MASTER THESIS

INTELLIGENT SYSTEMS

DETECTING HATE SPEECH IN MULTIMODAL MEMES USING VISION-LANGUAGE MODELS

RIZA VELIOGLU

Faculty of Technology
Bielefeld University

SUPERVISED BY
PROF. DR. BARBARA HAMMER

REVIEWED BY
PROF. DR. BARBARA HAMMER,
JEWGENI ROSE

APRIL 21, 2021

Acknowledgments

First and foremost, I would like to thank my supervisors, Professor Barbara Hammer and Jewgeni Rose for their valuable guidance through each stage of the process. Your insightful feedback pushed me to improve my thinking and brought my work to a higher level.

In addition, I would like to express my gratitude to my parents and my sister for their wise counsel and compassion. Thank you for your unconditional love and support under all circumstances.

I would also like to thank the Hammer Lab research group overall for their support, especially Johannes Brinkrolf and Valerié Vaquet. Thank you for the helpful feedback.

Finally, without the help of my friends, Benjamin Griffith, Dogan Hayirli, Furkan Arar, Kagan Dokuzoglu, and Sinan Kuscu I would not have been able to complete this dissertation. They offered stimulating conversations as well as enjoyable distractions to rest my mind outside of my research. This accomplishment would not have been possible without them. Thank you.

Detecting Hate Speech in Multimodal Memes Using Vision-Language Models

Abstract

Memes on the Internet are often harmless and sometimes amusing. The apparently innocent meme, though, becomes a multimodal form of hate speech when certain kinds of pictures, text, or variations of both are used – a *hateful meme*. The Hateful Memes Challenge[1] is a one-of-a-kind competition that focuses on detecting hate speech in multimodal memes and proposes a new data collection with 10,000+ new examples of multimodal content. We use VisualBERT, which is also known as "BERT for vision and language," and Ensemble Learning to boost the performance. In the Hateful Memes Challenge, our solution received an AUROC of 0.811 and an accuracy of 0.765 on the challenge test set, placing us third out of 3,173 participants. The code is available at `https://github.com/rizavelioglu/hateful_memes-hate_detectron`

1 `https://www.drivendata.org/competitions/70/hateful-memes-phase-2/`

I dedicate this thesis to my parents...

Declaration of Authorship

I, Riza Velioglu, declare that this thesis titled, Detecting Hate Speech in Multimodal Memes Using Vision-Language Models and the work presented in it is my own. I confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.

CONTENTS

1 INTRODUCTION

The world around us is multimodal – we see things, hear sounds, read sentences, feel textures, taste flavors and so on. Humans communicate with their environment by integrating information and creating connections between senses across the multiple sensory sources [6]. When a baby eats an apple, for example, she not only experiences the taste, but also she can hear the apple crunch, see its shiny skin, and feel its smooth surface on her skin [82]. These multiple simultaneous and time-locked stimuli, according to psychologists, are a critical enabler of human perceptual learning about the environment. Even when seen under drastically different circumstances, it takes just a fraction of a second to identify an individual or an object. In 2005, [69] conducted a research on how neurons in the human brain produce such a robust, high-level representation. They discovered that human brain possesses multimodal neurons. Rather than any particular visual feature, these neurons respond to clusters of abstract concepts centered around a common high-level theme. The "Halle Berry" neuron was the most well-known of these, that responds to photographs, drawings, or even images of the text "Halle Berry", but not other names.

In comparison, machine learning has historically concentrated on solving tasks with a single modality (e.g. in computer vision, speech recognition or natural language processing). Large, unimodal corpora containing millions of data samples, such as images (ImageNet [21]), text (BooksCorpus [117]) have been developed and studied by research communities in these domains. A particular promising strategy for exploiting these massive datasets is deep learning, which learns representations that map raw data formats to convenient and compact embedding vectors. This is mostly achieved by deep neural networks [47, 46] which minimize a suitable loss function on input-output data pairs to learn increasingly semantic, hierarchical representations. As a result of this paradigm, new neural network architectures have emerged which are especially effective at distilling images [45] and text [87] into concise, meaningful representations.

Learning from unimodal data in isolation, however, is becoming an increasingly unnatural and naive scenario, given the explosion of multimodal content. Videos are inherently multimodal, with audio accompanying visual material. Memes are naturally multimodal, too, images accompanied by contextual text in the form of captions. Therefore, artificial intelligence must be able to interpret and reason about multimodal signals to advance in understanding the world around us. In this work, we primarily focus on two input modes: vision which is in the form of images; and written natural language which is in the form of text.

1.1 MOTIVATION

Memes have grown in popularity in recent years, with over 180 Million posts on various social media sites as of 2018 [113]. While memes are mostly innocent and created for amusement, they have also been used to generate and disseminate *hate speech* in toxic cultures. Hate speech is described as "any communication that disparages a target group of people based on some characteristic such as race, colour, ethnicity, gender, sexual orientation, nationality, religion, or other characteristics" [61]. Large tech firms, such as Facebook, operate sites with millions of regular users, and they are obliged to delete a significant amount of content with regards to hate speech to protect their users. According to Mike Schroepfer, Facebook's current CTO, they took action on *9.6 million* pieces of content in the first quarter of 2020 [76] for breaching their hate speech policies. This indicates that the amount of malicious content on the internet today can not be dealt with by having humans reviewing every sample. Consequently, machine learning and in particular deep learning techniques would be needed to reduce the prevalence of online hate speech. However, hate speech detection is a difficult problem since it often depends on context, involves world awareness, and can be subtle. It is also a significant issue because of its ability to influence everyone in our society. Detecting hate speech in memes is even more challenging due to multimodality. As a result, these techniques must process content in the same manner that humans do: holistically. When viewing a meme, a human would not think of the words and the picture separately, but rather the *combined* context. Furthermore, although the visual and linguistic content of a meme is usually neutral or humorous on their own, however, they may create a hateful meme when combined. An example of such a case is shown in Figure 1.1. Consider the meme in the middle which does not contain hate speech. However, replacing the image in the meme with an image of a person, the meme becomes offensive.

There has been a surge of interest in multimodal problems since 2015 in visual question answering [4, 33], image captioning [44, 17], speech recognition [83, 63] and beyond. However, it is not obvious how much true multimodal reasoning and understanding are needed to solve today's problems. For example, language may unwittingly place strong priors on certain datasets, resulting in exceptional results without any comprehension of the visual material. To overcome this problem and to measure truly multimodal understanding and reasoning of the models, The Hateful Memes Challenge [42] is created. A crucial characteristic of the challenge is the so-called "benign confounders" (also called *contrastive* [25] or *counterfactual* [39] examples) which addresses the risk of exploiting unimodal priors by models: for every hateful meme, there are alternative images or text that flip the label to not-hateful. Figure 1.1 shows examples of benign confounders for a meme. Such image and text confounders require multimodal reasoning to classify the original meme and its confounders correctly. Thus, making the dataset challenging and appropriate for testing the true multimodality of a model.

1.2 THESIS OUTLINE

The remainder of this thesis is structured as follows: In Chapter 2 we give background on the topics required to follow the work presented in this thesis, specifically Transformer architecture and simple multimodal model architectures. In Chapter 3, we first present how (multi-)modality is defined in the research field and which one we adopted for our work. Then, we explain the role of deep learning in the field and we narrow down our focus on a specific branch, namely, vision-and-language (V&L). In Chapter 4, we give brief information about a challenge that aims to catalyze the research in the field and analyse the dataset proposed with the challenge in detail, as well as the benchmark models and their results. In Chapter 5, we explain our methodology to tackle the challenge, which received the third prize in the competition. We present the results and analyse them in Chapter 6. Finally, Chapter 7 discusses the impact of this work as well as potential research directions.

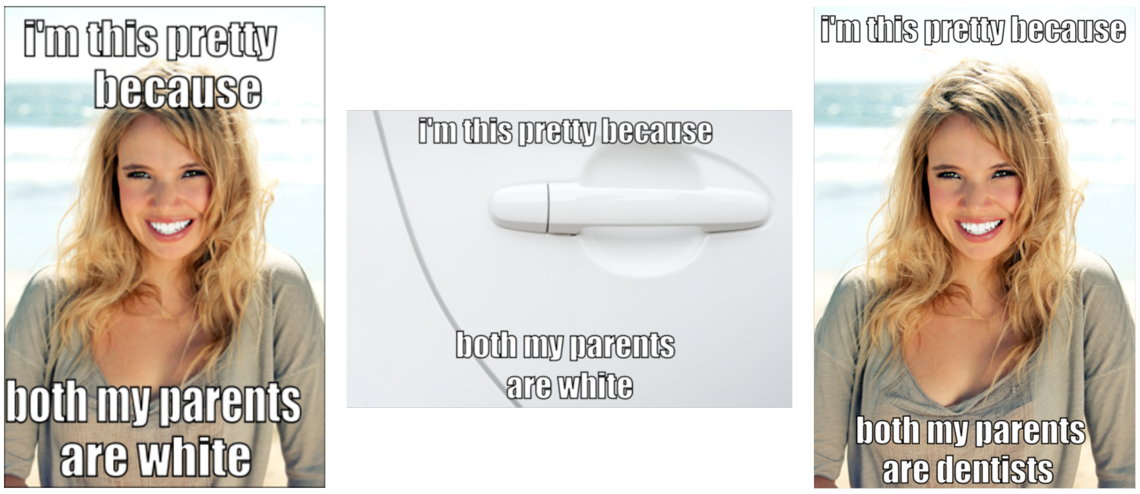

i'm this pretty because
both my parents are white
i'm this pretty because
both my parents are white
i'm this pretty because
both my parents are dentists

Figure 1.1: Three memes sampled from the dataset: (a) hateful (left), (b) not-hateful (center), and (c) another not-hateful (right). As the label flips by changing only the image, (b) is said to be the *image benign confounder* of (a). Similarly, (c) is said to be the *text benign confounder* of (a) as the label flips by changing only the text.

1.3 CONTRIBUTIONS

The main contributions of this thesis can be summarized as follows:

- We present a comprehensive empirical study to tackle detecting hate speech in multimodal memes which improves the provided benchmarks on the task.
- We open-source a multimodal model that could be used in a real-world application as the model performs reasonably well on detecting hate speech.

1.4 PUBLICATIONS

The research has led to two publications: (1) where the approach is explained, as a result of the competition, and (2) a competition report where the results of the competition are discussed:

1. **"Detecting Hate Speech in Memes Using Multimodal Deep Learning Approaches: Prize-winning solution to Hateful Memes Challenge"** [91]
   Riza Velioglu, Jewgeni Rose. At *NeurIPS Competitions*, 2020 (Oral Presentation)
2. **"The Hateful Memes Challenge: Competition Report"** (soon to be published)
   Douwe Kiela, Hamed Firooz, Aravind Mohan, Vedanuj Goswami, Amanpreet Singh, Casey A. Fitzpatrick, Peter Bull, Greg Lipstein, Tony Nelli, Ron Zhu, Niklas Muennighoff, **Riza Velioglu**, Jewgeni Rose, Phillip Lippe, Nithin Holla, Shantanu Chandra, Santhosh Rajamanickam, Georgios Antoniou, Ekaterina Shutova, Helen Yannakoudakis, Vlad Sandulescu, Umut Ozertem, Patrick Pantel, Lucia Specia, Devi Parikh. In *Journal of Machine Learning Research (JMLR) Special Issue on NeurIPS Competition*, 2021.

FOUNDATION/BACKGROUND 2

In this chapter, we give background information required to understand the work proposed in this thesis. We start with explaining the Transformer architecture [90] in detail in Section 2.1. Then, we present simple multi-modal models and illustrate their workflow in Section 2.2.

2.1 TRANSFORMERS

For sequence transduction tasks, the sequence-to-sequence (seq2seq) [87] encoder-decoder architecture is the foundation. It simply proposes encoding the entire sequence (source) at once and then using that encoding as a context for generating the decoded or target sequence. In seq2seq models[1] both the encoder and decoder are recurrent neural networks, i.e. use LSTM or GRU units. Machine translation between multiple languages in text or audio, question-answer dialogue generation, and even parsing sentences into grammar trees are examples of transformation tasks. This is, indeed, analogous to the human propensity to 'listening' a sentence (sequence) entirely before answering, whether in a conversation, translation, or other similar tasks. However, since a single vector must capture the entire sequence of information in the encoder-decoder architecture, it makes retaining knowledge at the start of the series and encoding long-range dependencies difficult. In other words, when the encoder processes the entire input sequence and compresses the information into a context vector, it often forgets about the beginning of the input. To solve this problem, the attention mechanism [5] was established.

Attention mechanisms are best known for their use in NLP because, as stated earlier, attention was created to address the problem of long sequences in machine translation,

1 In academia, seq2seq models are also known as the Encoder-Decoder models.

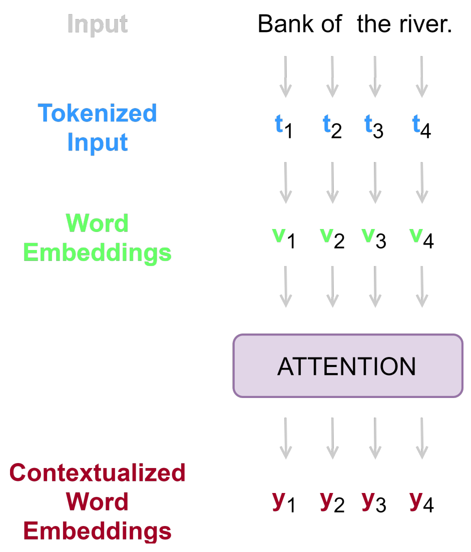


Figure 2.1: Attention mechanism reweighs the word embeddings in such a way that the resulting embeddings are better and have more context.

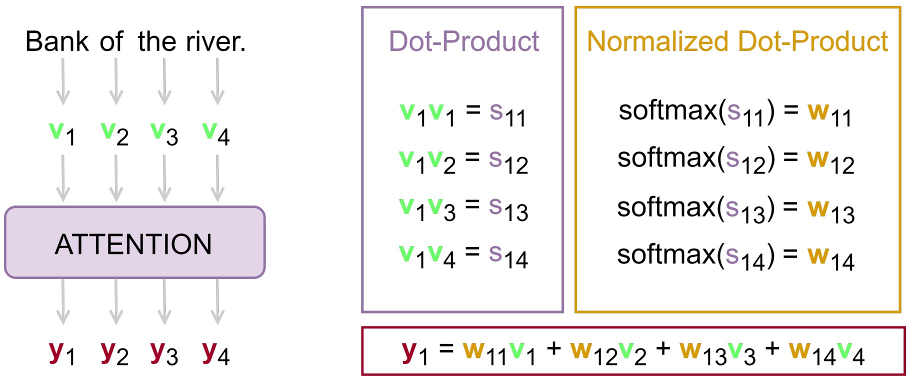


Figure 2.2: To produce the contextualized embedding for the first token, we first calculate the dot-product between the word "bank" and all the other words in the sequence. Then, we normalize each value, which produces the attention vector $\vec{w}_1$. Lastly, we calculate the dot-product between the attention vector, $\vec{w}_1$, and the input sequence, $\mathbf{V}$, which results in the contexualized embedding vector $\vec{y}_1$. [73]

which is also a problem in many other NLP tasks. However, given the big improvement in NLP tasks, its use extended to the computer vision [105] and further research on attention shed a light on different attention mechanisms, e.g. Content-based attention [31], Additive attention [5], Dot-Product attention [55], and Scaled Dot-Product attention [90]. We will only focus on Scaled Dot-Product, which is the building block of Transformer architecture.

In Deep Learning, attention can be thought of as a vector of importance weights: to predict or infer one element, such as a pixel in a picture or a word in a sentence, we estimate how strongly it is associated with (or "attends to") other elements using the attention vector, and take the sum of their values weighted by the attention vector as the target approximation. Figure 2.1 illustrates the idea of attention with an example. Given the sentence "Bank of the river." we tokenize the input and replace each of them with its corresponding word vector. But, considering the context of the word "bank" in the sentence, its meaning should be related to coast, beach or seaside rather than a financial institution. Current word vectors are lacking this capability and attention is used to address this problem. Attention layer applies the 'reweighing scheme' to word embeddings, $\vec{v}_i$, such that the output, $\vec{y}_i$ have better context.

Next, we explain the Scaled-Dot Product used in Transformers (Section 2.1.1) and how they are combined to form a Multi-Head Attention (Section 2.1.2). Lastly, we present the overall Transformer architecture which makes use of Scaled-Dot Product Attention and Multi-Head Attention (Section 2.1.3).

2.1.1 *Self-Attention*

Self-Attention in NLP, also known as intra-attention, is a type of attention that can reflect the relationships between words in a sentence. The model is able to look at other positions in the input sequence for hints that can help to produce a better encoding for each word in the sequence. Figure 2.2 shows an example attention mechanism that uses scaled dot-product on our previous example sentence. Note that the general formula for the dot-product can be written as follows:

$$s_{ij} = \vec{v}_i \cdot \vec{v}_j \qquad \forall i, j \in \{1, \cdots, n\} \tag{2.1}$$

where n is the sequence length. Or it can be written in matrix notation as follows:

$$\mathbf{S} = \mathbf{V} \cdot \mathbf{V}^T \tag{2.2}$$

where $\mathbf{V} \in \mathbb{R}^{n \times d_k}$, $\mathbf{S} \in \mathbb{R}^{n \times n}$, and d_k is the embedding dimension. Then, we normalize each $s_{ij} \in \mathbb{R}$ for all $i, j \in \{1, \cdots, n\}$ using a softmax function such that it holds:

$$\sum_{j=1}^{n} w_{ij} = 1 \tag{2.3}$$

This is done for numerical and interpretability reasons. After normalization, the attention matrix includes the information as to how much a particular token should attend to each of the other tokens in the sequence. To inject this information, we multiply the attention matrix $\mathbf{W}$ with the word embedding matrix $\mathbf{V}$:

$$\mathbf{W} \cdot \mathbf{V} = \mathbf{Y} \tag{2.4}$$

where $\mathbf{W} \in \mathbb{R}^{n \times n}$, $\mathbf{V} \in \mathbb{R}^{n \times d_k}$, $\mathbf{Y} \in \mathbb{R}^{n \times d_k}$, and d_k is the embedding dimension.

Figure 2.3 shows the calculations for a different token in the same sentence but with a different illustration. Notice that there are no learnable weights, parameters in this mechanism. In order to apply deep learning to this mechanism, Transformer architecture [90] introduces parameters which would be learned during training, with the hope that the resulting embeddings would benefit from the patterns that are learned. Also notice that all the word vectors $v_i \in \mathbf{V}$ are used exactly three times in dot-product calculations: (1) and (2) in the first dot-product (bottom-left in Figure 2.3), and (3) in the final dot-product (middle-top in Figure 2.3). These three places are where the parameters are inserted, as illustrated in Figure 2.4. The parameters $\mathbf{W}^Q$, $\mathbf{W}^K$, $\mathbf{W}^V$ are learned during training process and the matrix multiplications of $\mathbf{VW}^Q$, $\mathbf{VW}^K$, $\mathbf{VW}^V$ are given names as queries($\mathbf{Q}$), keys($\mathbf{K}$), values($\mathbf{V}$), respectively. These terms are normally used in database jargon but it is advantageous to use them here as they already give an idea of the variables. For instance, referring to Figure 2.3, to calculate $\vec{y}_3$ we need to calculate to dot-product between $\vec{v}_3$ and all the other word vectors. In other words, we *query* the $\vec{v}_3$ and what the query returns are the *keys*. Then, by combining the queries and the keys, we construct the *values*. Now, the output can be represented using these terms, as seen in Figure 2.4, the output matrix $\mathbf{Y}$ is computed as:

$$Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = softmax\left(\frac{\mathbf{QK}^T}{\sqrt{d_k}}\right)\mathbf{V} \tag{2.5}$$

This attention scheme is identical to dot-product attention, except for the scaling factor of $\frac{1}{\sqrt{d_k}}$. This is required to counteract the vanishing gradient problem.

2.1.2 *Multi-Head Attention*

The multi-head attention runs through the scaled dot-product attention several times in parallel, rather than computing it just once. The independent attention outputs are simply concatenated and converted linearly into the expected dimensions. Multi-head attention helps the model to simultaneously attend to details from various representation subspaces at various locations. This can be thought of as ensembling scaled dot-product

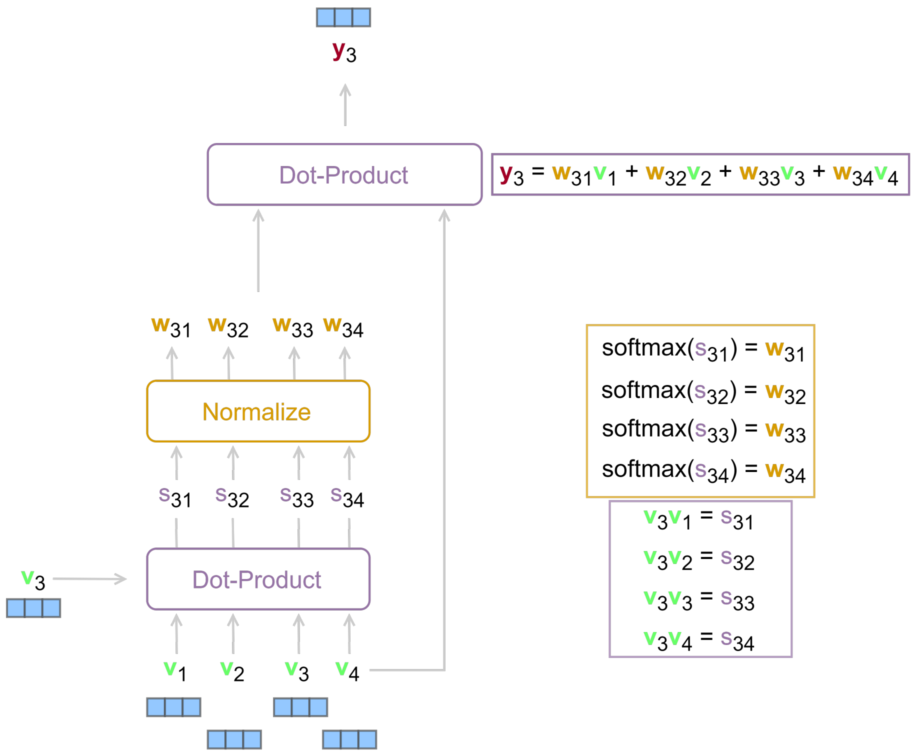


Figure 2.3: To produce the contextualized embedding for the third word vector, $\vec{v}_3$, we first calculate the dot-product between $\vec{v}_3$ and all the words in the sequence, $\mathbf{V}$. Then, we normalize each value, which produces the attention vector $\vec{w}_3$. Lastly, we calculate the dot-product between the attention vector, $\vec{w}_3$, and the input sequence, $\mathbf{V}$ which results in the contexualized embedding vector $\vec{y}_3$.

attention using the attention function. The resulting output values from each mechanism is then concatenated and projected using a Linear layer, as depicted in Figure 2.5.

The Multi-Head Attention can be written as follows:

$$MultiHead(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = Concat(head_1, \cdots, head_h)\mathbf{W}^O \tag{2.6}$$

where $head_i = Attention(\mathbf{QW}_i^Q, \mathbf{KW}_i^K, \mathbf{VW}_i^V)$, and where the projections are parameter matrices $\mathbf{W}_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $\mathbf{W}_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $\mathbf{W}_i^V \in \mathbb{R}^{d_{model} \times d_v}$ and $\mathbf{W}^O \in \mathbb{R}^{hd_v \times d_{model}}$. Originally there are 8 parallel attention layers, or heads ($h = 8$) and the parameters $d_k = d_v = d_{model}/h = 64$.

2.1.3 *Overall Architecture*

We feed the words sequentially to the model in the case of RNNs, and each token is aware of its order in the sequence. Multi-Head Attention, on the other hand, is permutation invariant. That is to say, it calculates the output of each token independently, with no regard for word order. To address this, the Transformer adds a vector called 'positional encoding' to each input embedding. These vectors follow a pattern that the model learns and uses to decide the location of each word in the sequence. The positional encoding $\mathbf{P} \in \mathbb{R}^{n \times d_k}$ has the same dimension as the word embeddings, hence, the two can be summed directly. The Transformer uses the sinusoidal positional encoding which is

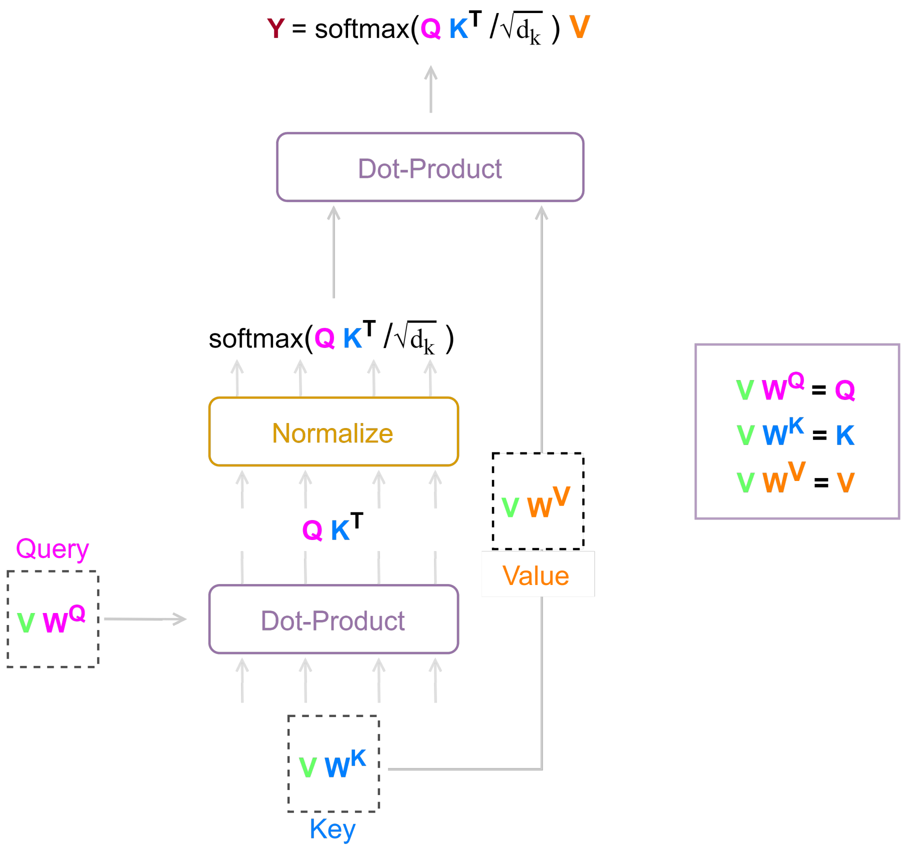


Figure 2.4: Scaled Dot-Product Attention: The input consists of queries and keys of dimension d_k, and values of dimension d_v. We compute the dot products of the query (**Q**) with all keys (**K**), divide each by $\sqrt{d_k}$, and apply a softmax function to obtain the weights on the values (**V**).

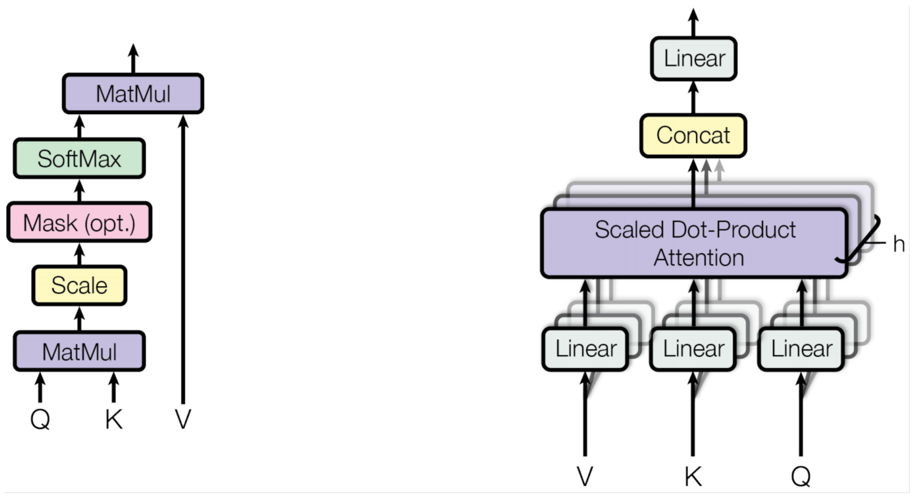


Figure 2.5: Scaled Dot-Product Attention (left) and Multi-Head Attention (right).

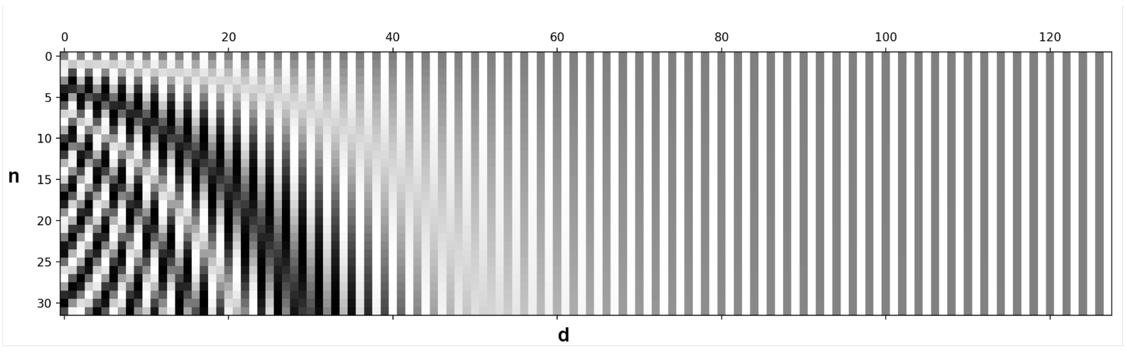


Figure 2.6: Sinusoidal positional encoding with $n = 32$ and $d = 128$. The value is between -1 (black) and 1 (white) and the value 0 is gray [96].

defined as follows:

$$\mathtt{PE}(i, \delta) = \begin{cases} sin(i/10000^{2\delta'/d}), & \text{if} \quad \delta = 2\delta' \\ cos(i/10000^{2\delta'/d}), & \text{if} \quad \delta = 2\delta' + 1 \end{cases} \tag{2.7}$$

where $i = \{1, \cdots, n\}$ is the token position and $\delta = \{1, \cdots, d\}$ is the dimension. That is, each dimension of the positional encoding corresponds to a sinusoid with various wavelengths in various dimensions, ranging from 2π to $10000 \cdot 2\pi$. Figure 2.6 an example of positional encoding for 32 words (rows) with an embedding size of 128 (columns). That is, the first row would be the vector added to the embedding of the first word in the sequence.

Figure 2.7 shows the Transformer model architecture with consists of encoder and decoder components. Both the encoder and decoder are made up of several similar encoder and decoders that can be stacked `Nx` times.

Encoder The encoder creates an attention-based representation capable of locating a single piece of information within an infinitely vast context. The block is made up of a stack of `Nx=6` equivalent layers. Each layer has a multi-head self-attention layer and a simple position-wise fully-connected feed-forward network. Each sub-layer has a residual connection and a layer normalization. All the sub-layers output data of the same dimension $d_{model} = 512$.

Decoder The decoder block is also a stack of `Nx=6` identical layers. Each layer has two sub-layers of multi-head attention mechanism and a simple fully-connected feed-forward network. Similarly, each sub-layer has a residual connection and layer normalization. The first multi-head attention sub-layer is modified such that the positions can not attend to subsequent positions, therefore, avoiding the mechanism looking into the future when predicting the current position.

2.2 BASIC MULTIMODAL MODELS

We will discuss multi-modal deep learning in detail later in Chapter 3. In this section, we explain the general representation of a multi-modal model for a vision-and-language (V&L) classification task and explore simple model architectures. We address

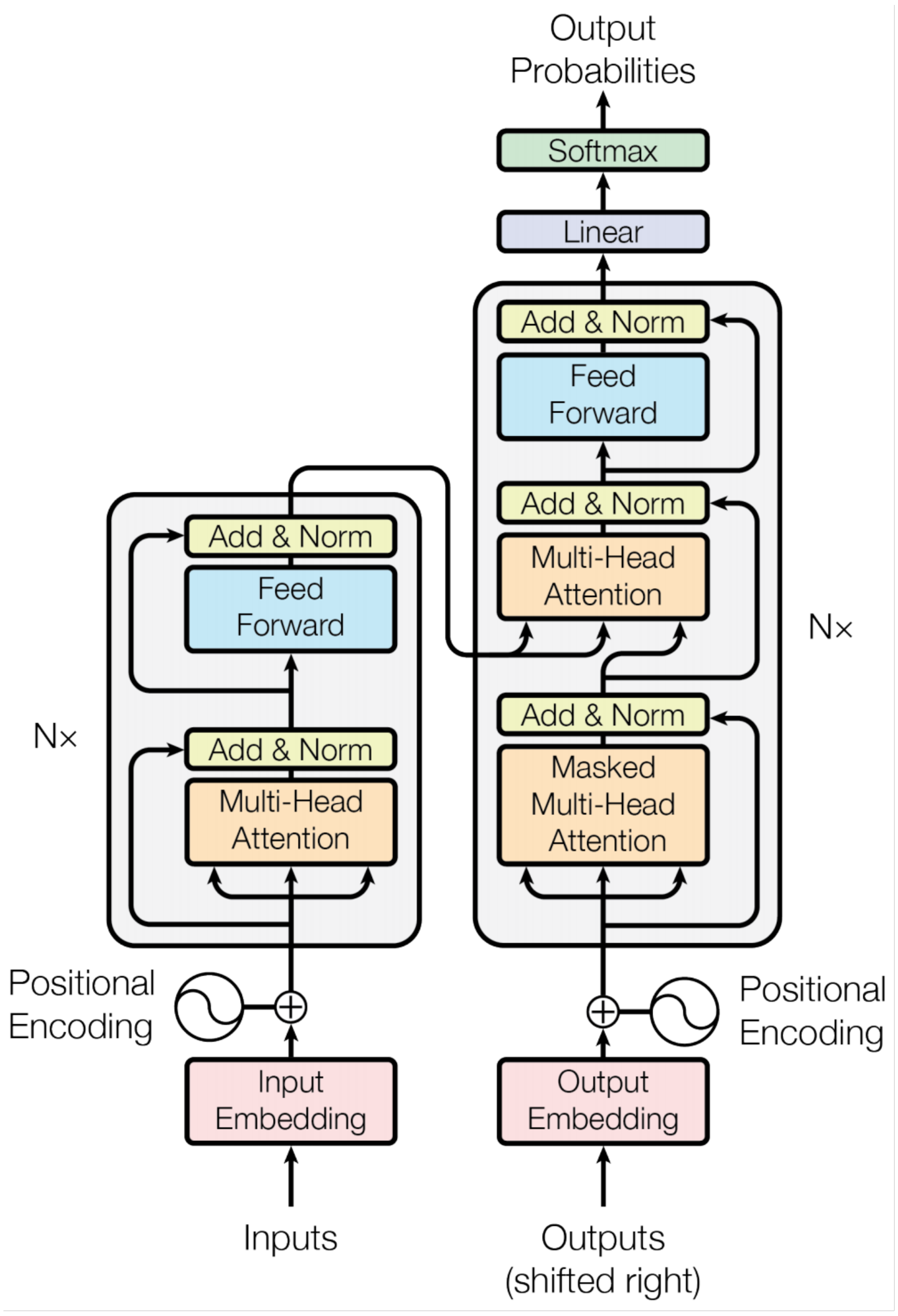


Figure 2.7: The Transformer - model architecture with encoder (left) and decoder (right) blocks. [90]

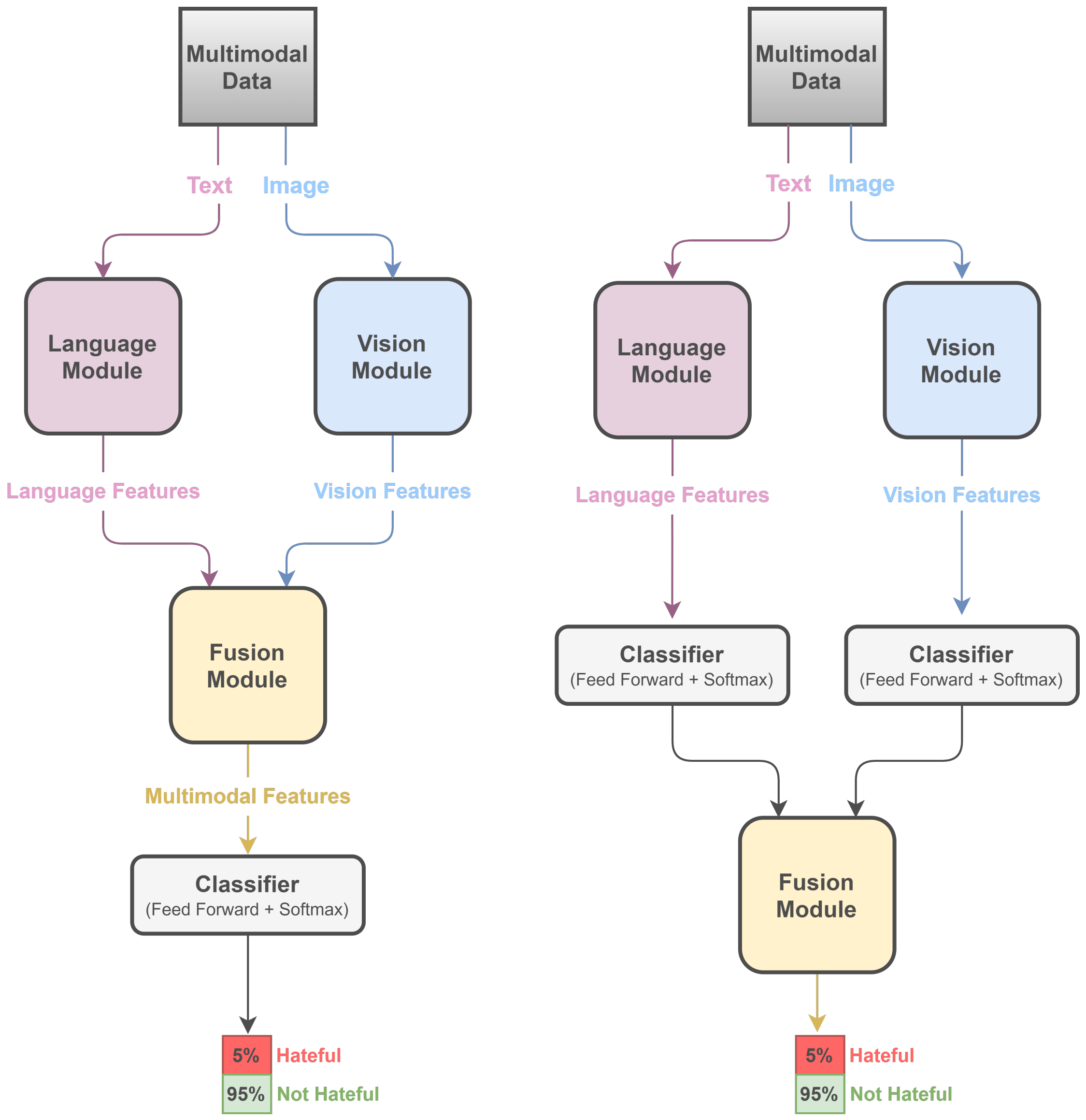


Figure 2.8: Generic representation of a multimodal model for vision-and-language classification tasks. In mid-fusion (left), both input modes pass through their own modules after which their features are fed to the fusion module, where the joint representation is produced and fed to a classifier which outputs class probabilities. In late fusion (right), class probabilities are computed per modality and the final decision is made by the fusion module.

the case of classification tasks, but any other task, e.g. regression, can be addressed in the same way.

After processing the two input modes, e.g. vision and language, we need to somehow combine both of the features from the two different modalities. In other words, we need to *fuse* their features in some way. There are multiple methods in multimodal fusion and the key characteristic of these models is the stage at which the features from different modalities are merged: close to the input data (early fusion), at the decision level (late fusion) or in between (middle fusion).

Early fusion (also known as feature-level fusion) refers to combining the information at the level of raw features. Late fusion (also known as decision-level fusion), on the other hand, processes each stream independently of others and then combining the output of the two. Middle fusion is where the features from different modalities are fused in the middle of the workflow. However, in literature, the terms early fusion and middle fusion are used interchangeably.

Figure 2.8 illustrates the generic representation of V&L model for binary classification task, where the task is to classify the memes (multi-modal data source) as *hateful* or not *hateful*. Both in middle and late fusion we process both text and image input with their respective modules, Language Module and Vision Module, respectively, which produce the respective input's features. In mid-fusion, these features are passed to the Fusion Module after which the generated multimodal features are passed through a classifier to output class probabilities. Whereas in late-fusion, processed features are directly passed through two disjoint classifiers (weights are not shared) to output class probabilities which are then passed to Fusion Module where the final decision is made.

Figure 2.9 shows an example for each mid-level and late fusion models. Note that the building designs of the models are derived from the general architecture shown in Figure 2.8. Memes are the multimodal data source with the image and the text they contain. For Language Module (in red) we used fastText's word embeddings which were mapped to 300-dimensional space with the help of a feed-forward layer. For Vision Module (in blue), images are encoded by ResNet-152 and mapped to the same vector space as language features. In the mid-level fusion model we used one of the simplest approach for the fusion module: concatenating both features and passing them through a feed-forward layer. Finally, a classifier is applied to the processed multimodal features. In the late fusion model, on the other hand, we directly pass the extracted features through corresponding classifiers which generate class probabilities. The final decision is made by the fusion module which simply takes the mean of the class probabilities. The mid-level concat fusion model is implemented and is available in the GitHub repository[2].

2 `https://github.com/rizavelioglu/hateful_memes-hate_detectron`

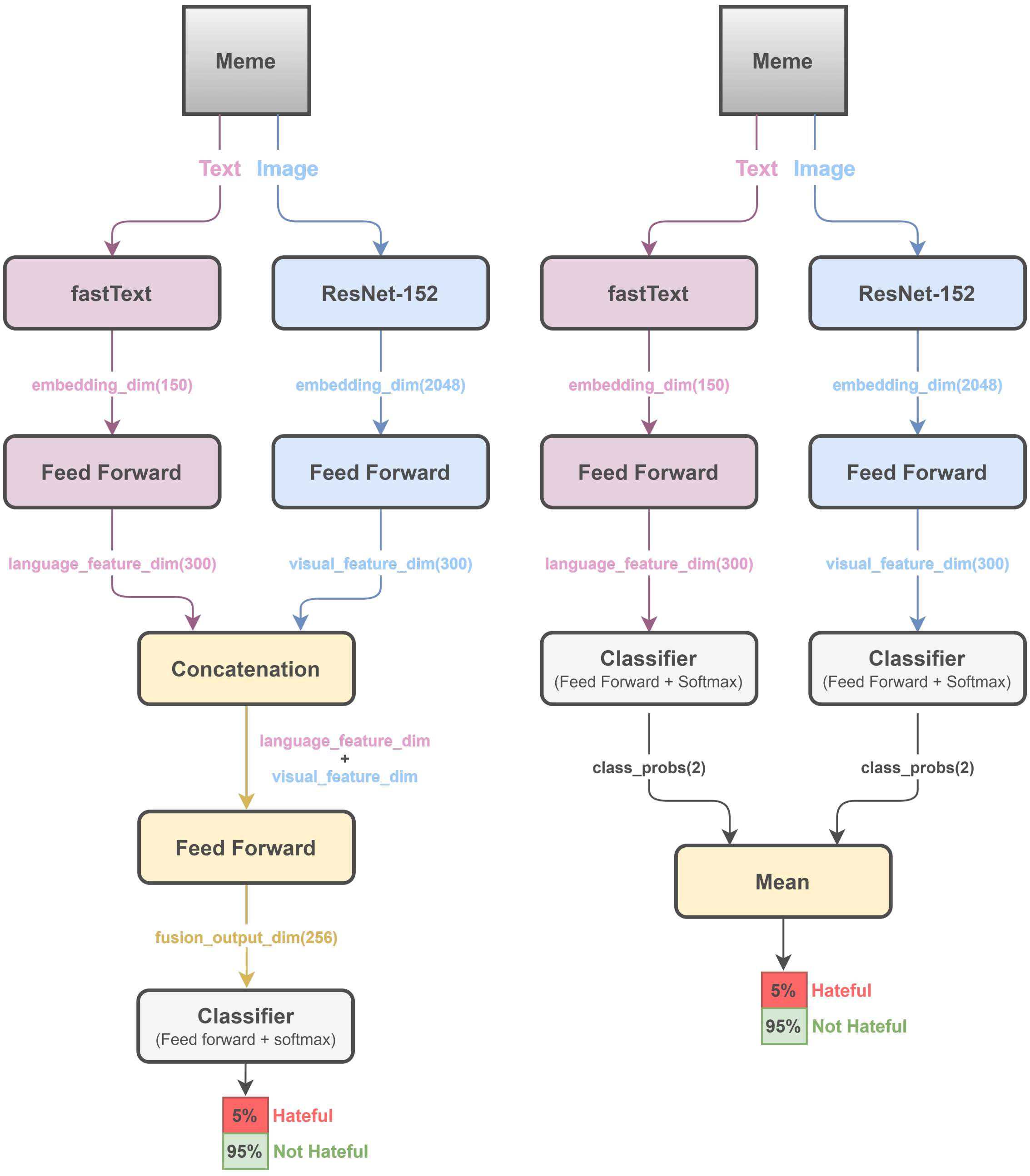


Figure 2.9: Mid-level concat-fusion model architecture (left) and late fusion model architecture (right).

3 MULTIMODAL RESEARCH

"A long-term objective of artificial intelligence is to build multimodal neural networks—AI systems that learn about concepts in several modalities, primarily the textual and visual domains, in order to better understand the world."

— ILYA SUTSKEVER[86]

In this chapter, we explain what (multi-)modality is and compare other definitions given in the field (Section 3.1). Then, we give a quick overview of the history of multimodal research and when it is introduced into the field of deep learning (Section 3.2). We then briefly summarize multiple fusion methods (Section 3.3) and, lastly, we focus on multimodal research in vision-and-language (V&L): we present the tasks, datasets, models, and their training (Section 3.4).

3.1 MODALITY AND MULTIMODALITY

In the field of media and communication, multimodal learning refers to the concept that various types of content and media can be used to improve the learner experience. Multimodal learning has the assumption that the learner can achieve a greater understanding of a concept when different learning modes are successfully used together. But, what exactly is a mode? A mode is a resource, something that conveys information. A mode could be textual (language), visual (image, video), aural (speech), symbolic (shape), etc. For example, the word "stop" denotes a term, but not necessarily a "context". When we add color (writing the word in red), the meaning becomes more clear. But when we add shape (putting the word "stop" in an octagon), then it represents a stop sign. Besides, the essence, or the context of the stop sign can still be transmitted even though one of these modes is absent or unavailable, for example, if someone is colorblind, or does not speak the language. This is the essence of multimodal learning: different modes collaborate to construct context, each contributing in its own unique way.

In the field of Machine Learning (ML), the terms *mode* and *modality* are used interchangeably and interpreted differently. For example, [7] defines multimodality as:

> Our experience of the world is multimodal – we see objects, hear sounds, feel texture, smell odors, and taste flavors. Modality refers to the way in which something happens or is experienced.

Whereas [32] defines it as:

> In the representation learning area, the word modality refers to a particular way or mechanism of encoding information.

On the other hand, [64] argues that the former definition relies on the human perceptual experience (human-centered) and the latter one focuses on the state in which the information is encoded before being processed by an ML system (machine-centered). They state that task-agnostic human- and machine-centered definitions fail to describe the term *multimodality* for multimodal ML and they propose a task-relative definition of *(multi)modality* as:

> A machine learning task is multimodal when inputs or outputs are represented differently or are composed of distinct types of atomic units of information.

They indicate that if a modality cannot be mapped unambiguously into one another by an algorithm, it is said to be an atomic unit of information. For example, if we are given natural images and images of text for a task, then the task is unimodal as both of the input modes are represented as the same atomic unit of information that is pixel values in stacks of matrices. However, if we use Optical Character Recognition (OCR) to obtain the text from the images of text, we are left with text and natural images which makes the task multimodal. In this work, we adopt the previously presented definition of multimodality by [64].

3.2 MULTIMODAL DEEP LEARNING

Research in multimodal learning dates back to the 1970s. In 1976, a paper in the field of psychology and linguistics affected computer scientists. The paper proposed the so-called *McGurk Effect* [57]. It is a perceptual phenomenon that shows how hearing and vision interact in speech perception: when the auditory component of one sound is combined with the visual component of another sound, the illusion occurs, causing the impression of a third sound. This study showed that one of the reasons why the speech recognition models failed at the time was, perhaps, that the models were unimodal. Although the demonstration of the need for multimodality led researchers to create a new field for multimodal speech recognition (audio-visual speech recognition), multimodal learning was not given the attention it deserved. With the Deep Learning era, from the 2010s and on, multimodal research did not gain importance right away. First, DL achieved state-of-the-art in single modalities where the task is unimodal. For instance, CNNs, which work with images, achieved SoTA accuracy for unimodal CV tasks whereas Transformer-like neural network architectures (e.g. BERT), which work with text, have played a game-changing role in unimodal NLP tasks. Only in the past few years, multimodal research has gained big popularity in the field. One of the few reasons for this delay, very similar to the delay of the DL era, are (1) new large-scale multimodal datasets (e.g. COCO, Conceptual Captions), (2) compute power where the models are trained in a parallelizable fashion, and (3) algorithmic advances in representation learning.

Since the key reason for multimodality is to use multiple information from different modalities to have a better representation, one could assume that adding more modalities would yield in a better performance. This may not necessarily hold under some conditions. A study [94] showed that in practice the best unimodal network performs better for video classifications. They identify two causes for this performance drop: first, multimodal networks are prone to overfitting. Second, training different modalities jointly with a single optimization strategy is not optimal because different modalities could overfit and generalize at different rates. They address these problems with a technique that weighs the individual modalities' loss and the joint loss. As the paper indicates, for some tasks or some dataset a single modality might be sufficient, however, for some other task or a dataset[1] multimodality is required. The key factor to achieve success in multimodal deep learning is how the information from different modalities are combined, in other words, how the features from different input modes are *fused*.

1 One such dataset is the Hateful Memes Dataset which is specifically built for testing the true multimodality, which we will discuss in Chapter 5.

3.3 MULTIMODAL FUSION

The goal of multimodal learning is to achieve a more robust, performant model using different modalities. While the features[2] of various modalities may have redundant information, for some tasks it might be complementary to use different modalities. First, learning from multimodal sources allows capturing the correspondences between modalities and a deeper understanding of natural phenomena. Second, having access to multiple modalities enables us to capture complementary information that is not visible in either of the modalities alone. Third, even if one of the modalities is absent, a multimodal system will still function. For example, identifying emotions from a visual signal when the individual is not speaking [19], also recall the 'stop sign' example presented in Section 3.1. The crucial factor to form multimodality is to integrate the features of modalities meaningfully. Multimodal fusion, in technical terms, is the process of combining data from several modalities and is one of the most studied aspects of multimodal machine learning, with works dating to 1980s [111].

As discussed briefly in Section 2.2 and illustrated in Figure 2.8, the vast majority of multimodal fusion has been done using early (i.e. feature-based), late (i.e. decision-based), and hybrid fusion approaches. Early fusion combines features as soon as they are extracted, mostly by concatenating the features and it has been used for product classification [112]. Late fusion, on the other hand, comprises integration after each of the modalities has made a decision. The decision is done by a fusing mechanism such as voting schemes [59], averaging [79]. One drawback of late fusion is that the low-level interaction between the modalities is discarded. Hybrid fusion seeks to combine the benefits of both of the aforementioned approaches into a common framework. It has been successfully used to identify multimodal speakers [100] and for large-scale video classification [37]. Also, neural networks have been used extensively for multimodal fusion [60, 92]. More recently, attention-based Transformer-like models gained significant popularity. However, the research areas multimodal deep learning and multimodal fusion are too involved for further discussion, therefore, we only focus on two modalities that our work is based on, that is, vision and language. We guide the reader to the surveys [7, 32, 115] for an in-depth analysis on multimodal learning and fusion.

3.4 VISION AND LANGUAGE

Significant advances in image processing and language understanding attracted substantial attention in the past few years based on the rapid development of deep learning algorithms. Recent years have seen considerable improvement in V&L tasks, with CNN and RNN fusion-based models increasingly improving the state-of-the-art on benchmarks. The tasks include visual-and-language navigation [3], Visual Question Answering (VQA) [1, 26], visual-based referred expression understanding and phrase localization [40, 110, 68], image and video captioning [38, 93, 36, 104, 62, 108], text-to-image generation [107, 74, 106]. In all of these tasks, we have a multimodal data source that provides image and text. As presented in Section 2.2 in Figure 2.8, we have memes as the multimodal data source and both modalities are processed individually to extract features, or embeddings, or a distributed representation. A distributed representation is a vector that distributes information related to a concept with several components, meaning that elements can be tuned separately to allow for the efficient encoding of more

2 Following [8] we use the term feature and representation interchangeably.

concepts in a comparatively low-dimensional space [9]. Symbolic representations, such as one-hot encoding, may be compared to such representations. The term "embedding" is used in deep learning to describe a mapping from a single-hot vector describing a word or image category to a distributed representation of real-valued numbers.

Text embeddings can be acquired from a Language Model (LM) that uses the chain rule for predicting the probability of a text sequence [9]. RNN based LMs (e.g. LSTM, GRU) have been used extensively for text embeddings. However, self-attention-based architectures, especially Transformers [90], have emerged as the model of choice in NLP. Pre-training on a large text corpus and then fine-tuning on a smaller task-specific dataset is the dominant method as introduced with BERT [22], which we will explain in detail in Section 5.2.

In computer vision, however, CNN architectures remain dominant and the image embeddings can be acquired from final layers of models like AlexNet [45], GoogLeNet [88], and ResNet [34] which won the ImageNet Large Scale Visual Recognition Competition for image classification in 2012, 2014, and 2015, respectively. Alternatively, object detection models may be used to extract visual embeddings which provide more precise semantic relationships with the associated labels from selected regions. Such models are known as the *R-CNN family*: Region-based CNN (R-CNN)[28], Fast R-CNN [27], Faster R-CNN [75], and Mask R-CNN [35]. Most recently, the success of Transformer in NLP inspired the researchers in computer vision: they apply a regular Transformer directly to images, with the fewest possible changes. The so-called Vision Transformer (ViT) [23] divides an image into patches and the linear embeddings of those patches are passed as input to a Transformer, as it is normally done with tokens (words) in an NLP application. ViT approaches or beats state of the art on multiple image classification tasks. Pyramid Vision Transformer (PVT) [95] broaden the scope and the impact of ViT by overcoming the difficulties of porting Transformer to various dense prediction tasks. Another model for dense prediction tasks is the Dense Vision Transformers (DVT) [72] where using a convolutional decoder, they combine tokens from different stages of the vision transformer into image-like representations at various resolutions and gradually combine them into full-resolution predictions. The Swin Transformer [52] is a new computer vision backbone that can be used for a variety of tasks. It uses a hierarchical Transformer whose representation is computed using a shifted windowing mechanism that restricts self-attention computation to non-overlapping local windows while still enabling cross-window communication. Convolutional Vision Transformer (CvT) [98] introduces convolutions for ViT to get the best of both approaches.

3.4.1 *Models*

Two major categories of architectures have arisen from the recent visio-linguistic pre-training approaches: single- and dual-stream architectures. Before passing through the transformer layers, single-stream architectures project and convert both visual and textual embeddings into a joint embedding space. Dual-stream architectures, on the other hand, extract the embeddings separately through different transformers and the resulting representations are fed into cross-modal transformer (`TRM`) layers. We will explain in detail the model we conducted our research in Section 5.3. Next, we give some example models for the two mentioned architecture designs and summarize the main differences between the models.

Examples of single-stream models are VisualBERT [49], OSCAR [50], UNITER [18]. VisualBERT is a BERT model with multiple `TRM` blocks. Features of both input modes' are concatenated and passed as input to the model in a similar fashion as BERT, but twice as wide. OSCAR makes use of the object tags detected in images as anchor points to learn semantic alignments between images and text, which improves the learning of cross-modal representations. They concatenate image and text features as well as the object tags' features, that is when the tags are removed from the input OSCAR reduces to VisualBERT. UNITER alters the pre-training objective of the former models and introduces a new objective.

Examples of dual-stream models are ViLBERT [54], LXMERT [89], ERNIE-ViL [109]. ViLBERT is made up of two paralleled BERT transformer streams that are linked by `TRM` block layers. The visual input is handled by one stream of `TRM` blocks while the linguistic input is handled by the other. LXMERT proposes a multi-component design for the cross-modality model and uses extended pre-training tasks (i.e. RoI-feature regression and image question answering). ERNIE-ViL introduces three new pre-training tasks using a scene graph parser.

According to recent findings [16], the discrepancies between different "Vision and Language BERTs" are largely due to training data and hyper-parameters. They discovered that random initialization has a major impact on the performance of the V&L BERTs in both pre-training and fine-tuning. They also discovered that when models are trained with the same hyper-parameters and data, they perform similarly. In particular, some models outperform others but they found that single- and dual-stream model families are on par.

3.4.2 *Training*

In visio-linguistic self-supervised pre-training both images and text are used. The model is trained to predict some hidden (masked) part of the input, whether it is a part of an image or a word from the text, using a pre-training proxy task with a self-supervised objective. Large image captioning datasets such as VQA 2.0 [30], COCO Captions [17], Conceptual Captions (CC) [78] has long been a popular pre-training dataset since they include comprehensive descriptions of images that can be used to learn task-agnostic and generic language grounding in images. Finally, the model is fine-tuned on a downstream task in an end-to-end fashion by replacing the pre-trained network's head with task-specific heads, e.g. classification head.

4 HATEFUL MEMES CHALLENGE AND DATASET

In this chapter, we present the Hateful Memes Challenge and the competition details in Section 4.1. Then, we do an in-depth analysis of the dataset in Section 4.2. Lastly, we present the baseline models for unimodal models, as well as for multimodal models with various degrees of sophistication and analyse their performance in Section 4.3.

4.1 THE COMPETITION

The competition was held based on the Hateful Memes Dataset at NeurIPS (2020). The winners were chosen based on their results on a separate "unseen" test set, which we will describe in Section 4.2. The "unseen" test set was created explicitly for testing solutions using new source content, ensuring that the participants would be evaluated on the actual task, which would in the real world contain entirely novel unseen examples, and also to reduce the possibility of participants leveraging inadvertent biases. The competition was divided into two phases: a first phase using the seen test set, which ran from May to October 2020 with one submission allowed per day; and a second phase using the unseen test set, which ran from October to November with three submissions allowed in total. For more information about the competition, please visit the website at https://hatefulmemeschallenge.com/.

4.1.1 *Task Formulation*

Hate speech is specifically described in the context of the challenge as follows:

> A direct or indirect attack on people based on characteristics, including ethnicity, race, nationality, immigration status, religion, caste, sex, gender identity, sexual orientation, and disability or disease. We define attack as violent or dehumanizing (comparing people to non-human things, e.g. animals) speech, statements of inferiority, and calls for exclusion or segregation. Mocking hate crime is also considered hate speech.

In this definition, there are some notable but subtle exceptions, such as attacking individuals/famous people if the attack is not based on any of the protected characteristics. Often, targeting hate groups (such as terrorist organizations) is not considered hate. This implies that detecting hate speech can also necessitate subtle world knowledge. The term parallels (but is a somewhat condensed version of) community standards on hate speech employed by Facebook[1].

The task is to classify a meme, which consists of an image and some text (the text is pre-extracted from the image to avoid having to use optical character recognition) based on whether or not it is hateful according to the above description.

1 https://www.facebook.com/communitystandards/hate_speech

Table 4.1: Hateful Memes Dataset characteristics

	Total	Not-hate	Hate	MM Hate	UM Hate	Img Conf	Text Conf	Random
Train	8500	5481	3019	1100	1919	1530	1530	2421
dev seen	500	253	247	200	47	100	100	53
dev unseen	540	340	200	200	0	170	170	0
test seen	1000	510	490	380	110	190	190	130
test unseen	2000	1250	750	750	0	625	625	0

4.1.2 *Metrics*

The primary metric for the competition is the area under the receiver operating characteristic curve (AUROC) [13]. The community is encouraged to report the accuracy as a secondary metric as it is simple to interpret and the development and test sets are not wildly unbalanced, so accuracy provides a fair signal of model success. The competition winners were decided based on AUROC, which provides a fine-grained sense of classifier performance.

4.2 DATASET

The dataset construction procedure is discussed in detail in [42]. In summary, it consists of four steps: 1) data filtering; 2) meme reconstruction; 3) hatefulness ratings; 4) benign confounder construction. The dataset is created in such a way that it encourages and measures truly multimodal understanding and reasoning of the models. A key point to achieve this objective is the so-called "benign confounders" (also known as *contrastive* [25] or *counterfactual* [39] examples) which addresses the risk of exploiting unimodal priors by models. For instance, a system might pick up on accidental biases where appearances of words like "black" or "white" might be strongly correlated with hate speech. To compete with this problem, and to make the dataset extra challenging, benign confounders are collected: for every hateful meme, there are alternative images or text that flip the label to not-hateful. Such image and text confounders require multimodal reasoning to classify the original meme and its confounders correctly because using just the text, or just the image, is going to lead to poor performance. Thus, benign confounders make the dataset challenging and appropriate for testing the true multimodality of a model.

The Hateful Memes dataset is not created for training models from scratch, but to fine-tune and test large-scale, pre-trained multimodal models. Thus, the size of the dataset (10K memes) is small compared to datasets that are used for pre-training such as Visual Genome (108K) [44], COCO (200K) [17], and Conceptual Captions (3.3M) [78]. Table 4.1 shows how the dataset breaks down into various categories and the size of each set – train, test, and development. Different sets were used in the competition: in phase 1, dev seen and test seen were used, and in phase 2 (the prize-winning phase), dev unseen and test unseen sets were used. One reason why different sets used in the final phase was the possibility of reverse-engineering the labels in test seen: as the participants could make 1 submission per day, one could gather quite useful information about the labels in the set. Thus, a new unseen set was generated. In addition, participants were allowed to make only three submissions in the final phase, which makes the exposition of the labels very challenging.

Table 4.2: Textual lexical analysis: Most frequent non-stopwords in the combined ‘dev seen‘ and ‘test seen‘ sets.

MM Hate	UM Hate	Text Conf	Img Conf	Not-Hate
like (0.05)	people (0.14)	like (0.05)	like (0.06)	want (0.08)
white (0.04)	like (0.07)	love (0.05)	dishwasher(0.05)	think (0.05)
people (0.04)	get (0.07)	people(0.05)	one (0.05)	get (0.05)
black (0.04)	i’m (0.05)	day (0.04)	get (0.04)	know (0.05)
get (0.04)	muslims(0.05)	time (0.04)	i’m (0.04)	people(0.05)
one (0.04)	black (0.05)	one (0.04)	way (0.03)	like (0.05)
dishwasher (0.03)	white (0.05)	take (0.03)	white (0.03)	take (0.04)
i’m (0.03)	us (0.03)	world (0.03)	islam (0.03)	always(0.04)
know (0.03)	america(0.03)	man (0.02)	black (0.03)	say (0.04)
back (0.02)	go (0.03)	look (0.02)	gas (0.03)	trump (0.04)

Table 4.3: Visual lexical analysis: Most frequent Mask-RCNN labels in the combined ‘dev seen‘ and ‘test seen‘ sets.

MM Hate	UM Hate	Text Conf	Img Conf	Not-Hate
person (3.43)	person (2.27)	person (3.33)	person (2.13)	person (2.11)
tie (0.16)	tie (0.21)	tie (0.18)	bird (0.16)	tie (0.23)
car (0.10)	cell phone (0.08)	dog (0.12)	chair (0.12)	cup (0.12)
chair (0.10)	dog (0.07)	book (0.12)	book (0.11)	chair (0.09)
book (0.07)	car (0.07)	car (0.11)	car (0.09)	car (0.07)
dog (0.07)	bird (0.06)	chair (0.08)	cup (0.08)	cell phone (0.06)
cell phone (0.05)	chair (0.05)	sheep (0.05)	dog (0.08)	dog (0.05)
handbag (0.05)	tennis racket (0.05)	cell phone (0.05)	bowl (0.08)	bottle (0.03)
sheep (0.04)	cat (0.03)	bottle (0.04)	sheep (0.08)	bed (0.03)
bottle (0.04)	cup (0.03)	knife (0.04)	bottle (0.07)	teddy bear (0.03)

Hateful memes can be multimodal in nature, meaning that the classification relies on both modalities, or unimodal, meaning that one modality is enough to obtain the correct classification label. Therefore, the dataset comprises five different types of memes: *multimodal hate*, where benign confounders were found for both modalities, *unimodal hate*, where one or both modalities were already hateful on their own, *benign image* and *benign text* confounders and lastly *random not-hateful* examples. Table 4.1 demonstrates how the dataset is divided into different categories. Unlike the train set, which is dominated by unimodal hate samples, dev seen and test seen sets are dominated by multimodal contents. In addition, the label distribution is balanced. For phase 2, unseen dev and unseen test sets were constructed such that there are no unimodal violating contents within.

Table 4.2 shows an analysis of the lexical (word-level) statistics of the seen sets: the top 10 most-frequent words by class and their normalized frequency. We observe that certain words used in dehumanizing based on gender (e.g. equating women with "sandwich makers" or "dishwashers") and colors ("black" and "white") are frequent. We also observe that these words are also frequent in the benign image confounder category which means that these words are not necessarily directly predictive of the label. In unimodal hate category, which is almost always text-only, we notice that the language is stronger and often targets religious groups (e.g. "Muslims").

Interpreting the properties of the visual modality is more challenging. However, we do the same analysis but this time for bounding box labels gathered from the object detector (Mask R-CNN [35]). The results are shown in Table 4.3.

4.3 BENCHMARKING MULTIMODAL CLASSIFICATION MODELS

The baseline scores are taken from [42] where they established the scores for diverse unimodal and several state-of-the-art multimodal models on the task. In what follows, we first introduce the baseline models and then present the results.

4.3.1 *Models*

A variety of models are evaluated belonging to one of the three following classes: unimodal models, multimodal models that were *unimodally* pre-trained, and multimodal models that were *multimodally* pre-trained. For example, when a pre-trained BERT model and a pre-trained ResNet model are combined in some way and pre-trained using a language model objective, then the model is unimodally pre-trained. On the other hand, if a multimodal objective[2] (e.g. visually-grounded language model objective) is used during pre-training, then it is multimodally pre-trained.

Two image encoders are evaluated: 1) **Grid** features: ResNet-152 [34] convolutional features from `res-5c` with average pooling, 2) **Region** features: features from `fc6` layer of Faster R-CNN [75] with ResNeXt-152 [103] as its backbone. The Faster R-CNN is trained on Visual Genome [44] and features from `fc6` layer are fine-tuned using the weights of the `fc7` layer (**Image-Region**). For the textual modality, the unimodal model is BERT (**Text BERT**).

In total, ten different models are evaluated including simple fusion methods as well as more sophisticated multimodal models. As discussed in Section 2.2, one of the simplest approaches is to process each modality on their own and taking the mean of the unimodal output scores (**Late Fusion**). Figure 4.1 shows the whole process, using ResNet-152 for the visual modality and BERT for the textual modality. Another simple fusion method is to concatenate both input modes' features and training a classifier on top (**Late Fusion**). Figure 4.2 shows the whole process where ResNet-152 and BERT features for vision and language modules are used, respectively. These can be compared to more complex multimodal methods such as supervised multimodal bi-transformers [41] using either Image-Grid or Image-Region features (**MMBT-Grid** and **MMBT-Region**), and versions of ViLBERT [54] and VisualBERT that were only unimodally pre-trained and not pre-trained on multimodal data (**ViLBERT** and **VisualBERT**).

Lastly, we have models that were pre-trained on a multimodal objective before fine-tuned on the downstream task: the official multimodally pre-trained versions of ViLBERT (trained on Conceptual Captions [78], **ViLBERT CC**) and VisualBERT (trained on COCO, **VisualBERT COCO**).

A grid search hyperparameter tuning over the learning rate, batch size, warm up and number of iterations is performed and the results are averaged over three random seeds, together with their standard deviation.

4.3.2 *Results*

The results for 'seen' and 'unseen' sets are shown in Table 4.4 and Table 4.5, respectively.

For seen dataset, both unimodal models, vision-only classifier (**Image-Region**) and text-only classifier (**Text BERT**), achieve the lowest scores, as expected. Though, **Text**

2 also known as multi-objective

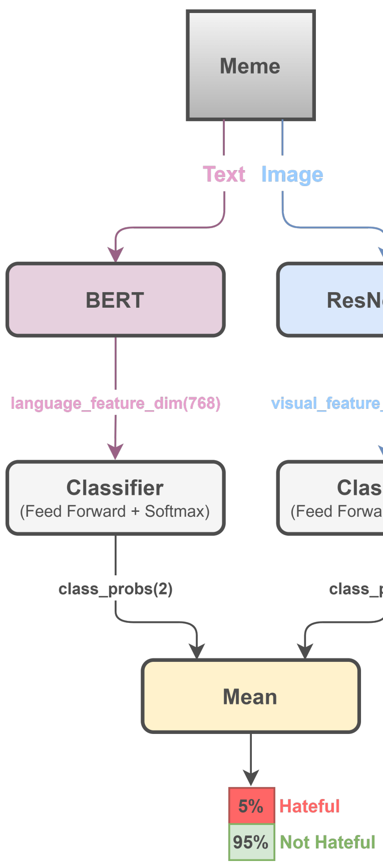


Figure 4.1: Late Fusion: input data modes pass through their respective modules and output scores. The mean of both scores is taken and the maximum class probability is selected.

Table 4.4: Seen dev and test set performance for baseline models.

		Seen Dev		Seen Test	
Type	**Model**	Acc.	AUROC	Acc.	AUROC
	Human	-	-	84.70	82.65
Unimodal	Image-Region	52.66	57.98	52.13±0.40	55.92±1.18
	Text BERT	58.26	64.65	59.20±1.00	65.08±0.87
Multimodal (Unimodal Pretraining)	Late Fusion	61.53	65.97	59.66±0.64	64.75±0.96
	Concat BERT	58.60	65.25	59.13±0.78	65.79±1.09
	MMBT-Grid	58.20	68.57	60.06±0.97	67.92±0.87
	MMBT-Region	58.73	71.03	60.23±0.87	70.73±0.66
	ViLBERT	62.20	71.13	62.30±0.46	70.45±1.16
	VisualBERT	62.10	70.60	63.20±1.06	71.33±1.10
Multimodal (Multimodal Pretraining)	ViLBERT CC	61.40	70.07	61.10±1.56	70.03±1.77
	VisualBERT COCO	**65.06**	**73.97**	**64.73±0.50**	**71.41±0.46**

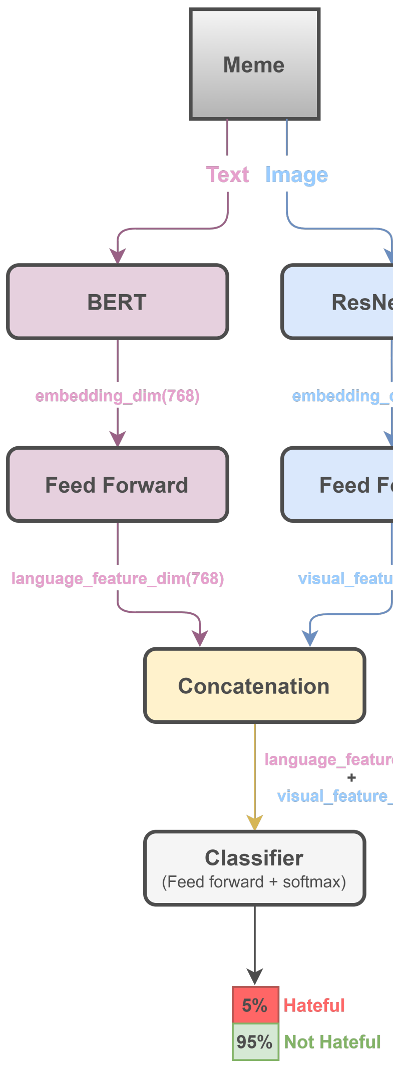


Figure 4.2: Mid-level Concat Fusion: input data modes pass through their respective modules after which their features are concatenated and passed through a feed-forward layer. The multimodal features are then passed through a classifier.

Table 4.5: Unseen dev and test set performance for baseline models.

		Unseen Dev		**Unseen Test**	
Type	**Model**	Acc.	AUROC	Acc.	AUROC
Unimodal	Image-Region	61.48	53.54	60.28±0.18	54.64±0.80
	Text BERT	60.37	60.88	63.60±0.54	62.65±0.40
Multimodal (Unimodal Pretraining)	Late Fusion	61.11	61.00	64.06±0.02	64.44±1.60
	Concat BERT	64.81	65.42	65.90±0.82	66.28±0.66
	MMBT-Grid	67.78	65.47	66.85±1.61	67.24±2.53
	MMBT-Region	70.04	71.54	70.10±1.39	72.21±0.20
	ViLBERT	69.26	72.73	70.86±0.70	73.39±1.32
	VisualBERT	69.67	71.10	**71.30±0.68**	73.23±1.04
Multimodal (Multimodal Pretraining)	ViLBERT CC	70.37	70.78	70.03±1.07	72.78±0.50
	VisualBERT COCO	**70.77**	**73.70**	69.95±1.06	**74.59±1.56**

BERT outperforms **Image-Region** by a large margin, demonstrating the strength of textual signal. In addition, **Text BERT** is on par with some unimodally pre-trained multimodal models, e.g. **Late Fusion**. Nevertheless, multimodal models outperform unimodal models in general. However, the difference between unimodally/multimodally pre-trained models' performance is relatively small, indicating that multimodal pre-training can be improved. Another indication that there is much room for improvement is the human accuracy: with 84.7% it is still much better than the best multimodal model, that is **VisualBERT COCO**, which is pre-trained on COCO caption dataset and achieves an AUROC score of 71.41, with an accuracy of 64.73.

For unseen dataset, **Text BERT** outperforms **Image-Region** by a smaller margin, unlike for seen set. Multimodal models outperform unimodal models in every setting. Moreover, the more advanced the fusion method, the better the model performs: the scores increase from top to bottom in the table, so as the complexity of fusion methods. However, the difference between unimodally/multimodally pre-trained models' performance is relatively small, similar to seen set. Moreover, the best performing model in terms of accuracy on the unseen test set is unimodally pre-trained **VisualBERT**. Even though multimodally pre-trained **VisualBERT COCO** outperforms the unimodally pre-trained **VisualBERT**, this still indicates the need for improvement in multimodal pre-training.

Also, we see that the score for each model on unseen set is improved compared to the scores for seen set. One reason for this performance boost may be the re-annotation of the training set. As is to be expected with a dataset of this size and nature, some of the examples in the training set were misclassified. Such discrepancies were mostly a consequence of noisy examples from the initial reconstruction process and annotator uncertainty. For the second phase of the competition, this was resolved by having the entire dataset re-annotated with improved training and tighter guidelines.

5 METHODOLOGY

In this chapter, we introduce the prize-winning solution to the Hateful Memes Challenge [42]. First, we explain how the challenge dataset is expanded to increase the size of the training set and analyse thoroughly those additional data in Section 5.1. Second, we describe how both input modes are encoded and prepared for training in Section 5.2. Third, we present the multimodal model we used in our work in Section 5.3. Lastly, we reveal the full training process in Section 5.4.

5.1 DATASET EXPANSION

In recent years, there has been a trend in NLP systems to use pre-trained language representations in increasingly flexible and task-agnostic ways for downstream tasks. Starting with task-specific architectures using word vectors [58, 67], then RNNs with multiple layers of representations [20, 56], and more recently pre-trained transformer language models [90] have been fine-tuned directly, obviating the need for task-specific architectures completely [22, 70]. This has resulted in significant progress on a wide range of tasks such as question answering, textual alignments, and many more. Yet, a major limitation to this approach preserved its existence, that is, the need for task-specific dataset and task-specific fine-tuning. That is to say, to achieve strong performance on the desired task typically requires fine-tuning on a dataset of thousands to hundreds of thousands of examples specific to that task. One possible approach to resolving this issue is meta-learning [15] – which in the context of the language models means that the model learns a wide range of skills and pattern recognition abilities during training and then uses those abilities at inference to easily adapt to or recognize the desired task at hand. However, this is beyond the scope of this thesis, hence, readers are guided to [15]. Furthermore, we are given such a fine-tuning dataset, that is Hateful Memes Dataset Section 4.2, and no need to deal with the aforementioned issue. Though, we still seek to expand the dataset to maintain stable training.

In Transfer Learning, pre-training plus fine-tuning paradigm, there are several properties fine-tuning dataset should have, but the two most important ones are the size of the new dataset – fine-tuning set, and its similarity to the original dataset – pre-training set where the model is pre-trained on. For the former, we expanded the dataset as more data delivers stable learning and achieves better performance. For the latter, we chose a data source that is similar to the pre-trained dataset in terms of the content of images. As a result, we expanded the training dataset by 428 additional samples. Although the number of samples added is not large, we observe a significant improvement in performance which indicates that the added samples have good *quality*, which indeed conforms to the common knowledge acquired in the field: quality matters more than the quantity and high-quality data yields a better performing model. In fact, [80] empirically showed that with the right pre-training and fine-tuning dataset match along with good quality, training on a smaller dataset can easily outperform training on a larger dataset.

Unused Data from Competition

As shown in Table 4.1 there are 500 samples in seen dev and 540 samples in unseen dev. By comparing the memes by their IDs in both set, we identified 400 overlapping samples:

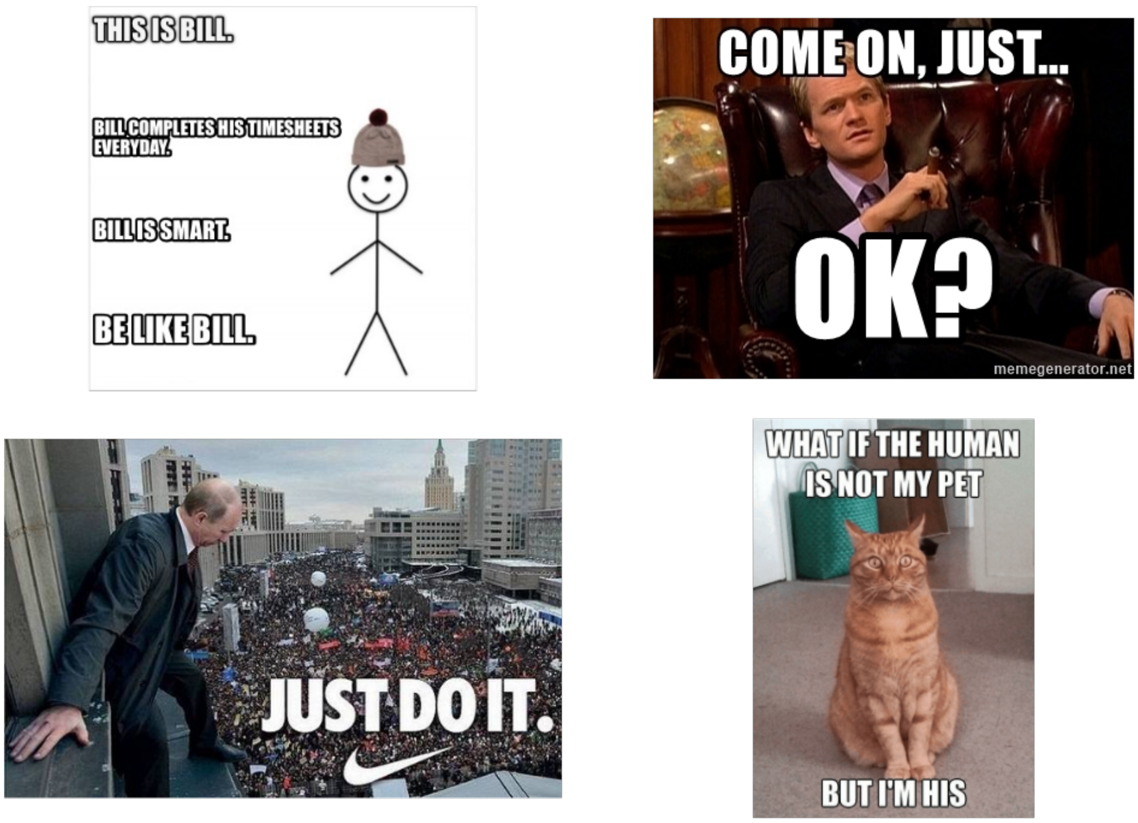


Figure 5.1: Four samples from the Memotion Dataset. A text-only meme **(top-left)** which we avoid adding because multi-modality is not required for classification of such memes. A wrongly labeled meme **(top-right)**: originally labeled as 'very_offensive' but re-labeled as not hateful. A wrongly labeled meme **(bottom-left)**: originally labeled as 'not_offensive' but re-labeled as hateful. A wrongly labeled meme **(bottom-right)**: originally labeled as 'very_offensive' but relabeled as not hateful.

$|\{dev\_seen\} \cap \{dev\_unseen\}| = 400$, which means that there are 100 samples that are not in dev unseen: $|\{dev\_seen\} \setminus \{dev\_unseen\}| = 100$. There is no information given about those left-out samples by the competition holders. Moreover, we were allowed to use dev and train set as train set, when needed. Therefore, we added those samples to the training set and evaluated the trained model on dev unseen.

Memotion Dataset

The Memotion Dataset [77] was released as part of the SemEval[1] 2020 task. The dataset consists of 6,992 memes which are collected through Google Images searching for specific categories. A total of 52 popular categories such as *Hillary Clinton, Donald Trump, Minions*, etc. were identified. The dataset was annotated through Amazon Mechanical Turk (AMT). AMT workers annotate the *emotion* class labels as *Humor, Sarcasm, Offensive, Motivation* and measure the intensity with which a specific impact of a class is conveyed, along with the overall *sentiments*: very negative, negative, neutral, positive, very positive. For example, a meme can be annotated as: "negative, not_funny, very_twisted, hateful_offensive, not_motivational". Table 5.1 shows the number of samples for each label in the dataset.

1 SemEval is a series of international NLP research workshops to advance the current state-of-the-art in semantic analysis.

Table 5.1: Characteristics of the Memotion dataset.

Category	Tag	Quantity
Sentiment Analysis	very_negative	151
	negative	480
	neutral	2201
	positive	3127
	very_positive	1033
Humor	not_funny	1651
	funny	2452
	very_funny	2238
	hilarious	651
Sarcasm	not_sarcastic	1544
	general	3507
	twisted_meaning	1547
	very_twisted	394
Offensive	not_offensive	2713
	slight_offensive	2592
	very_offensive	1466
	hateful_offensive	221
Motivation	not_motivational	4525
	motivational	2467

To ensure the annotation quality each meme was annotated five times and the final annotations are decided based upon the majority voting scheme. Each data sample consists of a meme and a text that was extracted from the image using Google vision OCR. Due to the inaccuracy of OCR, some noise could be induced to the data. For that reason, AMT workers were asked to correct the extracted text. We use the corrected text in our experiments.

We added all the "hateful_offensive" (221 samples) and "very_offensive" (1466 samples) to the training data as hateful memes and added "not_offensive" (2713 samples) ones as non-hateful memes. We discarded the "slight_offensive" memes because it might confuse the model. We then fine-tuned our models on this aggregated data (hateful memes + memotion) but the results even worsen. After an exploratory analysis of the dataset, we discovered that some of the samples are wrongly labeled, i.e. a hateful meme labeled not hateful and vice versa but most importantly the structure of the memes were different than the ones in the hateful memes dataset. For example, there are memes consisting of only text or an illustration of a sketch. One such example can be seen in Figure 5.1, top-left. Therefore, we manually re-labeled a part of the dataset using a script: we randomly iterate through the whole dataset and re-label the samples that are similar to the ones in the hateful memes dataset considering the meme style and the context of the image. Figure 5.1 shows 3 re-labeled samples which are then aggregated to the training data. After cherry-picking and re-labeling the "similar" memes, we added 328 new memes to the training data. Both the labeling script and the re-labeled data samples are available at the GitHub repository[2].

2 https://github.com/rizavelioglu/hateful_memes-hate_detectron/tree/main/utils

5.2 IMAGE AND TEXT ENCODING

There are a variety of methods to encode image and text. For textual input, some methods use simple statistics to convert the input mode into a vector representation (e.g. bag-of-words, tf-idf) while others use more complex mechanisms such as word-vectors (e.g. Word2Vec [58], GloVe [66], fastText [11]) or Transformer encoder architecture (e.g. BERT). Whereas for visual input, CNNs are used in some way to gather image features. Next, we present how we encoded both input modes.

Text Encoding

Transfer learning — pre-training a neural network model on a known problem, such as ImageNet, and then fine-tuning using the learned neural network as the basis of a new purpose-specific model — has been demonstrated numerous times in the field of computer vision. Those pre-trained networks have shown a groundbreaking performance in different areas of computer vision. We have also seen a similar approach in the field of NLP with the methods like Word2Vec, GloVe, fastText but with the release of BERT, a beginning of a new era has been marked in the field. BERT has shown superior performance compared to other embeddings methods like Word2Vec and GloVe and has become state-of-the-art in many NLP tasks. The main reason for this is that the other methods are context-independent, which means that they produce only one vector (embedding) for each word, integrating all of the word's different meanings into a single vector. For example, given the two sentences: "I paid the money straight into my *bank*.", and "A ferry had a crash on the east *bank* of the lake", the approaches like Word2Vec and GloVe would fail to capture that the word *bank* has different meanings in those two sentences and would return the same embedding for that word. BERT, on the other hand, is context-dependent, which means that each of the two words will have a different embedding because BERT considers the words around it while producing the embeddings. This is thanks to the bidirectional training of the model.

In comparison to other methods, which looked at a text sequence from left-to-right or a combination of left-to-right and right-to-left training, BERT looks at all the text input simultaneously. The findings [22] show that bidirectionally trained language models may provide a greater understanding of language context than single-direction language models. The bidirectional training alleviates the previously mentioned unidirectionality constraint by using a Masked Language Model (MLM) pre-training objective. The masked language model masks some tokens from the input at random, with the intention of predicting the original masked word based on its context. Unlike pre-training a left-to-right language model, the MLM objective allows the representation to fuse the left and right context, allowing to pre-train a deep bidirectional Transformer.

Figure 5.2 illustrates how MLM works given a text input *"My dog likes playing"*. The input is tokenized using WordPiece algorithm [99] which breaks a word into several sub-words (i.e. 'playing' to 'play', and '##ing'), allowing the model to represent frequently seen sub-words as well. This is done to deal with the out-of-vocabulary (OOV) problem: since the model is pre-trained with a fixed vocabulary, it is likely that certain tokens in a new data will not appear in the pre-trained model's fixed vocabulary. Those OOV words are replaced with a special token `[UNK]`, which stands for unknown token. Converting all unseen tokens to `[UNK]`, however, eliminates a lot of information from the input. As a result, WordPiece algorithm makes it possible to represent any word as a set of its individual characters, thereby, prevent the explosion of the `[UNK]` token. Besides, we have additional special tokens: `[CLS]`, `[PAD]`, and `[MASK]`. For classification tasks,

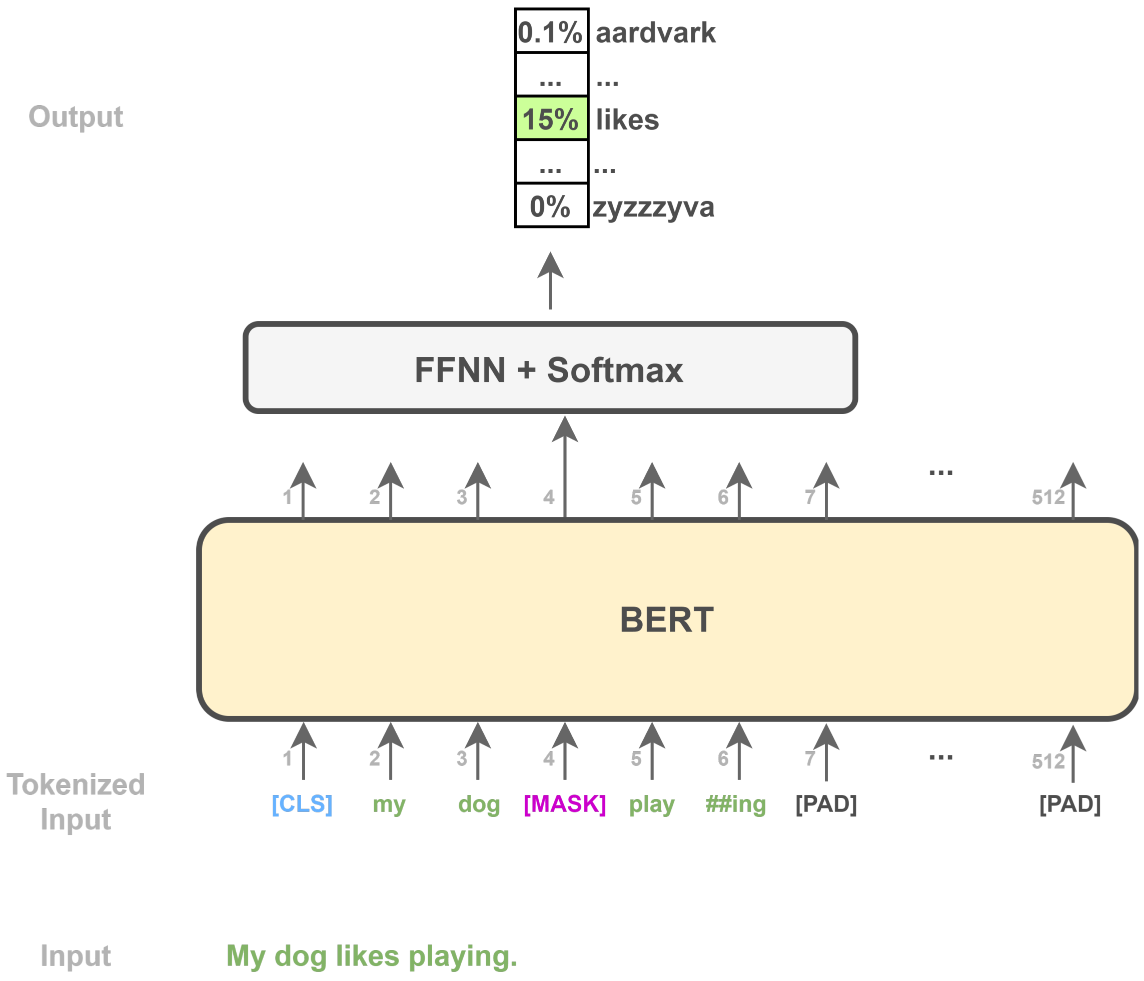


Figure 5.2: Illustration of the Masked Language Modeling (MLM) objective used in pre-training BERT for a given input.

a single vector representing the entire input sentence must be fed to a classifier. The final hidden state corresponding to `[CLS]` token (first token of BERT) is used as the aggregate sequence representation. BERT has a maximum processing capacity of 512 tokens at a time. To make up for sentences that are shorter than this maximum length, paddings (empty tokens) are added to those sentences. The `[PAD]` token is used in the original implementation to differentiate the paddings from other tokens. At last, `[MASK]` token is used for the tokens that are selected at random and masked for the MLM objective. While these modifications make it possible to perform a bidirectional pre-trained model, the `[MASK]` token does not appear during fine-tuning, resulting in a mismatch between pre-training and fine-tuning. To prevent this, `[MASK]` token is not always used to replace *masked* words. 15% of the token positions are sampled at random for prediction. The *i*-th token is substituted with (1) the `[MASK]` token 80% of the time, (2) a random token 10% of the time, (3) the unchanged *i*-th token 10% of the time. The output of the corresponding masked token (4-th token in Figure 5.2) is used to predict the masked word from a class of all English words.

To boost BERT's ability to manage multiple sentence relationships another objective, in addition to MLM, called Next Sentence Prediction (NSP) is used to jointly pre-train text-pair representations. In particular, when selecting sentences `A` and `B` for each pre-

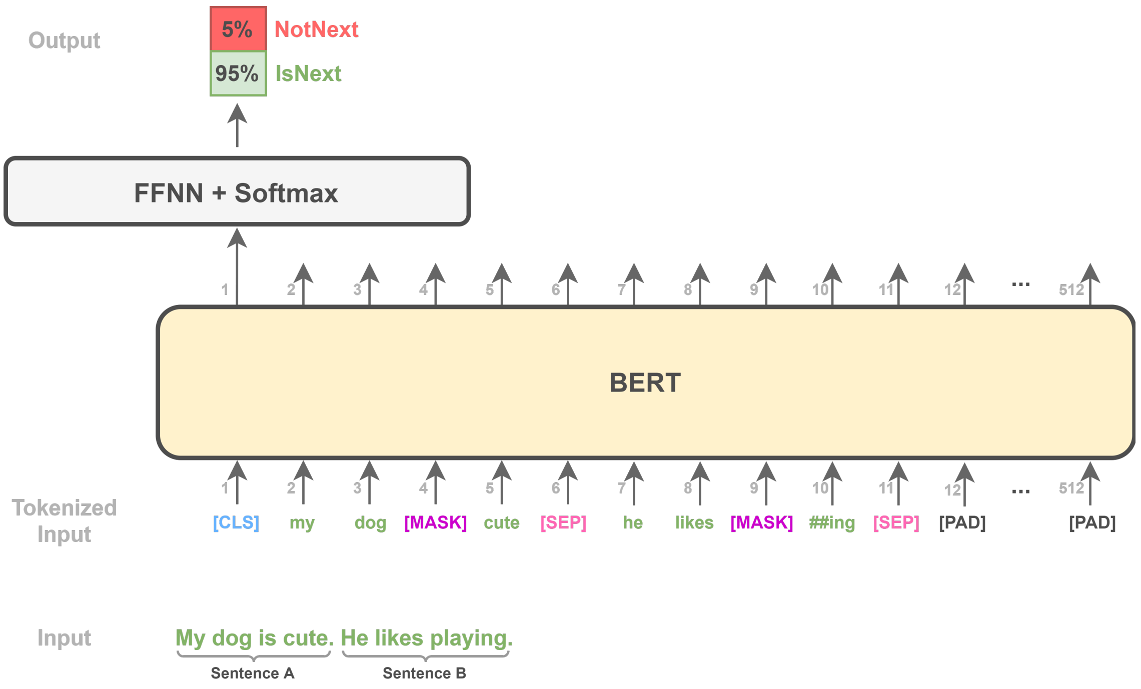


Figure 5.3: Illustration of the Next Sentence Prediction (NSP) objective used in pre-training BERT for a given input.

training example, 50% of the time `B` is the actual text sentence after `A` (labeled as `IsNext`), and 50% of the time is a random sentence from the corpus (labeled as `NotNext`). As we show in Figure 5.3, the `[CLS]` token is used for NSP task. Another special token `[SEP]` is used to indicate the separation between the two sentences.

Based on the original implementation of Transformer [90], BERT's model architecture is a multi-layer bidirectional Transformer encoder. As also explained in Section 2.1, Transformer is made up of two separate mechanisms: an encoder that reads the text input and a decoder that generates the prediction for the task. Particularly, only the encoder mechanism is used in BERT because the aim is to generate a language model, where the model encodes the text input and outputs features that can be used in various NLP tasks. BERT has two variants: **BERT$_{\text{BASE}}$** and **BERT$_{\text{LARGE}}$**, where the latter one is the larger version of the former in terms of the number of `TRM` blocks. Figure 5.4 shows an illustration of the two models. Furthermore, Figure 5.5 depicts the internal steps in an encoder and illustrates the architecture of BERT$_{\text{BASE}}$ in 3D.

In our approach, we used **BERT$_{\text{BASE}}$** model to encode textual input. Specifically, we used the pre-trained **BERT$_{\text{BASE}}$** model provided by the HuggingFace Transformers library [97].

Image Encoding

One of the most important advances following the introduction to deep learning and attention mechanisms to multi-modal vision and language research was the discovery of "bottom-up" attention [2]. Unlike normal attention, which focuses on particular parts of the visual input using 'top-down' linguistic inputs, bottom-up attention uses pre-trained object detectors to recognize relevant regions based on the visual input. Consequently, instead of using vanilla grid convolutional feature maps from Convolutional Neural

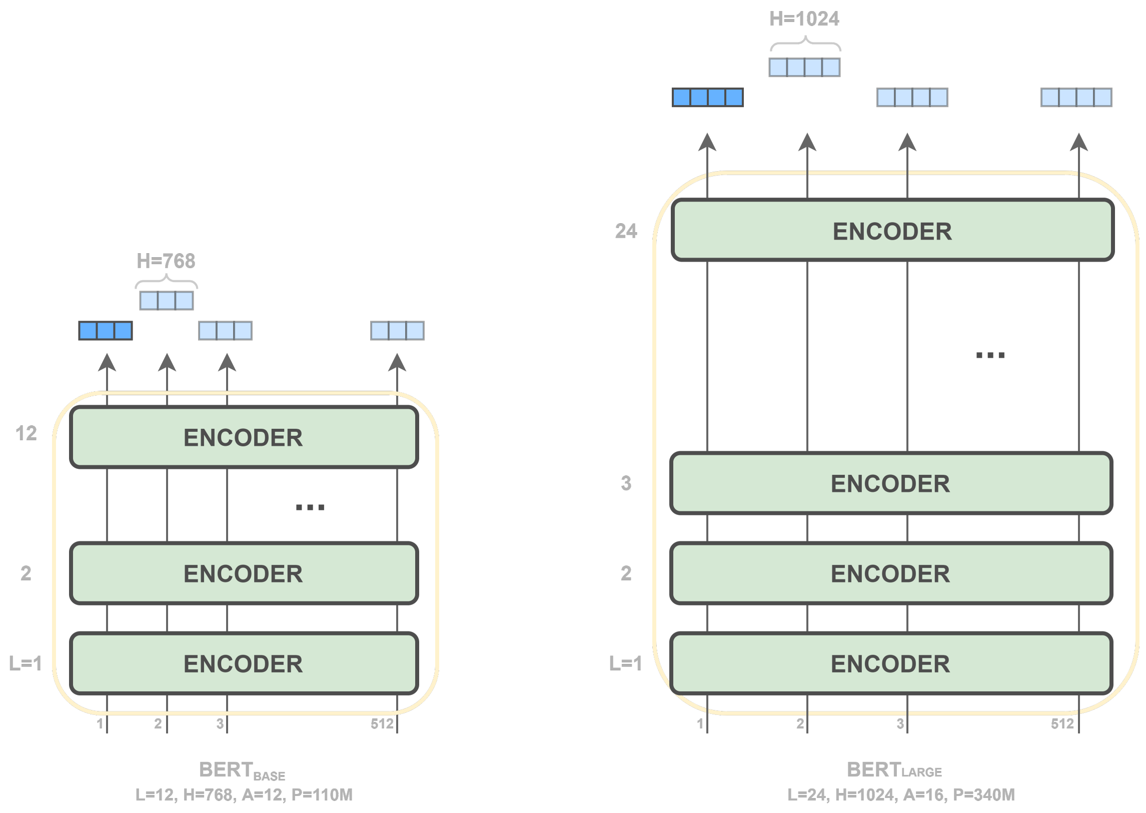


Figure 5.4: Illustration of two BERT models: **BERT$_{\text{BASE}}$** and **BERT$_{\text{LARGE}}$**, where we denote the number of layers (i.e. Transformer blocks or `TRM` blocks) as L, H as hidden size, A as the number of self-attention heads, and P as total number of parameters.

Networks (CNN), images are represented by a set of bounding box or region-based features.

Region-Based Convolutional Neural Networks (R-CNN) are a family of machine learning models for computer vision and specifically for object detection. The family of R-CNNs gained a huge population and has become the most prominent models in the field. The original purpose of R-CNN [28] is to take an input image and output a collection of bounding boxes, each of which contained an object as well as the object's category (e.g. animal or human). R-CNN independently computes the features on each of as many as two thousand regions-of-interest (ROI) (also known as 'region proposals' [28]), which is time-consuming. To make R-CNN faster, the training procedure is enhanced by running the model once on the whole image, resulting in Fast R-CNN [27]. While both R-CNN and Fast R-CNN used Selective Search [24] to come up with ROIs, Faster R-CNN [75] incorporates the ROI generation into the model itself which speeds up the process. Lastly, Mask R-CNN [35] extends previous versions of R-CNN to pixel-level image segmentation. Since pixel-level segmentation involves much more fine-grained alignment than bounding boxes, Mask R-CNN replaced ROIPooling with a new method called "ROIAlign" such that ROIs can be precisely mapped to the regions of the image.

For every image we extract 100 boxes of 2048$\mathcal{D}$ region-based image features from a `fc6` layer of a ResNeXT-152 based Mask R-CNN model [35], trained on Visual Genome [44] with the attribute prediction loss following [2]. Figure 5.6 shows an example of a processed image where the ROIs are proposed by Mask R-CNN. We project the visual embeddings into the textual embedding space before passing them through the trans-

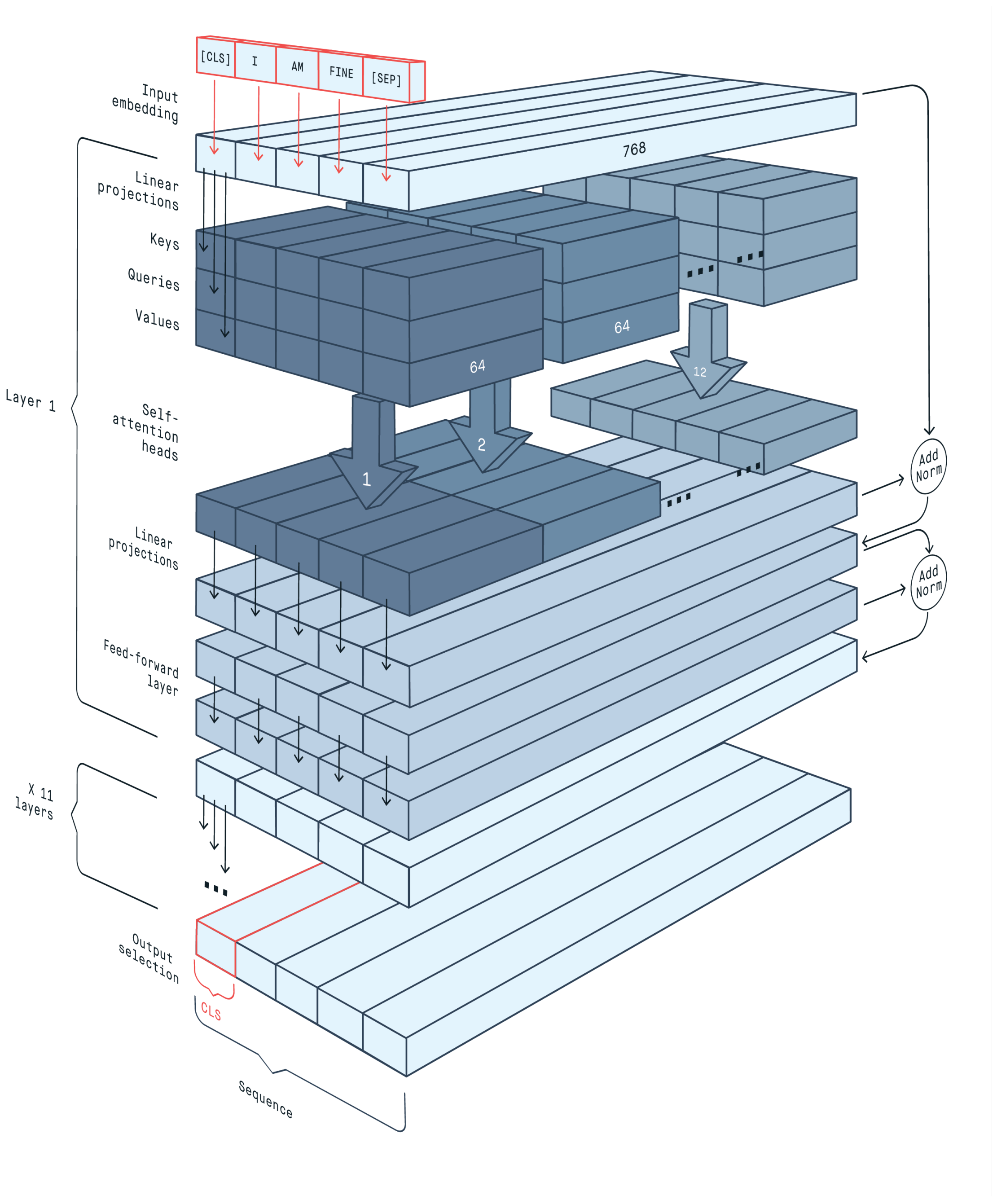


Figure 5.5: Visualization of $\textbf{BERT}_{\textbf{BASE}}$ in 3D (Brinne 2020) [14]

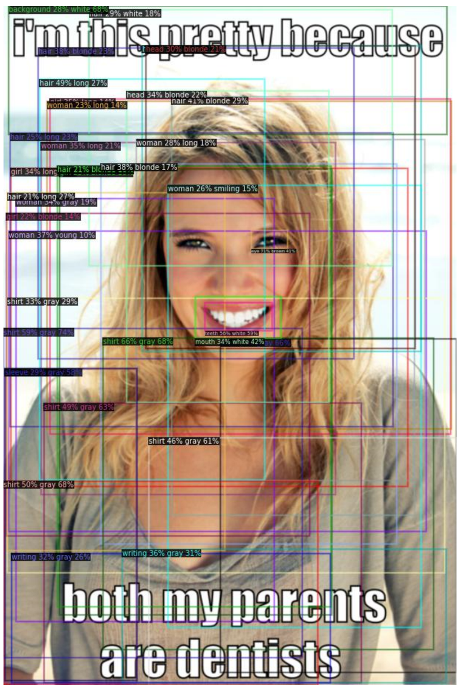


Figure 5.6: An example of a processed image where the regions are proposed by Mask R-CNN. Originally 100 boxes are extracted per image but for plotting purposes only 36 boxes are shown.

former layers. We learn weights $W_n \in \mathbb{R}^{PxD}$ to project each of the 100 image embeddings to D-dimensional token input embedding space:

$$I_n = W_n f(img, n), \tag{5.1}$$

where $P = 2048$, $D = 768$, and $f(\cdot, n)$ is the output of the n-th fully-connected layer in the image encoder.

5.3 MODEL – VISUALBERT

VisualBERT [49] is designed for capturing rich semantics in the image and associated text. Due to its simplicity and flexibility, it is used as a baseline for V&L tasks including VQA 2.0 [30], VCR [114], NLVR [85], Hateful Memes Challenge [42]. VisualBERT integrates a single-stream BERT model [22], with multiple Transformer blocks [90], and pre-trained object proposal systems such as Faster R-CNN [75] or Mask R-CNN [35]. Particularly, image features extracted from object proposals are treated as unordered input tokens and fed into the model along with the text, in a similar fashion as BERT but twice as wide. The concatenated text and image inputs are then jointly processed by multiple Transformer layers in VisualBERT. Hence, VisualBERT is also known as the 'BERT with vision for vision-and-language tasks'. The whole process is illustrated in Figure 5.8 which is derived from the general architecture seen in Figure 2.8.

After concatenation of both input modes' features, VisualBERT's input looks like:

$$\texttt{[CLS]},\ l_1,\ \ldots,\ \texttt{[MASK]},\ l_M,\ \texttt{[SEP]},\ v_1,\ \ldots,\ v_N$$

Similar to BERT, l_i represents a textual input token, `[MASK]` and `[SEP]` represent the masked input and separator tokens used in self-supervised pre-training, and v_i represents

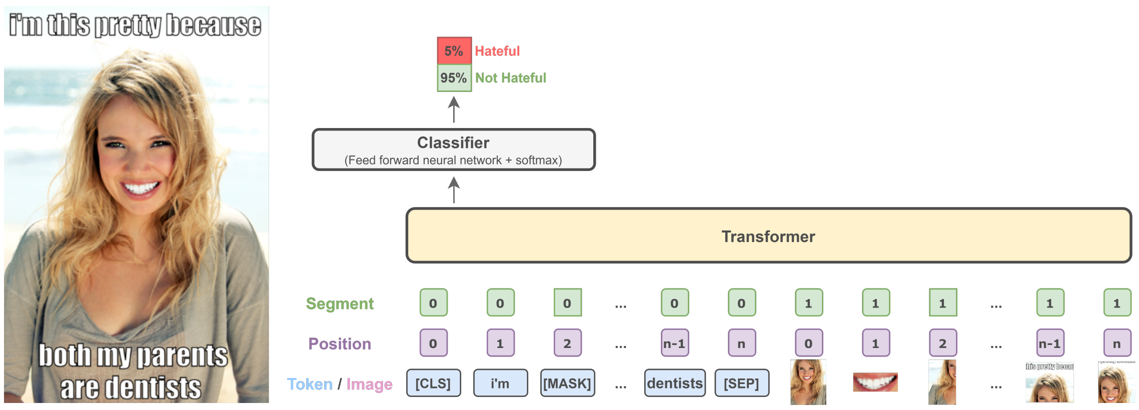


Figure 5.7: An example meme sampled from the dataset (left), and an illustration of the model (right). Image regions' embeddings and textual embeddings are concatenated and processed jointly with a Transformer to allow the self-attention to discover implicit alignments between language and vision.

an object embedding extracted from the image via object detector. This joint input is passed through the transformer blocks and the final representation corresponding to `[CLS]` token is used in downstream tasks. The BERT-like model architecture is illustrated in Figure 5.7. Image/Text embeddings are computed by adding image region/token embeddings, image/token positional embeddings (Position), and a specific embedding which distinguishes the image and text embeddings (Segment).

VisualBERT is trained in a very similar fashion as BERT but the model would have to learn to handle both language and visual input. Hence, VisualBERT is pre-trained on COCO dataset [17] which consists of around 200K image-caption pairs. VisualBERT is pre-trained using two visually-grounded objectives: masked language modeling with image and caption-image alignment prediction. Both of the objectives are very similar to objectives used while training BERT: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). VisualBERT's objectives extend BERT's objectives to accommodate both language and visual input. The former objective is the same as MLM but with vision: some tokens in text input are masked and the model learns to predict the masked tokens from the rest of the text and visual context. The latter objective is also inspired by NSP but with vision again: the model is given an image along with two captions where one of the captions is describing the image while the other has a 50% chance to be another corresponding caption and a 50% chance to be a randomly drawn caption. The model then needs to distinguish the two cases.

5.4 TRAINING

Recent advances in representation learning for natural language processing have led to drastic improvements in text-only classification problems. While other approaches are catalyzing the development of the task, BERT has been the most effective and the most successful one by far. Following BERT's success, diverse model architectures have been proposed including multimodal ones; such as ViLBERT [54], VisualBERT [49], LXMERT [89], VL-BERT [84], OSCAR [50], UNITER [18], and many more. All these models have very similar training scheme: pre-training on intermediary or proxy multimodal tasks before fine-tuning on the multimodal task at hand. Pre-training is performed on large image captioning datasets like COCO Captions [17], Conceptual Captions [78] and

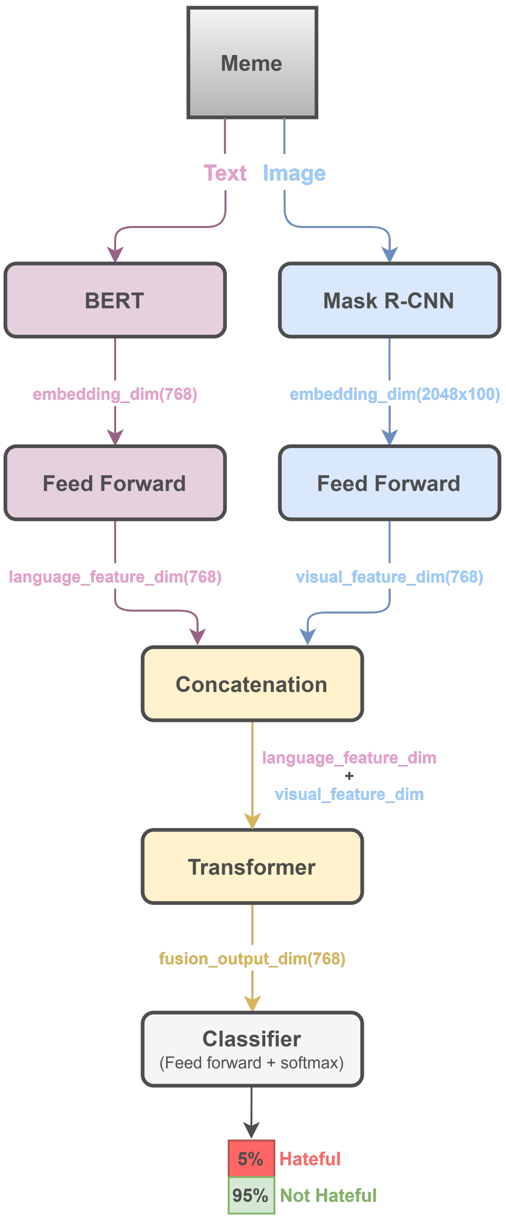


Figure 5.8: VisualBERT: input data modes pass through their respective modules after which their features are concatenated and passed through a Transformer. The multimodal features are then passed through a classifier to output class probabilities.

then transferred to a downstream task by fine-tuning the whole architecture end-to-end. Hence, this scheme is also known as Transfer Learning in the field of deep learning.

In this section, we explain how we leverage transfer learning: using a pre-trained model and adapting it to multimodal meme binary classification problem. First, we explain why we choose this specific pre-trained model and how it is trained in (Section 5.4.1). Then, in Section 5.4.2 we present how we adapt the higher-order feature representations in the base model to our task via fine-tuning. Lastly, we demonstrate how the classification is performed in Section 5.4.3.

5.4.1 *Pre-training*

We use MMF [81] which provides implementation of diverse multimodal models as well as their pre-trained weights for various datasets including Conceptual Captions [78], COCO [17], VQA 2.0 [30], and many more. The original implementation of VisualBERT is incorporated inside the framework ensuring no implementation differences. The default setting has the same configuration as BERT_{BASE}: 12 layers, a hidden size of 768, and 12 self-attention heads. The parameters of `TRM` layers are first initialized from pre-trained BERT_{BASE} weights provided by the HuggingFace Transformers library [97]. Region-based image features are extracted from `fc6` layer of a ResNeXT-152 based Faster R-CNN [75, 103] model trained on Visual Genome [44] with the attribute prediction loss following [2]. For BERT text encoder, the pre-trained BERT base-uncased weights provided by HuggingFace are used.

Pre-training datasets

Doing pre-training on large datasets reduces variance but increases the bias. Furthermore, increasing the dataset size does not always improve the performance. For instance, [80] observed that the dataset size is not the most important factor in visio-linguistic pre-training. They state that even a smaller dataset can easily outperform pre-training on a larger dataset when the right visual and textual domain match along with good quality data. For this reason, we use the models that were pre-trained on the following datasets, which we evaluate their performances by fine-tuning them on the challenge set and evaluating on seen dev set:

COCO Captions [17] Common Objects in Context [51] is a series of natural images from Flickr that depict everyday scenes. To advance the state-of-the-art in object detection and segmentation, COCO was introduced with bounding boxes, segmentation masks, and key-points for 80 common categories. COCO Captions [78] were later gathered to supplement multimodal AI development. COCO Captions comprises 200k labeled images, each with five captions, for a total of 1 million image-caption pairs.

Conceptual Captions [78] Conceptual Captions (CC) is a collection of 3.3 million image-caption pairs scraped from the internet by matching images with their alternative text. More precisely, images and their raw descriptions are harvested from the web, and therefore represent a wider variety of styles in terms of visual content. It is worth to mention that the whole pipeline is automated which extracts, filters, and transforms image-caption pairs with the goal of achieving a balance of cleanliness, informativeness, fluency, and learnability of the resulting captions.

VQA 2.0 [30] Understanding and reasoning about a picture to answer a question is a task of Visual Question Answering [1] (VQA). First, VQA 1.0 was released which collected COCO images [51]. VQA 2.0 was later implemented to counteract the language-biases caused by questions that could be answered without even looking at the picture (for example, "What color is the banana?"). For each question, complementary images were presented that were identical but had different responses to the same question. VQA 2.0 comprises 1.1 million questions based on 200k COCO images and there are 10 human-annotated answers to each question.

Pre-training objective

A similar training procedure as BERT is adopted but VisualBERT would have to learn to handle both textual and visual input with the previously stated image-caption datasets. Following [49], VisualBERT is originally trained with two visually-grounded language model objectives. (1) Masked language modeling with image and (2) Sentence-image prediction. The latter one is only possible with multiple image captions, which makes it impossible to be used with datasets like CC where there is a single caption. Therefore, only the former objective is used in pre-training VisualBERT model on different datasets.

Masked Language Modeling (MLM) Recall that we let image regions $\mathbf{v} = \{v_1, \cdots, v_N\}$ and let the input words $\mathbf{l} = \{l_1, \cdots, l_M\}$. The objective is to reconstruct $\mathbf{l}$ from a corrupted $\hat{\mathbf{l}}$ where some elements of text input are masked, i.e. replaced with a `[MASK]` token randomly with a probability p, whereas the vectors corresponding to image regions are not masked. Let θ be the trainable parameters. We minimize the negative log-likelihood:

$$\mathcal{L}_{MLM}(\theta) = -E_{(\mathbf{l},\mathbf{v})\sim\mathcal{D}} \log P_\theta(\mathbf{l} \mid \hat{\mathbf{l}}, \mathbf{v}) \tag{5.2}$$

5.4.2 *Fine-tuning*

As previously stated, we fine-tune several pre-trained VisualBERT models to find the best suitable pre-trained model for our task. We fine-tune the models on the Hateful Memes training set via back-propagation [48] and use binary cross entropy loss. We use the pre-extracted features[3] provided by the competition owners where they extract 2048$\mathcal{D}$ region based image features from `fc6` layer of a ResNeXT-152 based Faster R-CNN model [75, 103] trained on Visual Genome [49] with the loss following [2].

We evaluate every 100 updates and report the model with the best evaluation metric (AUROC) on the validation set (seen dev). We use the AdamW optimizer [43, 53] with cosine decay learning rate scheduler. For learning rate, it is common to use a small value for `TRM` weights that are being fine-tuned, in comparison to the randomly initialized weights for the linear classifier that computes the class probabilities. This is because we expect the `TRM` weights to be reasonably good and we do not want to distort them too much, especially while the classifier above them trained from random initialization. Therefore, we use a learning rate of 5e-5. We use batch size of 80 and set training update steps to 3000 for fine-tuning. There is a Jupyter notebook available in the GitHub repository[4] which includes all the code related to benchmarks. The results will be presented in Section 6.2.

3 Available at `https://dl.fbaipublicfiles.com/mmf/data/datasets/hateful_memes/defaults/features/features.tar.gz`

4 `https://github.com/rizavelioglu/hateful_memes-hate_detectron`

Downstream datasets

In addition to the pre-trained models, we also evaluate other models which are already fine-tuned on a downstream dataset. For example, pre-training a VisualBERT model on COCO and fine-tuning on a downstream dataset, which is then fine-tuned on the hateful memes dataset later. As the content of each dataset differs from one another, we hypothesize that a model that is trained on multiple datasets could leverage the diverse textual and visual domain available in the datasets used. Therefore, the model could learn more *general* features and could generalize and transfer its knowledge better.

SNLI-VE [102] The SNLI-VE (SNLI **V**isual **E**ntailment) dataset is built on top of SNLI [12] and Flickr30K [68] datasets. SNLI-VE is proposed for the Visual Entailment (VE) task [101] which aims to solve reasoning about the relationship between an image premise P_{image} and a text hypothesis H_{text}. Given an image as premise and a natural language sentence as hypothesis, the VE task involves classifying whether the statement is true (entailment), false (contradiction) or neutral with respect to the relationship conveyed by the (P_{image}, H_{text}). The dataset contains 550K image/statement pairs and evaluation is done using classification accuracy.

VizWiz [33] The VizWiz dataset consists of over 31K images collected by blind people who took a photo with their phone and reported a spoken question about it to address some of their daily needs. Each of the 32K questions are annotated by sighted human-annotators having 10 answers per visual question. The question-answering task from VizWiz is used as a downstream task and the VQA accuracy is used as the evaluation metric.

VQA 2.0 [30] We presented this dataset earlier as pre-training dataset, but it can also be used as a downstream VQA task in which for a given image and question pair (I, Q), an approach must predict an answer A, usually from a fixed-vocabulary. The VQA Accuracy metric[5] is used in the assessment.

5.4.3 *Classification*

The area under the receiver operating characteristic curve (ROC AUC) [13] has been selected as the measure of performance, which is given by the following formula:

$$AUROC = \int_{x=0}^{1} \texttt{TPR}(\texttt{FPR}^{-1}(x))dx \tag{5.3}$$

The labels indicating whether a meme is hateful or not-hateful are provided within the dataset, hence the task can be cast as a binary classification problem.

As illustrated in Figure 5.8, the input to the transformer is both the language and vision features. Then, we use the first output of the final layer (the vector corresponding to `[CLS]` token) as the input to a classification layer:

$$\texttt{clf}(x) = Wx + b \quad , \quad W \in \mathbb{R}^{DxC} \tag{5.4}$$

where D is the transformer dimensionality and C is the number of classes (also see Figure 5.7). We apply a softmax on the logits and train with binary cross-entropy loss.

5 For more details see: https://visualqa.org/evaluation.html

6 EXPERIMENTS AND RESULTS

In this chapter we first present the experimental setup, the hardware specifications used in the research environment, the setup for training in Section 6.1. Then we present the results in Section 6.2 and do a detailed analysis, both quantitative and qualitative in Section 6.3.

6.1 EXPERIMENTAL SETUP

We train our models with 4 NVIDIA RTX 2080Ti GPUs with 11GB memory each. We use the MMF[1] framework (also known as Pythia) [81] for our experiments that is based on PyTorch [65]. The framework provides several state-of-the-art model implementations as well as pre-trained models. We use the VisualBERT model that is pre-trained on Conceptual Captions [78]. We extract 2048$\mathcal{D}$ region based image features (100 regions per image) from `fc6` layer of a ResNeXT-152 based Mask R-CNN model [35], trained on Visual Genome [44] with the attribute prediction loss following [2]. We fine-tune the model on the aggregated (Hateful Memes + Memotion) training set via back-propagation [48] and use binary cross entropy loss. We evaluate every 50 updates and report the model with the best evaluation metric on the validation set. We use the AdamW optimizer [43], [53] with cosine decay learning rate scheduler. We set the parameters of AdamW as follows: ϵ as 1e-8 and (β_1, β_2) as (0.9, 0.999). We use a learning rate of 5e-5, batch size of 80 and set training update steps to 1000 for fine-tuning. The value of p, p_v, and p_l are set to follow masking probabilities as in original BERT paper [22].

We conducted an extensive hyper-parameter search and used the best configuration for the parameters including learning rate, warmup steps, warmup type, warmup factor, and warmup iterations. We evaluated the models with different hyper-parameters during the grid search and the top-27 models, according to their AUROC scores, are ensembled . Finally, the majority voting technique is applied to finalize the classification.

All the code is available at GitHub[2]. A Jupyter notebook is ready for use in the repository which loads the fine-tuned models and generates the predictions on a free, serverless Jupyter notebook environment provided by Google Colaboratory[3]. Another notebook is also provided in the repository where the whole pipeline –loading pre-trained models and fine-tuning– is documented in an end-to-end fashion with which the results could be reproduced.

6.2 RESULTS

In this section, we present our findings, results for various fine-tuned models in Section 6.2.1. Then, we show the results of ensemble learning in Section 6.2.2.

1 MMF: a framework for vision-and-language multimodal research from Facebook AI Research (FAIR). Available at `https://github.com/facebookresearch/mmf`

2 `https://github.com/rizavelioglu/hateful_memes-hate_detectron`

3 Google Colab: a free cloud service hosted by Google. Available at: `https://colab.research.google.com/`

Table 6.1: Performance of different pre-trained VisualBERT models on seen dev set after fine-tuned on the hateful memes training set with pre-extracted default features.

ID	Pre-trained on	Fine-tuned on	AUROC
1	CC_full	-	**73.58**
2	CC_fifty	-	73.21
3	CC_ten	-	72.98
4	VQA2_full	-	72.68
5	VQA2_fifty	-	72.07
6	VQA2_ten	-	67.33
7	VQA2_full	VizWiz	68.74
8	COCO_full	-	72.88
9	COCO_full	VQA2	71.55
10	COCO_full	Visual Entailment	73.18

6.2.1 *Fine-tuned models*

The results are shown in Table 6.1 where the scores are averaged over three random seeds. We have models pre-trained on CC, VQA 2.0, COCO, and on different proportions of them such as; fifty or ten percent. In other words, we clip the number of samples present in VQA 2.0 and CC to be the same as COCO for the sake of fair comparison. The chosen samples are randomly selected. In addition, as CC is significantly larger than COCO, we also experiment with various sizes of CC ranging from 10% (**CC_ten**), which is roughly the same size as COCO, to all of CC (**CC_full**). For example, the model with `ID=2` (**CC_fifty**) is pre-trained on the half of CC and achieves an AUROC score of 73.21. Similarly, the model with `ID=6` (**VQA2_ten**) is pre-trained on 10% of VQA 2.0 and achieves 67.33. Correspondingly, we note the models fine-tuned on downstream tasks with `ID ∈ {7, 9, 10}`. We discover that our hypothesis is proven to be wrong: the models that are further trained on downstream tasks not necessarily perform better. Consider the model with `ID=7` that achieved 68.74 AUROC and comparing it with its counterpart model (`ID=4`) which achieved a score of 72.68, it performed even worse.

Another example is the model fine-tuned on VQA 2.0 (`ID=9`) which consistently perform at-par with **COCO_full** (`ID=8`) but still rejecting our hypothesis. However, this issue does not persist for the model fine-tuned on VE (`ID=10`): we inspect that the model achieves a slightly higher score compared to **COCO_full** (`ID=8`), indicating that our hypothesis may hold under certain settings. To find out under which settings visio-linguistic representations are scalable, transferable, and task-agnostic, [80] made experiments with various settings and analyzed the empirical trends. They showed that if there is a discrepancy between the domain of pre-training and the downstream dataset, pre-training may not always function and that using in-domain pre-training datasets is preferable to using out-of-domain datasets. In this respect, we would expect that the models pre-trained on CC to perform better considering the diverse styles in terms of the visual content. The results meet our expectation: the models trained on different portions of CC with `ID ∈ {1, 2, 3}` perform perceivably better than all the other models. Therefore, we conducted our research on **CC_full**(`ID=1`) which received the highest AUROC score among the others.

6.2.2 *Ensemble Learning*

The role of supervised learning algorithms is to search through a hypothesis space [10] for a suitable hypothesis that will allow good predictions for a specific problem. Even if the hypothesis space includes hypotheses that are well-suited to a specific problem, finding a good one can be difficult. To form a stronger hypothesis, ensembles combine several hypotheses. The term "ensemble" refers to approaches that use the same base learner to produce several hypotheses. In other words, ensemble learning aims to enhance generalizability and robustness over a single model by combining the predictions of multiple base models. Specifically, we use the Majority Voting technique (also known as Hard Voting or Voting Classifier[4]), which combines various classifiers and predicts class labels using a majority vote. The resulting classifier is often used to balance out the shortcomings of several equally well-performing models. As a result, it performs better than any other model used in the ensemble.

On the dev unseen set, a grid search hyperparameter tuning over the learning rate, warmup factor, warmup iterations, warmup steps, and scheduler type is performed, which yielded multiple models with different AUROC scores. We set the number of updates to 2000 and evaluated every 50 steps. The base model is VisualBERT pre-trained on CC (previously mentioned **CC_full** in Table 6.1), and fine-tuning is done using our own extracted image features.

The top 27 models are chosen for ensemble learning after they are sorted by AUROC score (the number of models is chosen arbitrarily). Table 6.2 shows all the models in the ensemble with their hyper-parameter settings. The best model (`ID=60`) achieves 75.21 AUROC score where the worst model (`ID=26`) receives 73.68.

Predictions are collected from each of the models in the ensemble and the majority voting scheme is used, with the majority voted class determining the class of a data point. Furthermore, the probability of a data point being assigned to a class must be calculated to measure AUROC: if the majority voted for class 1 (hateful), the likelihood is the highest of all 27 models, and the lowest if the majority voted for class 0 (not hateful). Table 6.4 shows how the majority voting technique is applied. All the code, as well as the top-27 models, are available in the GitHub repository.

Finally, we evaluate our models by comparing them to the baseline approaches: Table 6.3 shows the results. Note that baseline models are fine-tuned using the image features provided by the competition holders which were extracted using Faster R-CNN, whereas our models are fine-tuned using our own image features extracted using Mask R-CNN. Also note that there is no entry for **VisualBERT CC**$_{\mathtt{ID=60}}$ for unseen test set. This is because participants were allowed to make maximum of three submissions and we did not use this specific model for submission. Next, in Section 6.3 we do an in-depth analysis and compare the results.

6.3 analysis

Table 6.3 shows both baseline and our models' performances where our two models are the best performing model in the ensemble (**VisualBERT CC**$_{\mathtt{ID=60}}$) and the ensemble itself (**VisualBERT CC**$_{ensemble}$). Our models improve the baselines significantly in both

4 Implementation of Voting Classifier from scikit-learn: `https://scikit-learn.org/stable/modules/ensemble.html#voting-classifier`

Table 6.2: The hyper-parameter settings for the best 27 models according to AUROC score on dev unseen after the grid search. The base model is VisualBERT pre-trained on CC with our image features.

ID	**lr ratio**	**warmup factor**	**warmup iterations**	**warmup steps**	**scheduler type**	**AUROC**	**Acc.**	**best iteration**
60	0.3	0.2	1000	2000	linear	**0.7521**	0.7093	750
1	0.3	0.3	1000	250	cosine	0.7516	0.6963	550
42	0.7	0.1	100	50	cosine	0.7502	0.7074	550
9	0.3	0.7	1000	500	cosine	0.75	0.7074	1000
0	0.6	0.3	1000	500	cosine	0.7497	0.7167	1000
57	0.6	0.3	1000	500	linear	0.7449	0.7037	700
23	0.3	0.05	500	250	cosine	0.7447	0.7037	550
20	0.6	0.3	1000	500	cosine	0.7444	0.7019	700
28	0.3	0.2	250	250	cosine	0.7437	0.7093	800
24	0.3	0.05	1000	250	cosine	0.7432	0.7019	450
27	0.3	0.1	1000	250	cosine	0.7427	0.7111	600
15	0.6	0.5	1000	250	cosine	0.7419	0.7	500
6	0.3	0.7	1000	250	linear	0.7402	**0.7185**	1200
22	0.3	0.05	250	250	cosine	0.7402	0.6833	550
34	0.3	0.1	250	250	linear	0.7402	0.7111	550
5	0.3	0.5	1000	250	linear	0.7401	0.6926	700
14	0.6	0.3	1000	250	cosine	0.7396	0.7111	450
41	0.3	0.05	500	500	cosine	0.7396	0.7037	450
43	0.7	0.1	250	50	cosine	0.7391	0.6963	650
52	0.7	0.3	250	50	linear	0.7391	0.6963	450
54	0.6	0.3	1000	250	cosine	0.7386	0.6926	550
16	0.6	0.7	1000	250	cosine	0.7381	0.7	500
31	0.3	0.05	250	250	linear	0.7381	0.7111	700
10	0.3	0.3	1000	500	linear	0.7378	0.7093	900
40	0.3	0.05	250	500	cosine	0.7376	0.7056	850
32	0.3	0.05	500	250	linear	0.7375	0.7093	800
26	0.3	0.1	500	250	cosine	0.7368	0.6981	550

Table 6.3: Model performances on unseen dev and unseen test.

		Unseen Dev		**Unseen Test**	
Type	**Model**	Acc.	AUROC	Acc.	AUROC
Baselines	Image-Region	61.48	53.54	60.28±0.18	54.64±0.80
	Text BERT	60.37	60.88	63.60±0.54	62.65±0.40
	Late Fusion	61.11	61.00	64.06±0.02	64.44±1.60
	Concat BERT	64.81	65.42	65.90±0.82	66.28±0.66
	MMBT-Grid	67.78	65.47	66.85±1.61	67.24±2.53
	MMBT-Region	70.04	71.54	70.10±1.39	72.21±0.20
	ViLBERT	69.26	72.73	70.86±0.70	73.39±1.32
	VisualBERT	69.67	71.10	71.30±0.68	73.23±1.04
	ViLBERT CC	70.37	70.78	70.03±1.07	72.78±0.50
	VisualBERT COCO	70.77	73.70	69.95±1.06	74.59±1.56
Ours	VisualBERT $CC_{\texttt{ID=60}}$	70.93	75.21	-	-
	VisualBERT $CC_{ensemble}$	**74.23**	**79.26**	**76.50**	**81.08**

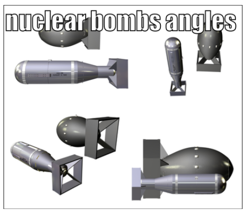
metrics – accuracy and AUROC. In addition, all the other models used in the ensemble (Table 6.2) improve the baselines in terms of AUROC, where in terms of accuracy only some models improve the best performing baseline model (**VisualBERT COCO**) while others are at par or the difference is insignificant. Next, we answer a few possible research questions.

Does using new image features help?
The VisualBERT model pre-trained on CC with pre-extracted default features (`ID=1` in Table 6.1) achieve AUROC score of 73.58, whereas the same model pre-trained on the same dataset but using our own image features (`ID=60` in Table 6.2) receives 75.21 AUROC. Although hyper-parameter settings may differ in these two models during fine-tuning, we observe that any model in the ensemble performs better than the aforementioned model, which indicates that our features are better and complementary to the task. In this regard, a recent study [116] presents a detailed study of improving visual representations for V&L tasks. They argue that previous V&L research has focused primarily on improving the multimodal fusion model and has neglected to improve the object detection model. They empirically show that the new visual features extracted using their proposed object detection model improve the performance across all V&L tasks, indicating that visual features are essential in V&L models.

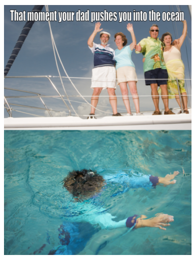
Does ensembling improve the score?
To answer this question we can compare the performances of the ensemble and best performing model in the ensemble on unseen dev set. As seen in Table 6.3, the ensemble performs significantly better than the best model in the ensemble, approximately an improvement of %4 in terms of both accuracy and AUROC. This strategy, we argue, effectively applies ensemble learning and produces one strong model from multiple *weak* models, analogous to the concept of "bringing the experts of the experts together". Consider a model that is effective at detecting hate speech directed at women (i.e., an expert in that regard), but not so good at detecting hate speech directed at religion. Then there is the possibility of another specialist with the exact opposite knowledge. Using the majority voting method, we can bring all of these experts together and benefit from them as a whole.

How fast does the model adapt to the downstream task?
During hyper-parameter tuning we experimented with a diverse number of updates (or iterations), ranging from 1k to 10k. Overall, the models did not improve after 2k iterations. Therefore, we set the maximum number of iterations to 2k and noted the best performing model as well as its best iteration. The right-most column in Table 6.2 shows the iteration in which the model performed the best during validation. For example, the best performing model (`ID=60`) achieves its best results after only 750 updates. The rest of the models also peak in a similar number of updates. For this reason, we observe that the models adapt to the downstream task fairly quick. Considering the fine-tuning dataset size, it is expected that the models converge fast.

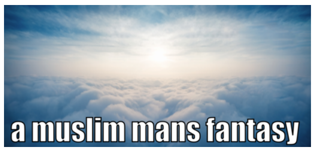
Lastly, we show a few model inferences on unseen test set. In Table 6.4, we show how the majority voting technique is applied for those samples. For example, for the meme with `ID=23706` all the 27 models in the ensemble predicted that the meme is hateful (class 1). Therefore, the meme is classified as class 1 (hateful) and the largest probability amongst the models in the ensemble is assigned. In contrast, the smallest probability is assigned for the samples that are classified as class 0 (not hateful).



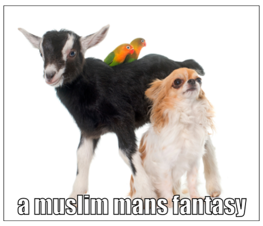
(a) ID=04398 | Pred=0



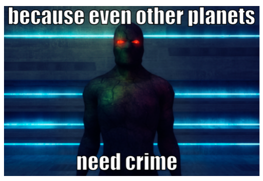
(b) ID=58239 | Pred=0



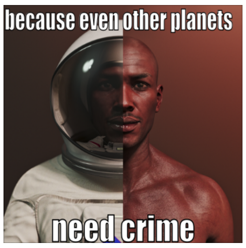
(c) ID=10697 | Pred=1



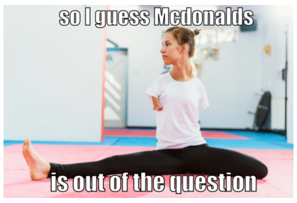
(d) ID=23706 | Pred=1



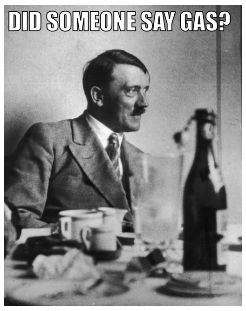
(e) ID=13457 | Pred=0



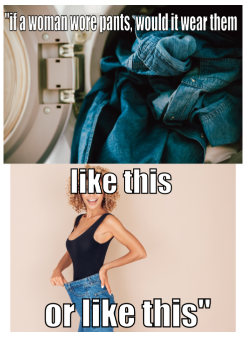
(f) ID=53640 | Pred=1



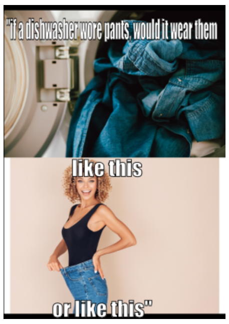
(g) ID=12394 | Pred=1



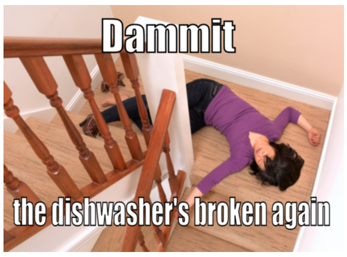
(h) ID=32698 | Pred=1



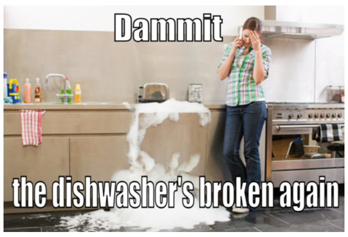
(i) ID=65307 | Pred=0



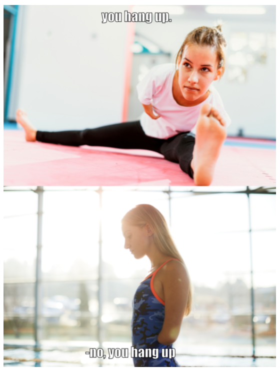
(j) ID=19382 | Pred=1



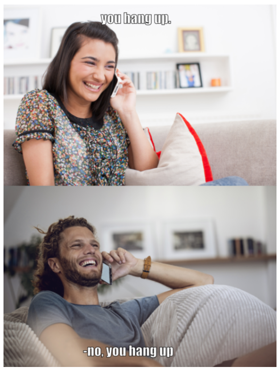
(k) ID=65107 | Pred=1



(l) ID=93125 | Pred=0



(m) ID=34762 | Pred=0



(n) ID=46879 | Pred=0

Figure 6.1: VisualBERT $CC_{ensemble}$ model inferences on unseen test set, where Pred=0 means the model predicts that the meme is not hateful, and 1 as hateful.

Table 6.4: Majority Voting technique for VisualBERT CC$_{ensemble}$ model on unseen test set. For example, for the meme with ID=04398 all the 27 models predicted that the meme is not hateful (class 0). Hence, the class of the sample is 0, with the smallest probability amongst the 27 models.

Meme ID	# of votes for not hateful (0)	hateful (1)	Probability Smallest	Largest	New Label	New Probability
04398	27	0	6.7773E-06	7.4641E-03	0	6.7773E-06
58239	27	0	7.3150E-06	3.0126E-02	0	7.3150E-06
10697	1	26	8.6104E-02	9.9998E-01	1	9.9998E-01
23706	0	27	9.9611E-01	9.9999E-01	1	9.9999E-01
13457	17	10	1.0984E-04	9.9973E-01	0	1.0984E-04
53640	0	27	7.9756E-01	9.9997E-01	1	9.9997E-01
12394	13	14	1.4593E-04	9.9996E-01	1	9.9996E-01
32698	13	14	8.9907E-04	9.9996E-01	1	9.9996E-01
65307	14	13	1.1288E-03	9.9470E-01	0	1.1288E-03
19382	1	26	4.5838E-01	9.9998E-01	1	9.9998E-01
65107	4	23	1.1642E-02	9.9996E-01	1	9.9996E-01
93125	27	0	1.2231E-05	1.6519E-02	0	1.2230E-05
34762	26	1	2.1944E-05	7.2206E-01	0	2.1944E-05
46879	27	0	1.9846E-05	6.0324E-02	0	1.9846E-05

Figure 6.1 shows the memes with their ID's and model predictions. We warn and apologize to the reader that the figure includes offensive content. In this respect, memes are shrunken to avoid any unexpected disturbance. Please zoom in for a clearer view.

Our model correctly classifies Figure 6.1a, Figure 6.1b, Figure 6.1d, Figure 6.1f, Figure 6.1l, and Figure 6.1n where all the models agree on the decision. Whereas for Figure 6.1c, Figure 6.1i, and Figure 6.1m model fails to predict the correct class. For Figure 6.1c, the model correctly classifies its image confounder (Figure 6.1d) but almost all the models in the ensemble wrongly classify itself. On the other hand, the model misclassified Figure 6.1i while correctly classifying its text confounder (Figure 6.1j). Model classifies Figure 6.1n correctly while misclassifying its image confounder (Figure 6.1m). However, the model is capable of detecting hate speech directed at women (Figure 6.1k, Figure 6.1l), and race (Figure 6.1e, Figure 6.1f).

7 CONCLUSION

The world around us and how we, as humans, perceive the world is multimodal. Building multimodal neural networks — AI systems that learn about concepts in multiple modalities, especially the textual and visual domains, in order to better understand the world — is a long-term goal of artificial intelligence. Hence, multimodal deep learning plays a crucial role in this regard.

In this work, we proposed an approach detecting hate speech in internet memes multimodally, i.e. considering visual and textual information holistically. We took part in the Hateful Memes Challenge and placed third out of 3,173 participants. Our approach utilizes a pre-trained VisualBERT (a BERT of vision and language), fine-tuned on an expanded training dataset, finally applying Ensemble Learning. Our approach achieves 0.811 AUROC with an accuracy of 0.765 on the challenge test set, which is a considerable result but also shows that we are still far from the accuracy of human judgement.

Understanding memes often requires subtle world knowledge. Rich intellectual information (for example, understanding that the object is a vehicle but also that it is a "Tesla Model S" or that the singer in the photo is "Dua Lipa") would be extremely beneficial. Given the dataset's context, explicit knowledge of hate speech subject characteristics; such as race and gender could also be beneficial; however, integrating such features in practice poses significant ethical concerns.

The VisualBERT model does not consider the object tags gathered from object detection models while extracting image features. Providing this information to the model would be beneficial (e.g. OSCAR [50]). Moreover, adding various vision-and-language models in the ensemble would also be complementary to the task.

Lastly, investigating more into vision-and-language models, doing an in-depth analysis (e.g., the analysis from OpenAI [29] for their model CLIP [71]) of what the models or neurons learn could help to create better designed, well-suited models.

BIBLIOGRAPHY

[1] Aishwarya Agrawal et al. *VQA: Visual Question Answering*. 2016. arXiv: 1505.00468 [cs.CL].

[2] Peter Anderson et al. *Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering*. 2018. arXiv: 1707.07998 [cs.CV].

[3] Peter Anderson et al. *Vision-and-Language Navigation: Interpreting visually-grounded navigation instructions in real environments*. 2018. arXiv: 1711.07280 [cs.CV].

[4] Stanislaw Antol et al. "Vqa: Visual question answering". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2425–2433.

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2016. arXiv: 1409.0473 [cs.CL].

[6] L.E. Bahrick and R. Lickliter. *Intersensory redundancy guides attentional selectivity and perceptual learning in infancy*. 2000.

[7] Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. *Multimodal Machine Learning: A Survey and Taxonomy*. 2017. arXiv: 1705.09406 [cs.LG].

[8] Yoshua Bengio, Aaron Courville, and Pascal Vincent. *Representation Learning: A Review and New Perspectives*. 2014. arXiv: 1206.5538 [cs.LG].

[9] Yoshua Bengio et al. "A Neural Probabilistic Language Model". In: *J. Mach. Learn. Res.* 3.null (2003), pp. 1137–1155.

[10] Hendrik Blockeel. "Hypothesis Space". In: *Encyclopedia of Machine Learning*. Ed. by Claude Sammut and Geoffrey I. Webb. Boston, MA: Springer US, 2010, pp. 511–513. ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_373. URL: https://doi.org/10.1007/978-0-387-30164-8_373.

[11] Piotr Bojanowski et al. *Enriching Word Vectors with Subword Information*. 2017. arXiv: 1607.04606 [cs.CL].

[12] Samuel R. Bowman et al. *A large annotated corpus for learning natural language inference*. 2015. arXiv: 1508.05326 [cs.CL].

[13] Andrew P Bradley. "The use of the area under the ROC curve in the evaluation of machine learning algorithms". In: *Pattern recognition* 30.7 (1997), pp. 1145–1159.

[14] Björn Brinne. *3D representation of a Transformer (BERT)*. [Online; accessed April 5, 2021]. 2020. URL: https://peltarion.com/blog/data-science/illustration-3d-bert.

[15] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL].

[16] Emanuele Bugliarello et al. *Multimodal Pretraining Unmasked: Unifying the Vision and Language BERTs*. 2020. arXiv: 2011.15124 [cs.CL].

[17] Xinlei Chen et al. "Microsoft coco captions: Data collection and evaluation server". In: *arXiv preprint arXiv:1504.00325* (2015).

[18] Yen-Chun Chen et al. *UNITER: UNiversal Image-TExt Representation Learning*. 2020. arXiv: 1909.11740 [cs.CV].

[19] Sidney K. D'mello and Jacqueline Kory. "A Review and Meta-Analysis of Multimodal Affect Detection Systems". In: *ACM Comput. Surv.* 47.3 (2015). ISSN: 0360-0300. DOI: 10.1145/2682899. URL: https://doi.org/10.1145/2682899.

[20] Andrew M. Dai and Quoc V. Le. *Semi-supervised Sequence Learning*. 2015. arXiv: 1511.01432 [cs.LG].

[21] J. Deng et al. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[22] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: (2019). arXiv: 1810.04805 [cs.CL].

[23] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2020. arXiv: 2010.11929 [cs.CV].

[24] Pedro F Felzenszwalb and Daniel P Huttenlocher. "Efficient graph-based image segmentation". In: *International journal of computer vision* 59.2 (2004), pp. 167–181.

[25] Matt Gardner et al. "Evaluating nlp models via contrast sets". In: *arXiv preprint arXiv:2004.02709* (2020).

[26] Donald Geman et al. "Visual Turing test for computer vision systems". In: *Proceedings of the National Academy of Sciences* 112.12 (2015), pp. 3618–3623. DOI: 10.1073/pnas.1422953112. eprint: https://www.pnas.org/content/112/12/3618.full.pdf. URL: https://www.pnas.org/content/112/12/3618.

[27] Ross Girshick. *Fast R-CNN*. 2015. arXiv: 1504.08083 [cs.CV].

[28] Ross Girshick et al. *Rich feature hierarchies for accurate object detection and semantic segmentation*. 2014. arXiv: 1311.2524 [cs.CV].

[29] Gabriel Goh et al. *Multimodal Neurons in Artificial Neural Networks*. https://openai.com/blog/multimodal-neurons/. 2021.

[30] Yash Goyal et al. "Making the V in VQA Matter: Elevating the Role of Image Understanding in Visual Question Answering". In: (2017). arXiv: 1612.00837 [cs.CV].

[31] Alex Graves, Greg Wayne, and Ivo Danihelka. *Neural Turing Machines*. 2014. arXiv: 1410.5401 [cs.NE].

[32] W. Guo, J. Wang, and S. Wang. "Deep Multimodal Representation Learning: A Survey". In: *IEEE Access* 7 (2019), pp. 63373–63394. DOI: 10.1109/ACCESS.2019.2916887.

[33] Danna Gurari et al. *VizWiz Grand Challenge: Answering Visual Questions from Blind People*. 2018. arXiv: 1802.08218 [cs.CV].

[34] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[35] Kaiming He et al. *Mask R-CNN*. 2018. arXiv: 1703.06870 [cs.CV].

[36] Justin Johnson, Andrej Karpathy, and Li Fei-Fei. *DenseCap: Fully Convolutional Localization Networks for Dense Captioning*. 2015. arXiv: 1511.07571 [cs.CV].

[37] A. Karpathy et al. "Large-Scale Video Classification with Convolutional Neural Networks". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1725–1732. DOI: 10.1109/CVPR.2014.223.

[38] Andrej Karpathy and Li Fei-Fei. *Deep Visual-Semantic Alignments for Generating Image Descriptions*. 2015. arXiv: 1412.2306 [cs.CV].

[39] Divyansh Kaushik, Eduard Hovy, and Zachary C Lipton. "Learning the difference that makes a difference with counterfactually-augmented data". In: *arXiv preprint arXiv:1909.12434* (2019).

[40] Sahar Kazemzadeh et al. "ReferItGame: Referring to Objects in Photographs of Natural Scenes". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 787–798. DOI: 10.3115/v1/D14-1086. URL: https://www.aclweb.org/anthology/D14-1086.

[41] Douwe Kiela et al. *Supervised Multimodal Bitransformers for Classifying Images and Text*. 2020. arXiv: 1909.02950 [cs.CL].

[42] Douwe Kiela et al. "The Hateful Memes Challenge: Detecting Hate Speech in Multimodal Memes". In: *arXiv preprint arXiv:2005.04790* (2020).

[43] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[44] Ranjay Krishna et al. "Visual genome: Connecting language and vision using crowdsourced dense image annotations". In: *International journal of computer vision* 123.1 (2017), pp. 32–73.

[45] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. Red Hook, NY, USA: Curran Associates Inc., 2012, pp. 1097–1105.

[46] Y. Lecun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[47] Yann LeCun, Y. Bengio, and Geoffrey Hinton. "Deep Learning". In: *Nature* 521 (May 2015), pp. 436–44. DOI: 10.1038/nature14539.

[48] Yann LeCun et al. "Handwritten Digit Recognition with a Back-Propagation Network". In: *Advances in Neural Information Processing Systems*. Ed. by D. Touretzky. Vol. 2. Morgan-Kaufmann, 1990. URL: https://proceedings.neurips.cc/paper/1989/file/53c3bce66e43be4f209556518c2fcb54-Paper.pdf.

[49] Liunian Harold Li et al. "Visualbert: A simple and performant baseline for vision and language". In: *arXiv preprint arXiv:1908.03557* (2019).

[50] Xiujun Li et al. *Oscar: Object-Semantics Aligned Pre-training for Vision-Language Tasks*. 2020. arXiv: 2004.06165 [cs.CV].

[51] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. 2015. arXiv: 1405.0312 [cs.CV].

[52] Ze Liu et al. *Swin Transformer: Hierarchical Vision Transformer using Shifted Windows*. 2021. arXiv: 2103.14030 [cs.CV].

[53] Ilya Loshchilov and Frank Hutter. *Decoupled Weight Decay Regularization*. 2019. arXiv: 1711.05101 [cs.LG].

[54] Jiasen Lu et al. *ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks*. 2019. arXiv: 1908.02265 [cs.CV].

[55] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. *Effective Approaches to Attention-based Neural Machine Translation*. 2015. arXiv: 1508.04025 [cs.CL].

[56] Bryan McCann et al. *Learned in Translation: Contextualized Word Vectors*. 2018. arXiv: 1708.00107 [cs.CL].

[57] H. MCGURK and J MACDONALD. "Hearing lips and seeing voices". In: *Nature* 264 (1976), pp. 764–786.

[58] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781 [cs.CL].

[59] Emilie Morvant, Amaury Habrard, and Stéphane Ayache. *Majority Vote of Diverse Classifiers for Late Fusion*. 2014. arXiv: 1404.7796 [stat.ML].

[60] Jiquan Ngiam et al. "Multimodal Deep Learning". In: *ICML*. 2011, pp. 689–696. URL: https://icml.cc/2011/papers/399_icmlpaper.pdf.

[61] J. T. Nockleby. "Hate speech". In: *Encyclopedia of the American Constitution*. Vol. 3. 2000, pp. 1277–79.

[62] Yingwei Pan et al. *Jointly Modeling Embedding and Translation to Bridge Video and Language*. 2015. arXiv: 1505.01861 [cs.CV].

[63] Georgios Paraskevopoulos et al. "Multiresolution and Multimodal Speech Recognition with Transformers". In: *arXiv preprint arXiv:2004.14840* (2020).

[64] Letitia Parcalabescu, Nils Trost, and Anette Frank. *What is Multimodality?* 2021. arXiv: 2103.06304 [cs.AI].

[65] Adam Paszke et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019. arXiv: 1912.01703 [cs.LG].

[66] Jeffrey Pennington, Richard Socher, and Christopher Manning. "GloVe: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. URL: https://www.aclweb.org/anthology/D14-1162.

[67] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "GloVe: Global Vectors for Word Representation". In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.

[68] Bryan A. Plummer et al. *Flickr30k Entities: Collecting Region-to-Phrase Correspondences for Richer Image-to-Sentence Models*. 2016. arXiv: 1505.04870 [cs.CV].

[69] Kreiman G. et al. Quiroga R. Reddy L. "Invariant visual representation by single neurons in the human brain". In: *Nature* 435 (2005), pp. 1102–1107. DOI: 10.1038/nature03687. URL: https://doi.org/10.1038/nature03687.

[70] A. Radford and Karthik Narasimhan. "Improving Language Understanding by Generative Pre-Training". In: 2018.

[71] Alec Radford et al. *Learning Transferable Visual Models From Natural Language Supervision*. 2021. arXiv: 2103.00020 [cs.CV].

[72] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. *Vision Transformers for Dense Prediction*. 2021. arXiv: 2103.13413 [cs.CV].

[73] Rasa. *Rasa Algorithm Whiteboard - Transformers & Attention 1: Self Attention*. [Online; accessed April 9, 2021]. 2020. URL: https://www.youtube.com/watch?v=yGTUuEx3GkA.

[74] Scott Reed et al. *Generative Adversarial Text to Image Synthesis*. 2016. arXiv: 1605.05396 [cs.NE].

[75] Shaoqing Ren et al. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2016. arXiv: 1506.01497 [cs.CV].

[76] Tekla S.Perry. *Q&A: Facebook's CTO Is at War With Bad Content, and AI Is His Best Weapon*. https://spectrum.ieee.org/computing/software/qa-facebooks-cto-is-at-war-with-bad-content-and-ai-is-his-best-weapon. July 2020.

[77] Chhavi Sharma et al. "SemEval-2020 Task 8: Memotion Analysis–The Visuo-Lingual Metaphor!" In: *arXiv preprint arXiv:2008.03781* (2020).

[78] Piyush Sharma et al. "Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2018, pp. 2556–2565.

[79] Ekaterina Shutova, Douwe Kiela, and Jean Maillard. "Black Holes and White Rabbits: Metaphor Identification with Visual Features". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, June 2016, pp. 160–170. DOI: 10.18653/v1/N16-1020. URL: https://www.aclweb.org/anthology/N16-1020.

[80] Amanpreet Singh, Vedanuj Goswami, and Devi Parikh. "Are we pretraining it right? Digging deeper into visio-linguistic pretraining". In: *arXiv preprint arXiv:2004.08744* (2020).

[81] Amanpreet Singh et al. *MMF: A multimodal framework for vision and language research*. 2020.

[82] L. Smith and M. Gasser. *The development of embodied cognition: six lessons from babies.* 2005. DOI: 10.1162/1064546053278973. URL: https://doi.org/10.1162/1064546053278973.

[83] Tejas Srinivasan et al. "Multimodal Speech Recognition with Unstructured Audio Masking". In: *arXiv preprint arXiv:2010.08642* (2020).

[84] Weijie Su et al. *VL-BERT: Pre-training of Generic Visual-Linguistic Representations*. 2020. arXiv: 1908.08530 [cs.CV].

[85] Alane Suhr et al. "A Corpus for Reasoning About Natural Language Grounded in Photographs". In: (2019). arXiv: 1811.00491 [cs.CL].

[86] Ilya Sutskever. *OpenAI Multimodal Research*. [Online; accessed April 10, 2021]. 2021. URL: https://openai.com/blog/tags/multimodal/.

[87] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. *Sequence to Sequence Learning with Neural Networks*. 2014. arXiv: 1409.3215 [cs.CL].

[88] Christian Szegedy et al. *Going Deeper with Convolutions*. 2014. arXiv: 1409.4842 [cs.CV].

[89] Hao Tan and Mohit Bansal. *LXMERT: Learning Cross-Modality Encoder Representations from Transformers*. 2019. arXiv: 1908.07490 [cs.CL].

[90] Ashish Vaswani et al. "Attention Is All You Need". In: (2017). arXiv: 1706.03762 [cs.CL].

[91] Riza Velioglu and Jewgeni Rose. “Detecting Hate Speech in Memes Using Multimodal Deep Learning Approaches: Prize-winning solution to Hateful Memes Challenge”. In: (2020). arXiv: 2012.12975 [cs.AI].

[92] Valentin Vielzeuf et al. *CentralNet: a Multilayer Approach for Multimodal Fusion*. 2018. arXiv: 1808.07275 [cs.AI].

[93] Oriol Vinyals et al. *Show and Tell: A Neural Image Caption Generator*. 2015. arXiv: 1411.4555 [cs.CV].

[94] Weiyao Wang, Du Tran, and Matt Feiszli. *What Makes Training Multi-Modal Classification Networks Hard?* 2020. arXiv: 1905.12681 [cs.CV].

[95] Wenhai Wang et al. *Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions*. 2021. arXiv: 2102.12122 [cs.CV].

[96] Lilian Weng. “The Transformer Family”. In: *lilianweng.github.io/lil-log* (2020). URL: https://lilianweng.github.io/lil-log/2020/03/27/the-transformer-family.html.

[97] Thomas Wolf et al. *HuggingFace's Transformers: State-of-the-art Natural Language Processing*. 2020. arXiv: 1910.03771 [cs.CL].

[98] Haiping Wu et al. *CvT: Introducing Convolutions to Vision Transformers*. 2021. arXiv: 2103.15808 [cs.CV].

[99] Yonghui Wu et al. *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. 2016. arXiv: 1609.08144 [cs.CL].

[100] Zhiyong Wu, Lianhong Cai, and Helen Meng. “Multi-level Fusion of Audio and Visual Features for Speaker Identification”. In: *Advances in Biometrics*. Ed. by David Zhang and Anil K. Jain. Springer Berlin Heidelberg, 2005, pp. 493–499.

[101] Ning Xie et al. *Visual Entailment Task for Visually-Grounded Language Learning*. 2019. arXiv: 1811.10582 [cs.CV].

[102] Ning Xie et al. *Visual Entailment: A Novel Task for Fine-Grained Image Understanding*. 2019. arXiv: 1901.06706 [cs.CV].

[103] Saining Xie et al. *Aggregated Residual Transformations for Deep Neural Networks*. 2017. arXiv: 1611.05431 [cs.CV].

[104] J. Xu et al. “MSR-VTT: A Large Video Description Dataset for Bridging Video and Language”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 5288–5296. DOI: 10.1109/CVPR.2016.571.

[105] Kelvin Xu et al. *Show, Attend and Tell: Neural Image Caption Generation with Visual Attention*. 2016. arXiv: 1502.03044 [cs.LG].

[106] Tao Xu et al. *AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks*. 2017. arXiv: 1711.10485 [cs.CV].

[107] Xinchen Yan et al. *Attribute2Image: Conditional Image Generation from Visual Attributes*. 2016. arXiv: 1512.00570 [cs.LG].

[108] Quanzeng You et al. *Image Captioning with Semantic Attention*. 2016. arXiv: 1603.03925 [cs.CV].

[109] Fei Yu et al. *ERNIE-ViL: Knowledge Enhanced Vision-Language Representations Through Scene Graph*. 2021. arXiv: 2006.16934 [cs.CV].

[110] Licheng Yu et al. *Modeling Context in Referring Expressions*. 2016. arXiv: 1608.00272 [cs.CV].

[111] B. P. Yuhas, M. H. Goldstein, and T. J. Sejnowski. “Integration of acoustic and visual speech signals using neural networks”. In: *IEEE Communications Magazine* 27.11 (1989), pp. 65–71. DOI: 10.1109/35.41402.

[112] Tom Zahavy et al. *Is a picture worth a thousand words? A Deep Multi-Modal Fusion Architecture for Product Classification in e-commerce*. 2016. arXiv: 1611.09534 [cs.CV].

[113] Savvas Zannettou et al. “On the origins of memes by means of fringe web communities”. In: *Proceedings of the Internet Measurement Conference 2018*. 2018, pp. 188–202.

[114] Rowan Zellers et al. “From Recognition to Cognition: Visual Commonsense Reasoning”. In: (2019). arXiv: 1811.10830 [cs.CV].

[115] Chao Zhang et al. “Multimodal Intelligence: Representation Learning, Information Fusion, and Applications”. In: *IEEE Journal of Selected Topics in Signal Processing* 14.3 (2020), pp. 478–493. DOI: 10.1109/jstsp.2020.2987728. URL: http://dx.doi.org/10.1109/JSTSP.2020.2987728.

[116] Pengchuan Zhang et al. *VinVL: Revisiting Visual Representations in Vision-Language Models*. 2021. arXiv: 2101.00529 [cs.CV].

[117] Yukun Zhu et al. *Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books*. 2015. arXiv: 1506.06724 [cs.CV].