights of the complete encoder network (cp. figure 4.9) and the subsequent classifier network (cp. figure 4.11) are modified by back-propagation. Hence, the feature extraction of the classifier is not fixed as originally proposed. In spite of the learning that has occurred and will be presented throughout the section, we want to mention that this learning scheme is unqualified to solve the initial task. Such a framework would require a massive amount of annotated data to noticeably improve the baseline laid by our experiments. This is due to the inherent complexity of the actual problem, i.e. the ability to generalize and map non-visual concepts. Moreover, we believe that such a system would underperform compared to the architecture proposed in chapter 4, even if enough annotated training data would be available. Two reasons might exclude a fully supervised scheme from solving the examined task. Firstly, a

system concentrating solely on the detection of features that do improve the current label prediction based on back-propagation, might not manage to find a global minimum, that requires the generalization of concepts as assumed by our explanations in chapter 4. This is due to the fact, that there is no necessity for the system to learn a complete representation of the semantic content. Another reason, that might restrict a fully supervised scheme from proper feature learning, is the noisy nature of the annotations themselves. This is because of the expected label variance among different annotators as stated in section 4.2.3.1.

5.2.1. Experimental Setup

As outlined in subsection 4.2.3, the dataset consists of 834 samples (100 from the *MS COCO* dataset, 205 from the *BBC News Corpora* and 529 from the *SimpleWiki* dataset). The samples have been distributed onto a training set consisting of 741 samples and a test set consisting of 93 samples using stratified sampling.

Similar to above, the model has been trained using SGD with mini-batches with an initial learning rate of 1.0, that has halved as soon as a complete batch of all training samples has been processed.

The classifier models, used to conduct experiments, has been initialized with the weights learned in the autoencoder model presented in the previous section. Although, the learned feature extraction algorithm does not fully encode the input, some general properties are identified by the model.

We will present result for 4 different classifier settings, which are described by the following listing.

- $\mathbb{E}_{MC}^{CE}$: The task of predicting *Semantic Correlation* is stated as a multiclass problem. MI and SC predictions are rated by a cross-entropy loss function.
- $\mathbb{E}_{MC}^{DA}$: The task of predicting *Semantic Correlation* is stated as a multiclass problem. MI and SC predictions are rated by the *Distance Aware Loss* function described in section 4.4.1.
- $\mathbb{E}_{R}^{CE}$: The task of predicting *Semantic Correlation* is stated as a regression problem. MI and SC predictions are rated by a cross-entropy loss function.
- $\mathbb{E}_{R}^{DA}$: The task of predicting *Semantic Correlation* is stated as a regression problem. MI and SC predictions are rated by the *Distance Aware Loss* function described in section 4.4.1.

A pre-superscript P will denote experiments that have been trained on samples containing only photographs (e.g. ${}^{P}\mathbb{E}_{MC}^{CE}$). Thus, the dataset used in this experiments comprises only samples labeled with the image type *Photograph* (100 from the *MS COCO* dataset, 200 from the *BBC News Corpora* and 397 from the *SimpleWiki* dataset).

Systems that have been used for comparison are:

- $\mathbb{E}_{\text{SVM}}^{AE}$: A multiclass SVM implementation [13], trained with article embeddings $\mathbf{a}_e$ as features that have been generated by the trained autoencoder model from section 5.1[3].

[3] A suitable value for the *weight-decay* hyperparameter, that trades-off noise incorporation, has been found via grid search.

- $\mathbb{E}^{C}_{\text{SVM}}$: A multiclass SVM implementation [13], trained with article embeddings $\mathbf{a}_e$ as features that have been generated by the trained classifier model $\mathbb{E}^{CE}_{MC}$, as it shows an acceptable performance on both measures.
- $\mathbb{E}^{MF}_{\text{RAND}}$: RAND denotes the calculated accuracy that would be reached by a random classifier. MF denotes a random classifier that consistently predicts the most frequent label. Hence, such a classifier exploits the strong label imbalance to achieve high accuracy values.
- $\mathbb{E}^{UD}_{\text{RAND}}$: RAND denotes the calculated accuracy that would be reached by a random classifier. UD denotes a random classifier that randomly outputs labels according to a uniform distribution. The distribution of target labels from input samples equals the label distribution $\mathcal{L}$ that can be derived from the label frequencies from the classifier dataset.
- $\mathbb{E}^{LD}_{\text{RAND}}$: RAND denotes the calculated accuracy that would be reached by a random classifier. LD denotes a random classifier that randomly outputs labels according to the label distribution $\mathcal{L}$. Target labels from input samples are drawn from the same label distribution $\mathcal{L}$.

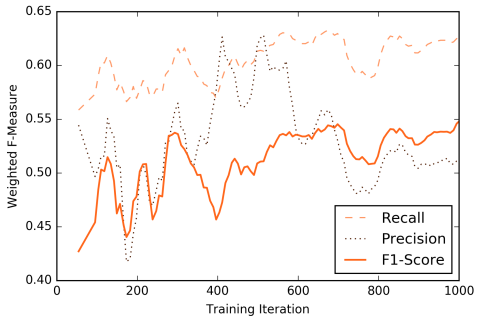
5.2.2. System Performance

Experiment	Accuracy MI	Accuracy SC
$\mathbb{E}^{CE}_{MC}$	0.6289	0.6875
$\mathbb{E}^{DA}_{MC}$	0.5391	0.1758
$\mathbb{E}^{CE}_{R}$	**0.6328**	–
$\mathbb{E}^{DA}_{R}$	0.5430	–
${}^{P}\mathbb{E}^{CE}_{MC}$	0.6172	0.6953
${}^{P}\mathbb{E}^{DA}_{MC}$	0.5352	0.6953
${}^{P}\mathbb{E}^{CE}_{R}$	0.4961	–
${}^{P}\mathbb{E}^{DA}_{R}$	0.2305	–
$\mathbb{E}^{AE}_{\text{SVM}}$	0.6125	0.6500
$\mathbb{E}^{C}_{\text{SVM}}$	0.5375	**0.7250**
$\mathbb{E}^{MF}_{\text{RAND}}$	0.5593	0.6563
$\mathbb{E}^{UD}_{\text{RAND}}$	0.1250	0.2000
$\mathbb{E}^{LD}_{\text{RAND}}$	0.3693	0.4772

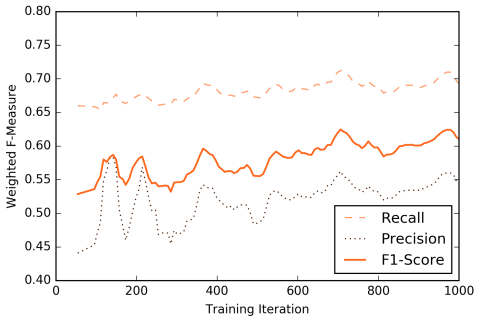
Table 5.1.: The overall accuracy of predicting the correct MI resp. SC label that has been achieved in our experiments.

The accuracy achieved in all the previously described experiments is depicted in table 5.1.

A classifier that would learn to predict a single label such as $\mathbb{E}^{MF}_{\text{RAND}}$ represents the most severe case, as the complete entropy of the input distribution has been vanished from the predictions. Therefore, we additionally provide F-measures for the conducted experiments to better account for systems achieving high scores by solely exploiting label imbalances. As



(a) Evolution of the weighted F1-Score when predicting MI labels.



(b) Evolution of the weighted F1-Score when predicting SC labels.

Figure 5.3.: The graphics show aggregated F-measures (weighted average to account for label imbalances), that have been recorded during training of $\mathbb{E}_{MC}^{CE}$ for MI resp. SC labels.

Experiment	**Weighted F1 MI**	**Micro F1 MI**	**Macro F1 MI**
$\mathbb{E}_{MC}^{CE}$	0.5540	0.6289	0.2260
$\mathbb{E}_{MC}^{DA}$	0.3776	0.5391	0.0876
$\mathbb{E}_{R}^{CE}$	**0.5951**	**0.6328**	**0.2536**
$\mathbb{E}_{R}^{DA}$	0.3821	0.5430	0.0880
${}^{P}\mathbb{E}_{MC}^{CE}$	0.5291	0.6172	0.2327
${}^{P}\mathbb{E}_{MC}^{DA}$	0.3731	0.5352	0.0872
${}^{P}\mathbb{E}_{R}^{CE}$	0.4375	0.4961	0.1510
${}^{P}\mathbb{E}_{R}^{DA}$	0.1571	0.2305	0.1250
$\mathbb{E}_{\text{SVM}}^{AE}$	0.4862	0.6125	0.1959
$\mathbb{E}_{\text{SVM}}^{C}$	0.4939	0.5375	0.1913

Table 5.2.: Several multiclass F-measures for MI predictions, that have been recorded for the conducted experiments

we are dealing with multiclass problems, a single F1-Score, that harmonizes precision and recall of label predictions, is not applicable to present a global view on the performance of the system. Therefore, table 5.2 and 5.3 provides a list of three different multiclass metrics, that aim to evaluate multiclass predictions with distinct ways of aggregating binary F1-scores. Weighted and Macro F1-Score are computed as the harmonic mean of the averaged individual label estimates for recall and precision. In addition, the Weighted F1-Score takes the biased label distribution into account by computing a weighted average. The Micro-F1 score is computed identically to the binary version, except that the underlying frequency estimates (true positives, false positives and false negatives) are accumulated among all labels.

Additionally, figure 5.3 shows the evolution of the Weighted F1-Score for experiment $\mathbb{E}_{MC}^{CE}$.

Most notably, the depicted results reveal that the *Distance Aware Loss* is always leading to result inferior to those classifiers using Cross-Entropy loss. We still struggle to find a

Experiment	Weighted F1 SC	Micro F1 SC	Macro F1 SC
$\mathbb{E}_{MC}^{CE}$	0.6081	0.6875	0.2613
$\mathbb{E}_{MC}^{DA}$	0.0526	0.1758	0.0598
${}^{P}\mathbb{E}_{MC}^{CE}$	0.6337	**0.6953**	0.2886
${}^{P}\mathbb{E}_{MC}^{DA}$	0.0504	0.5352	0.1719
$\mathbb{E}_{\text{SVM}}^{AE}$	0.6500	0.5326	0.2120
$\mathbb{E}_{\text{SVM}}^{C}$	**0.7250**	0.6841	**0.3513**

Table 5.3.: Several multiclass F-measures for SC predictions, that have been recorded for the conducted experiments

	0	1	2	3	4	5	6	7
$\mathbb{E}_{MC}^{CE}$	0.0000	0.0000	0.0000	0.0000	**0.7462**	0.2564	0.8056	0.0000
$\mathbb{E}_{R}^{CE}$	0.0000	0.0000	0.0000	0.0000	0.7297	**0.5047**	0.7941	0.0000
$\mathbb{E}_{\text{SVM}}^{AE}$	0.0000	0.0000	0.0000	0.0000	0.7300	0.0000	**0.8400**	0.0000
$\mathbb{E}_{\text{SVM}}^{C}$	0.0000	0.0000	0.0000	0.0000	0.6700	0.2800	0.5900	0.0000
# Samples	3	0	1	4	43	18	10	1

Table 5.4.: F1-Score of each individual MI label. The last row contains the number of representatives of each label in the test set.

explanation for that behavior. It might be that the used distance matrices from appendix C do not differentiate distinct labels sufficiently, leading to a too vaguely stated classification problem.

Strangely, *Distance Aware Loss* applied on MI predictions influences the performance of an SC classifier, that considers the prediction as regression problem rated via squared loss. For instance, when using Cross-Entropy loss for the MI classifier, the mean squared error on SC predictions decreases by 37.38% during training. On the other hand, if MI predictions are rated by the *Distance Aware Loss*, an actual increase of the squared error around 7.66% can be measured. The reasons for this behavior should be analyzed in future work by conducting further experiments with more carefully engineered distance metrics. Experiments using *Distance Aware Loss* are neglected by further considerations.

As the multiclass problems comprise only a few labels, it is feasible to consider each label separately. Table 5.4 and 5.5 present the F1-scores that have been reached for each label individually.

The tables reveal that the major cause of the poor classification results lay in the sparsity of certain labels. Future work should address this issue by incorporating suitable multimodal documents from datasets that either have a high amount of shared information or a negative semantic correlation.

Most surprisingly, the classifier is capable, in spite of poor initial encodings, of learning features necessary to judge *Semantic Correlation*. This effect is visible when comparing the results of $\mathbb{E}_{\text{SVM}}^{AE}$ and $\mathbb{E}_{\text{SVM}}^{C}$. Recall, that these experiments differ in the features they use. $\mathbb{E}_{\text{SVM}}^{AE}$ uses the feature vectors from the initial classifier network and $\mathbb{E}_{\text{SVM}}^{C}$ uses the feature vectors from the final classifier network (after training). Such improvements are not obviously visible when comparing their MI prediction results. This might be due to the

	-1.0	-0.5	0.0	0.5	1.0
$\mathbb{E}_{MC}^{CE}$	0.0000	0.0000	0.4691	0.0000	0.8373
$\mathbb{E}_{\text{SVM}}^{AE}$	0.0000	0.0000	0.2700	0.0000	0.7900
$\mathbb{E}_{\text{SVM}}^{C}$	0.0000	0.0000	**0.5900**	**0.2900**	**0.8800**
# Samples	1	3	11	15	50

Table 5.5.: F1-Score of each individual SC label. The last row contains the number of representatives of each label in the test set.

fact, that the feature vectors do not properly encode concepts appearing in the input, thus it is not possible to judge their alignment.

Moreover, the results show that high *Mutual Information*, in particular label 6, is easier to detect. Note, that all samples of label 6 are actually taken from image caption datasets and are therefore biased towards high visual concepts. Hence, it can be assumed that high visualness is much easier to learn, which matches our expectation.

In conclusion, one can state that neither the dataset nor the concrete architectural implementation are well suited to tackle the problem of mimicking human intuition when judging the relation of multimodal documents. However, all experiments have shown that the general expectations of the approach described in chapter 4 can be met, except for the loss function introduced in section 4.4.1. Section 5.1 has shown that an autoencoder is capable of encoding concepts from the input in an extremely dense feature vector in a fully unsupervised training regime. It can be assumed that a more expressive hidden representation will lead to the ability of encoding further concepts. The results presented in this section have illuminated that random classifiers can be outperformed by a deep neural network, setting an initial baseline, that should be easily surpassed by a more carefully engineered system (e.g. by incorporating some of the suggestions from chapter 7).

6. Applications

A variety of direct application can be derived from a system that correctly predicts *Semantic Correlation* and *Mutual Information*. A few of them will be listed in this chapter to illustrate the potential benefits arising from a working system and to motivate future research in this area.

To highlight the quality of learned feature representations, these vectors may be used to estimate the similarity between encoded input samples. For instance, an image-text retrieval task can be naturally formulated as the vector similarity of their embeddings. More precisely, the encoder network in figure 4.9 may be used to embed a sample consisting solely of text and one that only contains an image. The similarity between the resulting article embeddings does rate their semantic similarity, as the embeddings where generated with the goal to encode the whole input content. This is the same analogy that has been utilized by word vectors, since a proper mathematical representation of an abstract concept can easily be used to compute semantic distances.

However, as highlighted in chapter 4, similarity in the content of samples or entities does not necessarily imply a strong semantic correlation. Such entities may even be negatively correlated (cp. example 4.2). As subtle details can heavily change the correlation estimate, a learned vector encoding, that can significantly increase vectorial distances for certain changes but maintaining a similar position for others, is unlikely to be learned by any reasonable sized dataset. Therefore, a reliable image-text retrieval system does require further intervention, i.e. an algorithm that rates *Semantic Correlation* just as we proposed.

In general, an MI – SC estimator can increase the quality of search results. For the sake of illustration, we assume an MI – SC system that has been trained on several modality pairs (e.g. text-text, image-text, audio-text, ...) throughout the chapter. A user that uses an entity (e.g. an image or a text) as search input, might have different expectations regarding the outcome. One might look for documents containing the same information (i.e. high MI and SC) to retrieve a document that is possibly easier to understand or to perceive. Another possibility would be to gather as much information as possible about a certain topic (i.e. low MI but high SC, thus documents that complement one another).

New tasks in the field of machine translation may arise, but in terms of translating from one modality to another. For instance, co-occurring texts and information graphics could be automatically gathered to build an enormous dataset of high MI and high SC pairs [1]. Usually, such pairs would have to be manually rated or generated by humans. Similar, to the training of language models in machine translation, neural networks that translate one modality to another could be trained with such a dataset.

The same accounts for a field that has been heavily invested recently. People, who are visually impaired, could immensely benefit from a system that automatically describes a scene. Current algorithms rely on training data, that was human annotated and therefore only contains a couple of hundred thousand samples. This is not sufficient to help one through everyday situations. Collecting data from the Internet is hard, because neither

[1] Note, that no such dataset for information graphics exists currently.

the surrounding text nor the caption describes what is visually obvious most of the time. Though, as figure 2.1 depicts, current captioning algorithms already show outstanding results. It can be assumed that such systems achieve even better outcomes when provided with more data. As human annotation is a time consuming and expensive process, an automatic dataset generation would be preferable. As a simplistic image-caption retrieval from the Internet would not produce the desired outcome, a more sophisticated retrieval algorithm has to be designed to filter for captions that describe visual obvious content[2]. Hence, similarly to the retrieval of information graphics above, a Web crawler, that retrieves image-caption pairs of high MI and high SC could be used to generate more prosperous datasets.

Furthermore, an MI – SC estimator could be used to rank image-captioning results. As stated in chapter 2, evaluating the quality of automatically generated image captions without human interference is still an unsolved task. Hence, an MI – SC estimator may allow to design a fully unsupervised captioning framework, that learns from automatically gathered samples and terminates when proper outcomes are produced.

A further application would be to extract relevant passages in a text. For instance, almost all images that had been annotated in the course of this work had no direct references in co-occurring text (i.e. a stated image label). Yet, some texts contained passages that exclusively described image content (the relevant text snippets that have been extracted; cp. section 4.2). Identifying those sections could again be accomplished by rating image-sentence pairs according to their SC and MI estimate and extracting those with sufficiently high values.

This application could be particularly interesting in scientific publications, as such an algorithm could automatically highlight and assemble relevant text passages that describe a graphical representation. Once more, we want to highlight potential subsequent applications, that would be rendered possible as a result of those direct uses. As it is still computationally infeasible to train networks that generalize well on too noisy data while aiming to solve complex tasks (such as image captioning), large datasets of high accuracy and quality are required. Such a dataset can then be used to automatically generate textual descriptions of graphics in scientific publications similarly to recent captioning algorithms.

There is certainly a variety of unmentioned fields that would benefit from a reliable MI – SC estimator, as human judgment and intuition does prevent many systems from being able to operate fully automated. It is yet clear that the listed tasks can only be adequately solved with a system that better judges image-text relations. A reliably operating MI – SC estimator, as suggested in chapter 4, would be perfectly suited to provide the missing piece required for all of the above stated applications.

[2] It might be sufficient to automatically crawl image captions, that contain a relatively high number of *high visual* words. An algorithm to rate the visualness of words can be taken from [68]. However, a dataset retrieved via such a heuristic may still be too noisy.

7. Conclusion

This thesis has presented a novel approach to rate the relation of co-occurring image-text pairs. Recent approaches relied on the assumptions that co-occurring pairs have a high semantic correlation. We illuminated that this is not the case, their natural combined usage is rather complicated and highly dependent on the source and expected preexisting contextual knowledge. We showed that it is essential to acknowledge and handle the usage variations among co-occurring modalities in order to automatically generate task-specific datasets, necessary to fully automate a wide variety of current and future tasks located on the edge of NLP and CV, that will ease and improve everyday live.

More precisely, our approach relied on the assumption that extensive background knowledge is required in order to comprehend even marginal alignments between entities. Therefore, we gathered a new dataset from an online encyclopedia, because they often comprise knowledge about the world and its entities in a compact and nonredundant fashion. In combination with other datasets, encyclopedia articles have been utilized to train an unsupervised system, that learns a compression of the provided samples. The learning of a compression enforces the system to generalize concepts and to map related concepts from both modalities onto the same representation. Furthermore, we proposed a novel annotation scheme, which allows human judges to easily rate a multimodal relation based on two measurements. One measures the amount of shared information in order to estimate whether entities do complement one another. The more subtle measurement of *Semantic Correlation* evaluates the overall likelihood of co-occurrence, that requires to infer the meaning of all concepts appearing in an entity and to identify and rank mutual alignments between concepts. We have generated a human-annotated dataset based on those measurements, containing over 700 image-text pairs from a variety of distinct sources in order to ensure a sufficient variance of relation types. In addition, we implemented a deep learning architecture to tackle both problems, learning extensive background knowledge and judging intermodal relations. An autoencoder architecture based on recently developed image feature extraction models and a hierarchical text processing recurrent neural network, to allow learning of long-term-dependencies in long texts, has been proposed to generate a dense embedding space for concept encoding. A rating system that utilizes the learned encodings, and therewith the simplification of complex, highly nonvisual concepts, has been trained with the gathered annotations in order to quantify the relation of a text enclosed with an image. Subsequently, we have demonstrated that the system in principal may work as expected and we have set an initial baseline above random classifications for future work to improve on.

However, as the evaluation in chapter 5 has shown, the problem faced in this thesis is inherently more complex than recently encountered tasks, that have had extraordinary success using deep learning methods. In order to approach the problem successfully with the computational power currently available, a more carefully designed dataset and network are required. In account of this, we want to propose some improvements that can be integrated to yield to a better outcome.

The dataset for the autoencoder is not expressible enough yet to keep up with human

conceptualization skills. A larger news corpora, as presented in 4.1.2, might be more suitable to learn a sufficient amount of background knowledge regarding ongoing events, affairs and stories. Furthermore, the *SimpleWiki* crawler should be restarted to gather a complete encyclopedia copy. Otherwise, an autoencoder may only be able to learn an incomplete view on the world. Additionally, colloquial language can be included in the database to allow a wider range of applications. However, all these dataset enrichments do increase the required computational power and may not be necessary to achieve modest results.

On the other hand, improvements of the annotated dataset are desperately needed as the sparsity of certain labels have considerably worsen the overall results of our system in section 5.2. First of all, a system to distinguish image types can be invented in order to retrieve rare types such as charts. This could be done unsupervised by a clustering approach on a dataset having an approximately balanced type distribution or by providing a small annotated dataset[1]. A balanced dataset with respect to image types could illuminate the usage patterns of different graphical representations and therewith lead to a broader scope of used labels (e.g. it can be assumed that charts are typically well described in co-occurring text in contrast to photographs).

As addressed in section 4.2.3.1, it may be wise to further simplify the MI annotation scheme in order to better capture the actual distribution of relation types and to obtain an ordinal labeling. This can solve the data sparsity among labels capturing high amounts of shared information as captioning dataset could fill this gap.

To overcome the sparsity among negative SC labels, it may be necessary to assemble pairs from handpicked images and texts. For instance, a text that explains the physique of a horse and is enclosed with the image of a dog is a perfect example for negative semantic correlation. Though, negatively correlated samples are unlikely to occur in natural settings and therefore have to be artificially generated.

Additionally, the system architecture has to be tuned to train the underlying networks more effectively. Especially the autoencoder structure has to be reviewed to allow larger feature embeddings. For this reason, a system should be designed that allows varying embedding sizes for word, sentence, image and article embeddings. This is the most obvious architecture as words generally do not contain as much information as complete sentences or texts. This can be achieved by either one of two architectural modifications of the autoencoder, which has been proposed in section 4.3. Firstly, one may increase the number of output neurons in LSTM layers generating sentence or article embeddings. However, as it can be difficult to implement a model that holds only the output of the last element for each sequence in memory for a batch containing variable-sized sequences, a simpler hierarchical expansion might be more suitable. Therefore, additional LSTM layers between word-sentence and sentence-article embedding generation are inserted. For instance, a layer that takes a constant window of four word embeddings to generate inputs for an LSTM that creates the final sentence embedding, does increase the intermediate embedding size by a factor of 4. Such naive improvements might allow to learn the encoding of a lot more concepts during training and leads therewith to more expressive feature vectors.

Future work should also examine the influence of more complex loss functions to better rate the similarity of the predicted and true outcome. For instance, recently developed object, object-attribute and spatial-relation detectors can be utilized additionally to quantify the similarity of input and output images. For text similarity, embedding distances might be

[1] A small dataset should be sufficient as it might be not that complicated to distinguish image types. In addition, a noisy result is no problem when the retrieved documents are used for further human annotation.

regarded similar to recent paraphrase detection approaches. However, such enhancements may also impair the system performance as the learning problem could be too loosely stated analogous to the impact of a *Distance Aware Loss* (cp. section 4.4.1 and 5.2).

Heavy memory consumption in our autoencoder architecture is caused by the back-translation of word embeddings to tokens in the vocabulary as this step requires to compute a matrix that contains a similarity score for each token in a text compared with each token in the vocabulary. Hence, a reduction of the vocabulary size without a loss in expressiveness is desirable. In addition, this would extensively reduce the number of weights to be learned, as the matrix of all word embeddings shrinks. To achieve this, we want to suggest a more natural way of encoding word embeddings. The method is an hierarchical word encoding, where the encoding of words with the same stem is related. Therefore, words are inputted as pairs consisting of the word stem and the original word ending. An embedding matrix would only be learned for word stems, as endings usually modify those stems similarly. Hence, a final word embedding is retrieved via a fully connected layer that modifies a word embedding, such that the initial ending is taken into consideration.

In order to take artificial intelligence on the next stage, the more ambiguous and subjective facets of human intelligence, such as their intuition and judgment, have to be considered and investigated. This thesis is a first attempt to enter a field of exciting new tasks and possibilities.

Almost all presented formulations can be transferred to other constellations of modalities (e.g. text and video). We hope that future work, inspired by the methodology developed in this work, will include additional modalities and accomplish a prediction performance that allows to tackle the applications incentivized by chapter 6 or those that haven't even been imagined yet.

A. Specifics and Characteristics of the Datasets

The materials supplied in this appendix shall give some auxiliary information about the datasets described in section 4.1, especially the *SimpleWiki* dataset, that was newly created. Figures A.1 and A.2 give an intuition for the general sentence/text complexity of each individual dataset. Section A.1 explains the JSON structure of a single article from the *SimpleWiki* dataset.

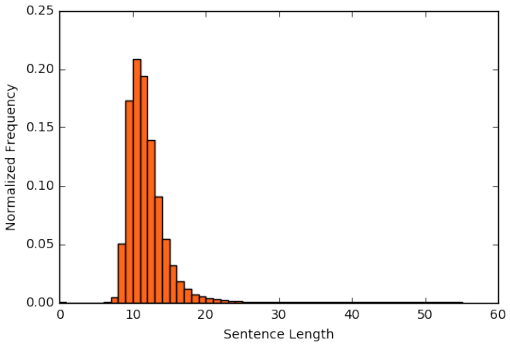
A.1. Article structure in the SimpleWiki dataset

The overall structure of a single article is depicted in the below JSON fragment A.1. The following list gives a brief description of each property.

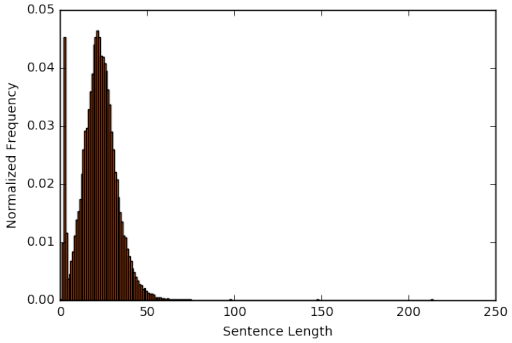
```
{
  "references":[ ... ],
  "categories":[ ... ],
  "sections":[ ... ],
  "summary":"...",
  "images":[ ... ],
  "id":1,
  "keyphrases":[ ... ],
  "title":"...",
  "url":"https://simple.wikipedia.org/wiki?curid=1"
}
```

Fragment A.1: JSON object that defines a *SimpleWiki* article

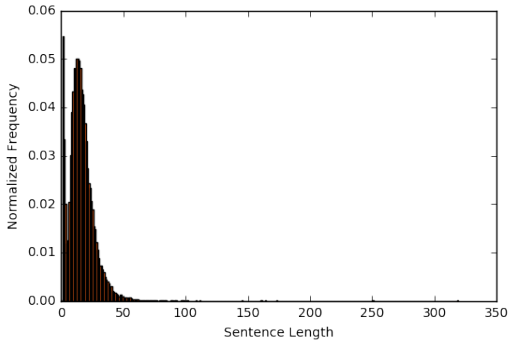
- **references:** A list of all external references that have been cited in the article.
- **categories:** A list of categories that have been assigned to the article.
- **sections:** A list of section objects as defined in JSON fragment A.2.
- **summary:** The summary of the article.
- **images:** A list of image objects as defined in JSON fragment A.3.
- **id:** The unique ID of the article.
- **keyphrases:** A list of keyphrases. These are all internal links of article, as they are assumed to be valid keywords/phrases.
- **title:** The title of the article.



(a) The sentence length distribution of the captions belonging to the validation set of the *MS COCO* dataset.

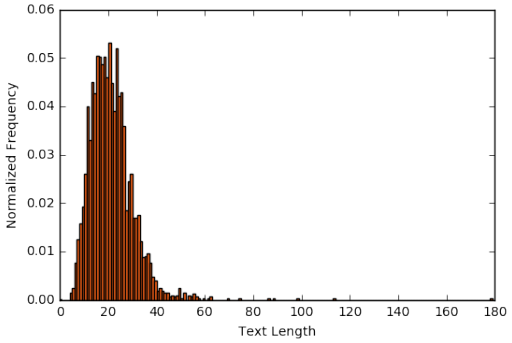


(b) The sentence length distribution of sentences drawn from article texts in the *BBC News database*. Headings are considered as sentences in this overview.

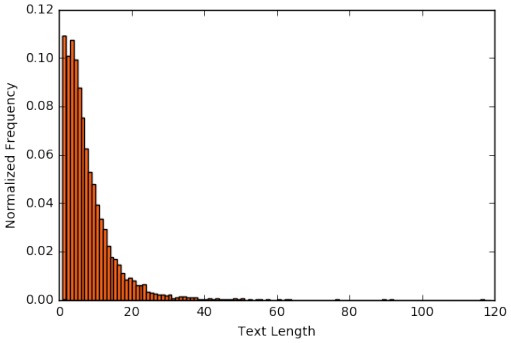


(c) The sentence length distribution for sentences in an article from the *SimpleWiki* dataset. Headings are considered as sentences in this overview.

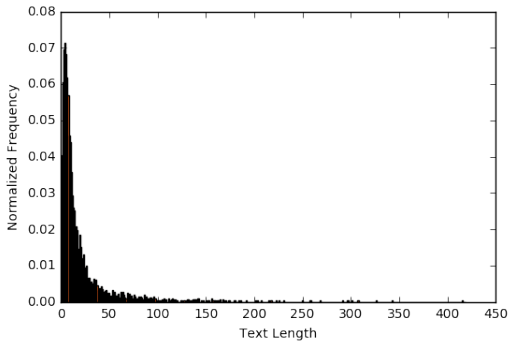
Figure A.1.: The histograms show the more complicated sentence structure of news or encyclopedia articles compared to artificially generated image captions.



(a) Shows the distribution of the number of sentences in a news article from the *BBC News database*.



(b) Shows the distribution of the number of sentences in a single section or an article summary for articles from the *SimpleWiki* dataset.



(c) Shows the distribution of the number of sentences in a complete article from the *SimpleWiki* dataset.

Figure A.2.: The histograms show that news articles have a minimum length, whereas encyclopedia articles can have any length.

- **url:** The URL of the article.

The structure of each section resp. subsection is defined by the below JSON fragment.

```
{
  "images":[ ... ],
  "lists":[ [ ... ], ... ],
  "subsections":[ ... ],
  "title":"...",
  "keyphrases":[ ... ],
  "text":"..."
}
```

Fragment A.2: JSON object that defines a *SimpleWiki* section

The properties of a section are defined as follows:

- **images:** A list of image objects as defined in JSON fragment A.3.
- **lists:** A section may contain several lists. Each list is a collection of list items.
- **subsections:** A list of section objects as defined in JSON fragment A.2.
- **title:** The heading of the section.
- **keyphrases:** Internal links, that occur within the section (ignoring subsections).
- **text:** The actual text of the section.

```
{
  "filename":"/wiki/File:filename.jpg",
  "origformat":"jpg",
  "metapath":"c/c2/filename.json",
  "keyphrases":[ ... ],
  "imgpath":"c/c2/filename.jpg",
  "caption":""
}
```

Fragment A.3: JSON object that defines a *SimpleWiki* image

The skeleton of an image object is depicted in fragment A.3. Some of its properties are optional, as images are not necessarily downloaded. For instance, an image that belongs to a section, that does not contain any text, is omitted. Its properties are defined as follows:

- **filename:** The filename of the image.
- **origformat:** (optional) The original format of the image, before it was converted into JPEG format.
- **metapath:** (optional) The relative path to a JSON file containing metadata about the image (e.g. license information).
- **keyphrases:** A list of keyphrases (internal links) appearing in the caption.
- **imgpath:** (optional) The relative path to the preprocessed image.
- **caption:** The image caption.

B. Label Distributions among different Image Types

Different image types (e.g. charts or photographs) are intentionally used to adequately present the desired information. The annotation statistics presented in this appendix should enrich the explanations from subsection 4.2.3 by considering individual image types. As the image type distribution among the two annotated datasets is extremely biased, we only consider those where enough representatives appeared to allow one to draw meaningful conclusions, namely photographs for both datasets and drawings for the *SimpleWiki* dataset.

Label	0	1	2	3	4	5	6	7
Total	17	–	1	6	85	91	–	–
Percentage	8.5	–	0.5	3.0	42.5	45.5	–	–

(a) Distribution of MI labels.

Label	-1.0	-0.5	0.0	0.5	1.0
Total	7	28	82	67	16
Percentage	3.5	14.0	41.0	33.5	8.0

(b) Distribution of SC labels.

Table B.1.: Label distribution of the annotated samples from the *BBC News Database* dataset for image type *photograph*.

Table B.1 shows the distribution of labels among photos in the *BBC News Database*. On average, the pairs have a semantic correlation of 0.14 (and a mean of 0.0). Among these 200 photographs only 8 pairs have been assigned with relevant text snippets.

In addition, table B.2 shows the distribution of labels among photos in the *SimpleWiki* dataset. They have an average semantic correlation of 0.88 (and a mean of 1.0). 117 out of 397 contain relevant text snippets.

Furthermore, table B.3 reveals the distribution among drawings appearing in the *Sim-*

Label	0	1	2	3	4	5	6	7
Total	22	–	5	28	292	48	–	2
Percentage	5.54	–	1.26	7.05	73.55	12.09	–	0.5

(a) Distribution of MI labels.

Label	-1.0	-0.5	0.0	0.5	1.0
Total	–	3	16	56	322
Percentage	–	0.76	4.03	14.11	81.11

(b) Distribution of SC labels.

Table B.2.: Label distribution of the annotated samples from the *SimpleWiki* dataset for image type *photograph*.

Label	0	1	2	3	4	5	6	7
Total	6	–	1	10	57	14	–	3
Percentage	6.59	–	1.1	10.99	62.64	15.38	–	3.3

(a) Distribution of MI labels.

Label	-1.0	-0.5	0.0	0.5	1.0
Total	–	–	9	10	72
Percentage	–	–	9.89	10.99	79.12

(b) Distribution of SC labels.

Table B.3.: Label distribution of the annotated samples from the *SimpleWiki* dataset for image type *drawing*.

pleWiki dataset. 33 out of 91 drawings are associated with relevant text snippets. These drawings have an average SC value of 0.85 (mean: 1.0). It can be seen that drawing have a similar label distribution as photographs. Though, we believe that a further distinction between artistic and technical drawings would lead to less resp. more shared information, such that it can be seen solely from such distributions, that there are certain differences between image types.

C. The defined Distance Metrics on MI and SC Labels

This appendix provides the distance metrics that have been used to penalize misclassifications according to the loss function described in subsection 4.4.1.

MI labels are structured according to equation C.1.

$$\mathbf{D}_{MI} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{pmatrix} 0.0 & 1.0 & 0.8 & 0.6 & 0.4 & 0.2 & 0.5 & 0.5 \\ 1.0 & 0.0 & 0.2 & 0.4 & 0.6 & 0.8 & 0.5 & 0.5 \\ 0.8 & 0.2 & 0.0 & 0.2 & 0.4 & 0.6 & 0.8 & 0.8 \\ 0.6 & 0.4 & 0.2 & 0.0 & 0.2 & 0.4 & 0.6 & 0.6 \\ 0.4 & 0.6 & 0.4 & 0.2 & 0.0 & 0.2 & 0.4 & 0.4 \\ 0.2 & 0.8 & 0.6 & 0.4 & 0.2 & 0.0 & 0.2 & 0.2 \\ 0.5 & 0.5 & 0.8 & 0.6 & 0.4 & 0.2 & 0.0 & 1.0 \\ 0.5 & 0.5 & 0.8 & 0.6 & 0.4 & 0.2 & 1.0 & 0.0 \end{pmatrix} \end{array} \tag{C.1}$$

As it has been already mentioned in subsection 4.2.3.1, MI labels 0 to 5 have a natural inclusion structure:

$$0 < 5 < 4 < 3 < 2 < 1 \qquad .$$

This structure is reflected by the distance metric. The rare cases 6 and 7 allow a wide range of possible samples, so the defined distance can only be a coarse estimate.

The distance between pairs of SC labels is defined by equation C.2.

$$\mathbf{D}_{SC} = \begin{array}{c} \\ -1.0 \\ -0.5 \\ 0.0 \\ 0.5 \\ 1.0 \end{array} \begin{array}{c} \begin{array}{ccccc} -1.0 & -0,5 & 0.0 & 0.5 & 1.0 \end{array} \\ \begin{pmatrix} 0.0 & 0.0 & 1.0 & 1.0 & 1.0 \\ 0.0 & 0.0 & 0.0 & 1.0 & 1.0 \\ 1.0 & 0.0 & 0.0 & 0.0 & 1.0 \\ 1.0 & 1.0 & 0.0 & 0.0 & 0.0 \\ 1.0 & 1.0 & 1.0 & 0.0 & 0.0 \end{pmatrix} \end{array} \tag{C.2}$$

Semantic Correlation is hard to quantify, such that it has to be presumed, that there is a substantial variance among annotators. Though, the general direction should be identical. For this reason, equation C.2 does not penalize close misclassifications.

Bibliography

[1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*, 2015.

[2] Kobus Barnard and Keiji Yanai. Mutual information of words and pictures. *Information Theory and Applications*, 2, 2006.

[3] 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(Feb):1107–1135, 2003.

[4] Yevgeni Berzak, Yan Huang, Andrei Barbu, Anna Korhonen, and Boris Katz. Anchoring and agreement in syntactic annotations. In *EMNLP*, 2016.

[5] Haixun Wang Bin Shao. Understanding tables on the web. Technical report, March 2011. URL `https://www.microsoft.com/en-us/research/publication/understanding-tables-on-the-web/`.

[6] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[7] Léon Bottou. From machine learning to machine reasoning. *CoRR*, abs/1102.1808, 2011. URL `http://arxiv.org/abs/1102.1808`.

[8] Léon Bottou. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade*, pages 421–436. Springer, 2012.

[9] Jake Bouvrie. Notes on convolutional neural networks. 2006.

[10] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO captions: Data collection and evaluation server. *CoRR*, abs/1504.00325, 2015. URL `http://arxiv.org/abs/1504.00325`.

[11] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. Natural language processing (almost) from scratch. *CoRR*, abs/1103.0398, 2011. URL `http://arxiv.org/abs/1103.0398`.

[12] Marc Corio and Guy Lapalme. Generation of texts for information graphics. In *IN PROCEEDINGS OF THE 7TH EUROPEAN WORKSHOP ON NATURAL LANGUAGE GENERATION EWNLG99*, pages 49–58, 1999.

[13] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *J. Mach. Learn. Res.*, 2:265–292, March 2002. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=944790.944813`.

[14] 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.

[15] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2625–2634, 2015.

[16] Stephanie Elzer, Sandra Carberry, Ingrid Zukerman, Daniel Chester, Nancy Green, and Seniz Demir. A probabilistic framework for recognizing intention in information graphics. In *IJCAI*, pages 1042–1047. Professional Book Center, 2005.

[17] Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh K Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C Platt, et al. From captions to visual concepts and back. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1473–1482, 2015.

[18] Ali Farhadi, Ian Endres, Derek Hoiem, and David Forsyth. Describing objects by their attributes. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR*, 2009.

[19] Yansong Feng and Mirella Lapata. Automatic image annotation using auxiliary text information. In *ACL*, volume 8, pages 272–280, 2008.

[20] Yansong Feng and Mirella Lapata. Automatic caption generation for news images. *IEEE transactions on pattern analysis and machine intelligence*, 35(4):797–812, 2013.

[21] Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Marc' Aurelio Ranzato, and Tomas Mikolov. Devise: A deep visual-semantic embedding model. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2121–2129. Curran Associates, Inc., 2013. URL `http://papers.nips.cc/paper/5204-devise-a-deep-visual-semantic-embedding-model.pdf`.

[22] Pavel Golik, Patrick Doetsch, and Hermann Ney. Cross-entropy vs. squared error training: a theoretical and experimental comparison.

[23] Yunchao Gong, Liwei Wang, Micah Hodosh, Julia Hockenmaier, and Svetlana Lazebnik. *Improving Image-Sentence Embeddings Using Large Weakly Annotated Photo Collections*, pages 529–545. Springer International Publishing, Cham, 2014. ISBN 978-3-319-10593-2. doi: 10.1007/978-3-319-10593-2_35. URL `http://dx.doi.org/10.1007/978-3-319-10593-2_35`.

[24] Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton. Speech recognition with deep recurrent neural networks. *CoRR*, abs/1303.5778, 2013. URL `http://arxiv.org/abs/1303.5778`.

[25] Hua He, Kevin Gimpel, and Jimmy Lin. Multi-perspective sentence similarity modeling with convolutional neural networks.

[26] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[27] Raphael Hoffmann, Congle Zhang, and Daniel S Weld. Learning 5000 relational extractors. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 286–295. Association for Computational Linguistics, 2010.

[28] Laura Hollink, Adriatik Bedjeti, Martin van Harmelen, and Desmond Elliott. A corpus of images and text in online news. 2016.

[29] Haixun Wang Hongsong Li. Probase: A probabilistic taxonomy for text understanding. In *ACM International Conference on Management of Data (SIGMOD)*, May 2012. URL `https://www.microsoft.com/en-us/research/publication/probase-a-probabilistic-taxonomy-for-text-understanding/`.

[30] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[31] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. Convolutional neural network architectures for matching natural language sentences. *CoRR*, abs/1503.03244, 2015. URL `http://arxiv.org/abs/1503.03244`.

[32] David H Hubel and Torsten N Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243, 1968.

[33] Andrej Karpathy and Fei-Fei Li. Deep visual-semantic alignments for generating image descriptions. *CoRR*, abs/1412.2306, 2014. URL `http://arxiv.org/abs/1412.2306`.

[34] Andrej Karpathy, Armand Joulin, and Fei-Fei Li. Deep fragment embeddings for bidirectional image sentence mapping. *CoRR*, abs/1406.5679, 2014. URL `http://arxiv.org/abs/1406.5679`.

[35] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, pages 1725–1732, Washington, DC, USA, 2014. IEEE Computer Society. ISBN 978-1-4799-5118-5. doi: 10.1109/CVPR.2014.223. URL `http://dx.doi.org/10.1109/CVPR.2014.223`.

[36] Andrej Karpathy, Justin Johnson, and Fei-Fei Li. Visualizing and understanding recurrent networks. *CoRR*, abs/1506.02078, 2015. URL `http://arxiv.org/abs/1506.02078`.

[37] Yoon Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014. URL `http://arxiv.org/abs/1408.5882`.

[38] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. Character-aware neural language models. *arXiv preprint arXiv:1508.06615*, 2015.

[39] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, page 2012.

[40] G. Kulkarni, V. Premraj, V. Ordonez, S. Dhar, S. Li, Y. Choi, A. C. Berg, and T. L. Berg. Babytalk: Understanding and generating simple image descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2891–2903, Dec 2013. ISSN 0162-8828. doi: 10.1109/TPAMI.2012.162.

[41] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 2267–2273. AAAI Press, 2015. ISBN 0-262-51129-0. URL http://dl.acm.org/citation.cfm?id=2886521.2886636.

[42] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[43] Yann LeCun, Leon Bottou, Genevieve B. Orr, and Klaus Robert Müller. *Efficient BackProp*, pages 9–50. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998. ISBN 978-3-540-49430-0. doi: 10.1007/3-540-49430-8_2. URL http://dx.doi.org/10.1007/3-540-49430-8_2.

[44] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015.

[45] Wei Liu and Xiaoou Tang. Learning an image-word embedding for image auto-annotation on the nonlinear latent space. In *Proceedings of the 13th Annual ACM International Conference on Multimedia*, MULTIMEDIA '05, pages 451–454, New York, NY, USA, 2005. ACM. ISBN 1-59593-044-2. doi: 10.1145/1101149.1101249. URL http://doi.acm.org/10.1145/1101149.1101249.

[46] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. ISSN 1573-1405. doi: 10.1023/B:VISI.0000029664.99615.94. URL http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94.

[47] Lin Ma, Zhengdong Lu, Lifeng Shang, and Hang Li. Multimodal convolutional neural networks for matching image and sentence. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2623–2631, 2015.

[48] Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, and Alan L. Yuille. Explain images with multimodal recurrent neural networks. *CoRR*, abs/1410.1090, 2014. URL http://arxiv.org/abs/1410.1090.

[49] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[50] Vibhu O. Mittal, Giuseppe Carenini, Johanna D. Moore, and Steven Roth. Describing complex charts in natural language: A caption generation system. *Comput. Linguist.*, 24(3):431–467, September 1998. ISSN 0891-2017. URL http://dl.acm.org/citation.cfm?id=972749.972754.

[51] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going deeper into neural networks. https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html, 2015. Accessed: 2016-12-04, Image License: https://creativecommons.org/licenses/by/4.0/.

[52] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 689–696, 2011.

[53] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL http://dx.doi.org/10.3115/1073083.1073135.

[54] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *ICML (3)*, 28:1310–1318, 2013.

[55] A Ramisa, F Yan, F Moreno-Noguer, and K Mikolajczyk. Breakingnews: Article annotation by image and text processing. corr abs/1603.07141 (2016).

[56] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[57] Hong Shen and Anoop Sarkar. *Voting Between Multiple Data Representations for Text Chunking*, pages 389–400. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. ISBN 978-3-540-31952-8. doi: 10.1007/11424918_40. URL http://dx.doi.org/10.1007/11424918_40.

[58] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[59] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[60] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015. URL http://arxiv.org/abs/1512.00567.

[61] Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 173–180, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. doi: 10.3115/1073445.1073478. URL http://dx.doi.org/10.3115/1073445.1073478.

[62] S. Viji. Term and document correlation and visualization for a set of documents. Technical report, Stanford University, 2002.

[63] Mauricio Villegas and Roberto Paredes. Image-text dataset generation for image annotation and retrieval. *Prometeo*, page 014, 2009.

[64] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. *CoRR*, abs/1411.4555, 2014. URL http://arxiv.org/abs/1411.4555.

[65] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[66] Jiao Xue, Youtian Du, and Hanbing Shui. Semantic correlation mining between images and texts with global semantics and local mapping. In *International Conference on Multimedia Modeling*, pages 427–435. Springer, 2015.

[67] Fei Yan and Krystian Mikolajczyk. Deep correlation for matching images and text. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3441–3450, 2015.

[68] Keiji Yanai and Kobus Barnard. Image region entropy: a measure of visualness of web images associated with one concept. In *Proceedings of the 13th annual ACM international conference on Multimedia*, pages 419–422. ACM, 2005.

[69] Yi Zhang, Jeff Schneider, and Artur Dubrawski. Learning the semantic correlation: An alternative way to gain from unlabeled text. In *Proceedings of the 21st International Conference on Neural Information Processing Systems*, NIPS'08, pages 1945–1952, USA, 2008. Curran Associates Inc. ISBN 978-1-6056-0-949-2. URL `http://dl.acm.org/citation.cfm?id=2981780.2982024`.

[70] Y. T. Zhuang, Y. Yang, and F. Wu. Mining semantic correlation of heterogeneous multimedia data for cross-media retrieval. *IEEE Transactions on Multimedia*, 10(2): 221–229, Feb 2008. ISSN 1520-9210. doi: 10.1109/TMM.2007.911822.

[71] C. Lawrence Zitnick, Devi Parikh, and Lucy Vanderwende. Learning the visual interpretation of sentences. In *Proceedings of the 2013 IEEE International Conference on Computer Vision*, ICCV '13, pages 1681–1688, Washington, DC, USA, 2013. IEEE Computer Society. ISBN 978-1-4799-2840-8. doi: 10.1109/ICCV.2013.211. URL `http://dx.doi.org/10.1109/ICCV.2013.211`.