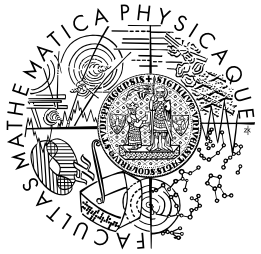


FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Hoa Trong Vu

Grounding Natural Language Inference on Images

Institute of Formal and Applied Linguistics, Faculty of Mathematics Physics

Supervisors:	Dr. Pavel Pecina
Study programme:	Master of Computer Science
Study branch:	Computational Linguistics

Prague 2018

I, Hoa Trong Vu, declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ..... signature of the author

Title: Grounding Natural Language Inference on Images

Author: Hoa Trong Vu

Institute: Institute of Formal and Applied Linguistics, Faculty of Mathematics Physics

Supervisor: Dr. Pavel Pecina, Associate Professor, Institute of Formal and Applied Linguistics, Charles University, Prague.

Abstract: Despite the surge of research interest in problems involving linguistic and visual information, exploring multimodal data for Natural Language Inference remains unexplored. Natural Language Inference, regarded as the basic step towards Natural Language Understanding, is extremely challenging due to the natural complexity of human languages. However, we believe this issue can be alleviated by using multimodal data. Given an image and its description, our proposed task is to determined whether a natural language hypothesis contradicts, entails or is neutral with regards to the image and its description. To address this problem, we develop a multimodal framework based on the Bilateral Multi-perspective Matching framework. Data is collected by mapping the SNLI dataset with the image dataset Flickr30k. The result dataset, made publicly available, has more than 565k instances. Experiments on this dataset show that the multimodal model outperforms the state-of-the-art textual model.

Keywords: Natural Language Inference, Multimodal, BiMPM.

Acknowledgements

In the summer of 2017, when I had no choice but to request for a study extension, I never thought I could finally get the thesis done as it is now. My health issues and other personal problems prevented me from getting the work done. However, I was immensely helped by many people for which it becomes one of the most unforgeable experiences in my life. Being able to reach this stage of the thesis and about to get a job make me more than happy. When I sit back and recall the difficult time as well as the people who helped me, I would like to express my gratefulness towards them.

First of all, I would like to deeply thank my supervisor Professor Pavel Pecina for being constantly patient towards my slow progress. Professor Pecina is actually my co-supervisor but ends up do the work of a main-supervisor, reading my write-up content and provide suggestion and feedback. I got more and more confident and finally get the work done.

I would like to thank the Erasmus Mundus Erasmus Mundus European Masters Program in Language and Communication Technologies (LCT) for providing a scholarship for the author. I would like to extend my gratitude to Professor Markéta Lopatková, Ms. Kristýna Kysilková and Ms. Solange Petracchi for keeping me in progress.

I am sincerely grateful to Aliia, Somayeh for your help with the tedious annotation work and Xiaoyu for the just-in-time proof reading. Another big thank to Zhiguo Wang who shared the code of the original BiMPM model.

I owe my Vietnamese Prague friends (Chuong, Hai, Bao Han, Hiep, Hanh, Thang ...) a huge thank for their mentally support, being around me during the hard time of my life, always be available whenever I need. I guess they have no idea how much they mean to me. They are really my second family in Prague.

Finally, I would like to express my deepest thank to my family members who stay in the other side of the world, trying their best to raise me up, so I can do my best in my career. Their love and words made me believe in myself when I believed in myself least.

Contents

Chapter 1

Introduction

Natural Language Inference (NLI) or *Recognizing Textual Entailment* (RTE) is the problem of determining whether a natural language hypothesis h can be inferred from a natural language premise p (entailment) or contradiction or whether it contradict to p (contradiction) or whether it is neutral with respect to p. This problem is extremely challenging due to the complexity of natural languages. However, we believe that the issue can be alleviated by using multimodal data.

Recognizing relation between sentences is difficult, but if it is solved, it would bring crucial benefits for many practical applications. For example, a question answering system could use an NLI system to examine whether a target question can be inferred from some candidate answers. A machine translation system can use NLI as an evaluation component to measure the semantic equivalence between generated candidates and gold-standard translations, instead of using BLEU score which works on the lexical surface. Recently, NLI has also become a crucial testing ground for the development of distributional semantic representation (Nangia et al. 2017). MacCartney (2009) regards NLI as a necessary step towards true natural language understanding.

Neural networks are the current dominant methods for NLI (Bowman, Gauthier, et al. 2016; Wang, Hamza, and Florian 2017; Chen et al. 2017b; Nangia et al. 2017). First, these methods are effective in modeling semantic representations, sentence structures. In fact, a simple multilayer perceptron can perform on par with traditional methods (Bowman, Angeli, et al. 2015) and later developed models perform remarkably better. Second, neural networks usually make use of pre-trained word embedding vectors which play a role similar to that of the knowledge base in a symbolic logic system. The difference is these vectors are trained from the vast amount of text from the Web which makes it less expensive and domain-restricted. Finally, it is feasible to train a joint multimodal model using neural networks as they are all represented as vectors and matrices.

Recently, as a result, there is a great interest among the Natural Language Processing community and Computer Vision community in problems that requires linguistic and visual data. The recent years have seen a surge in multimodal data with billions of videos and image posted everyday. In fact, videos and images have even become the dominant modality as in Vine, Instagram or Youtube.

Interestingly, in science, research also suggests that meaning of words and sentences in natural language should be grounded in physical objects and actions (Roy 2005; Andrews, Vigliocco, and Vinson 2009).

In light of previous research, in this thesis, we are interested in examining the multimodal settings for NLI. We would like to see if visual information could or should be used in order to boost performance of an NLI system.

1.1 Goal and Contribution

The purpose of this thesis is to explore an effective method to exploit visual information to improve the NLI task. The challenge is how to make an inference from an image or how to utilize both modalities in a model. To this end, we extend the textual BiMPM framework (Wang, Hamza, and Florian 2017) to work with images. Our proposed multimodal model outperforms the textual BiMPM.

We form a dataset for this purpose from SNLI and Flickr30k. The dataset contains more than 565k instances. We make this dataset publicly available together with the implementation. To the best of our knowledge, our work and the dataset is a first effort in this direction.

1.2 The Multimodal Natural Language Inference Task

We propose an extension of NLI called Multimodal Natural Language Inference Task (MNLI). Given an image I and its description as a premise $p[p_1, p_2...p_m]$, the task is to determine whether a hypothesis $q[q_1, q_2...q_n]$ entails, contradicts or is neutral with respect to the premise, where this process can be supported by information from the image.

The entailment relation between sentences is defined based on a likelihood condition: if a typical human reading the premise would infer that the conclusion (hypothesis) is most likely true (entailment), its negation is most likely true (contradiction) or the conclusion can be either true or false (neutral). This definition, as many other definitions in NLP, refers to human understanding of languages, their ability to observe, and assumes commonsense knowledge, on which inference judgment relies. For instance, the premise "People trying to get warm in front of a chimney" and the hypothesis "People trying to get warm at home" are highly likely to be in an entailment relation. Meanwhile, given "A girl playing a violin along with a group of people", the negation of "A girl is washing a load of laundry" is likely to be true, which makes the relation between the two sentences one of contradiction.

(A) P: *A family in front of a chimney* and H: *A family trying to get warm*

(B) P: *A girl playing a violin along with a group of people* and H: *A girl is washing a load of laundry.*

FIGURE 1.1: Premise, Hypothesis and Image examples

Our question is whether having an image that illustrates the event (e.g., Figure 1.1a, Figure 1.1b) can help a model to capture the relation. The proposed task in this thesis is to answer this question.

In this task, the image plays the role of a grounding source. For example, a normal human being can easily draw the same labels for those three hypothesis with regards to only the image. Indeed, for this simple task, it is probably easier, more intuitive for humans to work with visual data. However, whether computers can have a similar ability remains a question.

On the assumption that we have an ideal matching scheme between two modalities, visual information can help the inference process. First, there is less potential ambiguities in visual scenes than in natural languages. The fact that natural languages can have many linguistic phenomena makes it harder to deal with. Lynch, Aryafar, and Attenberg (2016) found that, in the automatic ranking search results, thumbnails seem to contain less "noise" and multimodal data are more reliable. Second, information on the image can be verified by the text and vice versa. This helps to strengthen the inference process. Third, the image is a richer input for the inference task. For instance, in the Figure 1.1a, we can conclude that the daughter is sitting next to the mother, but neither of these pieces of information is in the textual premise.

The inclusion of images can also alter the entailment relations between sentences. For example, to a "blind" model the sentences of the sentence pair in Figure 1.2 would seem to be unrelated, but when the two sentences are viewed as descriptions of the image, they do become related.

A suitable MNLI system therefore has to perform two sub-tasks: (a) first it has to ground its linguistic representations of P, H or both in the visual non-linguistic data; (b) it needs to reason about the possible relationship between P and H (modulo the visual information).

It is important to notice that NLI works with *directional* relationships which means if the A entails B, the reversed direction does not hold. If it does, the

FIGURE 1.2: P: *People trying to get warm* and H: *People are outside on a chilly day.* NLI: unrelated vs. Multimodal NLI: related

problem becomes *paraphrase identification* which can be solved by running an NLI system with input A, B and B, A.

1.3 Related work

In recent years, quite some work has been carried out on grounding language through vision since the ability to draw inferences from a given text has long been considered a milestone in formal and computational semantics. The major multimodal tasks that have been proposed include but are not limited to, multimodal translation (Specia et al. 2016) image captioning (IC, e.g. (Hodosh, Young, and Hockenmaier 2013; K. Xu et al. 2015)) and visual question answering (VQA, eg. (Malinowski and Fritz 2014; Antol et al. 2015)), and even more recently, visual reasoning (Johnson et al. 2017; Suhr et al. 2017) and Visual Dialog (Das et al. 2017). Our focus is on multimodal natural language inference (MNLI). While the literature on NLI or TE is rather vast, MNLI is still rather unexplored territory.

1.3.1 Textual Entailment

Early successful methods heavily rely on lexical surface representation and syntactic similarity, which is inspired from information retrieval and machine translation. In general, the possibility of entailment is determined based on lexical overlap between the premise and the hypothesis. Obviously the systems could not require all the text to be matched, they usually train to get a threshold of the proportion of the text to be able to predict (Jijkoun, Rijke, et al. 2005). Geffet and Dagan (2005) extends the above idea by introducing stemming, lemmatizing, lexical resources such as Wordnet and probabilistic mappings produced by distributional similarity approaches to increase the chance of matching. Glickman and Dagan (2005) compares the words of the premise and hypothesis using

the results from an Internet search engine. Their system achieved the best performance in the first PASCAL RTE Challenge (Dagan, Glickman, and Magnini (2006)).

For semantic researchers, applying formal logic is the most intuitive answer for NLI. This is done by translating natural language into logical formulas then use logical inference to determine the relation (Akhmatova 2005; Bayer et al. 2005; Bos and Markert 2006). In the mid-'90s, a group of formal semanticists developed FraCaS (Framework for Computational Semantics) Cooper et al. (1996)[1]. Their dataset contains logical entailment problems in which a conclusion has to be derived from one or more premises (but not necessarily all premises are needed). The entailments are driven by logical properties of linguistic expressions, like the monotonicity of quantifiers, or their conservativity property etc. Hence, the framework outputs 'entailment' if and only if in all the interpretations (worlds) in which the premises are true the conclusion is also true; otherwise the framework output 'contradiction'.

Figure 1.3 shows a simple example of applying formal logics where the Hypothesis is derived from the Premise and result in the *entailment* label. For purposes of simplicity, the example knowledge base here contains only two rules. If the Hypothesis was "John Smith owns a horse" then the result should be *not entail* because there is no rule in the knowledge base that allows the inference that owning a Honda Civic implies owning a horse. Therefore, after applying the existing statements in the knowledge base (neither of which is compatible with the Hypothesis) to the Hypothesis representation, the system would halt and output that the Premise does not entail the Hypothesis.

Formal logic methods have advantages dealing with negations, quantifiers, conditions; however, there are several challenges when applying these methods (Sammons 2015; MacCartney 2009). First, they are domain-restricted. The knowledge base must be extremely large to enable inference of arbitrary premises and hypotheses. Second, the system has to handle open challenging natural language phenomena including idioms, ellipsis, anaphora, paraphrase, ambiguity, vagueness, aspect, lexical semantics, the impact of pragmatics, and so on. Third,

In 2005, the PASCAL RTE (Recognizing Textual Entailment) challenge was launched, to become a task organized annually. In 2008, the RTE-4 committee made the task more fine-grained by requiring the classification of the pairs as "entailment", "contradiction" and "unknown" (Giampiccolo et al. 2008). The RTE datasets, unlike FraCaS, contain real-life natural language sentences and the sort of entailment problems which occur in corpora collected from the web. Importantly, the sentence pair relations are annotated as entailment, contradiction or neutral based on a likelihood condition: if typically a human reading the premise would infer that the conclusion (called hypothesis) is most likely true (entailment), its negation is most likely true (contradiction) or the conclusion can be either true or false (neutral). At SemEval 2014, in order to

[1] A cleanly processable version of it has been made available only recently: http://www-nlp.stanford.edu/~wcmac/downloads/fracas.xml

Premise: John Smith bought a Honda Civic.
Hypothesis: John Smith owns a car.
World Knowledge:
1. $\forall X, HondaCivic(X) \rightarrow car(X)$ (a Honda Civic is a car)
2. $\forall X, Y buy(X, Y) \rightarrow own(X, Y)$ (if X buys Y, then X owns Y)

Proof: Initial state of belief (representation of Text):
$\exists A, BJohnSmith(A) \wedge HondaCivic(B) \wedge buy(A, B)$
(John Smith bought a Honda Civic)

Target assertion (representation of Hypothesis):
$\exists C, DJohnSmith(C) \wedge car(D) \wedge own(C, D)$

1. Apply rule 1 with **premise** clause "Honda Civic(B)":
$HondaCivic(B) \rightarrow car(B)$

State of belief is now:
$\exists A, BJohnSmith(A) \wedge car(B) \wedge buy(A, B)$

2. Apply rule 2 with Premise clause "buy(A,B)":
$buy(A, B) \rightarrow own(A, B)$

State of belief is now:
$\exists A, BJohnSmith(A) \wedge car(B) \wedge own(A, B)$

State of belief entails target by substitution of variables, therefore T entails H.

FIGURE 1.3: An example of formal proof of an entailment pair (Sammons 2015)

evaluate Compositional Distributional Semantics Models focusing on the compositionality ability of those models, the SICK dataset (Sentences Involving Compositional Knowledge) was used in a shared entailment task (Marelli et al. 2014). Sentence pairs were obtained through re-writing rules and annotated with the three RTE labels via a crowdsourcing platform. Both in RTE and SICK the label assigned to the sentence pairs captures the relation between the two sentences.

Following the success of the RTE shared task, there has been an increasing emphasis on data-driven approaches which, given the hypothesis H and premise P, seek to classify the semantic relation (see Sammons, Vydiswaran, and Roth 2011 for a review). Specifically, neural approaches have come to dominate the scene, as shown by the recent RepEval 2017 task (Nangia et al. 2017), where all submissions relied on bidirectional LSTM models, with or without pretrained embeddings. This is due to the recently released dataset SNLI by (Bowman, Angeli, et al. 2015). The dataset contains 570k pairs of sentences together with their relation between each pair is either *entailment*, *neutral* or *contradiction*.

The premise is taken from an image corpus called Flickr30k. Before this release, the research community usually relied on datasets from the RTE challenges[2], the Sentences Involving Compositional Knowledge (SICK) from SemEval 2014 task (Marelli et al. 2014). Those datasets are very high quality, manually created but small in size which is not suitable for neural networks methods. The size of SNLI, however, makes it feasible for applying neural networks methods which usually require a large amount of data. Indeed, the success of this dataset has been demonstrated in a vast number of studies, some of them even surpass human performance[3].

While a simple neural network architecture proposed in Bowman, Angeli, et al. (2015) achieves 77.6% on SNLI and approach the state-of-the-art result on SICK dataset, later proposed models are more advanced. Mou et al. (2016) proposed a "sentence encoder" architecture where 2 encoders with shared parameters are employed to turn the hypothesis and premise into two vectors. By using a tree-based convolutional neural network in each encoder, their model obtains an accuracy of 82.1% on the SNLI dataset. These vectors are then concatenated with their element-wise product, sum, or difference. Figure 1.4 illustrates this architecture. This architecture has become common for testing sentence representation (Nangia et al. 2017).

Wang, Hamza, and Florian (2017) propose a far more elegant attention-based matching framework. In the framework, each context-aware token of either premise or hypothesis is matched with the tokens in other sentence using various weighting strategies. The multi-perspective matching operation is an extension of cosine similarity where inputs are projected to different spaces and each output value is cosine similarity of the inputs in each space. The single BiMPM yields 86.9% on SNLI. In this thesis, we make use of this framework to evaluate the effectiveness of visual information.

1.3.2 Multimodal and grounding tasks

In recent years, quite some models have been proposed to integrate the language and vision modalities; usually the integration is operationalized by element-wise multiplication between linguistic and visual vectors. These research has shown the benefit of using multi-modal data over text-only data. Specia et al. (2016) show improvements in translating image captions by using visual information from the images. Pasunuru and Bansal (2017) found that in a multitask setting, video information can help the entailment generation task and the entailment generation task to helps improve video captioning.

Though the interest in these multi-modal solutions has spread in an astonishing way thanks to various multimodal tasks proposed, including the IC, VQA, visual reasoning and visual dialogue tasks mentioned above, very little work has been done on grounding entailment. Interestingly, Young et al. (2014) have

[2] https://tac.nist.gov//2011/RTE/

[3] Statistics of performance on the dataset are available on https://nlp.stanford.edu/projects/snli

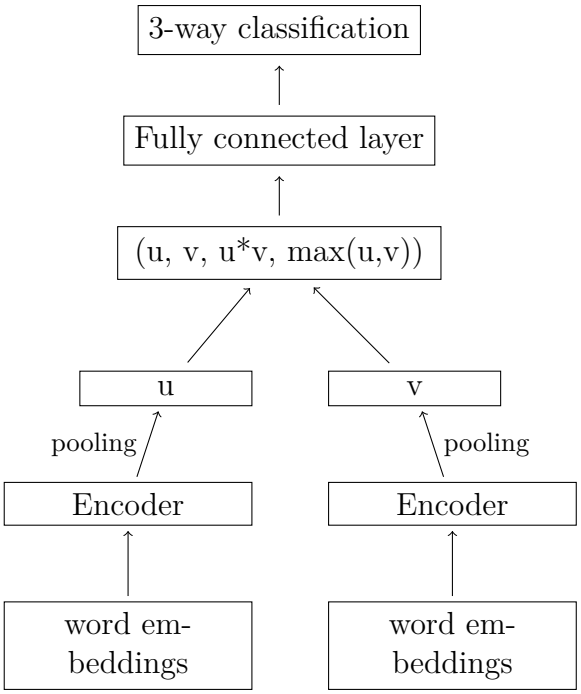


FIGURE 1.4: A "sentence encoder" architecture.

proposed the idea of considering images as the "possible worlds" on which sentences find their denotation. Hence, they released a "visual denotation graph" which associates sentences with their denotation (sets of images). The idea has been further exploited in Lai and Hockenmaier (2017) and Han, Gomez, and Mineshima (2017). Vendrov et al. (2016) look at hypernymy, textual entailment and image captioning as special cases of a single visual-semantic hierarchy over words, sentences and images, and they claim that modeling the partial order structure of this hierarchy in the visual and linguistic semantic spaces improves model performance on those three tasks. We share with this work the idea that the image can be taken as a possible world. However, we do not use sets of images to obtain the visual denotation of text in order to check whether entailment is logically valid/highly likely. Rather, we take the image to be the world/situation in which the text finds its interpretation.

1.4 Thesis structure

Chapter 2 gives the necessary basic models that are required for building the proposed models and delivers all important definitions. Section 2.1 explains how to build a feed-forward network, the basic component of neural networks. Subsequently, we present some successful models in Computer Vision and Natural Language Technology in Section 2.2 and Section 2.4. These models are the main modules in our proposed models.

Chapter 3 presents the methodologies we use to demonstrate the effectiveness of multimodal data. The textual framework BiMPM is presented in Section 3.2. Our proposed multimodal framework is presented in Section 3.3

Chapter 4 presents our experiments with our proposed model on the SNLI dataset. We first present our data collection procedure and experiment results then give some insights into how the model performs.

Chapter 5 gives a conclusion of the thesis and presents possible improvements that could be made.

Chapter 2

Background on Neural Networks

The first development of neural networks dates back in 1943 when McCulloch and Pitts developed a linear model that could test if output of $f(x, w)$ is positive or negative. This was followed by the perceptron (Rosenblatt 1958) which is the first model that can learn weights from data. After a long history with many efforts and inventions, neural networks made a recent comeback as Hinton, Osindero, and Teh (2006) and Bengio et al. (2007) came up with a strategy to train a deep neural network. Since then, the term neural networks aka *deep learning* started to become common in different communities studying different tasks including natural language processing, vision, motion planning and speech recognition.

In this chapter, we present the necessary basic models that are required for building the proposed models and deliver all important definitions. We first start with the basic Feed-forward networks in Section 2.1 then we describe some successful models in Computer Vision and Natural Language Technology, namely Convolutional Neural Network in Section 2.2 and Long Short-Term Memory in Section 2.4. In the end of this chapter, we introduce some common usage of neural networks in NLP which also appear in our proposed model.

2.1 Feed-forward Neural Network

Based on an excellent overview of neural networks by **bishop:2006:PRML** we present here a brief description of Feed-forward network and related techniques including backpropagation, activation functions and regularizations. We start with the common logistic regression as it is usually regarded as a one-layer neural network.

2.1.1 Logistic Regression

Logistic Regression for multiclass classification:

$$y(x, w) = f(\sum_{j=1}^{M} w_j \phi_j(x)) \tag{2.1}$$

Where w_j are trainable parameters. $\phi(x)$ are feature functions, one of these is set to constant, say $\phi_0(x) = 1$ so that w_0 is a bias. $f(.)$ is a softmax function, a nonlinear activation function turns numbers into proper probabilities. The posterior class probabilities are given by

$$p(\mathcal{C}_k|\phi) = y_k(\phi) = \frac{\exp(a_k)}{\sum_j \exp(a_j)} \tag{2.2}$$

where the 'activations' a_k are given by

$$a_k = w_k\phi \tag{2.3}$$

To train this model, we use maximum likelihood to adjust the parameters w. Let $\{\phi_n, T\}$ be the data set where $\phi_n = \phi(x_n)$ and T is the N x Kmatrix of one-hot target variables, each variable is a vector with all elements zero except the k^{th} value for the target class $\mathcal{C}_k$. $y_{nk} = y_k(\phi_n)$. The likelihood function is given by:

$$p(T|w_1, ..., w_K) = \prod_{n=1}^{N}\prod_{k=1}^{K} p(\mathcal{C}_k|\phi)^{t_{nk}} = \prod_{n=1}^{N}\prod_{k=1}^{K} y_{nk}^{t_{nk}} \tag{2.4}$$

We take the negative logarithm:

$$E(w_1, ..., w_K) = -\ln p(T|w_1, ..., w_K) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \ln y_{nk} \tag{2.5}$$

which is the *cross entropy* cost function for multiclass classification.

Now, to minimize this cost, we take the gradient of the function regarding one parameter w_j. First, we calculate the derivatives of y_k(result of softmax function) with respect to the activations:

$$\frac{\partial y_k}{\partial a_j} = y_k(I_{ki} - y_i) \tag{2.6}$$

where I_{kj} are elements of the identity matrix. Making use of equation 2.6 and $\sum_k t_{nk} = 1$, we obtain

$$\nabla_{w_j} E(w_1, ..., w_K) = \sum_{n=1}^{N}(y_{nj} - t_{nj})\phi_n \tag{2.7}$$

As we see in the Equation 2.7, the contribution to the gradient from data point n is given by the lost $y_{nj} - t_{nj}$ between the target value and the prediction value, times the feature function. For the sake of simplicity, we could formulate an on-line algorithm in which data are presented one at a time and the w_j is updated using

$$w^{(t+1)} = w^{(t)} - \eta\nabla_{w_j} E \tag{2.8}$$

in which η is the learning rate.

Linear models are simple, stable and easy to train; however their modeling ability is limited. For an M-dimensional feature space, this model requires only M trainable parameters. Linear operations are relatively cheap to compute. They usually are optimized into matrix multiplications and are executed really fast using GPUs. However, linear models can only model linear relations which do not always apply in practical applications. A win-win solution is to introduce non-linearities into these models. The way to do that is to stack multiple linear models together with non-linear functions and what we get are feed-forward neural networks. Logistic regression model itself is a one-layer feed-forward neural network.

2.1.2 Feed-forward Neural Network

As discussed above, the idea of these models is extended from logistic regression which take the form 2.1 to represent more complex relations. Our goal is to extend the feature functions $\phi_j(x)$ to depend on parameters which can be trained along with w. There are many ways to achieve this including using the function following the same form 2.1, that makes each feature function become a nonlinear combination of inputs with trainable parameters. The *activations* a_k in Equation 2.3 now are written as:

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} h_j + w_{k0} \tag{2.9}$$

$$h_j = g(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}) \tag{2.10}$$

where g is a nonlinear activation function which computes *hidden units* h_j from inputs. K, D, M are the numbers of outputs, hidden units and inputs respectively. w_{kj} and w_{ji} are trainable parameters.w_{k0} and w_{j0} are biases.

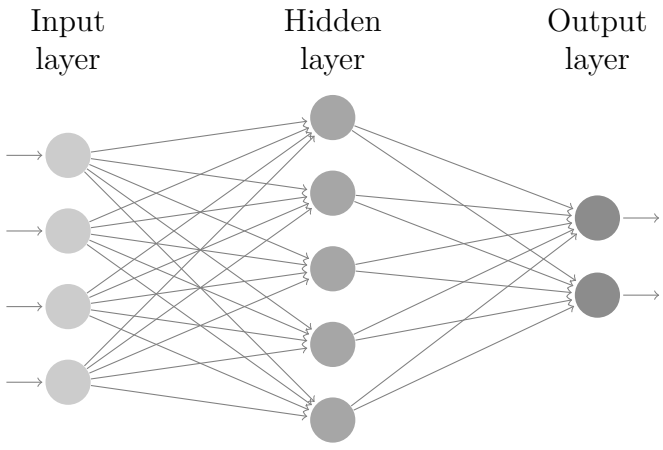


FIGURE 2.1: An example of a 2-layer feed-forward network where inputs, hidden units, output are denoted by nodes and weights are represented by arrows

The model above is called 2-layer feed-forward network(FFN), illustrated in Figure 2.1. The information flows through the network in the forward direction, from the input layer, through a hidden layer and finally to the output layer. There are no feedback connections connecting later stages back to the input. We will discuss some networks having feedback connection in Section 2.4 including Recurrent Neural Network and Long Short-Term Memory.

These models are called networks as the function they represent is a composition of other functions that can further be decomposed into other functions. This can be conveniently illustrated as a network of chain structure. The first function transforming the inputs is the *first layer*. The final layer is called *output layer*. The depth of the model is defined as the number of the computation layers not including the input layer. This is where the term "deep learning" comes from, although "deep learning" can also refer to methods for learning feature representation at successively higher, more abstract layers(Deng, Yu, et al. 2014). Values at the middle layers are not known for the training data and the training process needs to make use of these layers together with the output layer to give the desired output. These layers are called *hidden layers*, each of them contains many *hidden units*.

The term *neural* comes from the fact that these models are loosely inspired by the biological nature of the human brain. Each *unit* in a layer plays a similar role as a *neuron* as it also receives input from other units and computes its own activation value. However, Goodfellow, Bengio, and Courville (2016) points out the fact that neural networks nowadays are guided by many mathematics and engineering and it is better to treat feed-forward network as function approximation machines rather than as models of brain functions.

The simplest way to train this model is using gradient information to adjust the weights which take a small step in the direction of negative gradient:

$$w^{(\tau+1)} = w^{(\tau)} - \eta \nabla E(w^{(\tau)} \tag{2.11}$$

in which $\eta > 0$ is the learning rate. After each update, the weights are adjusted to the direction of decrease of the error function E and this method is known as *Gradient Descent*. Although it is very intuitive, it requires the whole training set to be processed at a time. The calculation of gradient is discussed in the section 2.1.3.

Goodfellow, Bengio, and Courville (2016) highlight that the difference between training a linear model and a feed-forward network is that nonlinearity turns most common loss functions to become non-convex. This means it is not guaranteed that applying stochastic gradient descent to these functions will find convergence. It is advised to initialize all weights to small random values and biases to either zero or small positive values. Stochastic gradient descent, in contrast to the on-line algorithm we mention in Section 2.1.1, updates parameters after seeing a few training examples instead of the full training set.

Despite the fact that feed-forward networks are straightforward, its computation ability is nontrivial. The universal approximation theorem (Hornik 1991)

claims that a standard feed-forward network with a single hidden layer that contains finite number of hidden units and with arbitrary activation function can approximate any Borel measurable function with any desired non-zero amount of error.

The universal approximation theorem means that neural networks can learn arbitrary functions; however, it is not guaranteed that the training algorithm will be able to find the right parameters or will not suffer from overfitting. The theorem also does not specify how large the network should be and in the worst case an exponential number of outputs may be required (Barron 1993). However, using deeper models would help to reduce number of units needed and reduce the errors. This explains how the term "deep learning" is getting more and more popular.

2.1.3 Error Backpropagation

As we already discussed in the previous section, we can optimize weight parameters by taking step by step toward the direction of negative gradient of the error function. We also knew the gradient of error function at the output layer but we do not know the gradient with regard to the weights at the hidden layers. The solution is using *error backpropagation* or simply *backprop* which alternately passes information forward and backward through the network. The formula is illustrated on a 2-layer FFN together with a sum of squares error function but can be applied to many other kinds of network, not just the FFN.

The backprop algorithm has two stages: forward and backward. In the forward stage, we apply input vector x_n to the network, forward it through the network to find the activation values of hidden units and output units using the Equation 2.9 and 2.10. Specifically, at the first layer:

$$y_k = \sum_i w_{ki} x_i \tag{2.12}$$

And at the hidden layers, activations are calculated by

$$a_j = \sum_i w_{ji} z_i \tag{2.13}$$

$$z_j = h(a_j) \tag{2.14}$$

In the backward stage, we need to calculate the derivatives of the error function with respect to each of the weights. Let us say we train the network using maximum likelihood and the error function is sum of square error of each data point:

$$E(w) = \sum_{n=1}^{N} E_n(w) \tag{2.15}$$

$$E_n = \frac{1}{2} \sum_k (y_{nk} - t_{nk})^2 \tag{2.16}$$

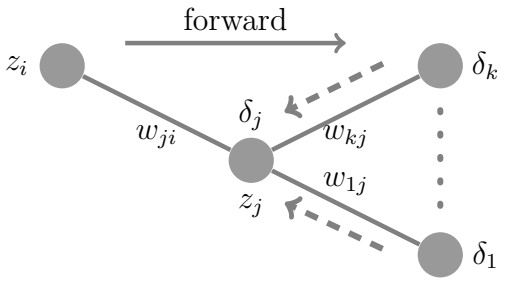


FIGURE 2.2: Calculate error δ_j at hidden unit j by backpropagate (dash arrows) errors from all the nodes that j sends information

where $y_{nk} = y_k(x_n, w)$ is output at data point n. Now let us consider the derivative of E_n with respect to a weight w_{ij}. We notice that E_n depends on the weight w_{ji} only by the sum a_j given by Equation 2.13. Apply chain rule, we get the derivative:

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j}\frac{\partial a_j}{\partial w_{ji}} \tag{2.17}$$

Using 2.13 we have:

$$\frac{\partial a_j}{\partial w_{ji}} = z_i \tag{2.18}$$

Let

$$\delta_j = \frac{\partial E_n}{\partial a_j} \tag{2.19}$$

By substituting 2.18 and 2.19 to 2.17, we have

$$\frac{\partial E_n}{\partial w_{ji}} = \delta_i z_i \tag{2.20}$$

Equation 2.20 tells us that we only need to calculate the error δ_j in order to obtain the required derivative. For the output unit, we have the error for each class:

$$\delta_k = \frac{\partial E_n}{\partial a_k} = y_k - t_k \tag{2.21}$$

For hidden units, let us use the chain rule again:

$$\delta_j = \frac{\partial E_n}{\partial a_j} = \sum_k \frac{\partial E_n}{\partial a_k}\frac{\partial a_k}{\partial a_j} \tag{2.22}$$

where the sum is over all the k units that the current unit sent connections. The backward process is shown in the figure 2.2. The units k could be either hidden units or output units. By substituting 2.19 into 2.22 and making use of 2.13 and 2.14, we have:

$$\delta_j = h'(a_j)\sum_k w_{kj}\delta_k \tag{2.23}$$

This is the *backpropagation* formula which tells us to calculate the error backward from the later units in the network and because we already know the error value for the output, all the error δ can be calculated recursively.

2.1.4 Activation functions

Activation functions are among the key components in neural networks. In general, activation functions are used to introduce nonlinearities into the model, making it possible to represent complex functions. In section 2.1.1, the nonlinear softmax function is used to turn numbers into proper probabilities. In theory, any differentiable non-linear function can be used as an activation function, however, in practice only a small fraction of them are used and *tanh, ReLU, sigmoid* are among the most prevalent.

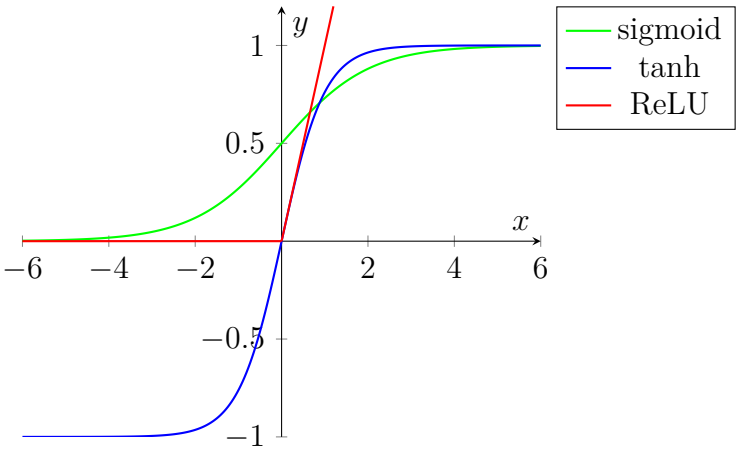


FIGURE 2.3: Commonly used activation functions

Sigmoid (Equation 2.24,2.25) compress its inputs into range $(0, 1)$, therefore it is used when we need to get proper probabilities. However, sigmoid suffers from the *vanishing gradient problem* issue as the derivative of the sigmoid function has near-zero value everywhere except for when the input has values of 0. This means that further layers in the backprobagation process will receive considerably smaller gradient. Consider using a fixed learning rate, parameters at the first layers may fluctuate largely, while parameters at further layers may only be changed slightly. Another issue with sigmoid is that its output is not zero-centered, which causes difficulties during optimization.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.24}$$

$$f'(x) = f(x)(1 - f(x)) \tag{2.25}$$

Tanh (Equation 2.26: 2.27) has range of $(-1, 1)$. Its output is zero-centered; however, it also suffers from *vanishing gradient problem* for the similar reason.

$$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{2.26}$$

$$f'(x) = 1 - f^2(x) \tag{2.27}$$

ReLU (Equation 2.28, 2.29): A rectified linear unit outputs zero if the input smaller than zero, and outputs the input otherwise. It is commonly used in convolutional neural networks. Particularly, ReLU has a constant derivative (1) for $x \geq 0$ which is more stable than that of *sigmoid* and *tanh*. Therefore, ReLU is less likely to suffer from *vanishing gradient problem*.

$$f(x) = \begin{cases} 0 \text{ for } x < 0 \\ x \text{ for } x \geq 0 \end{cases} \tag{2.28}$$

$$f'(x) = \begin{cases} 0 \text{ for } x < 0 \\ 1 \text{ for } x \geq 0 \end{cases} \tag{2.29}$$

2.1.5 Regularizations

One of the problems we usually encounter in machine learning is overfitting, especially when we train on a small dataset. Overfitting makes the trained model performs considerably worse on test set than it does on the training set. Basically, the model tends to only remember training examples and shows poor performance in generalization. In case training set remains unchanged, there are many strategies to overcome overfitting including regularization. Regularization discourages learning a more complex model by:

- Penalizing the weights of the model by adding the norm of the weights to the cost function. For example, use $J + \| \theta \|_2^2$ in stead of J.
- Using dropout (Hinton, N. Srivastava, et al. 2012).

To use dropout, we remove a portion of the activations at training time by randomly setting values of activations to 0. During the test time, however, we do not apply dropout, only scale the value of the activations properly. The reason why dropout works well is given by Hinton, N. Srivastava, et al. (2012), By dropping a portion of the activations, the network has to collect more evidence from other neurons rather than fitting to a particular pattern. The learned function is therefore more robust.

2.2 Convolutional Neural Networks

Covnets are a specialized kind of network for processing data that has a grid-like topology, for example 2D grid of image pixels. Convolutional neural networks

(CovNets) were first developed by LeCun et al. (1989) but were not popular due to hardware limitations. In the visual recognition challenge ILSVRC 2012[1], CovNets made a comeback by achieving a top-5 error rate of 16% while best methods based on standard vision features achieved 26% (Krizhevsky, Sutskever, and Hinton 2012). In 2015, ResNet (K. He et al. 2016) another CovNet architecture reduced this rate to 3.57%.

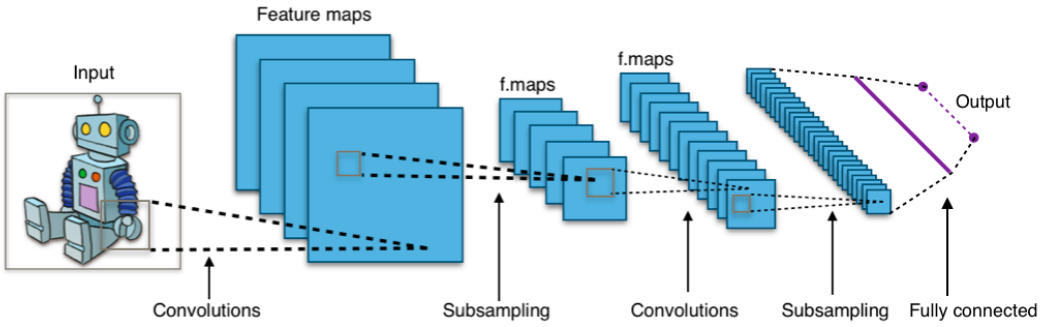


FIGURE 2.4: A typical convolutional neural network architecture[2]

There are many variations in CovNets architectures but a typical one are built from three main types of layers: *convolutional layer*, *pooling layer*, and *fully-connected layer*. Figure 2.4 shows an example of a typical CovNets architecture. This network contains a *convolutional layer* interacting with the input, followed by a *pooling layer*. The network goes on with another *convolutional layer* and a pooling layer before producing the final output through a *fully-connected layer*.
Convolutional Layer: This is the most important part of CovNets and its name is inspired by the "convolution operations" which are performed in this layer. Figure 2.5 illustrates an example of a convolution operation. There is a set of *filters* (or *kernels*) sliding along the image width and height. For example, a filter may be a tensor size 4x4x3 which is 4 pixels width, 4 pixels height and depth of 3 (3 channels red, blue and green of images). The value of the filter tensor is trainable. Every time we slide the filter, we compute the dot product between the filter tensor and the input area covered by the filter:

$$(\mathbf{I} * \mathbf{K})_{r,s} = \sum_{u=-h_1}^{h_1} \sum_{v=-h_2}^{h_2} \mathbf{K}_{u,v} \mathbf{I}_{r+u,s+v} \tag{2.30}$$

where K is a filter tensor, (r, s) is the current point on the input I. h_1, h_2 are size of the filter. The result of this process is a 2D feature map (see figure 2.4). As we apply a set of filter to the input, the output after this layer is a set of activation maps (see figure 2.4 and 2.5).

Pooling Layer (also called subsampling or downsampling): The purpose of this layer is to gradually reduce the size of the input and reduce the number of parameters which help to optimize the amount of computation and to control

[1] http://www.image-net.org/challenges/LSVRC/
[2] https://en.wikipedia.org/wiki/Convolutional_neural_network#/media/File:Typical_cnn.png
[3] http://cs231n.github.io/convolutional-networks/

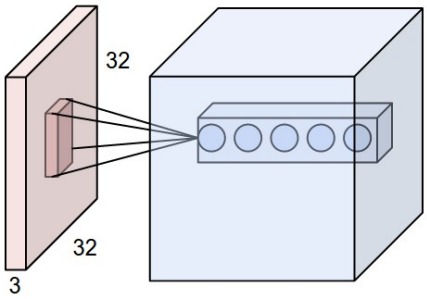


FIGURE 2.5: An example of a convolutional layer[3]. The input image having size 32x32 with 3 color channels. After applying a set of filter, each filter return a 2D feature map, we obtain a 3D output tensor.

overfitting. Pooling layers are usually used between successive convolutional layers (See figure 2.4). Various types of pooling layers can be implemented: *max, average, sum*. The most common form of is a pooling layer with filters having size 2x2. Every time we slide this filter, we take a sample of the region covered by the filter. For example, in case of max pooling and 2x2 filter, we take a max over 4 numbers. The depth dimension of the feature maps remains unchanged.

Fully-connected Layer: As discussed in previous section 2.1.2, hidden units in this layer are fully connected to all activations in the previous layer. That is also the only difference between this layer and the convolutional layers. In convolutional layer, each part of the outputs only connect with a region in the input and many parts of the output share parameters.

Zeiler and Fergus (2014) explained why Covnets have high performance on many computer vision tasks by visualizing feature maps of different layers of the fully-trained models and training models. First, they found out that each layer of the network learns different types of shapes and objects and these shapes and object are increasingly different at higher layers. Second, the networks are robust to feature variance. The network output is stable to translations and scalings.

VGGnet (Simonyan and Zisserman 2014) is one of the common Covnets architecture, the runner-up in ILSVRC 2014. This architecture shows us that the depth of the network is a crucial component for good performance. Their pre-trained models are widely used in computer vision research community as it is possible to use these models to extract features from images at different levels of abstraction.

2.3 Recurrent Neural Network

If Convnets are the dominant method in Computer Vision, Recurrent Neural Networks (RNN) play the similar role in Natural Language Processing. RNNs are a family of neural network architectures for dealing with sequential data. Given an ordered list of input vectors, at each point in time, RNNs process the

input vector with regards to the output of the previous state. We can represent RNNs as deterministic transitions from previous to current hidden states.

$$RNN : x_t, h_{t-1} \rightarrow h_t \tag{2.31}$$

Where x_t is the input at time t, h is the transition function. If we consider a vanilla RNN, h is given by

$$h_t = f(Wx_t + Uh_{t-1}); \quad f \in \{sigmoid, tanh\} \tag{2.32}$$

where W, U are trainable weight matrices.

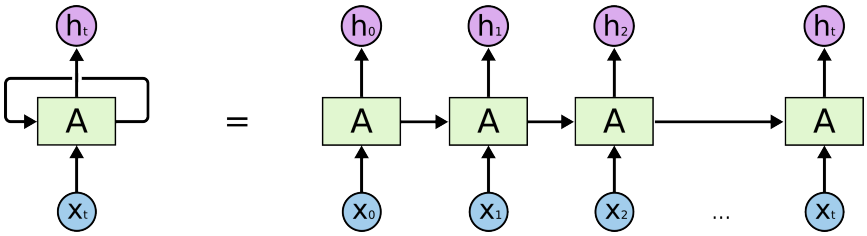


FIGURE 2.6: A recurrent neural network unrolled through time.

In comparison to feed-forward networks, RNNs take advantages of sharing parameters. For example, Figure 2.6 shows an unfolded form of RNN. At different points in time, the network may process different inputs but the parameters remain the same. As a result, the models have much less parameters, which allows the models to handle long sequences.

Backpropagation Through Time To train RNNs, we can use the Error Backpropagation discussed in 2.1.3 but on the model's unfolded form through time (Figure 2.6). In short, the Backpropagation Through Time (BPTT) first unfolds the network then calculates the errors at each time step. After rolling back the network, BPTT updates the weights with the accumulated error. This is where the *vanishing gradient problem* can happen as the weights can get close to zero after a long sequence of accumulation operations. Long short-term memory, discussed next section, is designed to solve this issue.

2.4 Long Short-Term Memory

Long short-term memory(LSTM), proposed by Hochreiter and Schmidhuber (1997), is one of the recurrent neural networks processing sequence data. For a long sequence, there is a well-known issue called vanishing gradient problem. LSTM addresses this issue by using few gate vectors and memory that "memorize" information. LSTM has *input, forget and output gates* helping it to decide to overwrite the memory cell, retrieve it, or keep it for the next time step. This allows LSTM to preserve long-range dependencies while processing long sequences. LSTM can be described using deterministic transitions from

previous to current hidden states:

$$LSTM : h_t^{l-1}, h_{t-1}^l, c_{t-1}^l \rightarrow h_t^l, c_t^l \tag{2.33}$$

Let $T_{n,m} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ be an affine transform ($Wx + b$ for some W and b)

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} sigm \\ sigm \\ sigm \\ tanh \end{pmatrix} T_{2n,4n} \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix} \tag{2.34}$$

$$c_t^l = f \odot c_{t-1}^l + i \odot g \tag{2.35}$$

$$h_t^l = o \odot tanh(c_t^l) \tag{2.36}$$

Here, i, f, c, o, g are respectively input, forget, memory, output and hidden state of LSTM. Let $\odot$ be element-wise multiplication and let h_o^t be an input word vector at timestep k, while $sigm, tanh, \odot$ are the logistic sigmoid, tanh activation and element-wise multiplication functions respectively.

Bidirectional LSTM (BiLSTM) (Graves and Schmidhuber 2005) is an extension of LSTM. As we see in equation 2.33, LSTM takes previous hidden states into account while processing current input. BiLSTM, on the other hand, read the input sequence from left to right and from right to left by employing two separate LSTMs. The output of BiLSTM at each point in time is the concatenation of the outputs by the two LSTMs.

2.5 Examples of applications in NLP

In NLP, as Goldberg (2016) pointed out, RNNs are the dominant architecture and their most successful type of RNN architecture LSTMs are responsible for many state-of-the-art sequence modeling results . There are various uses of RNN in NLP including *accepter*, *encoder*, *transducer* and *encoder-decoder*.

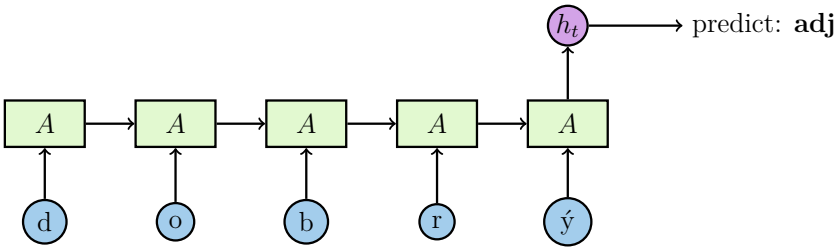


FIGURE 2.7: Example of Accepter RNNs

Accepter: First, RNNs can be used as *an accepter* where output of the model is decided on the final hidden state (Figure 2.7). For example, we could use an RNN to read characters of a word as an ordered sequence and use the last state

to predict the part-of-speech of that word (Ling et al. 2015), or use RNN to read words in a sentence and predict the polarity (ex. sentiment) of the sentence by the last hidden state.

Encoder: Again, we can make use of the last hidden state or concatenation of all hidden states and treat it as the encoding of the sequence. Unlike the *accepter*, decision is not made directly on this state but instead use it as additional information together with other signals. For example, Bowman, Angeli, et al. (2015) concatenate encoding of two sentences obtained by an LSTM and put it into a feed-forward network to classify the relationship of these two sentences. Figure 1.4 illustrates this architecture. In our proposed model, the use of RNN falls into this type.

Transducer: Third, we could use RNN as a transducer, giving an output for every input. An example of this architecture is a sequence tagger, where we use the hidden states to predict the tag at the corresponding positions. A CCG super-tagger by W. Xu, Auli, and Clark (2015) using this architecture produce state-of-the art results on CCG super-tagging. Another example of transduction setup is language models where previous words are used to predict a distribution over the next word. RNN-based language models by Tomas Mikolov et al. (2010), Sundermeyer, Schlüter, and Ney (2012), and Tomáš Mikolov (2012) outperform traditional methods.

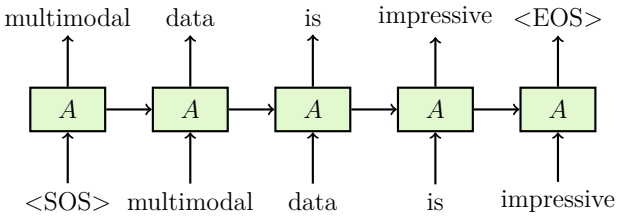


FIGURE 2.8: An example of RNN-based language model working as a transducer

Encoder - Decoder: This can be seen as a special case of *encoder* as, in fact, it employs one RNN to encode the sequence into a fixed-length vector. This vector is used as input to the other decoder RNN. In machine translation, for example, Sutskever, Vinyals, and Le (2014) employ an LSTM to encode a variable-length source sentence to a fixed length vector and employ another LSTM to decode the vector. The later LSTM works as a *transducer*.

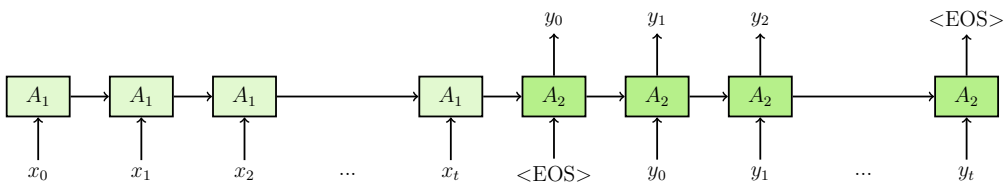


FIGURE 2.9: An encoder-decoder translating sequence "$x_0x_1x_2...x_t$" to sequence "$y_0y_1y_2...y_t$"

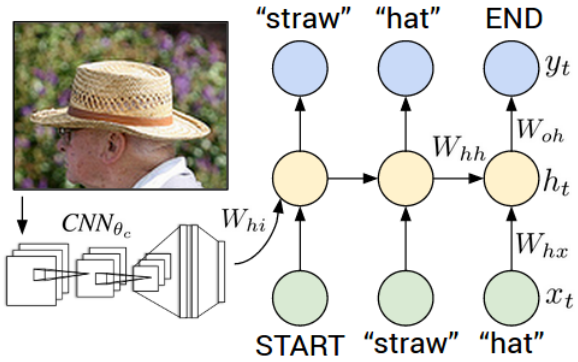


FIGURE 2.10: An example of image captioning system from Karpathy and Fei-Fei (2015)

Thanks to the development of neural networks, the fields of Computer Vision and Natural Language Processing are moving closer (Bernardi et al. 2016). In fact, *image captioning* and *visual question answering* have become key tasks in both communities. It is common that there is a CovNet to get the visual information and an RNN to process the textual data. Our proposed model has a similar architecture. Figure 2.10 demonstrates an example architecture to produce descriptions from images. The fixed-length vector extracted by the Covnet are used as the initial state for RNN. The generated descriptions are surprisingly realistic (Karpathy and Fei-Fei 2015).

We have now introduced all necessary information concerning how to build and train a neural network as well as how they process text and image. In the next chapter, we make use of the models and definitions given here to describe our proposed model.

Chapter 3

Methodologies

Following the success of neural networks in NLI and multimodal tasks, we opt to use neural networks as our approach. As we discussed in Section 1.3, there are two main types of architectures among neural networks approaches, namely encoders and pairwise models. In a sentence encoder architecture, the hypothesis and the premise are only compared after we get their encoded vector representations (Figure 1.4). In contrast, a pairwise model matches the two sentences in earlier stages, compares words to words and are capable of utilizing powerful attention mechanisms. In practice, pairwise models often achieve better performances than encoder models. However, as our ultimate goal is to effectively exploit both visual and linguistic information, we will examine both architectures in this work.

First, we start by giving details on a simple BiLSTM encoder and a variety of methods inspired by work on image captioning which address the question of how to incorporate the image (Section 3.1). Next, in Section 3.2, we describe the BiMPM framework, a pairwise model which achieves the state of the art performance on the SNLI dataset. Section 3.3 proposes an architecture grounding hypothesis and premise on to the image in the BiMPM framework.

In the next chapters, we will compare performances of the textual models to the multimodal models counterparts on a newly created multimodal NLI dataset.

3.1 BiLSTM Encoders

BiLSTMs were among the first few models proposed to solve the NLI problem and remain a strong baseline (Liu et al. 2016). The last RepEval sharedtask proved the efficiency of this model, showing the strength of BiLSTM combined with maxpooling (Nangia et al. 2017). In this model, the hypothesis sentence and the premise sentence are given to the same BiLSTM. The encoded vectors of the two sentences are then concatenated together with either their multiplication or summation which are fed in a fully connected layer to predict the relation. This model is relatively simple; however, it has been shown to be effective in not only NLI but also a wide range of tasks including image captioning. We adopt this model as our baseline encoder model.

Adapting this architecture in the multimodal NLI task requires us to answer few questions. First, whether we should incorporate the image after getting the sentence encoding or include it when the sentences are being processed by the recurrent networks? Second, if we include the image to every time step of the recurrent networks, should we pass a different representation of the image to each time step? Similar questions have been fairly well-studied in image captioning (refer to Tanti, Gatt, and Camilleri 2018 for an overview). In image captioning, the input consists of an image and its caption, which are encoded by a recurrent network and a convolution network respectively. Our encoder models are inspired by those work in image captioning. Precisely, we examine two main methods: one includes image features after the premise and the hypothesis are encoded (*merging* method), one lets the recurrent network exposed to the image while it is encoding the text (injection).

3.1.1 "Blind" BiLSTM

The "blind" BiLSTM is a textual encoder having one BiLSTM to encode both the hypothesis and the premise:

$$\begin{aligned} h_t^p &= BiLSTM(p_t, h_{t-1}^p) \\ h_t^q &= BiLSTM(q_t, h_{t-1}^q) \end{aligned} \tag{3.1}$$

p_t, q_t are the t^{th} word (token) of the premise and hypothesis respectively. BiLSTM is explained in section 2.4. h_t denotes the t^{th} hidden state of the BiLSTM.

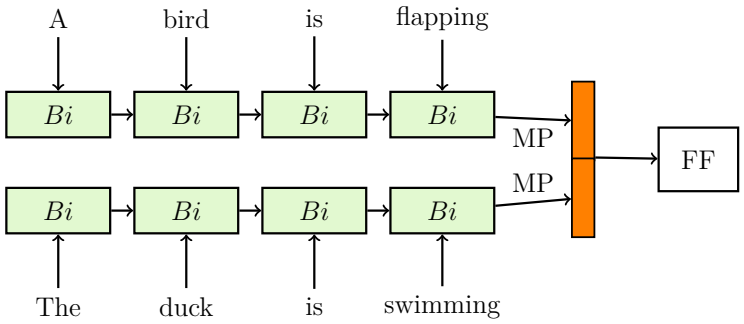


FIGURE 3.1: "Blind" BiLSTM: Representation of the premise and the hypothesis are concatenated with their point-wise multiplication and fed into a feed forward network. Legend: FF: Feed forward layer, MP: max pooling.

Representations of the premise and the hypothesis are obtained by applying maxpooling over the output of the BiLSTM. These representations and their multiplication are then fed into a two-layer feed-forward network (denoted *FF*),

followed by a softmax layer to predict the probability distribution of the class.

$$\begin{aligned} h^p &= Maxpooling([h_1^p, h_2^p, ..., h_m^p]) \\ h^q &= Maxpooling([h_1^q, h_2^q, ..., h_n^q]) \\ h &= CONCAT(h^p, h^q, h^p * h^q) \\ Pr(y|p, q, I) &= softmax(FF(h)) \end{aligned} \tag{3.2}$$

Where h^p and h^q are encoded vectors of the premise and the hypothesis respectively.

3.1.2 Merging

Figure 3.2 shows an overview of the *merging* architecture. The image is fed into a convolution neural network to get a fixed-length vector which serves as the image features. The vector is then shrunk to a desired size by a feed-forward layer:

$$h_I = \mathbf{W_I} Covnet(I) + b_I \tag{3.3}$$

h_I is the image vector, *Covnet* represents the convolutional neural network. W_m, b_m are weights and bias of the feed-forward layer respectively.

The premise and the hypothesis are encoded similarly as in the "blind" BiLSTM model. These representations are concatenated with their element-wise multiplication or addition with the image features. The combination is fed into a fed forward layer which followed by a softmax layer to predict the probability distribution of the class.

$$\begin{aligned} h &= CONCAT(h^p \odot h_I, h^q \odot h_I) \\ Pr(y|p, q, I) &= softmax(FF(h)) \end{aligned} \tag{3.4}$$

3.1.3 Injection

The injection methods introduce the image to the network when the premise and the hypothesis are being processed by the BiLSTMs. Following the success in image captioning, we examine three possible injection methods, namely *init-inject*, *par-inject*, *Attentional par-inject*.

Init-inject: Figure 3.3 illustrates the injection by initialization model. The image, the premise, and the hypothesis are processed similarly as in the *merging*

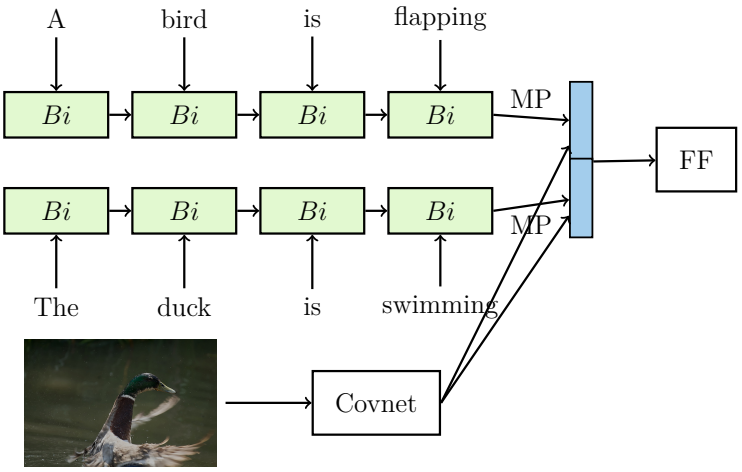


FIGURE 3.2: Merging: Image features are blended with representation of the premise and hypothesis before being fed to a fully connected layer. Legend: FF: Feed forward layer, Covnet: Convolutional neural subnetwork, MP: Maxpooling.

method (Equation 3.3, 3.1), except that the image features are now used to initialize the first hidden state of the recurrent network:

$$h_0^p = h_0^q = h_I \tag{3.5}$$

By default, the hidden state of the recurrent network is initialized randomly. The Equation 3.5 also implies that Equation 3.3 shrink the image features to the same size as the size of BiLSTM hidden states. The class decision is then made by only the representation of the premise and the hypothesis:

$$\begin{aligned} h &= CONCAT(h_m^p, h_n^q) \\ Pr(y|p,q,I) &= softmax(FF(h)) \end{aligned} \tag{3.6}$$

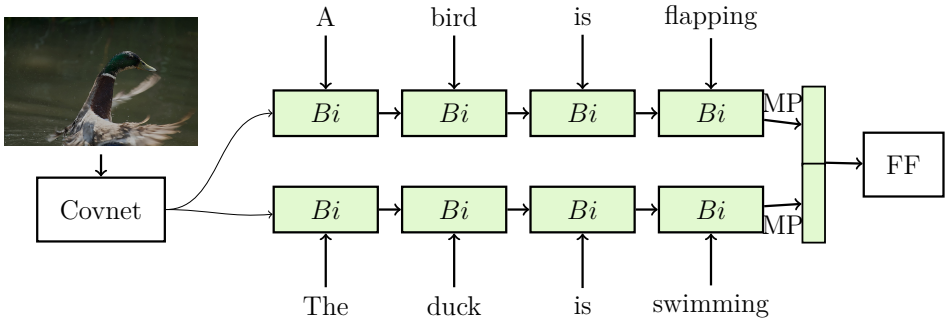


FIGURE 3.3: Init-inject: The image features are used to init the BiLSTM cells of the recurrent neural network.

Par-inject: Figure 3.4 shows an overview of the par-inject model. In this model, the image features are injected to every step of the recurrent network. Input to BiLSTM in this model consists of a word, the image, and the previous

hidden state. In order for BiLSTM to take three inputs, we modify the standard BiLSTM (Section 2.4, Equation 2.34). We keep all the parameters but change the input:

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} sigm \\ sigm \\ sigm \\ tanh \end{pmatrix} T_{3n,4n} \begin{pmatrix} x_t^1 \\ x_t^2 \\ h_{t-1} \end{pmatrix} \tag{3.7}$$

Here, x_t^1, x_t^2 are two separated inputs at time t which take values of image features and a word token:

$$\begin{aligned} h_t^p &= BiLSTM(h_I, p_t, h_{t-1}^p) \\ h_t^q &= BiLSTM(h_I, q_t, h_{t-1}^q) \end{aligned} \tag{3.8}$$

The class decision is then made by only the representation of the premise and the hypothesis:

$$\begin{aligned} h &= CONCAT(h_m^p, h_n^q) \\ Pr(y|p,q,I) &= softmax(FF(h)) \end{aligned} \tag{3.9}$$

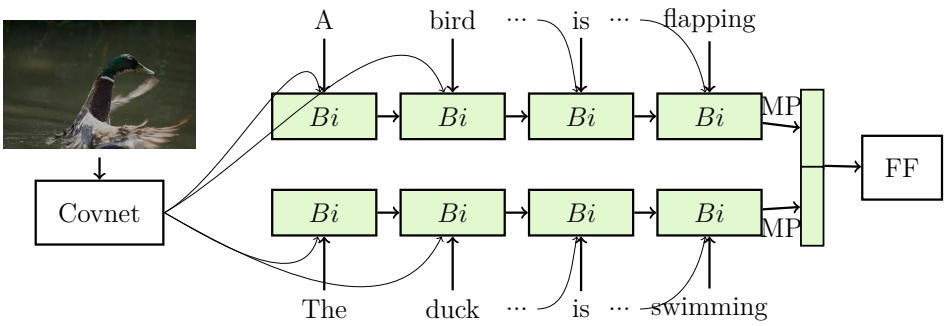


FIGURE 3.4: Par-inject: At every step of encoding the hypothesis and the premise, the recurrent network receive a word and the image as an input.

Attentional par-inject: This method is similar to the *par-inject* method because the image features are injected to every step of the recurrent network. However, this method passes different representation of the same image to different steps of the recurrent process. The modified image representations come from attention mechanisms. In this work, we define a mechanism that computes a dynamic representation of the image depending on the last hidden state of the BiLSTM cell. This mechanism is, to some extent, similar to the *soft attention* proposed by Bahdanau, Cho, and Bengio (2014) in machine translation and K. Xu et al. (2015) in image captioning.

Instead of extracting one image vector after the feed-forward layer of the Covnet, we extract vectors from lower layer which results in a set of vector. Each vector

corresponds to different feature of the image:

$$[a_1, a_2..., a_L] = \mathbf{W_I} Covnet(I) + b_I \tag{3.10}$$

Again, W_I, b_I are used to shrink the features to the desired size.

The BiLSTM is also modified to take three separated inputs (Equation 3.7). The hidden state of the BiLSTM is computed as follows:

$$\begin{aligned} h_t^p &= BiLSTM(\hat{z}_t, p_t, h_{t-1}^p) \\ h_t^q &= BiLSTM(\hat{z}_t, q_t, h_{t-1}^q) \end{aligned} \tag{3.11}$$

Where $\hat{z}_t$ is the dynamic image representation at time step t. We denote the attention mechanism ϕ as the function calculating $\hat{z}_t$ from the set of image vectors $a_i, i = 1, ..., L$. For each image feature a_i, the attention mechanism generates a positive weight α_i which can be interpreted as how much the network should focus on feature a_i, given the last hidden state. The weight α_i is computed by an attention model f_{att} for which we use a feed-forward layer conditioned on the last hidden state h_{i-1}.

$$e_{ti} = f_{att}(a_i, h_{i-1}) \tag{3.12}$$

$$\alpha_{ti} = \frac{\exp(e_{ti})}{\sum_{k=1}^{L} \exp(e_{tk})} \tag{3.13}$$

$$\hat{z}_t = \phi(\{a_i\}, \{\alpha_i\}) = \sum_{k}^{L} \alpha_{ti} a_i \tag{3.14}$$

We examine two options for the function f_{att}, where a_i and h_{i-1} could either interact by addition (Bahdanau, Cho, and Bengio 2014) or multiplication (Luong, Pham, and C. D. Manning 2015):

$$f_{att}(a_i, h_{i-1}) = \begin{cases} v_{att}^{\intercal}(\mathbf{W_{att}} a_i + \mathbf{U_{att}} h_{i-1}) \\ a_i^{\intercal} \mathbf{W_{att}} h_{i-1} \end{cases} \tag{3.15}$$

$v_{att}, \mathbf{W_{att}}, \mathbf{U_{att}}$ are learnable parameters.

The class decision is again made by only the representation of the premise and the hypothesis:

$$\begin{aligned} h &= CONCAT(h_m^p, h_n^q) \\ Pr(y|p, q, I) &= softmax(FF(h)) \end{aligned} \tag{3.16}$$

3.2 Bilateral Multi-Perspective Matching

Wang, Hamza, and Florian (2017) propose the Bilateral Multi-Perspective Matching framework (BiMPM) to solve various tasks in natural language matching including NLI. Figure 3.5 shows the overview of the architecture. Their ensemble models achieve the best result [1] on the SNLI dataset at the time of this thesis.

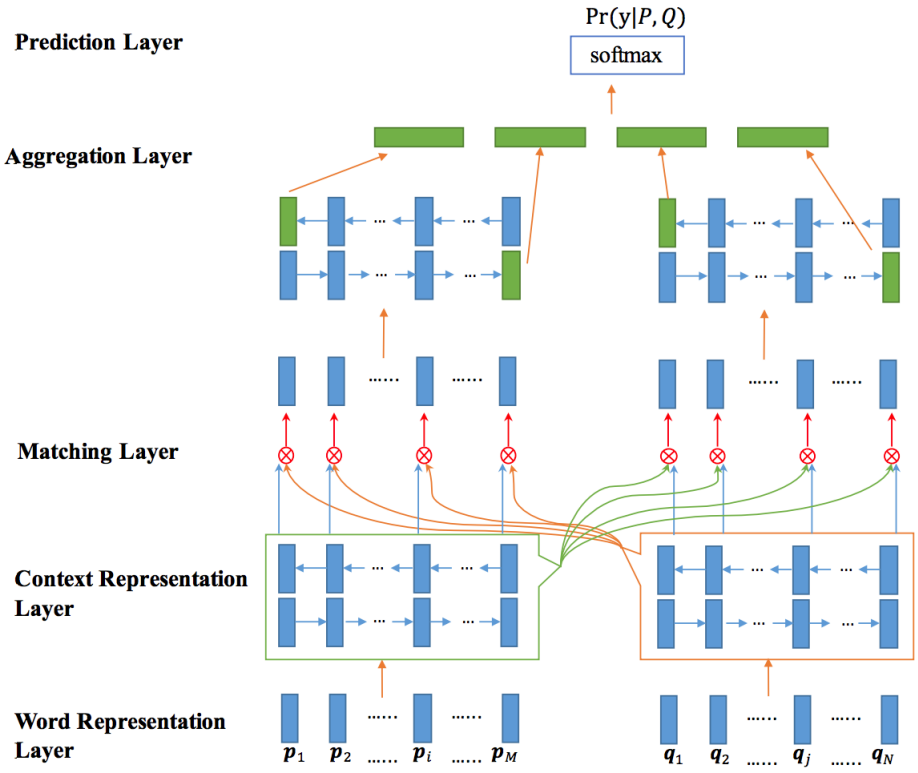


FIGURE 3.5: Overview of BiMPM architecture (Wang, Hamza, and Florian 2017).

Given a premise (P), a hypothesis (Q), the model predicts whether the hypothesis entails the premise by calculating the probability distribution of the output class $P(y|P, Q)$. The model receives P, Q as inputs and forwards them through five layers to calculate the probability distribution. In the first layer, words are turned into vectors. In the second layer, word vectors are incorporated with their context information and matched with other sentences in the next layers, using different strategies. The aggregation layer turns matching result vectors into fixed-length vectors which are concatenated and fed into a softmax to get the probability distribution. This 5-layers architecture is adopted from Wang, Hamza, and Florian (2017).

Embedding layer. This layer not only turns words into vectors but also preserves their semantic relations. Specifically, the framework makes use of the pre-trained words embeddings from Pennington, Socher, and C. Manning

[1] Results on SNLI dataset are listed here: https://nlp.stanford.edu/projects/snli/

(2014). The motivation behind these kind of embedding is the distributional hypothesis by Harris (1954) which states that words occurring in similar contexts have similar meaning. In fact, word embeddings can capture many linguistic regularities, for example "*paris*" - "*france*" ≈ "*rome*" - "*italy*". The word embedding plays a similar role as the knowledge base in systems using a logic-based approach.

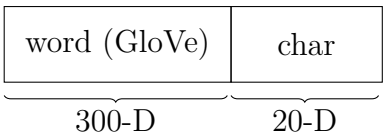


FIGURE 3.6: Word embedding components in BiMPM

To deal with out-of-vocabulary words, the word embedding is supported by the character embedding. The character embedding is obtained by running an LSTM over the sequences of characters and getting the last hidden state. The vector of each character is initialized randomly and trained together with the whole framework. Figure 3.6 shows the components of the embedding. After the first layer, the premise and hypothesis are represented as arrays of word embedding vectors.

$$\begin{aligned} \textit{Premise} \quad P: \quad & [p_1, ..., p_M], \quad p_i \in \mathbb{R}^D \\ \textit{Hypothesis} \quad Q: \quad & [q_1, ..., q_N], \quad q_i \in \mathbb{R}^D \end{aligned} \tag{3.17}$$

Context layer. The goal of this layer is to incorporate each time step of the premise and hypothesis with its contextual information. To this end, a bi-directional LSTM (BiLSTM) is employed. It receives a sequence of embedding vectors as input and outputs corresponding number of output contextual vectors in each direction. The same BiLSTM is used to encode both premise P and hypothesis Q. The layer is represented by equations 3.18 and 3.19 where p_i, q_i are inputs at time i to the $LSTM$ and h_i is the hidden state at time i.

$$\begin{aligned} \overrightarrow{h_i^p} &= \overrightarrow{LSTM}(p_i, \overrightarrow{h_{i-1}^p}) \\ \overleftarrow{h_i^p} &= \overleftarrow{LSTM}(p_i, \overleftarrow{h_{i-1}^p}) \end{aligned} \tag{3.18}$$

$$\begin{aligned} \overrightarrow{h_i^q} &= \overrightarrow{LSTM}(q_i, \overrightarrow{h_{i-1}^q}) \\ \overleftarrow{h_i^q} &= \overleftarrow{LSTM}(q_i, \overleftarrow{h_{i-1}^q}) \end{aligned} \tag{3.19}$$

Matching layer. This is the most crucial part of the BiMPM framework where each contextual embedding (time-step) of a sentence is matched against all contextual embedding of the other sentence. As shown in Figure 3.5, the hypothesis and premise are compared in two directions: compare each time-step of the hypothesis (Q) to all time-steps of the premise (P) and compare

each time-step of the premise to all time-steps of the hypothesis. The textual matching operation is denoted by $\otimes$ in Figure 3.5 and is defined in the section 3.2.1 in contrast to the multimodal matching operation defined in section 3.3.2 which which we shall turn to in due course.

Aggregation Layer. The purpose of this layer is to obtain a fixed-length vector from the sequence of matching vectors. Another BiLSTM is used to encode those matching sequences separately. The fixed-length vector is then constructed by concatenating the last time step of the outputs from BiLSTM models.

Prediction Layer. This is the last layer in the framework where the probability distribution is calculated. The framework employs a two-layer feed-forward network to transform the fixed-length vector and apply a softmax function to the output vector to obtain a proper probability distribution which indicates probabilities of the label y to be *entailment*, *neutral* or *contradiction*.

3.2.1 Textual Matching Operations

This is similar to cosine similarity except that the result of these operations are vectors instead of scalars. We keep it as it is from BiMPM (Wang, Hamza, and Florian 2017).

$$m = f_m(v_1, v_2; \mathbf{W}) \tag{3.20}$$

Each vector input is replicated l times and multiplied with different weight vectors W_k before obtaining a cosine similarity.

$$m_k = cosine(W_k \circ v_1, W_k \circ v_2) \tag{3.21}$$

where v_1 and v_2 are d-dimensional vectors. $\mathbf{W}$ is a $l \times d$ trainable weight matrix. m is a l-dimensional matching result vector $m = [m_1, ..., m_l]$. The idea of this operation is very similar to the idea from self-attentive sentence embedding (Lin et al. 2017) where each sentence has various meaning channels. Each channel encodes a different part of the sentence.

3.2.2 Matching Strategies

As we discussed, in the matching layer, each time-step of one sentence is matched with the other sentence in various strategies. Specifically, the strategy can be *fully-matching*, *maxpooling-matching*, *attentive-matching* and *max-attentive-matching*. In Wang, Hamza, and Florian (2017)'s work, the best-performing system uses all these strategies, therefore, in the experiments we will keep the same setting.

Full-Matching: This strategy is shown in Figure 3.7 (a) and Equation 3.22 where each forward (backward) time-step $\overrightarrow{h_i^p}$ (or $\overleftarrow{h_i^p}$) of P is matched with the last forward (backward) time-step $\overrightarrow{h_N^q}$ (or $\overleftarrow{h_1^q}$) of Q. The matching operation f_m is presented in Section 3.2.1 above. Consequently, the outputs of this matching

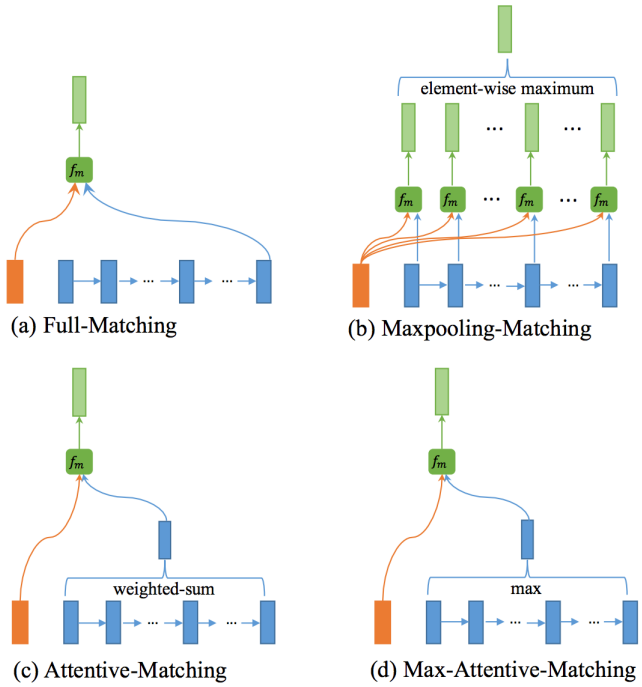


FIGURE 3.7: Diagram for 4 matching strategies (Wang, Hamza, and Florian 2017)

strategy would be a sequence of vectors having the same length as P if we match P against Q in one direction. The directions refers to the components of BiLSTMs, namely forward and backward. Later strategies describe bellow also output similar sequences, which allows us to easily concatenate all the outputs for subsequent processing steps.

$$\begin{aligned} \overrightarrow{m_i^{full}} &= f_m(\overrightarrow{h_i^p}, \overrightarrow{h_N^q}; \boldsymbol{W}^1) \\ \overleftarrow{m_i^{full}} &= f_m(\overleftarrow{h_i^p}, \overleftarrow{h_1^q}; \boldsymbol{W}^2) \end{aligned} \tag{3.22}$$

Maxpooling-Matching: Figure 3.7(b) and Equation 3.23 illustrate this strategy where each time-step of the premise is matched against each time-step of the hypothesis; then a maxpooling operation is applied to retain only the maximum value of each dimension. Again, as we explained above, applying each strategy on one direction will output a sequence of matching vectors.

$$\begin{aligned} \overrightarrow{m_i^{max}} &= \max_{j \in 1 \ldots N} f_m(\overrightarrow{h_i^p}, \overrightarrow{h_j^q}; \boldsymbol{W}^3) \\ \overleftarrow{m_i^{max}} &= \max_{j \in 1 \ldots N} f_m(\overleftarrow{h_i^p}, \overleftarrow{h_j^q}; \boldsymbol{W}^4) \end{aligned} \tag{3.23}$$

Attentive-Matching: Figure 3.7 (c) gives the diagram of this matching strategy. This strategy works in similar fashion to the normal attention mechanism (Bahdanau, Cho, and Bengio 2014; Luong, Pham, and C. D. Manning 2015). Equation 3.24 shows the weight functions. We first calculate the weights for each time-step of hypothesis Q by calculating the cosine similarities between

each forward (or backward) time-step $\overrightarrow{h_i^p}$ (or $\overleftarrow{h_i^p}$) and every forward (or backward) time-step of the other sentence $\overrightarrow{h_j^q}$ (or $\overleftarrow{h_j^q}$). In another word, given a time step of P how much attention we should pay for each time-step of Q:

$$\begin{aligned} \overrightarrow{\alpha_{ij}} &= cosine(\overrightarrow{h_i^p}, \overrightarrow{h_j^q}) \qquad j = 1, ..., N \\ \overleftarrow{\alpha_{ij}} &= cosine(\overleftarrow{h_i^p}, \overleftarrow{h_j^q}) \qquad j = 1, ..., N \end{aligned} \tag{3.24}$$

After getting the weights, we can now calculate an attentive representation vector of the hypothesis Q. Equation 3.25 shows how the attentive vector is computed in both backward and forward direction:

$$\begin{aligned} \overrightarrow{h_i}^{mean} &= \frac{\sum_{j=1}^{N} \overrightarrow{\alpha_{ij}} \cdot \overrightarrow{h}_j^q}{\sum_{j=1}^{N} \overrightarrow{\alpha_{ij}}} \\ \overleftarrow{h_i}^{mean} &= \frac{\sum_{j=1}^{N} \overleftarrow{\alpha_{ij}} \cdot \overleftarrow{h}_j^q}{\sum_{j=1}^{N} \overleftarrow{\alpha_{ij}}} \end{aligned} \tag{3.25}$$

Finally we match each forward (or backward) time-step with its attentive vector (Equation 3.26):

$$\begin{aligned} \overrightarrow{m_i^{att}} &= f_m(\overrightarrow{h_i^p}, \overrightarrow{h}_N^{mean}; \boldsymbol{W}^5) \\ \overleftarrow{m_i^{att}} &= f_m(\overleftarrow{h_i^p}, \overleftarrow{h}_1^{mean}; \boldsymbol{W}^6) \end{aligned} \tag{3.26}$$

Max-Attentive-Matching: This strategy is shown in Figure 3.7 (d). This is similar to the Attentive-Matching strategy except instead of taking the weighted sum, we take only the time-step having the highest weight.

3.3 Multimodal Bilateral Multi-perspective Matching

In light of Wang, Hamza, and Florian's findings, we propose the Multimodal Bilateral Multi-Perspective Matching framework (M-BiMPM) to solve our proposed task, namely multimodal NLI. Figure 3.8 shows the overview architecture of the framework. As has been noted, we keep the BiMPM architecture and factor in the image. For this purpose, we define a multimodal matching operation to match the image with the text. We represent image as a sequence of image features which makes it possible to match image and text in the same manner as textual matching.

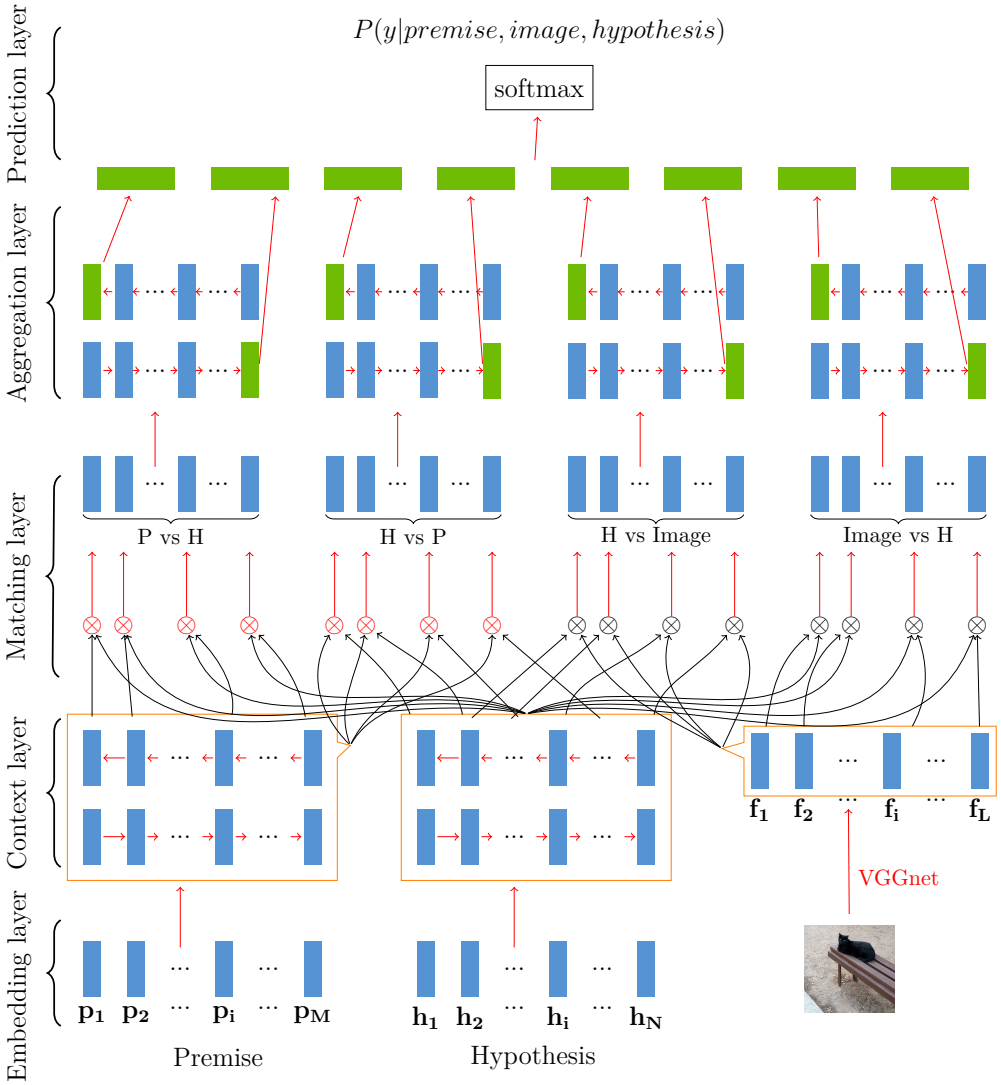


FIGURE 3.8: Architecture of Multimodal Bilateral Multi-perspective Matching framework. Due to space limitation, the diagram only shows the hypothesis matched with the image.

3.3.1 Framework overview

Figure 3.8 shows the overview of the framework. Specifically, given an image I, a premise P, a hypothesis Q, the framework estimates the probability distribution $P(y|I, P, Q)$. To this end, the text and image are forwarded through the same five layers as in the BiMPM framework. The difference is in the *embedding layer* and the *matching layer* where we handle the image.

Embedding layer. Inspired by work in image captioning (Karpathy and Fei-Fei 2015; K. Xu et al. 2015; Fang et al. 2015), we represent image as a set of feature vectors. We utilize the VGG16 model from Simonyan and Zisserman (2014) and extract features before the fully-connected layer which results in a set of feature vectors (Equation 3.27). It is believed that this allows the framework to focus on particular parts of the image in the matching process.

$$Image \quad I: \qquad \{f_1, ..., f_L\}, \qquad f_i \in \mathbb{R}^e \tag{3.27}$$

Matching layer. Here, we perform the textual and multimodal matching operations. Before this layer, text is turned into sequences of contextual vectors and image is turned into a set of image features. Initially, the premise is matched against the hypothesis and vice versa as defined in the BiMPM framework. In the same way, the image, the hypothesis and the premise are in turn matched with each other. In Figure 3.8, the textual and multimodal matching operations are denoted by $\otimes$ and $\otimes$ respectively. We use all the matching strategies described in BiMPM framework.

3.3.2 Multimodal Multi-perspective Operation

This is the crucial part in the model where we match the textual vector to visual vectors and vice versa. In other words, we need to define a similarity function giving high scores to similar input concepts, regardless their modality. For example, a vector of the word *cat* should be more similar to a *visual vector of a cat* than a *visual vector of a dog*. Second, matching vectors across modality is not trivial. These vectors not only have different dimensions but also belong to different spaces.

In this thesis, we project these vectors into a mutual space using affine transformation. This operation is simple and fast to compute. Weight matrices and bias are learned together with the whole model during the training process. In fact, experiments in Chapter 4 show that affine transformation performs very well. Investigation on more effective methods is a promising research direction. After projection, we calculate their similarity by the multi-perspective matching defined in Section 3.2.1.

$$v_i = \mathbf{W}_t f_i + b_t; \quad f_i \in \mathbb{R}^e; \mathbf{W}_t \in \mathbb{R}^{e \times d}; b_t, v_i \in \mathbb{R}^d \tag{3.28}$$

Where $\mathbf{W}_t$, b_t, f_i, and v_i are the weight matrix, the bias, the input features and output features respectively.

Similar to the textual matching operation, output of this operation is a l-dimension vector. l is the number of perspectives.

$$m = f_m(v_t, v_i; \mathbf{W}; \mathbf{U}) \tag{3.29}$$

where v_t and v_i are textual and visual vectors. $\mathbf{W} \in \mathbb{R}^{l \times d}$ and $\mathbf{U} \in \mathbb{R}^{l \times d}$ are weight matrices for textual and visual spaces. m is a l-dimensional matching result vector $m = [m_1, ..., m_l]$. Each element in m is calculated by

$$m_k = cosine(W_k \circ v_t, U_k \circ v_i) \tag{3.30}$$

3.4 Conclusion

In this chapter, we have discussed two approaches in NLI and how we adapt them to Multimodal NLI. First, we introduce the *encoder* approach together with three possible methods to include the image features. Second, we choose BiMPM to represent pairwise models. This framework implements a strong matching operation together with elegant attention strategies. We propose to extend this framework, augmenting it with visual information. This integration requires a cross-modality matching between textual and visual vectors. In this work, we use affine transformation to convert both modalities into a common space before matching them.

In the next chapter, we shall compare the performance of the multimodal to their textual model on two multimodal NLI dataset.

Chapter 4

Experiment results

In this chapter, we carry out several experiments to find out the advantages of multimodal data over the textual data. In section 4.1, we augment the large SNLI dataset with Flickr images to create the multimodal dataset. The first experiment compares different fusing modalities methods. Experiments 2 sheds some light on the question of when the image should be introduced to the model. These two experiments are carried out on encoder models while Experiment 3 will be performed on a pairwise model. In the fourth experiment, we test our model against the newly released subset of SNLI which has been shown to be better in quality. At the end of the chapter, we carry out some analysis to examine the effectiveness of the grounding process.

4.1 Dataset

4.1.1 SNLI and SNLI$_{hard}$

The Stanford Natural Language Inference (SNLI) Corpus was created by Bowman, Angeli, et al. (2015) for the purpose of NLI benchmarking. Modern methods on NLI, or NLP in general are data-intensive and not domain-restricted. However, the currently available datasets for NLI are relatively small and are not suitable for those methods. For example, the most recent data set, the Sentences Involving Compositional Knowledge (SICK) from the SemEval 2014 task (Marelli et al. 2014) contains 10,000 sentence pairs which are augmented from 1500 original pairs. Meanwhile, SNLI contains 570,152 sentence pairs generated by 2500 annotators (Bowman, Angeli, et al. 2015).

SNLI has been manually created by employing crowdsourcing. The premises are image captions borrowed from Flickr30k (Young et al. 2014). This Flickr30k dataset contains 160k captions and has also been created by crowdsourcing. Given the premise, workers from Amazon Mechanical Turk are asked to provide *true*, *neutral* and *false* descriptions to the photo. The workers are encouraged to produce novel sentences without sentence length limitation.

SNLI solved the problem of indeterminacies of event and entity coreference. This problem reduces the agreement rate on determining the correct semantic labels. For example, consider the two sentences *cheerleaders are on the field*

cheering and *some people are cheering on a field*. If we assume that *"cheerleaders"* and *"some people"* refer to the same people then the pair will be labeled as entailment, otherwise, it could be neutral. Likewise, we also have to choose whether we should make an assumption about event coreference. SNLI solved this issue by grounding the premise and hypothesis sentences in a specific scene. In other words, the objects or the entities mentioned in the text are supposed to refer to objects or entities in the photo. As a result, 58% of cases show a unanimous consensus from all five annotators and 98% of cases see a three annotator consensus (Bowman, Angeli, et al. 2015).

The dataset is balanced with respect to the three labels. The test set and development set contain 10k instances each. In addition, each Flickr caption appears in only one of the three sets, and all the instances in the development and test sets have been validated.

Despite the careful annotation process, it has been shown recently that the SNLI dataset contains annotation patterns on which a trivial classifier can achieve a high accuracy (67%) by only looking at the hypothesis (Gururangan et al. 2018). A closer investigation by the authors suggests that crowd workers adopt heuristics in order to generate hypotheses faster. For example, the entailment class tends to contain gender-neutral references to people, negation is highly correlated with contradiction and purpose clauses are a sign of neutral hypotheses. These authors also created a subset of SNLI test set on which the simple classifier fails (hereafter SNLI$_{hard}$). This subset contains 3261 instances.

4.1.2 M-SNLI and M-SNLI$_{hard}$

Our multimodal version of the SNLI dataset, M-SNLI[1], was built by matching each sentence pair to the corresponding image in the Flickr30k dataset. A small subset of the SNLI dataset where the caption is borrowed from VisualGenome cannot be match with any image, therefore our final dataset is slightly smaller than SNLI. The main statistics of the splits of the dataset are reported in Table 4.1 together with statistics for the visual counterpart of SNLI$_{hard}$, namely M-SNLI$_{hard}$. As shown in this table, the data is distributed evenly among all the classes. Meanwhile, figure 4.1 shows that the length of premise sentences vary considerably and hypothesis sentences tend to be short. Similar figures and statistics for SNLI are reported in Bowman, Angeli, et al. (2015). Finally, Table 4.2 shows some examples taken from our final training set.

Adding the image gives us more information but occasionally could make the gold standard label of the neutral instances questionable at the same time. Consider the last example in Table 4.2, *A big dog catches a ball on his nose* and *A big dog is sitting down while trying to catch a ball.* Without the image, one would classify this sentence pair as neutral but it is clearly contradictory if we take the corresponding image into account. This is because information about the posture of the dog is present in the image but not in the premise. This is also an example of how images can convey more than what is in the text.

[1]Submitted to http://lindat.mff.cuni.cz

Dataset	Entailment	Neutral	Contradiction	Total
Train	182,167	181,515	181,938	545,620
Test	3,368	3,219	3,237	9824
Dev	3,329	3,235	3,278	9842
M-SNLI$_{hard}$ (test)	1,058	1,135	1,068	3,261

TABLE 4.1: Data distribution on the obtained dataset

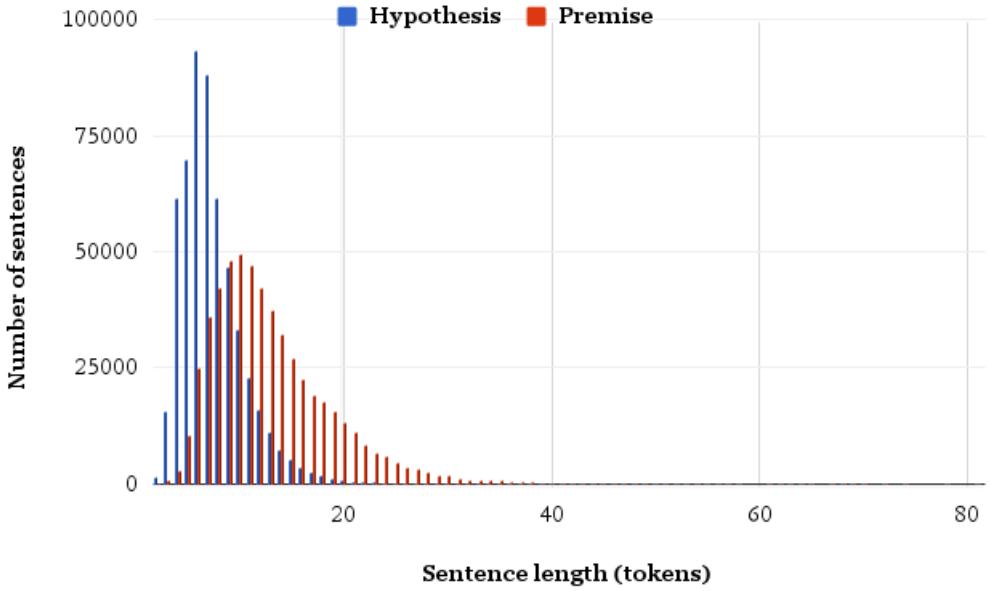


FIGURE 4.1: Distribution of sentence lengths for premise and hypothesis sentences

4.2 Experiments

In this section, we describe how we test our models proposed in Chapter3, the results of our experiments and several analyses.

All models have been implemented using Python 2.7 and Tensorflow 1.1. We train our models on an Ubuntu 14.04 machine, it has 60GB RAM and 24 cores of Intel(R) Xeon(R) CPU E5-2630 v2 @ 2.60GHz. Our models were trained in 10 iterations and the best performing models on the development set are picked.

4.2.1 Experiment 1: Fusing modalities

In this experiment, we examine different ways of combining the representations of the premise and hypothesis with the representation of the image. We choose the *merging* BiLSTM encoder (Section 3.1.2) to be the fixed architecture in this experiment. We expect to understand how much each modality contributes to the joint models and find the better method of fusing two modalities.

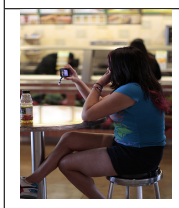
Flickr30k id	Premise	Label	Hypothesis
	A woman sitting at a table , taking a picture .	entailment	The woman is sitting down
	People sit and relax next to a pool in a plaza .	neutral	The pool is in a hotel , and these are guests
	A man parasails in the choppy water .	contradiction	The man parasailed in the calm water
	Busy ChinaTown street corner where people are walking past an open front store .	entailment	There are people in ChinaTown
	A big dog catches a ball on his nose	neutral	A big dog is sitting down while trying to catch a ball.

TABLE 4.2: Example pairs taken from the training set

Table 4.3 shows list of hyperparameters we used for Experiment 1 and Experiment2. Size of the final image vector after being resized is 200 which equals to size of the hidden states of BiLSTMs encoding the text. We examine three possibilities of activation functions. For regularization, we use dropout with keep probability rate 0.6 for feed forward layers and $L2$. We train models on SNLI and M-SNLI dataset.

Type	Hyperparameter	Options
Model	Image feature size	200
	Activation functions	ReLU, tanh, sigmoid
Training	Cost function	cross entropy
	Learning rate	0.001
	Optimizer	Adam
	LSTM dimension	100
	Concatenation size	400
	Regularization	L2, Dropout
	Iterations	10

TABLE 4.3: Hyperparameters for Experiment 1 and Experiment 2.

Results

System	Accuracy			
	Multiplication		Addition	
	Train	Test	Train	Test
"Blind" BiLSTM	86.32	81.49	_	_
Merging + Hypo ⊗ Image	87.72	78.24	76.38	71.73
Merging + Prem ⊗ Image	84.30	77.08	76.59	70.68
Merging + Hypo ⊗ Image + Prem ⊗ Image	85.37	80.07	76.28	72.90

TABLE 4.4: Results for experiments on fusing modalities on the M-SNLI dataset. *Hypo* stands for *hypothesis*, *prem* stands for *premise*.

Table 4.4 shows the results of four systems. "Blind" BiLSTM refers to the textual BiLSTM encoder, in which representations of the hypothesis and the premise are concatenated and fed into a feed forward layer. *Hypo* ⊗ *Image* refers to the merging system where only the hypothesis is grounded on the image. More precisely, input to the final feed forward layer consists of the premise representation, the hypothesis representation and the fusion representation between the hypothesis and the image. This fusion representation can either be generated by multiplication or addition. Similarly, *Prem* ⊗ *Image* grounds only the premise to the image and *Hypo* ⊗ *Image* + *Prem* ⊗ *Image* grounds both the sentences to the image. The final representation concatenations (for example the darker color bar in Figure 3.3) before fed into the feed forward layers in all systems have the same size (concatenation size).

Results for the "blind" model are put into the Multiplication column for ease of illustration; however, this model does not contain any grounding operation. From the table, we could see that introducing the image to encoder models hurts the performance in all cases. Multimodal models perform the worst when we use the addition grounding operation and show a decrease around 7% compared to the multiplication counterparts. Interestingly, for this encoder model, matching the image to the hypothesis gives a better performance than matching it to the premise and it is better to ground both the hypothesis and the premise to the image than grounding to each of them.

We also noticed the benefit of using *ReLU* over *tanh* and *sigmoid*, in both training time and accuracy. Training for the settings *Prem* ⊗ *Image* does not converge after 10 iterations when using *tanh* and *sigmoid*, giving accuracies around 33%. In case of the "blind" model, *ReLU* give almost 1% higher accuracy than using *tanh* and *sigmoid*. All the results reported in these experiments and onwards are from models using *ReLU*.

4.2.2 Experiment 2: "Merging versus Injection"

In these experiments, we compare different encoder architectures to answer two questions raised in Section 3.1: When should the image be introduced to the model? Should we use multiplication or addition for attention mechanism? Hyperparameters are kept identical to those in experiment 1.

System	Accuracy	
	Train	**Test**
"Blind" BiLSTM	86.32	81.49
Merging	85.37	80.07
Par-inject	84.81	78.49
Attentional Par-inject + Multiplication	87.17	79.64
Attentional Par-inject + Addition	84.05	79.12
Inject by initialization hypothesis	86.91	81.07
Inject by initialization premise	86.73	80.48
Inject by initialization both	87.29	81.42

TABLE 4.5: Results for experiments on fusing modalities on M-SNLI. *Hypo* stands for *hypothesis*, *prem* stands for *premise*

.

Our implementations for *par-inject* and *inject by initialization* are based on Tensorflow 1.1 *BasicLSTMCell* source code which implements LSTM cells with dropout (**zaremba2014recurrent**). We do not use dropout for LSTMs but only for feed forward layers.

Table 4.5 illustrates the results of the experiment. "Blind BiLSTM" is described in the previous experiment, The *merging* model is the best performing model in the previous experiment, which grounds both the hypothesis and the premise to the image using multiplication.

Again, introducing the visual information hurts the performance of encoder models; however, we can see that *injection by initialization* is by far the best method to combine visual and textual information for encoder models. First, *par-inject* decreases the performance by 3%. Despite a boost of roughly 1% by multiplicative attention and 0.6% by additive attention, the performance of *par-inject* systems remains worse than the "blind" BiLSTM and the merging system. Interestingly, multiplication operation continues to surpass addition operation in multimodal interaction. Second, *injection by initialization* gives the best accuracies among various methods of grounding the text to the image, it performs on par with the "blind" model.

The fact that *inject by initialization* only affects the content of the hidden states of LSTMs partially explains its good performance. So far, introducing the image to the encoder models always hurts the performance and the more visual information is involved the less competitive the multimodal models are. *Par injection* and *Merging* let the image representation make a large contribution to the model by concatenating the image representation to every step of the text

input or merging the image representation to the text representation. *Inject by initialization*, on the other hand, only initialize the hidden states of LSTMs which would otherwise be randomly initialized. Results on the first two experiments on encoders models motivate us to seek more elegant ways of utilizing the image.

4.2.3 Experiment 3: The pairwise models

In this experiments, we compare the performance of the BiMPM and Multimodal BiMPM on the M-SNLI dataset.

Hyperparameters

Following Wang, Hamza, and Florian (2017)'s implementation, we implement our model on a newer version of Tensorflow and also make it publicly available[2]. Table 4.6 explains some parameters and their values. To focus on the impact of visual information, the values of these parameters are kept unchanged across models.

Parameter	Explanation	Value
max_char_per_word	Character limit for character embedding	10
max_sent_length	Sentence length limit	100
lambda_l2	Regularization L2 value	0.0
fix_word_vec	Whether to train word embedding with the model	false
char_lstm_dim	Dimension of LSTM when encoding character embedding	20
context_lstm_dim	Dimension of LSTM in the Context layer	100
aggregation_lstm_dim	Dimension of LSTM in the Aggregation layer	300
vocab_size	Vocabulary size	300,000
learning_rate	Learning rate	0.001
dropout_rate	Dropout rate	0.4
max_epochs	Number of epochs	10
MP_dim	Number of perspectives in the matching operation	10
batch_size	Number of instances in each batch	180
cost_function	Cost function	cross entropy
optimizer	Method to minimize the cost function	Adam

TABLE 4.6: Hyperparameters for experiments

[2]https://github.com/hoavt-54/nli-images

In this thesis, we choose the cost function to be the cross entropy error function. Each subset or batch, in our experiments, contains 180 instances with premises having similar lengths. Each model is trained in 10 epochs, the best-performing model on development set is picked. Dropout rate and learning rate are fixed to 0.6 and 0.001 respectively. We repeat the training process for each setting 3 times and report the average performance.

Results

Table 4.7 shows the average accuracies for the different models over three train and test iterations. The first model, denoted by Hypo ⊗ Prem, is the textual BiMPM model. Our implementation of BiMPM gets 86.41% accuracy which is slightly lower than one reported by Wang, Hamza, and Florian (2017). This can be explained by the fact that we have to restrict the vocabulary size due to hardware limitation. Note that the vocabulary size is the same across all our implementations. The best-performing model which matches the hypothesis with the premise and the image yields an accuracy of **87%** approximately. Thus, the inclusion of the image in the BiMPM architecture yields roughly 0.6% increase in accuracy. Some examples picked from the best-performance textual and multimodal model are shown in Table 4.10 and Table **??**.

System	**Accuracy**	
	Train	**Test**
Hypo ⊗ Prem	92.11	86.41
Hypo ⊗ Image	83.32	73.50
Prem ⊗ Image + Hypo ⊗ Image	87.43	82.19
Hypo ⊗ Prem + Hypo ⊗ Image	91.15	**86.99**
Hypo ⊗ Prem + Hypo ⊗ Image + Prem ⊗ Image	91.23	86.81

TABLE 4.7: Average accuracies of different pairwise models after 3 times running. *Hypo* stands for *hypothesis*, *prem* stands for *premise*

.

A related point to consider is training the multimodal pairwise models takes extremely more time than training its textual counterparts. As we have to train the models on CPUs due to the large memory requirements, it usually took around 80 hours to train the pairwise models and around 10 hours in case of the textual models. Besides the fact that introducing the image to the model increases the number of parameters to train, training the multimodal model has to use memory from swap memory which makes the training process vastly slower.

We performed an ablation study to highlight different components in the models. First, we removed the textual matching operations between the premise and the hypothesis. Then, we continue to remove the matching operation between the premise and the image. Interestingly, the model that does not include

textual matching still performs quite well (73.5% and 82.19%). This proves that visual information plays an important role in these models. However, the low accuracies in the training set suggest that these models are underfitting, which is reasonable due to simplicity.

We further break down the results by analyzing outputs from the textual and multimodal systems. Table 4.8 shows the performance of these two models on each class over there evaluation runs. As clearly seen, the multimodal outperforms the textual model on the contradiction and entailment class. This shows that visual information is helpful to remedy the ambiguities of natural language. A more detailed analysis is given in Section 4.2.5.

On the neutral class, however, the multimodal model only performs slightly better. One of the possible reasons is what we discussed in Section 4.1 namely that adding visual information occasionally can make the gold standard label questionable. We investigate this possibility in a linguistic analysis in Section 4.2.5.

Label	**Prediction**	**Occurrence**	
		BiMPM	**M-BiMPM**
contradiction	contradiction	85.3	86.5
contradiction	entailment	4.3	4.0
contradiction	neutral	9.3	8.3
entailment	contradiction	1.6	2.1
entailment	entailment	92.6	93.0
entailment	neutral	8.7	7.8
neutral	contradiction	6.0	6.3
neutral	entailment	10.9	10.5
neutral	neutral	81.4	81.5

TABLE 4.8: The confusion matrix shows detailed performance on the test set of textual and multimodal system on each class. Numbers are percentages of the test set.

4.2.4 Experiment 4: The "hard" dataset

As mentioned in Section 4.1, M-SNLI$_{hard}$ contains non-trivial examples from M-SNLI. In this section, we evaluation both the encoder model, the pairwise model and their textual model counterparts on this dataset.

As shown in Table 4.11, the obtained results on M-SNLI$_{hard}$ are consistent with results found in previous experiments. It is clear that all models suffer a large decrease in their performance. This further confirms the findings about annotation issues of SNLI by Gururangan et al. (2018). On this hard subset, including the image to encoder models continues to hurt the performance (from

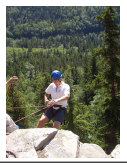
Image	Premise	Hypothesis	Label
	A young boy in suspenders .	The boy has suspenders attached to his pants .	Entailment
	A mountaineer about to descend down a mountain with a blue helmet on .	A mountaineer is standing on a mountain .	Entailment
	Two boys reading superhero books	Two boys watching a superhero show .	Contradiction

TABLE 4.9: Examples where adding the image improves the performance of the model (The multimodal guesses them correctly while the textual model does not.)

61% to 49%), especially on neutral cases (around 20% decrease). Meanwhile, the multimodal pairwise model consistently performs better than the textual model, across labels, reaching **73.75%** overall accuracy. This is also the best reported performance on this subset (Gururangan et al. 2018).

4.2.5 Linguistic analysis of the results

The first motivation for this analysis is that, as we mentioned in Chapter 1, we expect visual information to alleviate the difficulty in dealing with linguistic phenomena. Therefore, we carry out this analysis to test this assumption. Inspired by the linguistic analysis from Nangia et al. (2017) and Williams, Nangia, and Bowman (2017), we evaluate the M-BiMPM model and the "blind" BiMPM model on a linguistically annotated subset of the test set. We would like to see which model performs better in dealing with a given linguistic phenomenon.

Second, we noticed that adding the image gives more information but at the same time can change the label of the pair. For example, *Premise: A dog sitting in the front of a building* and *Hypothesis: A black dog sitting in the front of a building*. The example should be classified as *neutral* as we have no information regarding the color of the dog. However, as we add the image, it is clear what color of the dog is, which makes the label of the example is either *contradiction* or *entailment*.

For the annotation of linguistic phenomena, we randomly sample a subset of the SNLI test set containing 527 sentence pairs (185 entailment, 171 contradiction, 171 neutral), out of which 176 were from the hard test set (56 entailment, 62 contradiction, 58 neutral). Three annotators (including the author) were independently asked to read both the hypothesis and the premise and decide

Image	Premise	Hypothesis	Label
	A group of people are walking by a van .	People getting ready to enter a vehicle .	Neutral
	Several bicyclers are carrying their bikes up a grassy hill .	Bikers racing downhill .	Contradiction
	Basketball referee watching a team player hang from the hoop .	A woman watching the game .	Contradiction

TABLE 4.10: Examples where adding the image hinders the performance of the model

	BiLSTM	Merging BiLSTM	BiMPM	M-BiMPM
Entailment	72.12	69.09	80.43	81.38
Contradiction	60.79	46.34	77.62	76.12
Neutral	50.19	32.02	59.36	63.67
Overall	60.99	49.03	72.55	**73.75**

TABLE 4.11: Accuracies (%) for M-SNLI$_{hard}$ test set.

the most reasonable tags that explains the entailment relations. Details of the tags and examples are shown in Table 4.13. Each example was labeled by at least two annotators and could be assigned to more than one tag. As it is a multi-class annotation, we used Scott's π and Krippendorff's α to calculate annotation agreement as suggested by Passonneau (Passonneau, Habash, and Rambow 2006). The inter-annotator agreement was $\pi = 0.63$, $\alpha = 0.61$, and $\kappa = 0.64$[3].

We also label examples automatically. Automatic tags included SYNONYM and ANTONYM, which were detected using WordNet (Miller 1995). QUANTIFIER, PRONOUN, DIFF TENSE, SUPERLATIVE and BARE NP were identified using Penn treebank labels (Marcus, Santorini, and Marcinkiewicz 1993), while labels such as NEGATION were found with a straightforward keyword search. The tag LONG has been assigned to sentence pairs with a premise containing more than 30 tokens, or a hypothesis with more than 16 tokens.

[3]Inter-rater agreement was calculated using the NLTK implementation, http://www.nltk.org

System	Accuracy
DAM Parikh et al. (2016)	69.4
DIIN Gong, Luo, and Zhang (2018)	71.3
ESIM Chen et al. (2017a)	72.7
Our M-BiMPM	**73.75**

TABLE 4.12: Accuracies on SNLI$_{hard}$ (Gururangan et al. 2018).

To investigate how adding the image changes the labels, we analysed a subset of the M-SNLI$_{hard}$ containing 207 examples where adding the image hurts the performance. Two annotators (raw inter-annotator agreement: 96%) were independently asked to (a) read the Hypothesis and Premise carefully and check if the relation annotated in the original SNLI dataset actually held; (b) in case it held, check whether including the image altered the relation.

In Table 4.14, accuracies for the blind and multimodal version of BiMPM are broken down by linguistic labels. We only observe a significant difference in the *Entity* case, that is, where the referents in P and H are inconsistent. Here, the blind model outperforms the grounded one, an unexpected result, since one would assume a grounded model to be better equipped to identify mismatched referents.

Table 4.15 displays the proportions of image changing label and incorrect annotations. As the table suggests, there is a large amount of errors made by M-BiMPM in neutral cases due to relation changing by adding the image.

4.2.6 Experiment 5: The "foil" images

We continue to investigate the contribution of the image in all the multimodal models by evaluating these models on the M-SNLI$_{hard}$ with a "foil" image. These "foil" or wrong images are the most different images found in the test set from the correct images. The similarity is computed by using cosine on the image feature vectors. Our idea is that, if matching the text to the image is really helpful, we would see a drop in performance when the image is not related to the content of the text.

Table 4.16 shows some examples of the proximity between two images based on the cosine similarity between their feature representations. It is surprising that this simple similarity measurement can capture various semantic aspects of the image. For example, an image of two children having fun outside is closer to another image showing two children walking and less close to an image showing an indoor scene.

Table 4.17 shows the results, also containing results from Table 4.11 for easier reference. The "foil" image appears to hurt the performance; although not as much as one would expect. The performance of the M-BiMPM overall drops by 0.67% whereas that of Merging-BiLSTM is somewhat higher (-2.11%) which suggests that the grounding information plays a more important role in Merging-BiLSTM than it does in M-BiMPM.

Tag	Description	Example
Paraphrase	Two-way entailment, i.e., H entails P and vice versa.	P: *A middle eastern marketplace*, H: *A middle eastern store*
Generalisation	One-way entailment, i.e., H entails P but not necessarily vice versa.	P: *A group of people on the beach with cameras*, H: *People are on a beach.*
Entity	P and H describe different entities (e.g., subject, object, location) or incompatible properties of entities (e.g., color).	P: *A dog runs along the ocean surf*, H: *A cat is running in the waves.*
Verb	The sentences describe different, incompatible actions.	P: *Military personnel are shopping*, H: *People in the military are training.*
Insertion	H contains details and facts not present in P (e.g., subjective judgments and emotions.)	P: *Woman reading a book in a laundry room*, H: *The book is old.*
Unrelated	The sentences are completely unrelated.	P: *A woman is applying lip makeup to another woman*, H: *The man is ready to fight.*
Quantifier	The sentences contain numbers or quantifiers (e.g., *all, no, some, both, group*).	P: *A group of people are taking a fun train ride*, H: *People ride the train.*
World knowledge	Commonsense assumptions are needed to understand the relation between sentences (e.g., if there are named entities).	P: *A crowd gathered on either side of a Soap Box Derby*, H: *The people are at the race.*
Voice	The premise is an active/passive transformation of the hypothesis.	P: *Kids being walked by an adult*, H: *An adult is escorting some children.*
Swap	The sentences' subject and object are swapped from P to H.	P: *A woman walks in front of a giant clock*, H: *The clock walks in front of the woman.*

TABLE 4.13: Tags used in manual annotation of a subset of the SNLI test set.

4.2.7 Experiment 6: The significance test

Experiments 4.2.3, 4.2.4, 4.2.6 on the M-SNLI and M-SNLI$_{hard}$ has showed an improvement of the multimodal model over the textual model; however, it does not reflect in the linguistic analysis. Given the fact that the training data is considerably large, our concern is the improvement comes from the internal randomness of these models. Specifically, training neural networks critically depends on the initial value of the network parameters. Therefor, we carry out a statistical significance to further validate the advantage of adding visual information to the model.

Our ultimate goal in this experiment is to double-check the best result found in Experiment 4.2.3 (Table 4.7). Ideally, each experiment in this thesis should be accompanied with a test of statistical significance; however, given the huge

Manual tags	BiMPM	M-BiMPM	Automatic tags	BiMPM	M-BiMPM
Insertion	58	63	Antonym	84	84
Generalisation	93	89	Bare NP	79	75
Entity	95	↓78	Quantifier	73	73
Verb	77	68	Diff Tense	72	73
World knowledge	79	71	Pronoun	69	70
Quantifier	78	70	Synonym	69	71
Paraphrase	75	75	Long	67	73
Unrelated	50	50	Superlative	64	63
Swap	0	0	Negation	51	56

Table 4.14: Accuracies obtained by BiMPM and M-BiMPM models on SNLI$_{hard}$, by annotation tags. Arrows ↑↓ indicates a statistically significant difference.

Relation	Image mismatch (%)	Incorrect annotation (%)
Entailment	6.82	15.91
Neutral	44.58	1.20
Contradiction	3.80	22.78
Overall	24.76	12.62

Table 4.15: Cases where images hurt the M-BiMPM's performance: % of cases in which including the image modifies the original SNLI relation (Image mismatch), and cases in which the original SNLI relation is incorrectly annotated (reevaluated by our annotators).

computational time and cost, we could only afford to do the test for Experiment 4.2.3.

We choose the $k-$fold Cross-Validated Paired t Test with $k = 10$ (Dietterich 1998), where the training set is divided into 10 parts. On each step, the M-BiMPM and the BiMPM are trained on 8 parts, 2 parts are kept for validation and testing.

Table 4.18 shows the accuracies and numbers of instances misclassified by M-BiMPM and BiMPM on each fold. The t value is 11.73 (The value of p is $<$ 0.00001. The result is significant at $p < 0.05$) suggests that the value found in Experiment 4.2.3 is statistically significant.

4.3 Summary

In this section, we have seen that grounding on images is useful for NLI task; however, it requires an elegant and complex attention mechanism. First, we investigated different methods to incorporate the image into the BiLSTM models. The results were not promising as the accuracy kept declining as we introduce the image to the "blind" models. However, we found out that $ReLU$ is the good activation function for the task. Furthermore, grounding on the premise is less effective than grounding on the hypothesis or both of them.

Image	The closest image	The most different image
		
		

TABLE 4.16: Image closeness by computing cosine similarity on image representation. Images are taken from the test set.

	Merging-BiLSTM		M-BiMPM	
	Correct	Foil	Correct	Foil
Entailment	69.09	65.03	81.38	80.81
Contradiction	46.34	30.92	76.12	74.98
Neutral	32.02	31.46	63.67	63.39
Overal	49.03	46.92 (-2.11)	73.75	73.08 (-0.67)

TABLE 4.17: Accuracies of the multimodal models on M-SNLI$_{hard}$ with correct or foil image.

Second, introducing the image to the pairwise models show encouraging results. The multimodal M-BiMPM outperform the state of the art models on both M-SNLI dataset and M-SNLI$_{hard}$ dataset. However, the linguistic analysis showed that the multimodal M-BiMPM does not handle any linguistic phenomena better than the "blind" model. The linguistic analysis also suggest that a large amount of the errors made by the multimodal M-BiMPM is due to the fact that adding the image alters the original entailment relation.

Step(Fold)	Acc of M-BiMPM (%)	Miss by M-BiMPM	Acc of BiMPM (%)	Miss by BiMPM
1	85.04	8179	84.35	8556
2	85.30	7986	84.47	8437
3	85.32	7994	84.24	8583
4	85.58	7864	84.35	8535
5	85.13	8110	84.27	8579
6	85.42	7967	84.56	8437
7	84.89	8234	84.49	8452
8	85.37	7955	84.61	8369
9	85.19	8091	84.33	8561
10	85.11	8139	84.40	8527

TABLE 4.18: Results of the significance test. Acc: Accuracy, Miss: Number of instances misclassified by a model

Chapter 5

Conclusions and Future work

5.1 Conclusion

In this thesis, we proposed the *Multimodal Natural Language Inference* task which is an extension of Natural Language Inference. Given an image and its description as the premise, the task is to determine whether a hypothesis contradicts, entails or is neutral with regards to the image and its description. The task is challenging not only due to the natural complexity of human languages, but also because it is not obvious how to incorporate visual information into a textual model.

The dataset for this task was based upon the SNLI (Stanford Natural Language Inference) dataset. SNLI was manually created by asking annotators to come up with hypotheses which either entail, contradict or are neutral with respect to an image description. For our purpose, we mapped each sentence pair of SNLI to the original image dataset Flickr30k. The resulting dataset, made publicly available, has more than 565k instances.

Experiments show the usefulness of visual information in solving natural language inference. Furthermore, pairwise models are better than encoder models in processing multimodal data. Our extended multimodal version of BiMPM surpass the state of the art models on both M-SNLI dataset and M-SNLI$_{hard}$ dataset.

5.2 Contribution

We would like to highlight some of the main contributions of this thesis. These contributions include:

- We carried out one of the first experiments on including visual information for NLI.
- Our experiments on encoder models showed some interesting findings. First, it seems that the more visual information is involved in the models, the more it hurts the performance of the encoder models. Second, the

multiplication attentional function is largely better than its summation counterpart. Third, grounding both the hypothesis and the premise on the image is more effective than grounding on each of them.

- Integrating the image into the pairwise model shows higher results than integrating it into the encoder models.
- Linguistic annotation does not necessarily show any improvement of the multimodal model over the textual counterpart.
- We created a multimodal dataset for NLI based on SNLI.
- We achieve an advance over the current, best performing system in the NLI task on both the $MSNLI$ and $MSNLI_{hard}$.

5.3 Future work

Possible future work includes extending the current proposed model and applying BiMPM to other multimodal tasks. In this thesis, the two modality text and images are matched using affine transformation and cosine similarity. Despite the relatively simple similarity metric which directly compares vector in different spaces, experiments show an encouraging improvement. Having this result at hand, we would like to explore other methods to deal with multimodality, for example those proposed by Socher et al. (2014) and Karpathy and Fei-Fei (2015) in developing multimodal spaces.

Applying the BiMPM framework to other multimodal tasks is a promising research direction. In fact, this framework achieve 73.5% by matching only the hypothesis to the image. Therefore, a BiMPM-like model can be applied to other tasks similar to Multimodal NLI like visual question answering or image captioning.

Bibliography

Akhmatova, Elena (2005). "Textual entailment resolution via atomic propositions". In: *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*. Vol. 150. Southampton, UK.

Andrews, Mark et al. (2009). "Integrating experiential and distributional data to learn semantic representations." In: *Psychological review* 116.3, p. 463. American Psychological Association.

Antol, Stanislaw et al. (2015). "VQA: Visual Question Answering". In: *International Conference on Computer Vision (ICCV)*. Santiago, Chile.

Bahdanau, Dzmitry et al. (2014). "Neural machine translation by jointly learning to align and translate". In: *preprint arXiv:1409.0473*.

Barron, Andrew R (1993). "Universal approximation bounds for superpositions of a sigmoidal function". In: *IEEE Transactions on Information theory* 39.3, pp. 930–945.

Bayer, Samuel et al. (2005). "MITRE's Submissions to the EU Pascal RTE Challenge". In: *Proceedings of the Pattern Analysis, Statistical Modelling, and Computational Learning (PASCAL) Challenges Workshop on Recognising Textual Entailment*. Southampton, UK.

Bengio, Yoshua et al. (2007). "Greedy layer-wise training of deep networks". In: *Advances in neural information processing systems*. Vancouver, B.C., Canada, pp. 153–160.

Bernardi, Raffaella et al. (2016). "Automatic Description Generation from Images: A Survey of Models, Datasets, and Evaluation Measures." In: *J. Artif. Intell. Res.(JAIR)* 55, pp. 409–442. Elsevier.

Bos, Johan and Katja Markert (2006). "When logical inference helps determining textual entailment (and when it doesn't)". In: *Proceedings of the Second PASCAL RTE Challenge*. Venice, Italy, p. 26.

Bowman, Samuel R, Gabor Angeli, et al. (2015). "A large annotated corpus for learning natural language inference". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Lisbon, Portugal: Association for Computational Linguistics.

Bowman, Samuel R, Jon Gauthier, et al. (2016). "A fast unified model for parsing and sentence understanding". In: *arXiv preprint arXiv:1603.06021*.

Chen, Qian et al. (2017a). "Enhanced LSTM for natural language inference". In: *Proc. ACL*. Vancouver, Canada.

– (2017b). "Recurrent neural network-based sentence encoder with gated attention for natural language inference". In: *arXiv preprint arXiv:1708.01353*.

Cooper, Robin et al. (1996). *Using the framework*. Tech. rep. Technical Report LRE 62-051 D-16, The FraCaS Consortium.

Dagan, Ido et al. (2006). "The PASCAL recognising textual entailment challenge". In: *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*. Springer, pp. 177–190.

Das, Abhishek et al. (2017). "Visual Dialog". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, Hawaii.

Deng, Li, Dong Yu, et al. (2014). "Deep learning: methods and applications". In: *Foundations and Trends® in Signal Processing* 7.3–4, pp. 197–387.

Dietterich, Thomas G (1998). "Approximate statistical tests for comparing supervised classification learning algorithms". In: *Neural computation* 10.7, pp. 1895–1923.

Fang, Hao et al. (2015). "From captions to visual concepts and back". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. Boston, Massachusetts, USA, pp. 1473–1482.

Geffet, Maayan and Ido Dagan (2005). "The distributional inclusion hypotheses and lexical entailment". In: *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics. Ann Arbor, Michigan, USA, pp. 107–114.

Giampiccolo, D. et al. (2008). "The fourth PASCAL Recognising Textual Entailment Challenge." In: *Proceedings of the TAC 2008 Workshop on Textual Entailment*. Gaithersburg, Maryland, USA.

Glickman, Oren and Ido Dagan (2005). "Web based probabilistic textual entailment". In: *In Proceedings of the 1st Pascal Challenge Workshop*. Southampton, UK.

Goldberg, Yoav (2016). "A Primer on Neural Network Models for Natural Language Processing." In: *J. Artif. Intell. Res.(JAIR)* 57, pp. 345–420. Elsevier.

Gong, Yichen et al. (2018). "Natural language inference over interaction space". In: *arXiv preprint arXiv:1709.04348*.

Goodfellow, Ian et al. (2016). *Deep Learning*. MIT Press.

Graves, Alex and Jürgen Schmidhuber (2005). "Framewise phoneme classification with bidirectional LSTM and other neural network architectures". In: *Neural Networks* 18.5, pp. 602–610.

Gururangan, Suchin et al. (2018). "Annotation Artifacts in Natural Language Inference Data". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. Vol. 2. New Orleans, Louisiana, USA, pp. 107–112.

Han, Dan et al. (2017). "Visual Denotations for Recognizing Textual Entailment". In: *EMNLP*. Copenhagen, Denmark.

Harris, Zellig S (1954). "Distributional structure". In: *Word* 10.2-3, pp. 146–162.

He, Kaiming et al. (2016). "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. Las Vegas Valley, Nevada, USA, pp. 770–778.

Hinton, Geoffrey E, Nitish Srivastava, et al. (2012). "Improving neural networks by preventing co-adaptation of feature detectors". In: *arXiv preprint arXiv:1207.0580*.

Hinton, Geoffrey E et al. (2006). "A fast learning algorithm for deep belief nets". In: *Neural computation* 18.7, pp. 1527–1554. MIT Press.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural computation* 9.8, pp. 1735–1780.

Hodosh, Micah et al. (2013). "Framing Image Description as a Ranking Task: Data, Models and Evaluation Metrics". In: *Journal of Artificial Intelligence Research* 47, pp. 853–899. Elsevier.

Hornik, Kurt (1991). "Approximation capabilities of multilayer feedforward networks". In: *Neural networks* 4.2, pp. 251–257.

Jijkoun, Valentin, M Rijke, et al. (2005). "Recognizing textual entailment using lexical similarity". In: *Recognizing Textual Entailment*, p. 73.

Johnson, Justin et al. (2017). "CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning". In: *Proceedings of CVPR 2017*. Honolulu, Hawaii.

Karpathy, Andrej and Li Fei-Fei (2015). "Deep visual-semantic alignments for generating image descriptions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Santiago, Chile, pp. 3128–3137.

Krizhevsky, Alex et al. (2012). "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. Lake Tahoe, Nevada, USA, pp. 1097–1105.

Lai, Alice and Julia Hockenmaier (2017). "Learning to Predict Denotational Probabilities For Modeling Entailment". In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Valencia, Spain: Association for Computational Linguistics, pp. 721–730.

LeCun, Yann et al. (1989). "Backpropagation applied to handwritten zip code recognition". In: *Neural computation* 1.4, pp. 541–551.

Lin, Zhouhan et al. (2017). "A structured self-attentive sentence embedding". In: *arXiv preprint arXiv:1703.03130*.

Ling, Wang et al. (2015). "Finding function in form: Compositional character models for open vocabulary word representation". In: *arXiv preprint arXiv:1508.02096*.

Liu, Yang et al. (2016). "Learning natural language inference using bidirectional LSTM model and inner-attention". In: *arXiv preprint arXiv:1605.09090*.

Luong, Minh-Thang et al. (2015). "Effective approaches to attention-based neural machine translation". In: *arXiv:1508.04025*.

Lynch, Corey et al. (2016). "Images don't lie: Transferring deep visual semantic features to large-scale multimodal learning to rank". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. San Francisco, California, USA, pp. 541–548.

MacCartney, Bill (2009). "Natural language inference". PhD thesis.

Malinowski, M. and M. Fritz (2014). "A multi-world approach to question answering about real-world scenes based on uncertain input". In: *Advances in Neural Information Processing Systems*. Montreal, Canada.

Marcus, Mitchell P et al. (1993). "Building a large annotated corpus of English: The Penn Treebank". In: *Computational Linguistics* 19.2, pp. 313–330. ISSN: 08912017. DOI: 10.1162/coli.2010.36.1.36100. URL: http://aclweb.org/anthology/J93-2004.

Marelli, Marco et al. (2014). "A SICK cure for the evaluation of compositional distributional semantic models." In: *LREC*. Reykjavik, Iceland, pp. 216–223.

McCulloch, Warren S and Walter Pitts (1943). "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4, pp. 115–133. Springer.

Mikolov, Tomáš (2012). "Statistical language models based on neural networks". In: *Presentation at Google, Mountain View, 2nd April*.

Mikolov, Tomas et al. (2010). "Recurrent neural network based language model." In: *Interspeech*. Vol. 2. Makuhari, Chiba, Japan, p. 3.

Miller, George A. (1995). "WordNet: a lexical database for English". In: *Communications of the ACM* 38.11, pp. 39–41. ISSN: 00010782. Association for Computing Machinery.

Mou, Lili et al. (2016). "Natural language inference by tree-based convolution and heuristic matching". In: *The 54th Annual Meeting of the Association for Computational Linguistics*. Berlin, Germany, p. 130.

Nangia, Nikita et al. (2017). *The RepEval 2017 Shared Task: Multi-Genre Natural Language Inference with Sentence Representations*. Copenhagen, Denmark.

Parikh, Ankur P et al. (2016). "A decomposable attention model for natural language inference". In: *arXiv preprint arXiv:1606.01933*.

Passonneau, R. et al. (2006). "Inter-annotator agreement on a multilingual semantic annotation task". In: *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, pp. 1951–1956.

Pasunuru, Ramakanth and Mohit Bansal (2017). "Multi-Task Video Captioning with Video and Entailment Generation". In: *arXiv preprint arXiv:1704.07489*.

Pennington, Jeffrey et al. (2014). "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. Doha, Qatar, pp. 1532–1543.

Rosenblatt, Frank (1958). "The perceptron: A probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6, p. 386. American Psychological Association.

Roy, Deb (2005). "Semiotic schemas: A framework for grounding language in action and perception". In: *Artificial Intelligence* 167.1-2, pp. 170–205.

Sammons, Mark (2015). "Recognizing Textual Entailment". In: *The Handbook of Contemporary Semantic Theory*. Ed. by Shalom Lappin and Chris Fox. John Wiley & Sons, Ltd, pp. 523–557. ISBN: 9781118882139.

Sammons, Mark et al. (2011). "Recognizing textual entailment". In: *Multilingual Natural Language Applications: From Theory to Practice. Prentice Hall, Jun.*

Simonyan, Karen and Andrew Zisserman (2014). "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556*.

Socher, Richard et al. (2014). "Grounded compositional semantics for finding and describing images with sentences". In: *Transactions of the Association for Computational Linguistics* 2, pp. 207–218. MIT Press.

Specia, Lucia et al. (2016). "A Shared Task on Multimodal Machine Translation and Crosslingual Image Description." In: *WMT*. Berlin, Germany, pp. 543–553.

Suhr, Alane et al. (2017). "A Corpus of Natural Language for Visual Reasoning". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Vancouver, Canada: Association for Computational Linguistics, pp. 217–223.

Sundermeyer, Martin et al. (2012). "LSTM neural networks for language modeling". In: *Thirteenth Annual Conference of the International Speech Communication Association*. Portland, Oregon, USA.

Sutskever, Ilya et al. (2014). "Sequence to sequence learning with neural networks". In: *Advances in neural information processing systems*. Montréal, Canada, pp. 3104–3112.

Tanti, M et al. (2018). "Where to put the image in an image caption generator." In: *Natural Language Engineering* 24 (3), pp. 467–489. Cambridge University Press.

Vendrov, Ivan et al. (2016). "Order-Embeddings of Images and Language". In: *Proceedings of the International Conference of Learning Representations (ICLR)*. San Juan, Puerto Rico.

Wang, Zhiguo et al. (2017). "Bilateral Multi-Perspective Matching for Natural Language Sentences". In: *arXiv preprint arXiv:1702.03814*.

Williams, Adina et al. (2017). "A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference". In: *arXiv preprint arXiv:1704.05426*.

Xu, Kelvin et al. (2015). "Show, attend and tell: Neural image caption generation with visual attention". In: *International Conference on Machine Learning*. Lille, France, pp. 2048–2057.

Xu, Wenduan et al. (2015). "CCG Supertagging with a Recurrent Neural Network." In: *ACL (2)*. Beijing, China, pp. 250–255.

Young, Peter et al. (2014). "From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions". In: *Transactions of the Association for Computational Linguistics* 2, pp. 67–78.

Zeiler, Matthew D and Rob Fergus (2014). "Visualizing and understanding convolutional networks". In: *European conference on computer vision*. Springer. Zürich, Switzerland, pp. 818–833.

List of Figures

List of Tables

List of Abbreviations

BiMPM	**Bi**lateral **M**ulti-**P**erspective **M**atching
BiLSTM	**Bi**directional **L**ong-**S**hort **T**erm **M**emory
BPTT	**B**ackpropagation **T**hrough **T**ime
CCG	**C**ombinatory **C**ategorial **G**rammar
CovNets	**Cov**olutional **N**eural **N**etworks
CV	**C**omputer **V**ision
FFN	**F**eed-**f**orward **Networks**
NLI	**N**atural **L**anguage **I**nference
NLP	**N**atural **L**anguage **P**rocessing
SICK	**S**entences **I**nvolving **C**ompositional **K**nowledge
SNLI	**S**tanford **N**atural **L**anguage **I**nference
RNNs	**R**ecurrent **N**eural **N**etworks
RTE	**R**ecognizing **T**extual **E**ntailment