

# **DETECTING HATE IN MULTIMODAL MEMES USING MACHINE LEARNING**

PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR THE AWARD OF DEGREE OF

**MASTER OF TECHNOLOGY  
IN  
COMPUTER SCIENCE & ENGINEERING**

Submitted By

**VAISHALI SHARMA  
2K21/CSE/25**

under the supervision of

**Dr. ARUNA BHAT  
(Associate Professor)**



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

May 2022

# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

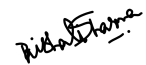
Bawana Road, Delhi-110042

## CANDIDATE'S DECLARATION

I, **Vaishali Sharma**, Roll No. 2K21/CSE/25 student of M.Tech (Computer Science and Engineering), hereby declare that the Project Dissertation titled “**Detecting Hate in Multimodal Memes Using Machine Learning**” which is being submitted by me to Delhi Technological University, Delhi, in partial fulfillment of requirements for the degree of Master of Technology in Computer Science and Engineering is a legitimate record of my work and is not copied from any source. The work contained in this report has not been submitted at any other University/Institution for the award of any degree.

Place: Delhi

Date: 20 May 2023



Vaishali Sharma

(2K21/CSE/25)

# **DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

## **CERTIFICATE**

I, hereby certify that the Project titled “**Detecting Hate in Multimodal Memes Using Machine Learning**”, submitted by Vaishali Sharma, Roll No. 2K21/CSE/25, Department of Computer Science & Engineering, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of degree of M.Tech in Computer Science and Engineering is a genuine record of the project work carried out by the student under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree to this University or elsewhere.

**Place: Delhi**

**20 May 2022**

**Dr. Aruna Bhat**

**Associate Professor**

# **ABSTRACT**

Social media has ingrained itself into our everyday lives, yet with its extensive use also comes the possibility of seeing offensive material. Artificial intelligence (AI) has established itself as an effective tool in the detection and removal of such information, which is a topic that is becoming more and more significant. However, because humans cannot distinguish between a picture and the text it contains because we interpret the combined meaning, it can be particularly difficult to identify nasty material in memes. Therefore, an AI tool designed to identify harmful memes has to have a thorough knowledge of their substance and context, just like people do.

A project was started to automatically categorise memes as hostile or not by integrating text, picture feature information, and extra data from online entity recognition in order to solve this problem. The Multimodal dataset from the Hateful Meme Detection Challenge 2020 was used in the study. Modern visual language models struggle to perform accurately in comparison to non-expert people on this dataset because it contains confounding instances such as memes that are unimportant, contrastive, or counterfactual, demonstrating the difficulty of the task. Models need a richer knowledge of language acquisition, image, current affairs, and the interactions across several modalities if they are to attain high accuracy.

The suggested method comprises categorising memes using text, images, and data gathered from the online entity identification procedure. The paper examines ways to strengthen the suggested technique going forward as well as the approach's flaws. The algorithms struggle with effectively detecting people's traits and classifying racial or religious groupings due to their lack of real-world experience. The models also have trouble recognising memes that are evocative of pain, abuse, and incapacity. The models also have trouble comprehending religious practises, traditional clothing, political and social allusions, and cultural norms.

The suggested architecture processes text and images simultaneously using two parallel streams and cross-attention training. The bidirectional multi-head attention paradigm serves as the foundation for both streams. The preprocessing pipeline necessary for the suggested design is also included in the study. The study was carried out in two stages, the first of which produced an Area Under the Receiver Operating Characteristic (AUROC) of 0.71 and an accuracy of 0.74 on the dataset of vile memes. The expanded Hateful Meme Detection Dataset showed that the model had an AUROC of 0.8108 on the test unseen data and 0.7555 on the dev unseen data, with an accuracy of 0.7352 for the test unseen and 0.7650 for the dev unseen data. The Hateful Meme Detection dataset was increased to include in order

to better broaden the dataset including more memes in Phase-2.

The paper recognises the project's limits even if the recommended method has shown promising outcomes. The method relies mainly on linguistic and visual characteristics, which restricts its ability to identify offending memes with delicate or intricate content. In order for the models to become more accurate, they also require an enormous quantity of training data, which can be challenging to get in real-world situations. The study highlights the importance of impartiality, accountability, and openness in the development and use of algorithms as well as the ethical conundrums raised by the application of AI to content moderation.

The study's conclusion offered a technique for automatically detecting offensive memes that made use of language, picture feature data, and internet object recognition. Despite the approach's positive results, problems still need to be resolved before the efficacy and accuracy of the models can be raised. Future research in this area may examine the inclusion of other modalities, such audio or video, to improve model performance. The algorithms' understanding of sociological and cultural influences may also be improved in order to boost the models' precision in recognising unfriendly content. Artificial intelligence (AI) must eventually be used with caution and openness in the moderation of material if we are to ensure that these innovations are exploited morally and responsibly.

# **ACKNOWLEDGEMENT**

I am extremely grateful to my project guide, **Dr. Aruna Bhat**, Associate Professor, Department of Computer Science and Engineering, Delhi Technological University, Delhi for providing invaluable guidance and being a constant source of inspiration throughout my research. I will always be indebted to her for the extensive support and encouragement she provided.

I am highly indebted to the panel faculties during all the progress evaluations for their guidance, constant supervision and for motivating me to complete my work. They helped me throughout by giving new ideas, providing necessary information and pushing me forward to complete the work.

Vaishali Sharma

2K21/CSE/25

# CONTENTS

CANDIDATE’S DECLARATION .....	1
CERTIFICATE.....	2
ABSTRACT .....	3
ACKNOWLEDGEMENT .....	5
CONTENTS .....	6
CHAPTER 1: INTRODUCTION.....	11
1.1 Problem definition.....	12
1.2 Objective .....	15
1.3 Contribution.....	16
CHAPTER 2: LITERATURE REVIEW .....	17
2.1 An overview of the research related to the detection of hateful content ....	17
2.2 An overview of nascent Visual-Linguistic Tasks .....	18
2.3 An overview of state-of-the-art Visual-Linguistic Models and Tasks .....	21
2.3.1 Visual Question Answering.....	23
2.3.2 Visual Commonsense Reasoning.....	26
2.3.3 VL-BERT.....	28
2.3.4 Vil-BERT .....	30
2.3.5 UNITER.....	32
CHAPTER 3: RELATED WORK .....	34

CHAPTER 4: METHODOLOGY .....	35
4.1 Optical Character Recognition .....	36
2.1 Inpainting Images .....	39
2.1 Image Patch Extractor.....	41
4.3.1 Meme Image Feature Extraction with Faster R-CNN .....	42
4.3.2 Meme Image Feature Extraction with Detectron2.....	45
4.4 Web Entity Detection .....	46
4.5 Single Stream Hateful Meme Classification.....	48
4.6 Two Stream Hateful Meme Classification .....	49
CHAPTER 5: EXPERIMENTS AND RESULTS .....	52
5.1 Analysis of VADER on Hateful Meme Dataset Textual Data .....	52
5.2 Analysis of TextBlob on Hateful Dataset Textual Data.....	54
5.3 Analysis of SGD on Hateful Meme Dataset Textual Data.....	55
5.4 Hyperparameter Tuning and Model Selection.....	56
CHAPTER 6: HATEFUL MEME DETECTION DETECTION .....	60
CHAPTER 7: SUMMARY .....	64
CHAPTER 8: CONCLUSION .....	66
REFERENCES .....	67



# LIST OF FIGURES

Figure 1.1: An example of a Meme .....	11
Figure 1.2: Benign Meme, Contrapositive Meme, Counterfactual Meme .....	14
Figure 2.1: Visual Question Answer Task Pipeline .....	23
Figure 2.2: Visual Commonsense Reasoning Task.....	27
Figure 2.3: Recognition to Cognition Network .....	28
Figure 2.4: Architecture for pre-training VL-BERT.....	30
Figure 2.5: Co-attention transformer layer .....	31
Figure 2.6: Architecture of UNITER model.....	32
Figure 4.1: Proposed Preprocessing Pipeline.....	35
Figure 4.2: Easy OCR Framework .....	36
Figure 4.3: Text Recognition Pipeline.....	37
Figure 4.4: CNN based Feature extraction .....	38
Figure 4.5: Sequence Labelling.....	38
Figure 4.6: Pipeline for Inpainting .....	40
Figure 4.7: Results of Inpainting on Hateful Memes Dataset .....	40
Figure 4.8: Architecture of Faster R-CNN model.....	43
Figure 4.9: R-CNN module for Classification and Regression .....	44
Figure 4.10: Architecture of Detectron2 model .....	45
Figure 4.11: Feature maps with different receptive field .....	47

Figure 4.12: Example of Google Web Detection On Hateful Meme Dataset.....	47
Figure 4.13: OCR and Inpainting Performed on Meme .....	48
Figure 4.14: Architecture of Modified VL-BERT .....	49
Figure 5.1: Results of VADER on Meme Text.....	54
Figure 5.2: (a) Confusion matrix for Rule based classification of VADER Polarity scores and (b) Confusion matrix for Rule based classification of VADER compound scores.....	54
Figure 5.3: Results of TextBlob on Meme Text .....	54
Figure 5.4: Confusion Matrix for Rule Based Classification on TextBlob Polarity Scores .....	55
Figure 5.5: Coefficients for Sentiment scores, Polarity and Subjectivity .....	56
Figure 5.6: Results of Single Stream methodology on Hateful Meme Detection Dataset.....	58
Figure 5.7: Results of Hyperparameter Training .....	61

# **LIST OF TABLES**

Table 5.1: Baseline Performance on Hateful Meme Detection Dataset .....	58
Table 5.2 : Hateful Meme Detection Dataset Splits .....	59
Table 6.1: Hateful Meme Detection Dataset Splits.....	60
Table 6.2: Data Distribution in the Hateful Meme Dataset.....	62

# CHAPTER 1: INTRODUCTION

Online communication via memes has grown in popularity. They frequently include innocuous humour and circulate quickly among internet users with minor alterations. However, some types of visuals, texts, or the mix of the two might become offensive memes. Hate is defined as an attack on an individual or group that revolves around traits such as racial, ethnic, national, immigrant, religious, caste, sex, sexuality, gender identity, disability, or illness. Memes, particularly nasty ones, have the ability to proliferate quickly and with just minor alterations, leading to the distribution of false information, hatred, and the misappropriation of our fallibility. They are therefore toxic and harmful, especially when directed towards specific people or groups.

Online, assaults can take the form of threatening or dehumanising remarks, as well as claims of inadequacy exclusion, or segregation. Hate speech includes mocking hate crimes. It is challenging to define hate speech since it can take many different forms, including harsh language, assault, cyberbullying, and online hostility.



*Figure 1.1: An example of a Meme*

The Hateful Meme Detection challenge presents a unique set of characteristics that make it challenging for models to learn. Only multimodal models have proven to be effective at learning, which negates the potential of models that take use of unimodal priors. Every image from a hostile meme in the collection has a replacement image or description that turns it into a neutral meme. Similar to how there is an image with a flipped label for every ugly meme while keeping the original image but changing the wording. The baseline performance of unimodal and multimodal models and the performance of non-expert humans for the Hateful Meme Detection Dataset are shown in Table 5.1. Non-expert human-trained annotators have an accuracy of 84.7%. This proves that even the best of the multimodal models are nowhere near human performance. These models were trained on datasets that were in context, unlike the hateful memes.

Existing pretrained computer vision models such as VGG, ResNet, etc. are trained on ImageNet, which contains image labels. This results in the CNN model learning the visual features, relations, and descriptions more than the content and context of the image. The models are not suitable for cross-modal tasks on their own. Multi-layer transformer models with multi-head attention are the hallmark for Natural Language Processing tasks. Models such as XLNET, BERT, ROBERTa, and GPT are trained using a massive corpus of textual data. They learn textual representations by predicting word tokens using the context and are used for downstream tasks with further fine-tuning. However, the text is sequential, unlike the image, which poses a challenge for combining the two modalities. Images can be included using autoencoding, but deciding how to do so is crucial.

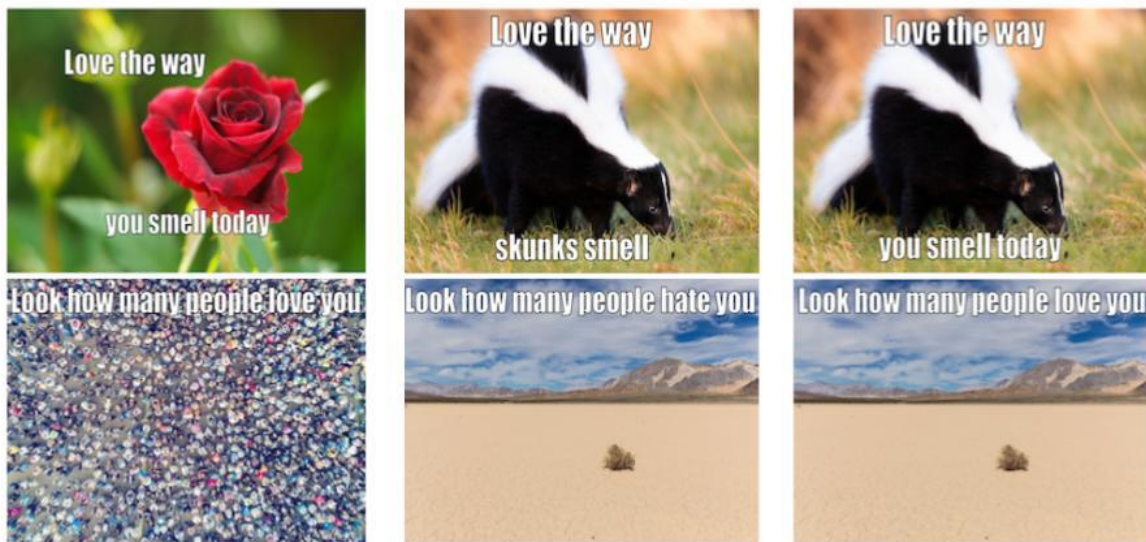
The Hate Meme Detection Dataset has revealed that state-of-the-art (SOTA) multimodal models have performed poorly in comparison to human performance. These models, which include text and visual information, were unable to match the precision of untrained individual annotators. In contrast, the models were trained on datasets that were used in context, meaning they were trained on data that resembled real-world scenarios. The results showed that non-expert human-trained annotators achieved an accuracy of 84.7% in identifying hate memes, indicating that these models still have a long way to go in matching human-level performance. This

performance gap highlights the complexity of hate meme detection and the challenges involved in developing accurate automated systems. To address this issue and effectively combat the propagation of offensive memes online, it is crucial to continue research in this field. By investing in further research, we can develop better models and methods for identifying and mitigating offensive content. This ongoing research will allow us to refine existing models, improve their performance, and bridge the gap between machine and human capabilities in hate meme detection. Moreover, research efforts can help in understanding the nuances of hate speech and offensive content, including cultural context, sarcasm, and implicit messages that may be difficult for machines to interpret accurately. Researchers may improve the effectiveness of hate meme identification models by creating more complex algorithms and training methods by developing a greater grasp of these features. Additionally, research in this field should focus on the ethical implications and potential biases associated with automated hate meme detection. It is crucial to ensure that these systems are fair, unbiased, and considerate of the diverse perspectives and cultural nuances present in different online communities.

In summary, despite the poor performance of current multi-modal models in hate meme detection, continued research is necessary to advance the development of more accurate and reliable methods. Through ongoing efforts, we can strive towards creating models that approach or even surpass human-level performance, ultimately contributing to a safer and more inclusive online environment.

## **1.1 Problem definition**

The hateful meme detection dataset is significantly smaller than other large datasets such as COCO (330K), Visual Genome (108K), and Conceptual Captions (3.3M). It's important to note that this dataset is not intended to be trained from scratch but to be fine-tuned and tested on a large pretrained model. The task's primary goal is to categorise meme pictures as hostile or not using a binary system.



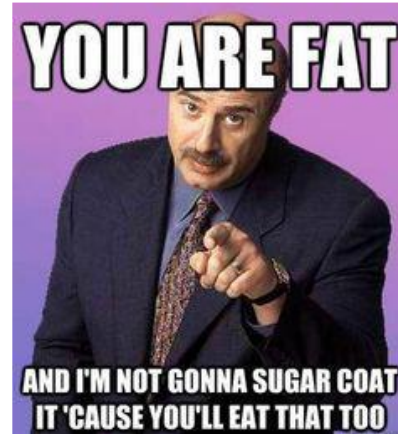
*Figure 1.2: Benign Meme, Contrapositive Meme, Counterfactual Meme*

Currently, state-of-the-art visual linguistic models are trained with datasets where the image and the textual content are correlated, such as the Visual Commonsense Reasoning (VCR) and Visual Question Answer (VQA) tasks, which use multimodal datasets for pre-training and downstream tasks. The Hateful Meme Detection data set, however, has a special quality that makes it useful for our purpose. The data set is introduced with confounder examples that are either benign, contrastive or counterfactual, as shown in Figure 1.2.

This dataset is diverse and contains objects that existing classifiers cannot identify. For instance, visual hints such as injury to oneself, physically challenged conditions, historical references, political references, and traditional attires are difficult to recognize and therefore hard to find by off-the-shelf classifiers. Additionally, the placement of the text in memes is also important, which means that performing just multimodal classification is not sufficient. It's crucial to incorporate additional information from different sources to improve the accuracy of the classification problem.



(a) Non-Hateful Meme example



(b) Hateful Meme example

## 1.2 Objective

This project seeks to advance the state-of-the-art (SOTA) already available multimodal models by incorporating additional sources of information and features for the Visual-Linguistic task of meme classification as hateful or not. Despite recent advancements in multimodal classification, the current SOTA models achieve an AUROC of 0.71 on the dataset. This performance is inferior to non-expert human performance. The project aims to highlight the difference between meme classification and multimodal reasoning tasks.

$$\text{AUROC} = \int_{x=0}^{x=1} \text{TPR}(\text{FPR}^{-1}(x)) dx \quad (1)$$

The primary metric utilized to evaluate model performance is AUROC, which penalizes models for incorrectly ordering the hatefulness of memes. In other words, it doesn't matter if the probabilities themselves are correct, but rather how well they are ordered. The formula for AUROC is given by the integral of TPR (true positive rate) multiplied by FPR-1 (inverse of the false positive rate) over the range of x from 0 to 1.



## 1.3 Contribution

The goal of this work is to create AI tools capable of analyse memes and identify whether or not they include offensive material, just like humans can. In order to do this, the tools must consider not only the memes' content and picture, but also any contextual information that may be present. It may be difficult for the AI technologies to understand the significance behind the memes without a substantial quantity of contextual data, though. In order to provide the tools more context and increase their accuracy and efficacy, the initiative attempts to collect new data from the internet.

The project's contribution is twofold: First, the team combines text, image, and web entity detection results to train a model for classifying memes as hateful or not. This is achieved by using two parallel streams of a multi-head attention transformer model and training them using co-attention. Second, they extend the Hateful Meme Detection dataset to include more memes in their second approach.

After hyperparameter search, the team selects the top-27 models for majority voting to come up with the final classification results. The model achieved an AUROC of 0.8108 on test unseen data and 0.7555 on dev unseen data of the extended Hateful Meme Detection Dataset with an accuracy of 0.7352 for test unseen and 0.7650 for dev unseen data. By combining text, image, and context, the project's approach goes beyond the traditional multimodal classification models used in other Visual-Linguistic tasks. The model's high performance demonstrates the effectiveness of using additional information from the web to add context to the memes, allowing for more accurate classification.

## **CHAPTER 2: LITERATURE REVIEW**

A substantial amount of research is being done in the fields regarding machine learning on the topic of identifying hate speech in recent years [3, 4]. However, the majority of these research have only employed text-based datasets, which may not be enough to fully reflect the problem's extent. The reason for this is that hate speech can take many various forms, from online hostility to harsh language to cyberbullying and harassment. Therefore, it is challenging to develop a thorough and widely recognised definition of hateful speech. Despite this difficulty, academics have worked to create efficient ways to detect and stop offensive language in online forums.

### **2.1 An overview of the research related to the detection of hateful content**

In recent years, there has been a growing interest in the research community to detect and mitigate the problem of online hate speech and cyberbullying. Several studies have been conducted to address this issue using different approaches and modalities. Homa Hosseinmardi et al. [5] focused on detecting cyberbullying in Instagram images and their corresponding comments, and achieved an accuracy of 87.5% by incorporating media information in addition to the text and images.

Z. Waseem et al. [6] proposed a topology for the task of detecting abusive language that could be used in data collection, annotation, feature engineering, and modeling. Ribeiro et al. [7] used a graph-based approach that collected and annotated hateful users and those who were banned from Twitter. Their node embedding algorithm outperformed the content-based hateful speech detection. Similarly, Zhong et al. [8] conducted an experiment using Instagram posts and comments to classify bullying and non-bullying.

The performance of non-expert individuals as well as the baseline results for

unimodal and multimodal models are displayed in Table 5.1. Modern models do not outperform humans, highlighting the difficulty of the classification problem. The accuracy of non expert trained annotators is 84.7%, demonstrating that even the finest multimodal models fall short of human ability. There is therefore a lot of room for development in this area.

## **2.2 An overview of nascent Visual-Linguistic Tasks**

A notable source for picture captioning research, the million-captioned photo collection from Flickr amassed by Vicente Ordonez et al. [9] has around one million photographs and the captions that go with them. The data set is perfect for testing and refining image captioning algorithms since it is a huge collection of diverse photographs that capture different situations and objects.

A denotational graph, essentially reflects a semantic representation of the text, was produced by P. Young et al. [10] using a sizable corpus of pictures with numerous captions. Each image in the corpus includes an average three-paragraph captions, and there are about 100,000 photos and captions altogether. A significant source of knowledge for visual anchoring and linguistic comprehension, the denotational graph displays the visual and verbal ideas in the subtitles and their relationships.

Another huge dataset that seeks to close the gap between language and video is MSR-VTT (Video-To-Text) [11]. It has more than 10,000 films and more than 200K descriptions in natural language. The dataset is helpful for tasks like video captioning as well as video retrieval since it contains a wide variety of video categories, including as news, sports, music, and food videos.

Microsoft COCO captions is a dataset with over 1.5 million captions describing 330K images [12]. It is a great resource for creating models that can comprehend visual information and explain it in natural language since the photos were hand-selected to represent a wide variety of item types and activities. The captions in the

dataset are diverse, covering various aspects of the image, such as objects, scenes, and actions.

Sahar Kazemzadeh et al. proposed an innovative approach to collect expressions for real-world natural scenes through a two-player game, which resulted in 130K expressions referring to 97K distinct objects in 19K images [13]. Their game involved one player describing a scene using natural language, and the other player had to identify the object in the image that matched the description. Researchers working on tasks including object identification, picture captioning, and visual question answering have found the data that was generated to be a useful tool.

Similarly, Harm de Vries et al. developed a two-player game to locate objects in an image using a sequence of visual-linguistic questions [14]. Their approach resulted in a large dataset of 800K Visual Question Answer (VQA) pairs on 66K images. This dataset has been used extensively in research on visual question answering, image captioning, and other multimodal tasks. Despite the recent progress in the field of multimodal learning, there is a lack of standard datasets or benchmark tasks [15]. Oleksii Sidorov et al. attempted to address this issue by collecting 145K captions for 28K images [16]. They tied the text to the objects in the image to develop a dataset for image captioning and reading comprehension. Their work has provided an excellent resource for researchers to evaluate and compare different models for these tasks.

Danna Gurari et al. introduced the Vizwiz dataset to address the challenges faced by visually impaired individuals when using mobile devices[17]. The dataset contains close to 31K visual questions, recorded by the blind using their mobile phones. Unlike other VQA datasets, the quality of the images in Vizwiz is poor, and the visual questions are more conversational in nature. This makes it a more realistic representation of the types of questions that the visually impaired ask in their day-to-day lives. Along with the Vizwiz data set, Danna Gurari et al. [18] gathered over 39K photos from blind persons, with five descriptions for each image. The "Describe" dataset was developed to enhance models for picture captioning for those

who are blind or visually impaired. The captions provided in this data set are more in-depth and descriptive, with the goal of giving visually impaired people who rely on these models to comprehend images as much knowledge as possible.

To increase the availability of visual resources for the blind, many research projects have combined the Vizwiz as well as Describe databases. VQA models have been trained on the Vizwiz dataset, while picture captioning models have been trained on the Describe dataset. These statistics have assisted academics in creating models that are better able to understand the concerns and requirements of the community of people who are blind, making technology more inclusive and accessible.

A. Suhr et al. introduced a new dataset with 100K English sentences paired with images from the web for the task of visual reasoning using multimodal data [19]. To overcome the shortcomings of the previous VQA and compositional questioning tasks, Drew A et al. came up with a robust corpus of data that uses the graph structure of the scene to come up with 22 million reasoning questions [20]. N. Xie et al. created the SNLI-VE using the Stanford Natural Language Inference corpus and Flickr30 dataset and have performed baselining for VQA task and built a model for Explainable Visual Entailment (EVE) [21].

Language and vision problems have become increasingly prevalent over time (Mogadala et al. [22]). In their study, Cesc C. Park and Gunhee Kim used picture streams to produce captions rather than only using one image and one caption [23]. With remarkable progress on important issues like visual inquiry responding to and picture subtitle age and recovery [10, 12, 24, 16, 18], diagnostic test datasets with little bias for the task of visual reasoning [25], visual narrating [23], visual exchange [26, 14], multimodal machine interpretation, visual thinking, and visual sound judgement thinking [19, 20, 27, 21, 17].

The dataset introduced by F. Alam et al. [28] is unique in that it focuses on natural disasters and aims to aid in crisis management. By collecting images and

corresponding tweets during such events, this multimodal dataset can provide valuable insights into the experiences and needs of those affected. The dataset is an important resource for researchers and policymakers alike, as it can help identify areas of need and improve response efforts.

On the other hand, UPMC-Food101 is a huge data set that includes 100K recipes for 101 food categories. The studies of Xin Wang et al. [29] developed an automated method for detecting culinary recipes based on photos using this dataset. The food sector will be significantly impacted by the capacity to correctly recognise recipes from photos, from recipe systems that offer recommendations to photography for food and advertising. For academics working on picture recognition and interpretation in the food domain, the UPMC-Food101 dataset is an invaluable resource.

The research [30] suggests a unique deep convolutional neural network design for sentiment analysis that takes into consideration the hierarchical structure of texts. On a number of benchmark datasets, including the well-known Stanford Sentiment Treebank dataset, our method produced cutting-edge results. The suggested architecture can capture the intricate links between words and phrases in a sentence, which has the potential to result in significant advancements in sentiment analysis. The study's conclusions open the door for further sentiment analysis research, which may result in applications like social media analysis and customer feedback analysis that are more accurate.

## **2.3 An overview of state-of-the-art Visual-Linguistic Models and Tasks**

A similar study that used state-of-the-art (SOTA) models that were pretrained on common Visual-Linguistic (VL) tasks has showed excellent results using the Hateful Meme Detection Dataset, a relatively new dataset in the field of multimodal analysis. These models can successfully learn the association between the dataset's picture

and textual content, which is necessary for the precise detection of offensive memes. Specifically, the pretraining tasks used by the SOTA VL models include Visual Commonsense Reasoning (VCR) and Visual Question Answer (VQA), both of which involve multimodal datasets. By leveraging these pretraining datasets and downstream tasks, the models are able to learn representations that effectively capture the relationship between visual and textual information.

However, the problem setting and dataset used in the Hateful Meme Detection Dataset are unique and have different characteristics than existing multimodal datasets. As such, it is important to carefully consider the limitations and challenges of the dataset when developing and evaluating models for this task. Future research in this area may involve developing new pretraining tasks and datasets that are better suited for the specific challenges posed by the Hateful Meme Detection Dataset. Among the models frequently used for Visual Linguistics problems are ViLBERT, VisualBERT, VLBERT, and UNITER. These models rely on the BERT model for linguistic modelling and the Faster RCNN model, which has been previously trained to extract visual data. Modern generalised pretrainable models whose are very successful for tasks linked to visual linguistics have been produced as a result of their use.

Below are the comparisons of the models:

ViLBERT [40] is a popular model for visual-linguistic tasks that uses two streams for the two modalities, i.e., text and images, followed by a Transformer training to combine them. They use a unique approach called the co-attentional transformer layer (Co-TRM) to train this cross-modality. A third transformer is used to combine the two streams. The authors claim that their architecture can better understand the interaction between the visual and linguistic contents.

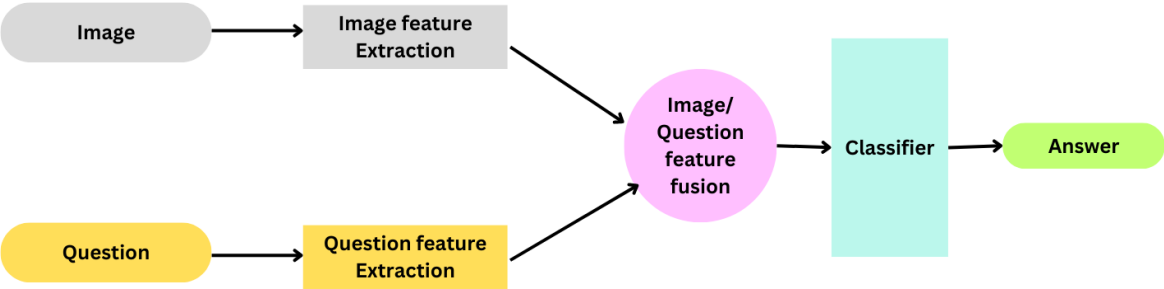
On the other hand, VisualBERT [24], Unicoder-VL [41], VL-BERT [42], and UNITER [43] use early fusion with a single stream model on both modalities and

employ early fusion. These single stream models use the masked language model pre-training task. However, VisualBERT lacks the object detection task or the visual-linguistic matching task. UNITER predicts the objects with the labels and performs masking of one of the modalities at a time to avoid misalignment. They employ the KL divergence loss within the variances of the input and the output.

Regarding the pre-training datasets, VisualBERT used the MS COCO Captions dataset, while ViLBERT and VLBERT utilized the Conceptual Captions dataset. In contrast, UNITER added one million image-captions along with the conceptual captions, MS COCO, and Genome Dense Captions data. UNICODER produced SOTA for image-to-text and text-to-image retrieval and VCR.

### 2.3.1 Visual Question Answering

Answering questions in natural language based on visual material, such as pictures or movies, is known as "visual question answering" (VQA). Building intelligent devices that can comprehend and analyse both visual and literary information is the aim of VQA. In order to complete the VQA job, a model must examine the input picture or video and the relevant natural language question before producing an appropriate response. The model must be able to make sense about the link between the visual and written knowledge as well as have a thorough comprehension of both.



*Figure 2.1: Visual Question Answer Task Pipeline*



The scientific community has made considerable strides in creating algorithms using deep learning that excel at the VQA challenge during the past few years. Many of these models use a combination of convolutional neural networks (CNNs) for visual processing and recurrent neural networks (RNNs) or transformer models for language processing. These models have been trained and evaluated using a number of large-scale datasets, including VQA v1 [44], VQA v2 [45], and GQA [46]. These datasets include a large number of pictures, questions, and ground truth responses given by human annotators.

VQA has several uses, including in robotics, intelligent personal assistants, and visual aids for individuals with impairments. For the creation of really intelligent systems, machines must be able to comprehend and interpret both textual and visual input.

VAQ pipeline:

#### 1. Feature Extraction:

The feature extraction phase of the VQA process is the initial step. This stage entails separating the characteristics from the query and the image. The very last convolutional layer that makes up a Convolutional Neural Network (CNN) can be utilised to gather visual features in the case of pictures. The data that is collected about the objects, colours, and other visual aspects present in the image is contained in the retrieved image features. The question, on the other hand, is supplied as a list of words that must be translated into numerical forms. Recurrent neural networks that can capture the sequential character of the issue, such Long Short-Term Memory (LSTM) or Gated Recurrent Units (GRU), are used for this.

#### 2. Multi-Modal Fusion:

The second stage in the VQA pipeline is multi-modal fusion. This stage combines the extracted image and question features to produce multimodal features. The fusion technique used can significantly impact the accuracy of the VQA model. Some of the commonly used fusion techniques include element-wise multiplication,

bilinear pooling, concatenation, attention- based pooling, compositional approach, and Bayesian- based methods. Element-wise multiplication is used when both image and question features are of the same dimension. Bilinear pooling is used when the dimensions of the image and question features are different. Concatenation and attention-based pooling are also popular fusion techniques.

### 3. Classification:

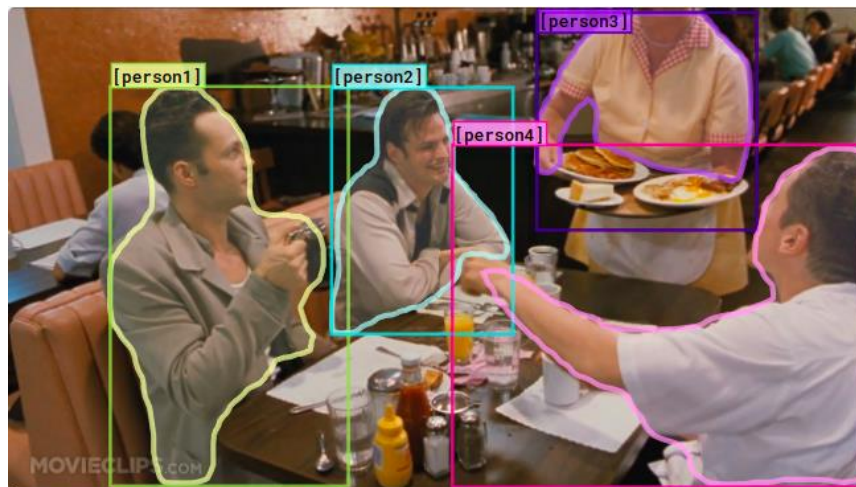
The final stage in the VQA pipeline is classification. This block takes the fused feature vector and performs classification. The type of classification depends on the nature of the VQA task. Questions with one -word answers can be thought of as a classification problem, where all answers are converted to labels. For open-ended questions or multiple- choice questions, the task becomes a multi- label classification problem. The classifier can be implemented using various machine learning algorithms such as support vector machines, random forests, or neural networks. The authors of the paper [44] use a VGG network to extract the image features. They extract the features from the last hidden layer of the network and apply the L2 norm to the activations of the last hidden layer. This step provides a set of image features that can be used for the subsequent steps of the pipeline.

Three alternative approaches are used to extract the question characteristics, giving the questions various embeddings. The top 1000 words in the questions are collected into a bag of words using the first technique, the Bag-of-Words Question (BoW Q), which gives the question a 1030 embedding. The second approach, LSTM Q, produces an embedding of length 1024 using a single-layer LSTM. The most recent hidden state and the LSTM cell state were concatenated to create this embedding. The third technique, deep LSTM Q, produces an embedding with a length of 2048 by using a two-layer LSTM. The last hidden state and the cell state of the last two hidden layers are combined to create this embedding. A fully connected layer and a tanh activation function are used to reduce the dimension from 2048 to 1024.

The question word embeddings for both the LSTMs are of length 300. They are encoded using a fully connected layer and a tanh layer. The input vocabulary

includes all the words from the questions in the training dataset. The BoW Q embedding and the image embeddings are concatenated, and the image embedding is transformed to a 1024 dimension using a fully connected layer and a tanh activation function. This transformed image embedding is fused with the LSTM Q embedding and deep LSTM Q embedding using element-wise multiplication. The fused embeddings are passed to a fully connected neural network with two hidden layers of 1000 neurons, with tanh activation. The last layer is followed by a softmax layer to get the probabilities over K classes. The objective function used is Cross Entropy loss. Here,  $K = 1000$ , which is the top frequent answers and it covers 82% of the answers in the training and validation set. The dataset used in this study consists of 200K images from Microsoft Common Objects in Context (MS COCO) with 50K abstract scenes. The dataset includes over 760K questions with 10M answers. The authors' approach provides an effective pipeline for the task of visual question answering, achieving state-of-the-art performance on the MS COCO dataset.

### 2.3.2 Visual Commonsense Reasoning



(a) Multiple Choice Question and Answers

Why is [person4] pointing at [person1]?

- |   |
|---|
| a) He is telling [person3] that [person1] ordered the pancakes. |
| b) He just told a joke.   |
| c) He is feeling accusatory towards [person1].                  |
| d) He is giving [person1] directions.                           |

(b) Multiple Choice Question and Answers

Rationale: I think so because...

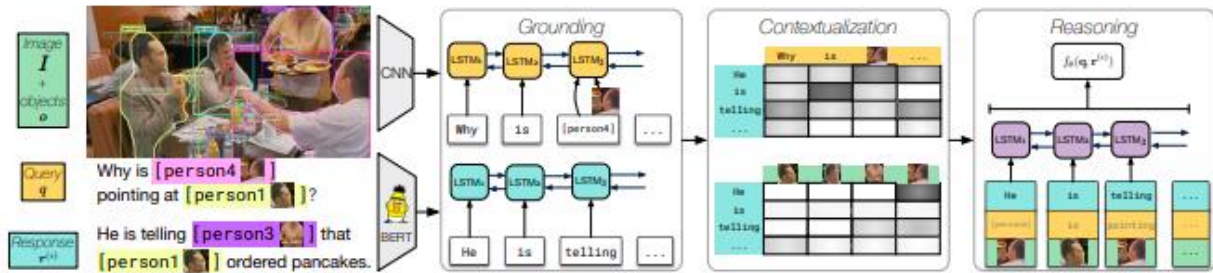
- |  |
|--|
| a) [person1] has the pancakes in front of him.   |
| b) [person4] is taking everyone's order and asked for clarification.                       |
| c) [person3] is looking at the pancakes both she and [person2] are smiling slightly.       |
| d) [person3] is delivering food to the table, and she might not know whose order is whose. |

(c) Multiple Choice Rationale

Figure 2.2: Visual Commonsense Reasoning Task

The VCR data set is intended to evaluate a computer vision model's capacity to comprehend pictures at a cognition- level, comparable with human cognition and rationality. The collection includes 110,000 photos and 290,000 questions with multiple-choices, all of which have the right justifications and solutions. Various and difficult issues must be answered by recognising and comprehending visual data expressed in natural language.

The Recognition-to-Cognition Network (R2C), which the authors of the research introduce, outperforms the most advanced visual question-answering systems now available. The R2C model generates an answer using three various kinds of reasoning: first, it creates a natural language passage that utilises the objects found in the image; second, it considers the question and answer's context even if the object is not viewable in the image; and third, it explains the reasoning process that was followed to arrive at the answer. Impressively, the model had a 67% accuracy in reasoning and a 65% accuracy in answering questions.



*Figure 2.3: Recognition to Cognition Network*

The ground module, the contextualization module, and the reasoning module make up the R2C model. The ground module trains a shared representation for the picture and text tokens using bidirectional LSTM. With the ROI aligned with the bounding box providing the representation of the objects, the module utilises CNN to learn the characteristics of the objects. The contextualization module uses the attention mechanism to represent the context between the query and the response. For every position of the response, an attended query is defined. The module that teaches reasoning does so while taking the answer, attended inquiry, and objects into account. The output of a bidirectional LSTM is the rationale. The question and response embeddings for each timestep are concatenated with the LSTM module's output. The multi-class cross-entropy between each answer and the actual label is reduced by training the R2C model to minimise it.

### 2.3.3 VL-BERT

The pretrainable VL-BERT model was created to handle a variety of downstream visual-linguistic tasks, including comprehension reading, visual question answering, and visual commonsense reasoning. By employing the Masked Language Modelling (MLM) BERT loss and pre-training on the enormous MS COCO data set and a text-only corpus, the model has learned a standard illustration for visual-linguistic tasks.

The core of VL-BERT consists of transformer attention modules that accept input in

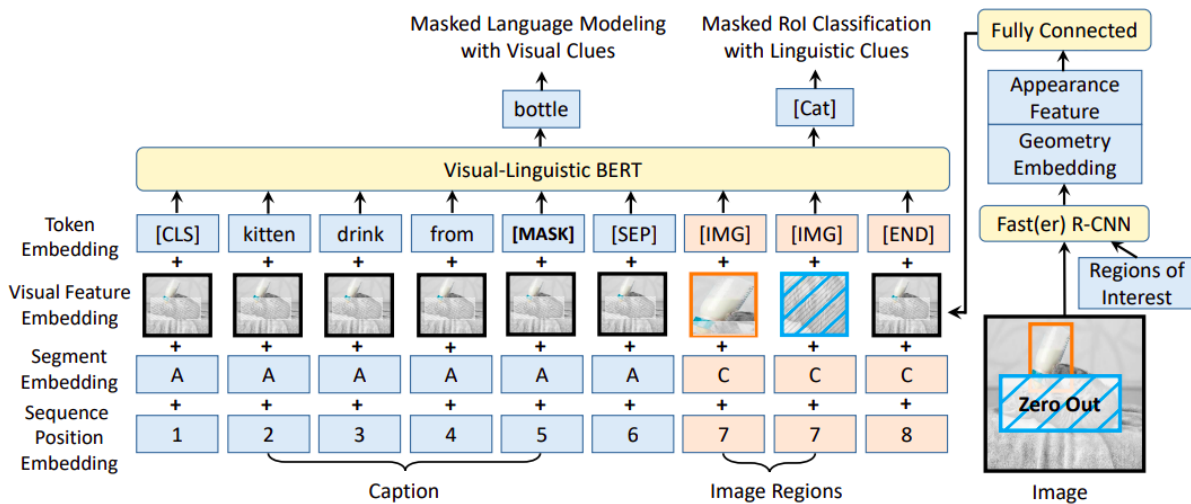
the form of either a word from the input phrase or a region-of-interest for either images or text embeddings. To gather and obtain visual-linguistic cues, these transformer attention modules are layered into many layers. New branches may be added to the model to meet additional visual-linguistic tasks, making it versatile. The authors of the publication demonstrate that VL-BERT yields cutting-edge outcomes on a variety of downstream tasks, making it a potential model for further studies in the area.

A series of embeddings that are a synthesis of various sorts of information serve as the model's input and are used to solve visual-linguistic problems. Linguistic embeddings come after a particular categorization token [CLS] in the input sequence. After the separator token, the linguistic embeddings and the visual embeddings are concatenated, and a special token [SEP] is used to separate them. The sequence concludes with the special token [finish] to mark the finish.

There are four types of embeddings used in the input sequence:

1. Token Embedding: The [IMG] token is assigned to the image feature embeddings to indicate the presence of a visual element. Special tokens such as [CLS] and [SEP] are used to mark the beginning and end of sentences or segments within the input sequence. Linguistic words are embedded using word piece embeddings, which break down words into smaller subwords for better generalization and to handle out-of-vocabulary words.

2. Visual Feature Embedding: Each visual element's visual geometry and visual appearance are combined to provide the visual embedding feature. Fast R-CNN, a well-known object identification method, is applied to the visual components in a region of interest (ROI) to produce the visual appearance. Fast R-CNN is used to the ROI that encompasses the entire picture to detect non-visual components. Each visual element's visual geometry embedding is represented by a 4-dimensional vector.



*Figure 2.4: Architecture for pre-training VL-BERT*

3. Segmentation Embedding: Three different segment kinds are used by VL-BERT: A, B, and C. The first and second phrases are assigned to segments A and B, accordingly. Segment C, on the other hand, is reserved especially for the picture areas of interest (ROIs).

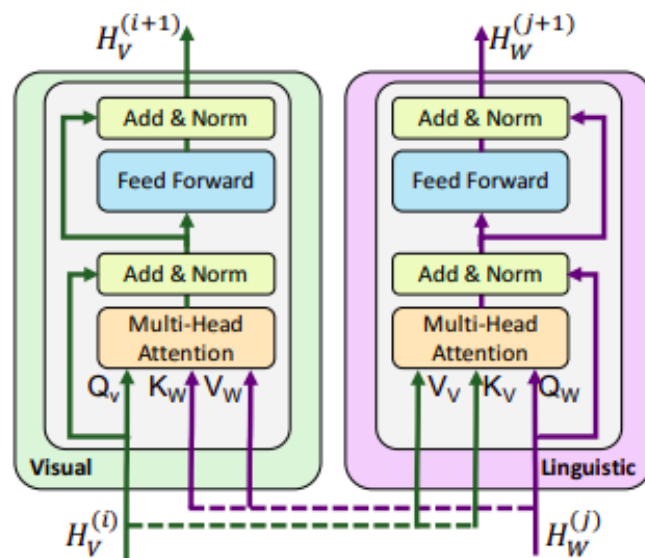
4. Position Sequence Embedding: To show the order of the input items in the sequence of inputs, this form of embedding is added. It is relevant only to the linguistic elements and not the visual elements. The position sequence embeddings help the model understand the relative positions of words within a sentence and the order of sentences within a document.

### 2.3.4 ViL-BERT

ViL-BERT is a model that combines two streams for processing text and vision, which run parallel and interact with each other using co-attention layers. This allows

each stream to decide on the depth of training and the sparsity due to attention. For two main tasks, the model is trained using MS-COCO: forecasting the semantics of input image and masked words and predicting the relationship between input text and image. ViL-BERT's primary novelty is the way that co-attention transformer layers are used to interact between the two parallel streams.

Four VL tasks are covered by the ViL-BERT model: caption-based image retrieval, visual question and answer, visual commonsense reasoning, and referring expressions. To build ViL-BERT, the authors use BERT as the backbone model, which is a bidirectional multi-head attention-based transformer model that has been successful in transfer learning for various natural language processing tasks. To create an intermediate depiction, the input tokens are first transformed into new embeddings and then sent through an encoder transformer. The attention distribution over the values is calculated using the dot product between the key and query after computing the key, query, and value from the intermediate representation. The weighed mean of the data is the result of the attention block.



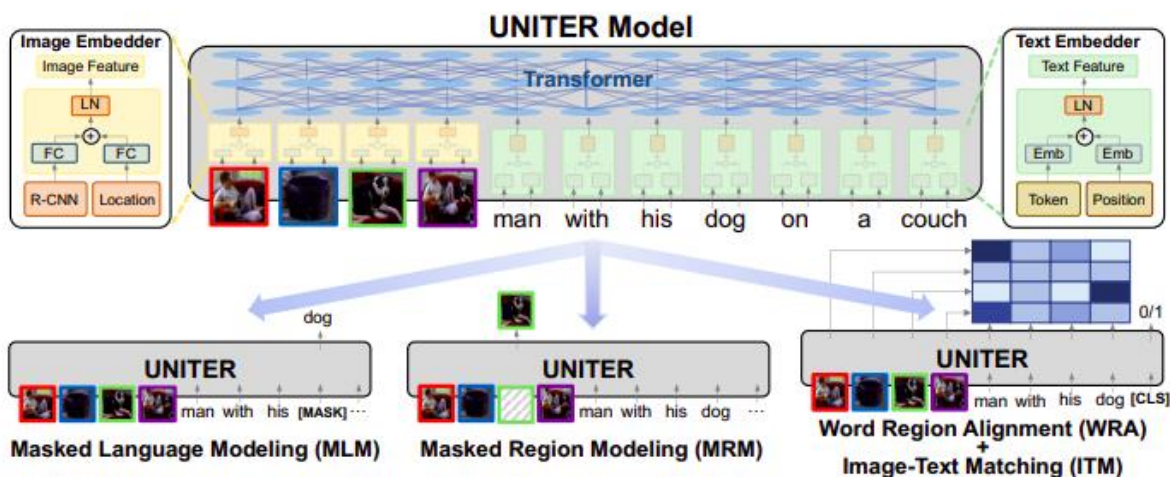
*Figure 2.5: Co-attention transformer layer*



There are two BERT models used in ViL-BERT: one for vision and the other for language. The key-value pairs in the multi-headed attention are exchanged so that language is incorporated into vision, and vision is incorporated into language. The BERT models are trained end-to-end on a large corpus for masked language modeling and next sentence prediction.

### 2.3.5 UNITER

A self-attention based transformer is at the heart of the pre-trained multimodal model known as the Universal Image-Text Representation (UNITER). The model was already pre-trained on four tasks: Image-Text Matching, Masked Region Modelling, Word Region Alignment, and Masked Language Modelling conditioned on photos. Masked Region Classification, Masked Region Feature Regression, and Masked Region Classification with KL-divergence are the three subtypes of Masked Region Modelling. Nine datasets, comprising VQA, VCR, Visual Entailment, Image-Text Retrieval, and Referring Expression Comprehension, were used to test the model on six V+L tasks. UNITER uses the picture Embedder to encode picture areas like visual and bounding box characteristics while the Text Embedder encodes tokens and locations for the words in the sentences.



*Figure 2.6: Architecture of UNITER model*

In contrast to other multimodal models that employ random masking, the model applies conditional masking to both pictures and words. By reducing the amount of money spent on transportation from one distribution to another, optimal transport is employed to align the word and picture areas. The picture embedder locates visual features using Faster R-CNN and utilises a 7-dimensional vector to encode their locations. The final embedding is created by adding the outputs of the fully connected (FC) layers and applying a normalisation layer after both the visual and location data have been projected into the same embedding space. The text embedder divides input words into tokens using the BERT model and wordpieces, and for each sub-word token, the final embedding is the sum of its word embedding and position embedding, followed by a normalisation layer. The [MASK] token for the word being masked is used for word masking, while visual elements are replaced by zeros for region masking. Only one modality is concealed at a time to avoid misalignment.

## CHAPTER 3: RELATED WORK

Ron Zhu, the author of paper [48], won the Hateful Meme Detection Challenge with an AUROC of 0.845 by enhancing the performance of existing visual-linguistic models. Zhu used a bottom up approach to extract features from clean meme images after inpainting and detecting human race and gender. These features were fed as input to a transformer model and trained using VL-BERT, UNITER-ITM, VILLA-ITM, and ENRIE-ViL. The predictions from these models were averaged to classify the meme, and a rule based approach was also applied to detect racism using text, race tag, skin tone, and gender.

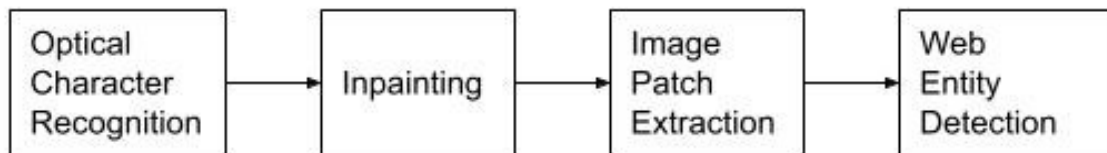
Niklas Muennighoff, in paper [49], used a transformer model for both text and images, with Faster R-CNN used to extract image features. The results from text and image features were combined and passed through a separate single-stream transformer model. The ensemble of five VL models - ERNIE ViL Large, ERNIE ViL Small, UNITER, OSCAR, and VisualBERT - achieved an AUROC of 0.81 and an accuracy of 82.52%, earning second place in the challenge.

Riza Velioglu and Jewgeni Rose [50] used VisualBERT as a single-stream transformer and Detectron2 to extract image features. They used hard voting from various base models to classify memes, achieving an AUROC of 0.811 with an accuracy of 0.765 and placing third in the challenge.

Phillip Lippe et al. [51] used the pretrained UNITER model and upsampled text confounders, giving more weight to hateful memes to address class imbalance. The models were trained using cross-validation, and an ensemble of UNITER models was used to classify memes. They achieved an AUROC of 0.8053, placing fourth in the challenge. The weights for ensembling were optimized using an evolutionary algorithm (EA) on the training set predictions.

## CHAPTER 4: METHODOLOGY

The proposed approach for detecting hateful memes is based on a single stream of processing using a transformer model called BERT, which is capable of handling both image and text inputs. The first step of the pipeline is to locate the text within the image and extract the textual information. This is achieved by creating image masks that cover the region of the image containing text, using the bounded boxes obtained from an Optical Character Recognition (OCR) module.

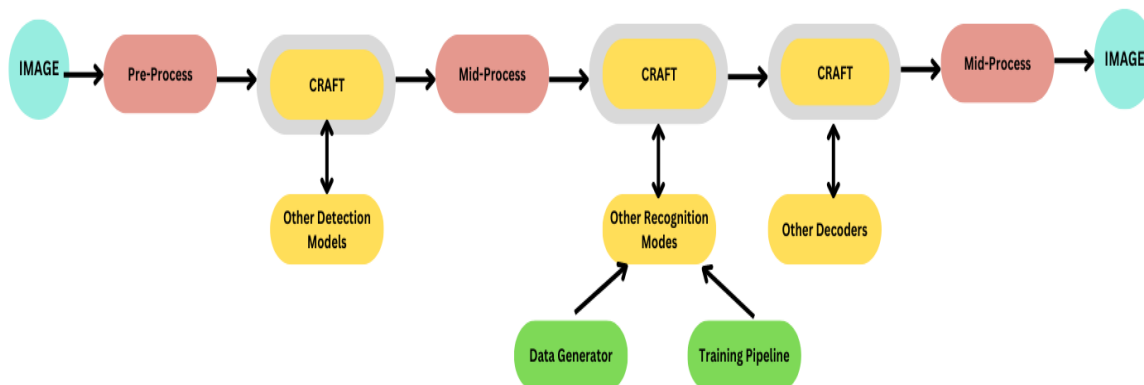


*Figure 4.1: Proposed Preprocessing Pipeline*

Once the text is extracted, the image masks are used to fill the region of text using a process called inpainting. Inpainting is a technique that fills in missing or damaged parts of an image with plausible information derived from the surrounding areas. If the image contains multiple patches, the inpainted image is passed through an Object Detection Model to extract those patches. Utilising new data sources to boost the model's performance is the next process in the pipeline. To do this, a Web Entity Detection Module is used to extract keywords from the picture patches. The generated multimodal data combines a picture with text and image-related keywords.

By using this method, the model is able to include context from the retrieved keywords as well as visual and linguistic clues from the meme. The suggested strategy seeks to increase the precision of hostile meme detection by merging these many sources of data.

## 4.1 Optical Character Recognition



*Figure 4.2: Easy OCR Framework*

Optical Character Recognition (OCR) is a process that involves using software or algorithms to recognize and convert text from an image into machine-readable text. This can be done for a variety of types of images including handwritten text, digits, signs, traffic symbols, invoices, and bank documents. Text extraction in OCR is generally done in two steps: text detection and text recognition. However, some deep neural network models can perform both steps simultaneously.

EasyOCR is a popular Python program that can be used to perform OCR, with support for over 50 languages. It has two phases: text detection and text recognition.

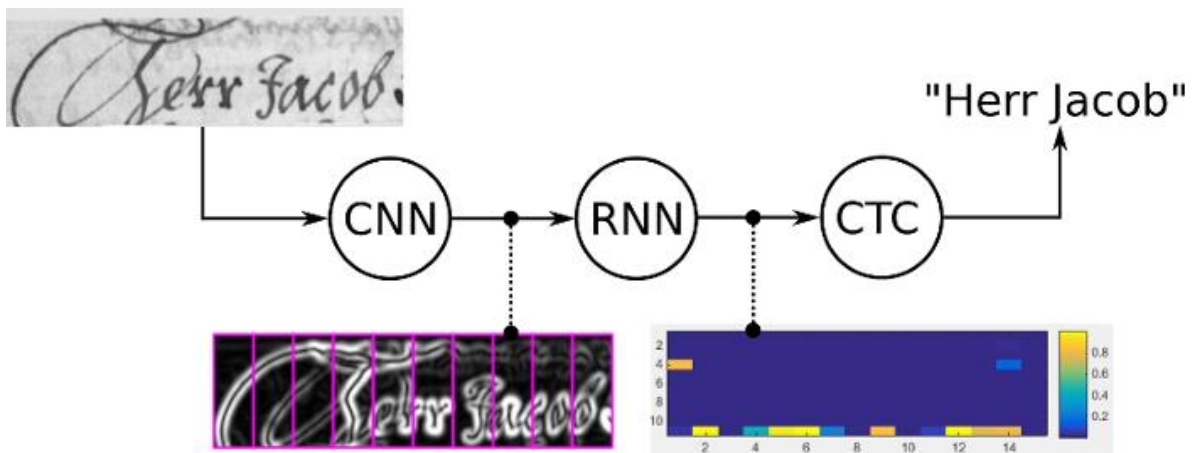
### 1. Text Detection

In the text detection phase, EasyOCR uses a pluggable detection model to locate the region on the image where the text is present. This step produces bounding boxes as output. In the text recognition phase, EasyOCR uses a pluggable recognition model to extract text from the image using the bounding boxes from the previous step. This phase involves three components: feature extraction, sequence labeling, and

decoding. EasyOCR uses ResNet for feature extraction, LSTM for sequence labeling, and Connectionist Temporal Classification (CTC) for decoding.

## 2. Text Recognition

EasyOCR is equipped with pluggable recognition models to extract text from an image using bounding boxes obtained from the text detection phase. The text recognition phase involves three key components - Feature Extraction, Sequence Labelling, and Decoding. Resnet is utilized for feature extraction, while LSTM is



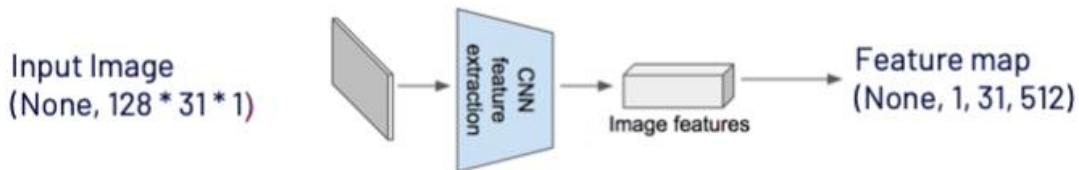
*Figure 4.3: Text Recognition Pipeline*

employed for Sequence Labelling, and Connectionist Temporal Classification (CTC) for decoding. The output of the recognition phase is raw text extracted from the input image and bounding box. The overall text recognition pipeline is depicted in Figure 4.3.

### **Text Recognition Pipeline**

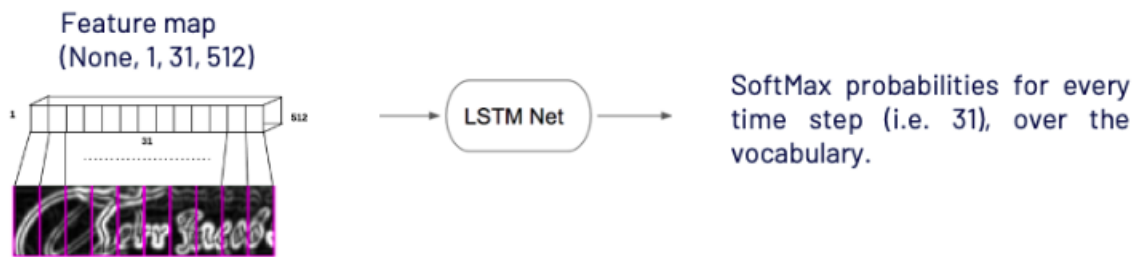
The text detection module of the EasyOCR system is used to pinpoint the areas of the picture that contain text. As seen in Figure 4.4, this module's output is a bounding box surrounding the content, which is subsequently fed into a CNN for feature

extraction. As illustrated in Figure 4.5, the resultant profile map has the dimensions (1, 31, 512), where 31 is the number of timesteps and 512 is the embedding size for each timestep. With the first timestep representing the leftmost portion of the picture and the 31st timestep representing the rightmost portion of the image, this feature map is utilised for sequence labelling using LSTM.



*Figure 4.4: CNN based Feature extraction*

To obtain the raw text from the image, the outputs from different timesteps of the LSTM are sent to a CTC decoder. Choosing the right loss function for this task can be tricky, as the length of the ground truth text may not be the same as the number of timesteps. Cross entropy loss is not a suitable choice for this task, as aligning the ground truth with the timestep count can be time-consuming. Annotating at the character level can also be time-consuming and may not be useful for obtaining character scores from the neural network.



*Figure 4.5: Sequence Labelling*

In optical character recognition (OCR) systems that handle duplicate characters in the input text, the CTC decoder is a vital component. The decoder adds special characters, such as a blank space, to distinguish between repetitive characters in order to solve this problem. The decoder determines the sum of probability for each alignment of the basic truth content in the picture after taking into account all potential alignments. Only if the total over the specific alignment has a high probability is the rating of the ground truth content regarded significant. Decoding is done using a beam search technique to obtain the raw text.

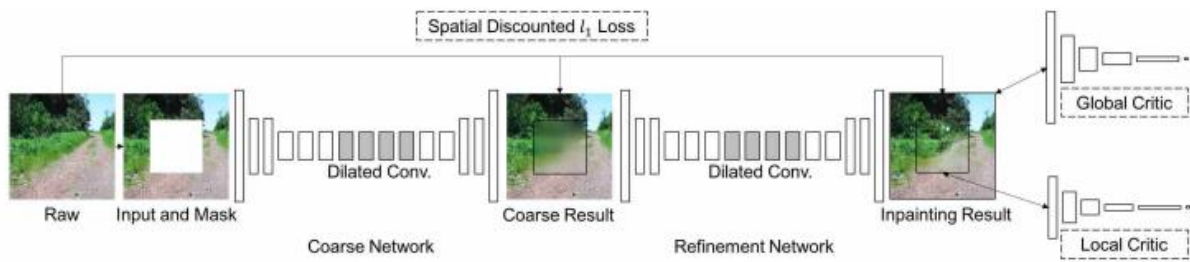
## 4.2 Inpainting Images

Image inpainting is a technique used to complete missing pixels of an image with visually realistic and semantically probable pixels that are coherent with the surrounding pixels in the image. Traditionally, this has been done by copying pixels from background regions and pasting them onto holes, starting from low to high resolution. However, this method fails when the missing regions are complex and do not contain repetitive structures or objects.

To overcome this limitation, inpainting using deep Generative Adversarial Networks (GANs) has been proposed. This approach involves formulating conditional image generation from high-level recognition using an encoder-decoder convolutional network that is jointly trained with adversarial networks to adopt to the coherency of the synthesized and existing pixels. However, using convolutional neural networks (CNNs) in this context has been found to produce artifacts, distorted structures, and blurry textures that are not consistent with nearby areas.

To address these issues, inpainting is done in two stages as shown in Figure 4.6. The first stage involves using a dilated convolutional neural network trained with reconstruction loss, while the second stage uses a contextual attention layer. The contextual attention mechanism makes use of the features of the known patches as the filters for convolution in order to process the generated regions. This layer uses convolutions to match the generated pixel regions with the existing patches, and





*Figure 4.6: Pipeline for Inpainting*



*Figure 4.7: Results of Inpainting on Hateful Memes Dataset*

takes in softmax probabilities per channel to find the weightage of the relevant patches. Deconvolution is performed to reconstruct the generated patch using the

contextual regions. A spatial propagation layer is also used to encourage spatial coherency of attention. In addition, there is an auxiliary path of convolutional layers running in parallel to the contextual attention path, which is used to generate novel contents. The output of the two paths is aggregated and given as input to a decoder for the final output. End-to-end training takes place with reconstruction loss and Wasserstein GAN (WGAN) losses. There are two WGAN losses, one for global image inpainting and the other for local region inpainting.

An example of the results of inpainting using the DeepFillv2 model on the Hateful Meme Dataset is shown in Figure 4.7. This approach has proven to be effective in generating visually realistic and semantically probable pixels that are coherent with the surrounding pixels in the image.

### **4.3 Image Patch Extractor**

The technique of locating and removing tiny image patches within a larger image is known as image patch extraction. Numerous computer vision applications, including object identification, segmentation, and classification, make extensive use of this method. The workflow for object detection as well as localization and picture patch extraction are both comparable. One of the most utilised object detection models for picture patch extraction is the Faster R-CNN method. A rapid R-CNN detector and a region proposal network (RPN) make up the two primary parts of this model. The quick R-CNN detector divides the candidate areas into several classes after the RPN creates a collection of potential areas in the picture.

The Detectron2 model may be utilised for picture patch extraction in addition to the Faster R-CNN. The PyTorch deep learning library serves as the foundation for the open-source object recognition and segmentation system known as Detectron2. Numerous training setups, support for unique datasets, and different backbone networks are just a few of the many capabilities offered by this model. Overall, picture patch extraction is a critical step in many automated vision applications, and using cutting-edge models like Faster R-CNN and Detectron2 may greatly increase

this procedure's precision and effectiveness.

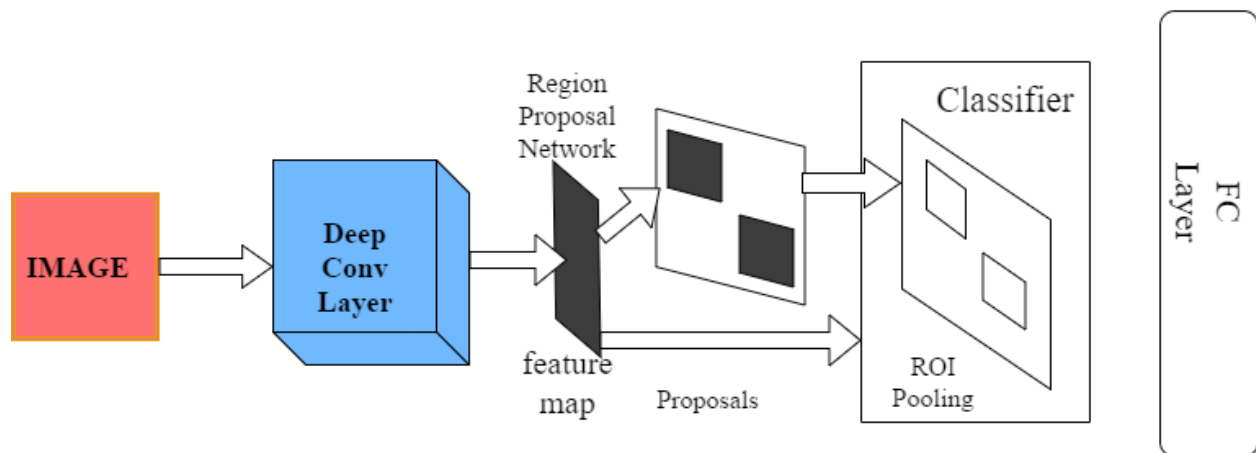
### **4.3.1 Meme Image Feature Extraction with Faster R-CNN**

The R-CNN and Fast R-CNN models have been enhanced by the Faster R-CNN model. The model comprises a number of parts and is fully differentiable. An picture is used as the model's input, and its output is a list of bounding boxes, each having labels and probabilities associated with them. To obtain feature maps using an intermediate layer, the pictures are first run through a pre-trained CNN model. Then, areas that could contain items are discovered using a Region Proposal Network (RPN). The RPN makes use of anchors, which are fixed-size bounding boxes distributed evenly over the picture. A completely convolutional layer is used by the RPN, a fully convolutional network, which is accompanied by two parallel convolutional layers. Each anchor receives two values from the classification layer, each representing how well the anchor scores as a background and foreground item. Four values that represent the deltas are generated by the regression layer, and these values are afterwards applied to the anchors to produce the bounding boxes. Anchors that have an Intersection over Union (IoU) of 0.5 or greater and intersect the ground truth object are used to identify the foreground. The anchors that share an IoU of less than 0.1 with the ground truth are regarded as background when calculating the background. To determine the binary cross-entropy loss for classification, the RPN randomly picks a small batch of size 256 that includes both foreground and background information. The regression loss - Smooth L1 loss is found using the anchors from this mini-batch that are designated as foreground.

To prevent duplication of overlapping sections, non-maximum suppression (NMS) is utilised. The NMS method takes a list of the suggested areas' scores in sorted order and eliminates any regions whose IoU exceeds a certain threshold and are associated with an alternative with an increased score. After NMS, the top-N suggestions are kept in order of score. After the RPN network has produced the bounding boxes, Region of Interest (RoI) pooling is used to acquire the characteristics of the objects included inside that specific bounding box. Utilising the area of interest pooling, the

RoI pooling reuses the convolutional feature maps for each proposal.

The R-CNN module, which is the last component of the Faster R-CNN architecture, is used to categorise the objects in the bounding box or to ignore them by applying the "background" label. The two Fully Connected (FC) layers with ReLU activation are sent by the R-CNN module after it flattens the suggestions.  $N+1$  units make up one FC layer, where  $N$  is the total number of classes and 1 represents the background class. To obtain the regression result for the bounding boxes, the other FC layer contains  $4N$  units. R-CNN training is similar to RPN training, but with R-CNN, we must take object class considerations into account. The proposals with IoU between 0.1 and 0.5 are allocated as background, and the proposals with IoU larger than 0.5 are assigned with the ground truth. The suggestions that don't intersect are rejected. Over 25% of the foreground items in a mini-batch of size 64 are given a class, and the remaining 75% are background. Cross-entropy loss characterises the categorization loss. The offset of the suggestions and the ground truth box serve as the regression FC's aim. For the suggested regions denoted as foreground, smooth L1 loss is employed.



*Figure 4.8: Architecture of Faster R-CNN model*

In the post-processing, NMS is applied class-wise by sorting the class's objects into

groups based on softmax probabilities. This is employed to set a maximum number of items per class. The assessment metric is the Mean Average Precision (mAP) at a certain IoU threshold. The R-CNN module, which is used to categorise the objects in the bounding box or ignore them by utilising the " background " label, is the last component of the Faster R-CNN architecture. To correctly suit the items, the bounding box coordinates might be further modified.

The R-CNN module in Figure 4.9 flattens the suggestions before sending them to two Fully Connected (FC) layers with ReLU activation.  $N+1$  units make up one FC layer, where  $N$  is the total number of classes and 1 represents the background class. To obtain the regression result for the bounding boxes, the other FC layer contains  $4N$  units. R-CNN training is similar to RPN training, except with R-CNN, we must take object class information into account. The suggestions with IoU between 0.1 and 0.5 are allocated as background, whereas the proposals with IoU larger than 0.5 are assigned with the ground truth. The suggestions that don't intersect are rejected.

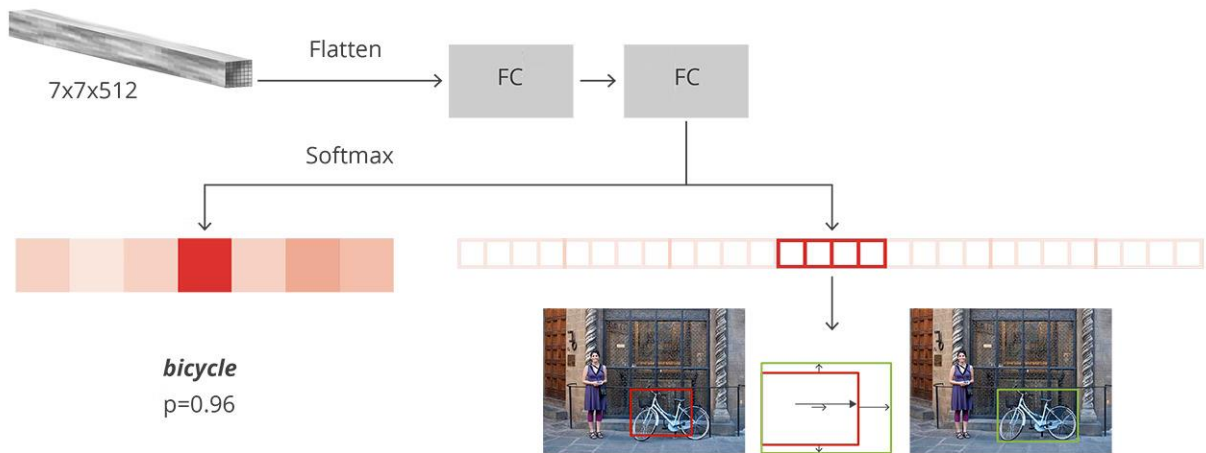


Figure 4.9: R-CNN module for Classification and Regression

A mini batch of size 64 is randomly sampled to contain over 25% of foreground with objects assigned to a class and the rest as background. The classification loss is a cross entropy loss. The target for the regression FC is the offset of the proposals and the ground truth box. Smooth L1 loss is used for the proposed regions marked as foreground. The post processing included applying NMS class wise by grouping the

objects of the class based on the softmax probabilities in sorted order. We can use this to limit the number of objects for each class. Mean Average Precision (mAP) at a certain threshold of IoU is used as the metric for evaluation. Overall, Faster R-CNN has been widely used for object detection tasks and has achieved state-of-the-art performance on various benchmark datasets.

### 4.3.2 Meme Image Feature Extraction with Detectron2

Detectron, developed by Facebook AI Research (FAIR), is a highly efficient and widely used object detection algorithm that is based on the Mask R-CNN benchmark, which is currently considered the state-of-the-art in object detection. Detectron is coded using Python and utilizes the Caffe deep learning framework, also developed by Facebook.

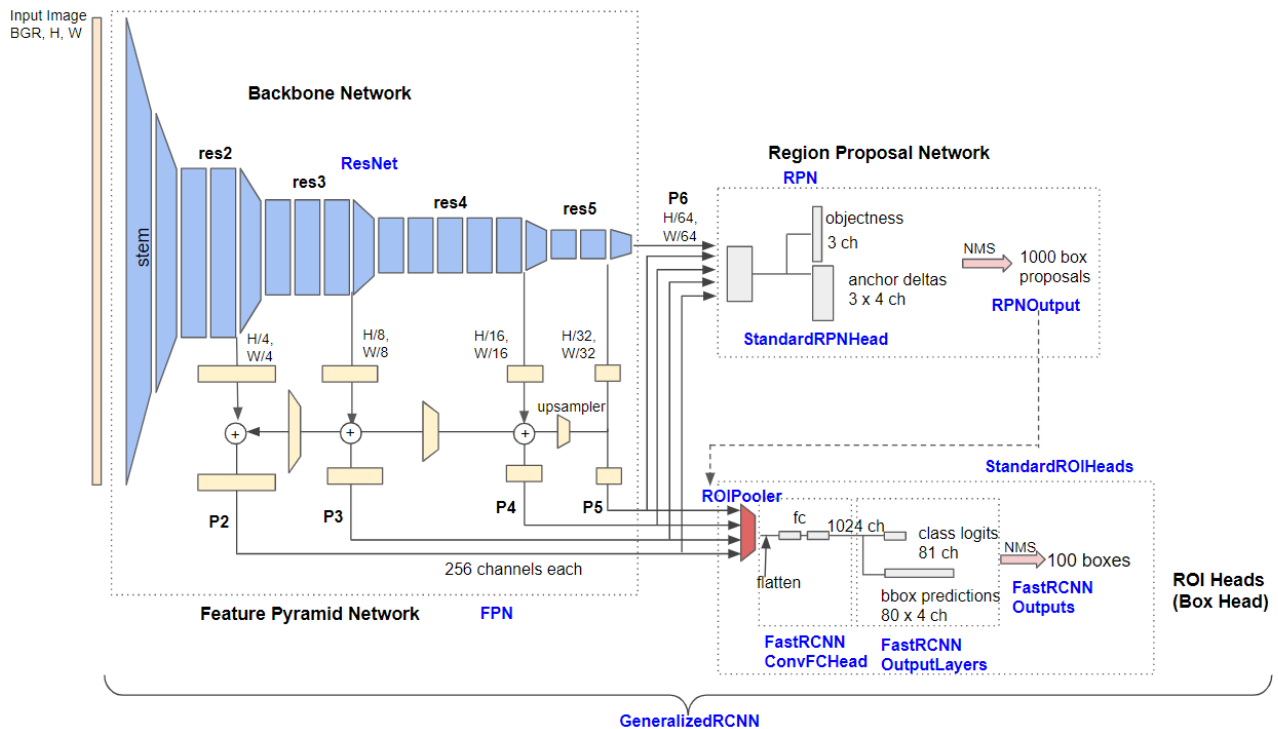


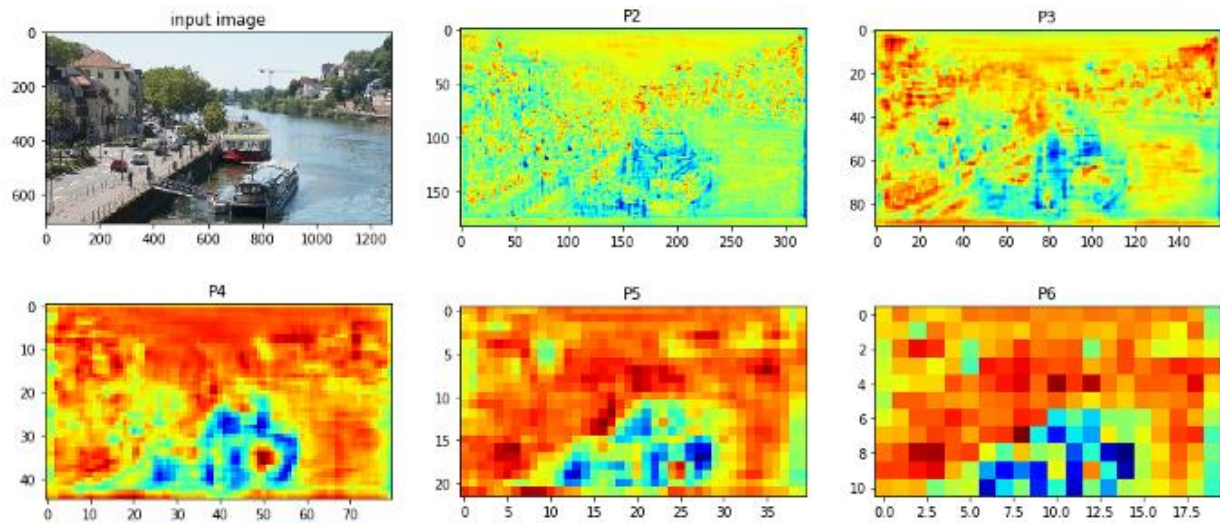
Figure 4.10: Architecture of Detectron2 model

Detectron's bounding box detection relies on the base Faster R-CNN with Feature Pyramid Network (FPN), which has become the industry standard. Detectron employs an FPN as its backbone network as shown in Figure 4.10, which is capable of extracting features at multiple scales with different fields of reception. The FPN extracts features at scales of 1/4, 1/8, and 1/16, while the res4 block utilizes 1/16 scaled features network as shown in Figure 4.11.

The RPN network in Detectron, similar to the one in Faster R-CNN, produces bounding box proposals with probability scores assigned to them, indicating foreground or background (no object) regions. In this case, 1000 proposals are used. The box-head network is then used to crop the region proposals and ensure that the object fits within the proposal regions. The box-head network outputs the fine-tuned locations for the bounding boxes and the classification results. Finally, non-maximum suppression (NMS) is used to obtain only those boxes whose intersection over union (IoU) is greater than the threshold value. This ensures that there are no redundant detections of the same object in the scene. Overall, Detectron is a powerful and efficient object detection algorithm that has become a popular choice for many computer vision tasks.

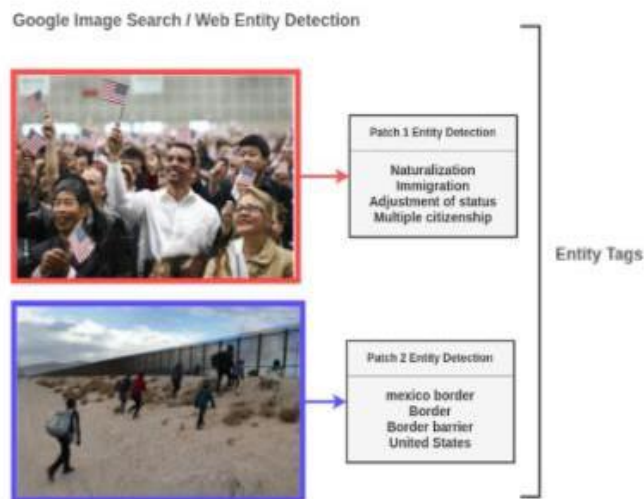
## **4.4 Web Entity Detection**

The Google Vision API is an image analysis service that utilizes machine learning to identify objects, faces, text, logos, and other entities present within images. This cloud-based service also offers optical character recognition (OCR) for text in images and can detect explicit content for image moderation.



*Figure 4.11: Feature maps with different receptive field*

Using the labels and entities detected by the API, it can find relevant information on the web or output web links. This can be leveraged by developers to create applications that search for similar images or related content on the web. Besides, Google also provides other APIs such as the Custom Search API and the Knowledge Graph API that can be utilized to find web links and information based on the content of an image.



*Figure 4.12: Example of Google Web Detection On Hateful Meme Dataset*

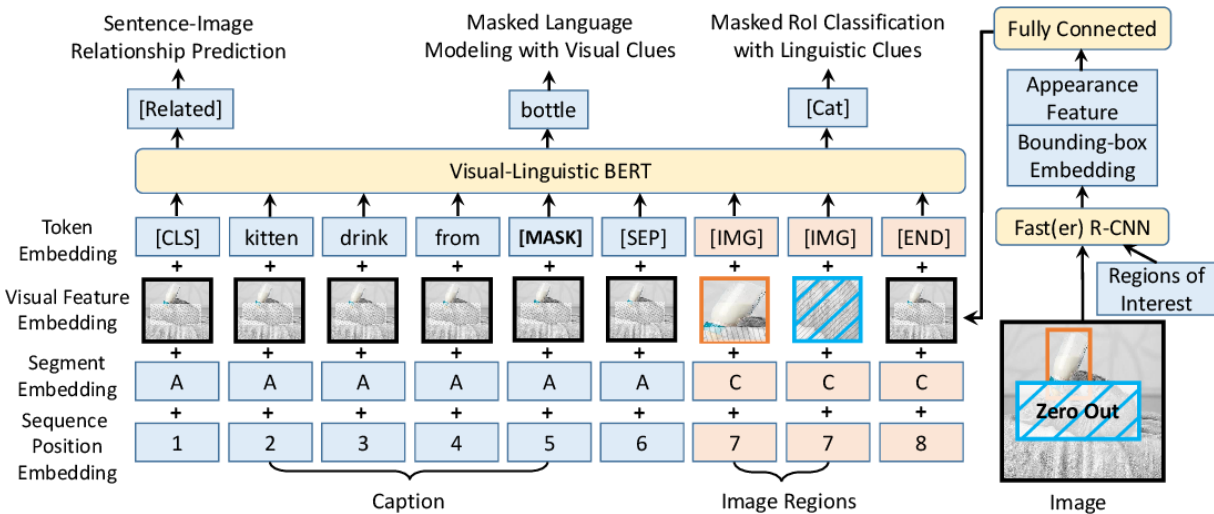


## 4.5 Single Stream Hateful Meme Classification



*Figure 4.13: OCR and Inpainting Performed on Meme*

Figure 4.1 shows the preprocessing pipeline used in the approach. The EasyOCR model is used to extract text and the corresponding bounding boxes from the OCR module. Then, masks are created in the regions of text using the bounding boxes. The DeepFillV2 model is used to inpaint the memes by taking in the image mask and the image with masked regions of text. After that, a modified Faster-RCNN model is used to extract image patches from the inpainted image. If the image is made up of multiple patches, the model splits them into multiple samples. The Google web entity detection API is used to obtain additional information for the multimodal data. This API provides information on the image patches extracted, and the keywords used in the classifier module along with the text and inpainted image. This additional source of information is used to capture the context of the image.



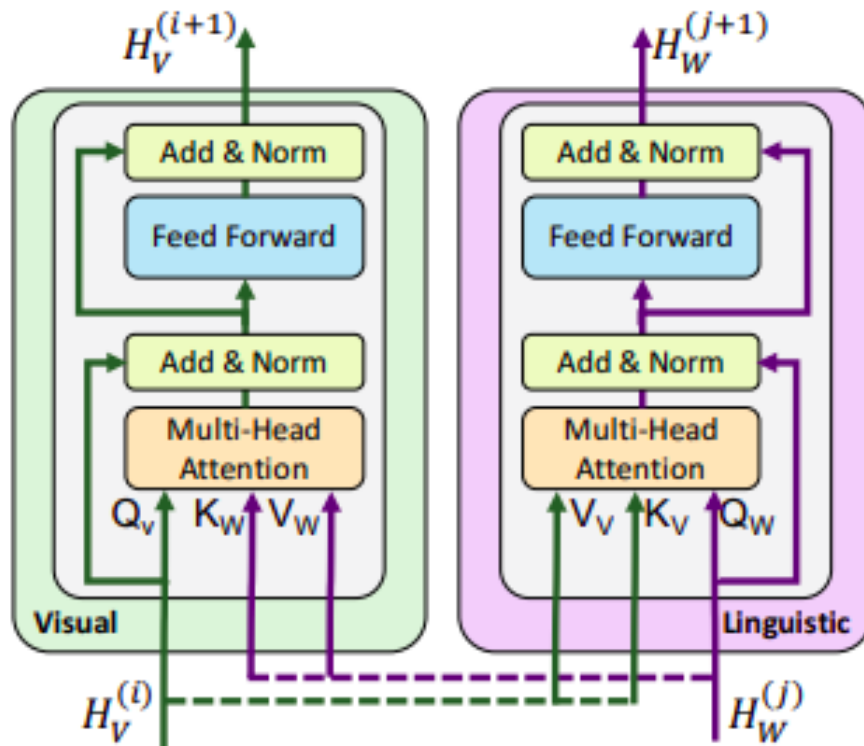
*Figure 4.14: Architecture of Modified VL-BERT*

Finally, the VL-BERT model is used to process both the image and text modalities and learn the characteristics of the images implicitly. The modified VL-BERT module takes in a sequence of the image with the embedding of keywords from the web entity detection module, image features with the text embedding, and image patches with image feature embedding. The [CLS] token is used to indicate the beginning of the sequence, [END] to indicate the end of the input sequence, and [SEP] as the delimiter between the two modalities. The model uses Masked Language Modeling by masking the text and retaining the image feature, and vice versa. The visual features are tagged either with special tokens or the textual information. The textual data in this scenario is the caption on the meme and the entity tags from the web entity detection API, and the image features are extracted from the Faster RCNN model. Both the visual features and the geometrical features are inferred from the object detection module.

## 4.6 Two Stream Hateful Meme Classification

The researchers proposed a new approach for visual-linguistic tasks that is based on the ViL-BERT architecture. However, their approach is different in that it employs

two parallel streams for processing text and image modalities. The textual stream includes tokens for the captions and data from the web entity detection module, while the image stream extracts visual features. The backbone of the model is a multi-head attention transformer, and to facilitate cross-modal learning, the architecture employs a co-attention transformer. This involves connecting the key from the textual transformer to the query of the vision transformer and vice versa.



*Figure 4.15: Co-attention transformer layer*

Through this process, the model can learn to connect the visual and textual information, enabling it to perform cross-modal training. The rule-based classification on the Polarity scores from Textblob, as shown in Figure 4.15, resulted in an accuracy of 0.55 on the test data. However, the algorithm has a very low precision on the positive class or non-hateful memes, while it has a very high precision on hateful memes. This indicates that the algorithm is better at identifying

hateful memes than non-hateful memes using the polarity scores.

New embeddings are learnt for the input tokens, which are then passed to an encoder transformer to produce an intermediate representation. This intermediate representation is used in the computation of the key, query and value. The dot product between the key and query is used to determine the distribution of attention over the values. A weighted average of the values is the output of the attention block. There are two BERT models used in this approach, one for vision and the other for language. In the multi-headed attention, the main value pairings are switched, integrating language into vision and vision into vision. For masked language modelling and next sentence prediction, a huge corpus is used to train the BERT models end-to-end.

In summary, this approach uses a new architecture that incorporates two streams of processing for the text and image modalities, using co-attention transformer learning and two BERT models trained end-to-end on a large corpus. This allows for cross-modal learning and better integration of visual and textual information.

## **CHAPTER 5: EXPERIMENTS AND RESULTS**

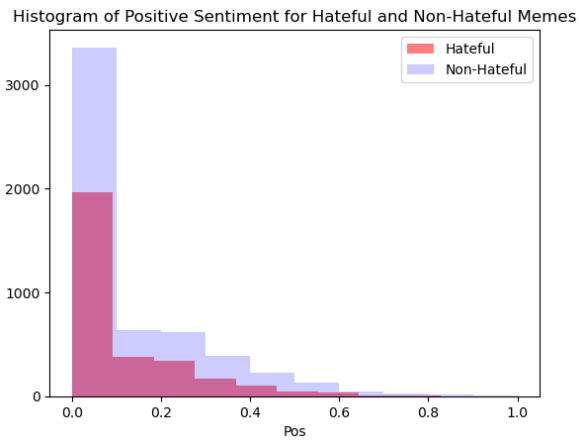
A sentiment analysis tool called VADER was created primarily to examine sentiment in online social networking posts. To ascertain the mood conveyed in a particular piece of text, it combines rules and lexicons. The programme offers polarity scores, which range from -1 (most negatively) to 1 (most positively), depending on how strongly a feeling is portrayed in the text. The separate scores represent the percentage of text that falls into each category, and the polarity scores are further divided into positive, neutral, and negative categories; they add up to 1. The compound score, which goes from -1 to 1, gives readers an overall assessment of the text's sentiment, with -1 denoting the most negative and 1 the most positive feeling.

Natural language processing features offered by TextBlob, a Python API, include sentiment analysis, noun extraction, parts of speech tagging, and translation. TextBlob's sentiment analysis tool offers two scores: polarity and subjectivity. Polarity, which goes from -1 (most negatively) to 1 (most positively), is a measurement of the sentiment conveyed in the text. As opposed to objectivity, which goes from 0 to 1, subjectivity measures how much factual information and how much personal opinion are represented in the text. A score of 0 indicates extremely objective language, while a score of 1 represents extremely subjective language.

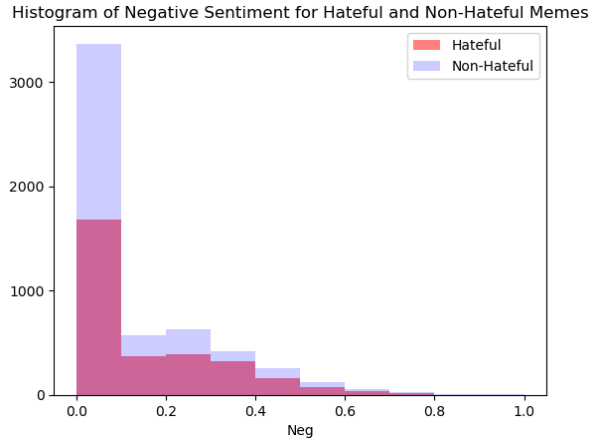
### **5.1 Analysis of VADER on Hateful Meme Dataset Textual Data**

In Figure 5.1a, the histogram corresponds to texts that VADER identifies as positive sentiment on the Hateful Meme Dataset. However, this does not necessarily indicate that the memes are non-hateful, as the image itself could still be hateful. Therefore, relying solely on text data for classification may not be accurate. It can be seen that the dataset is imbalanced when considering the text alone, with more non-hateful memes than hateful memes predicted as positive by VADER. In Figure 5.1b, for text

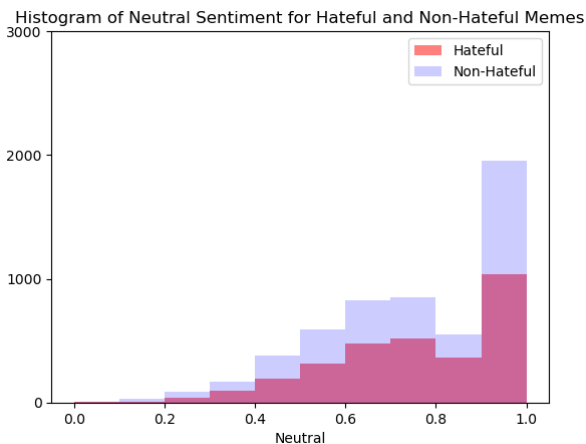
with negative sentiments, there are also more non-hateful memes. This suggests that positive text has been duplicated onto a hateful image. Rule-based classification using the positive and negative polarity resulted in an accuracy of 0.55 on the test data, as shown in Figure 5.2a. Similarly, using the compound score of VADER for rule-based classification resulted in an accuracy of 0.53 on the test data, as shown in Figure 5.1d. In both cases, precision and recall for the positive class (non-hateful) were low, likely due to positive text being duplicated on negative images.



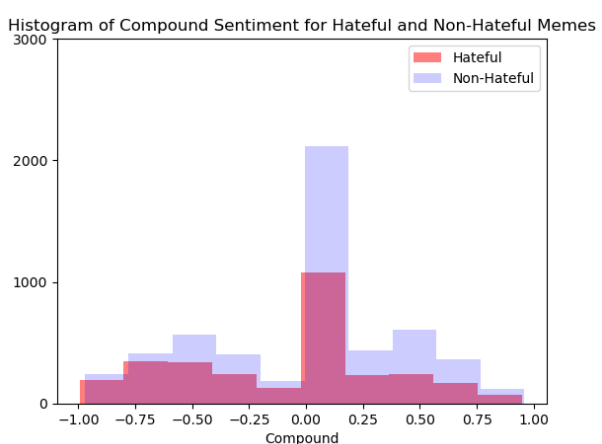
(a) Text Predicted as Positive Sentiment



(b) Text Predicted as Negative Sentiment

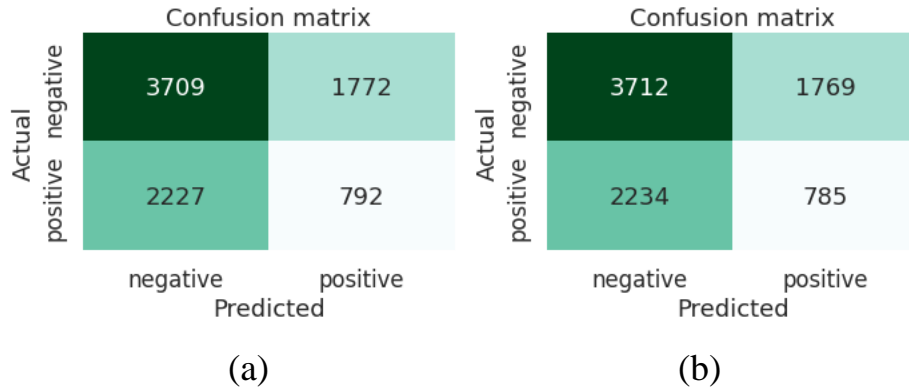


(c) Text Predicted as Neutral Sentiment



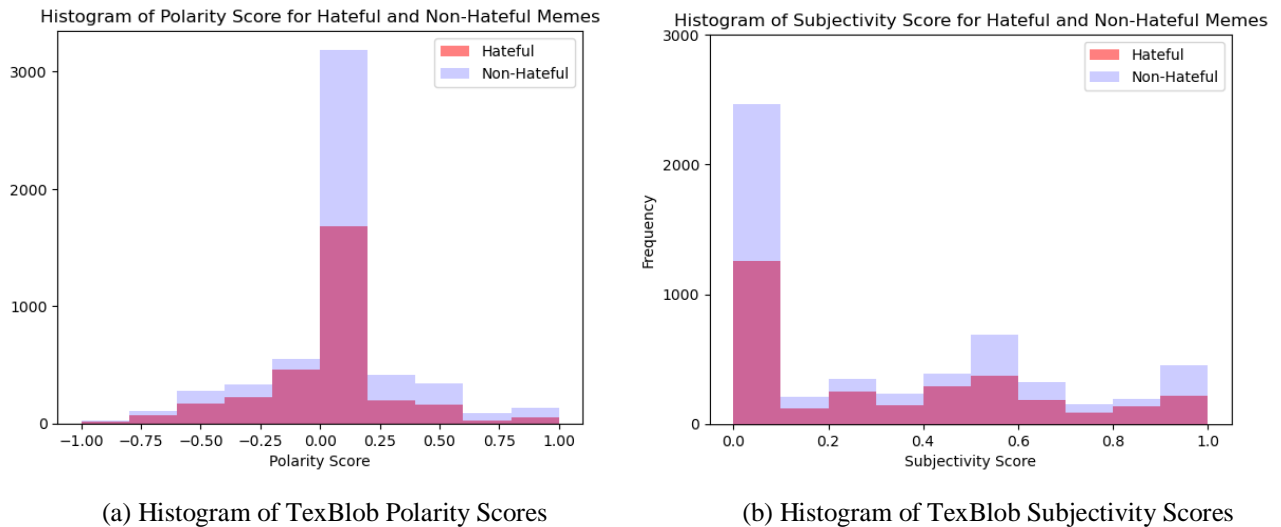
(d) Histogram for the compound score

*Figure 5.1: Results of VADER on Meme Text*



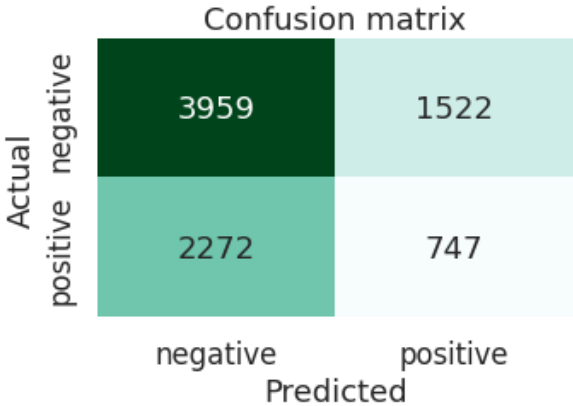
*Figure 5.2: (a) Confusion matrix for Rule based classification of VADER Polarity scores and (b) Confusion matrix for Rule based classification of VADER compound scores*

## 5.2 Analysis of TextBlob on Hateful Dataset Textual Data



*Figure 5.3: Results of TextBlob on Meme Text*

The rule-based classification on the Polarity scores from Textblob, as shown in Figure 5.4 , resulted in an accuracy of 0.55 on the test data. However, the algorithm has a very low precision on the positive class or non-hateful memes, while it has a very high precision on hateful memes. This indicates that the algorithm is better at identifying hateful memes than non-hateful memes using the polarity scores.

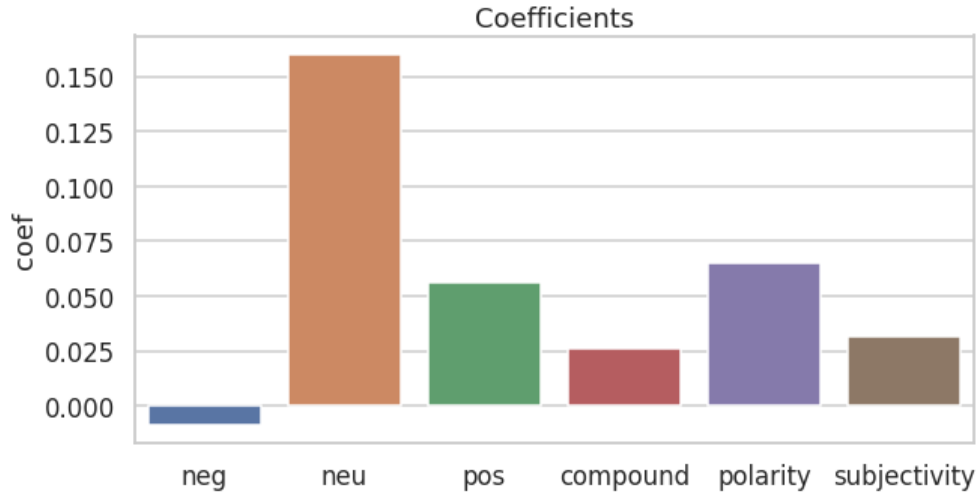


*Figure 5.4: Confusion Matrix for Rule Based Classification on TextBlob Polarity Scores*

### 5.3 Analysis of SGD on Hateful Meme Dataset Textual Data

In the natural language processing (NLP) pipeline, the term frequency inverse document frequency (TF-IDF) approach is used to convert textual input into a matrix of TF-IDF characteristics. In this experiment, TfidfVectorizer was used in combination with randomized hyperparameter search and cross-validation. The contribution of each feature to the





*Figure 5.5: Coefficients for Sentiment scores, Polarity and Subjectivity*

prediction was shown in 5.5, and the SGD classifier acquired an accuracy of 0.64 on the test data. However, although hateful memes had higher recall and accuracy, non-hateful memes had lower levels of both. The baseline performances of several unimodal and multimodal models, both with and without pretraining, are displayed in Table 5.1. Modern multimodal models pretrained on COCO and CC were able to reach a test accuracy of 75.44 percent. The main assessment metric was AUROC, which penalises models that do a poor job of ranking hatred. On the unseen test data, the initial strategy using a single-stream model had an AUROC of 0.71 and an accuracy of 74%. With an accuracy of 0.7352 for the test and 0.7650 for the dev, the second technique produced an AUROC of 0.8108 on the test unseen data and 0.7555 on the dev unseen data. Performance was improved by assembling. Figures 5.6 show the training and validation graphs after 3000 epochs of training.

## 5.4 Hyperparameter Tuning and Model Selection

Hyperparameter search involves exploring different combinations of hyperparameters to find the best configuration that results in the highest performance of the model. This process takes into account various factors such as batch size,

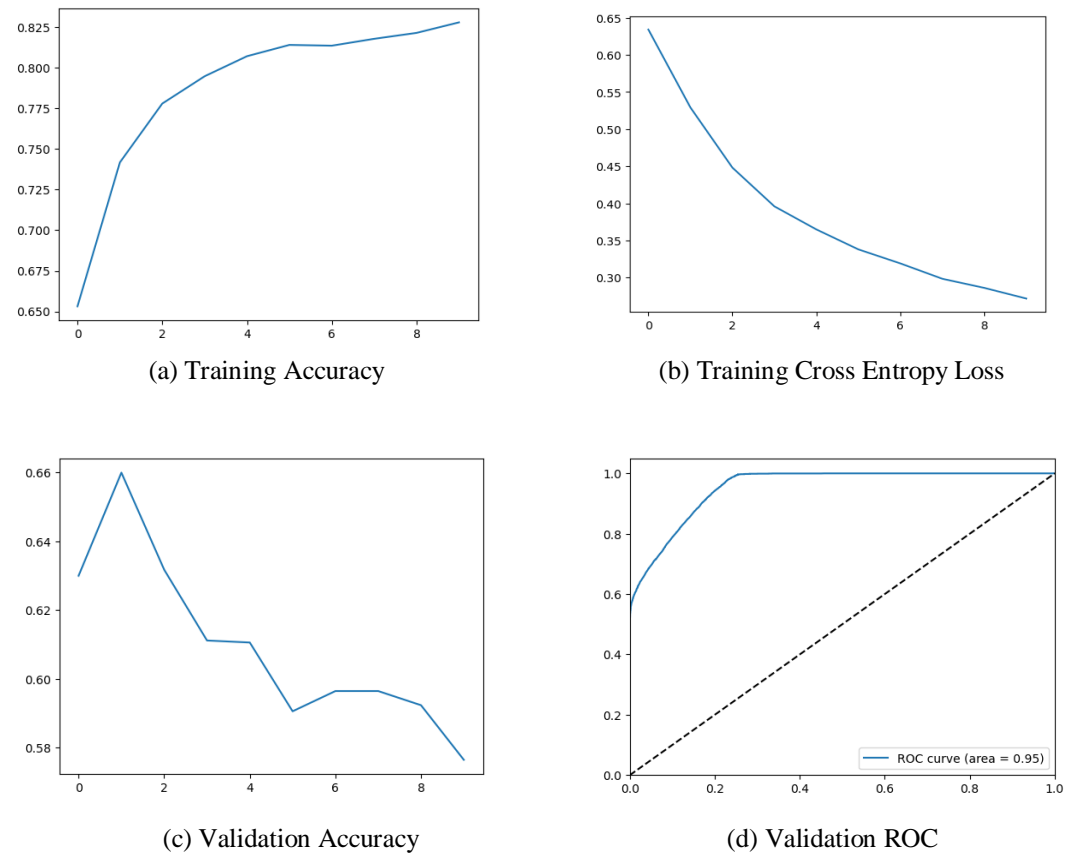
learning rate, warmup, scheduler type, and the number of training iterations. The performance of several models for the test dataset is evaluated, and only models with an ROC score of 0.76 or higher are selected based on a predefined threshold.

The majority class label is used as the final prediction in the Majority Voting approach, which then combines the predictions from the chosen models. This method is based on ensemble learning, which makes use of numerous predictors to enhance model performance. The ROC scores and hyperparameters of each of the 27 models used in the study are shown in Figure 5.8.

Type	Model	Validation		Test	
		Accuracy	AUROC	Accuracy	AUROC
	Human	-	-	84.70	-
Unimodal	<i>Image-Grid</i>	50.67	52.33	52.73±0.72	53.71±2.04
	<i>Image-Region</i>	52.23	57.24	52.36±0.23	57.74±0.73
	<i>Text BERT</i>	58.27	65.05	62.80±1.42	69.00±0.11
Multimodal (Unimodal Pretraining)	<i>Late Fusion</i>	59.39	65.07	63.23±1.09	69.30±0.33
	<i>Concat BERT</i>	59.32	65.88	61.53±0.96	67.77±0.87
	<i>MMBT-Grid</i>	58.27	66.73	62.83±2.04	69.49±0.59
	<i>MMBT-Region</i>	64.75	72.62	67.66±1.39	73.82±0.20
	<i>ViLBERT</i>	63.16	72.17	65.27±2.40	73.32±1.09
	<i>Visual BERT</i>	65.01	74.14	66.67±1.68	74.42±1.34

Multimodal	<i>ViLBERT CC</i>	66.10	73.02	65.90±1.20	74.52±0.06
(Multimodal Pretraining)	<i>Visual BERT COCO</i>	65.93	74.14	69.47±2.06	75.44±1.86

*Table 5.1: Baseline Performance on Hateful Meme Detection Dataset*



*Figure 5.6: Results of Single Stream methodology on Hateful Meme Detection Dataset*

Method	AUROC	Accuracy
VADER	0.64	0.53
TextBlob	0.63	0.55
VADER+SGD	0.64	0.64
Single Stream	0.71	0.74
Two Stream	0.81	0.76

*Table 5.2 : Hateful Meme Detection Dataset Splits*

# CHAPTER 6: HATEFUL MEME DETECTION DATASET

The paper [2] provides detailed information about the construction and annotation of the hateful meme dataset. The dataset includes 10,000 images of 5 categories: multimodal hateful memes, unimodal hateful memes, non-hateful text confounders, non-hateful image confounders, and random non-hateful examples. Confounding examples are used to create non-hateful memes that appear hateful, by replacing either the text or image with a hateful content. . The dataset aims to categorise memes into two categories, and testing measures like AUROC and accuracy are employed. The dataset is intended for fine-tuning and testing substantial pre-trained models rather than for initial training. In contrast to other large-scale picture datasets like COCO, Visual Genome, and Conceptual Captions, the size of the dataset is really rather tiny.

Input Type	Splits
Dev Images	5%
Test Images	10%
Multimodal Input with Hate	40%
Unimodal Input with Hate	10%
Benign Input and Text Confounder	20%
Random non-hateful Images	10%

*Table 6.1: Hateful Meme Detection Dataset Splits*

Hyperparameter Sweeps									Phase 2-Validation (dev_unseen)		
id	lr_ratio	use_warmup	warmup_factor	warmup_iterations	lr	batch_size	scheduler_num_warmup_steps	scheduler.type	ROC-AUC	Acc.	best iteration
18	0.6	TRUE	0.2	1.00E+03	5.00E-05	80	500	warmup_cosine	0.7757	0.7315	3150
19	0.3	TRUE	0.2	1.00E+03	5.00E-05	80	250	warmup_cosine	0.7739	0.7241	900
16	0.6	TRUE	0.2	1.00E+03	5.00E-05	80	2000	warmup_linear	0.7695	0.7167	2100
8	0.3	TRUE	0.7	1.00E+03	5.00E-05	80	250	warmup_cosine	0.7695	0.7241	700
7	0.3	TRUE	0.6	2.00E+03	5.00E-05	80	250	warmup_cosine	0.7678	0.7259	2100
21	0.6	TRUE	0.2	1.00E+03	5.00E-05	80	500	warmup_cosine	0.7677	0.7389	1950
6	0.3	TRUE	0.6	1.00E+03	5.00E-05	80	250	warmup_cosine	0.7675	0.7352	2350
15	0.3	TRUE	0.2	1.00E+03	5.00E-05	80	500	warmup_cosine	0.7671	0.713	2650
0	0.6	TRUE	0.1	1.00E+03	5.00E-05	80	500	warmup_cosine	0.7663	0.7241	1000
2	0.6	TRUE	0.1	2.00E+03	5.00E-05	80	500	warmup_cosine	0.7654	0.7259	1800
20	0.6	TRUE	0.2	1.00E+03	5.00E-05	80	2000	warmup_linear	0.7652	0.7259	3400
9	0.3	TRUE	0.3	5.00E+02	5.00E-05	80	250	warmup_cosine	0.7652	0.7204	700
13	0.3	TRUE	0.2	1.00E+03	5.00E-05	80	500	warmup_cosine	0.7651	0.7315	
23	0.3	TRUE	0.2	1.00E+03	5.00E-05	80	250	warmup_cosine	0.765	0.7222	2350
17	0.6	TRUE	0.2	1.00E+03	5.00E-05	80	500	warmup_linear	0.7649	0.7185	1400
22	0.6	TRUE	0.2	1.00E+03	5.00E-05	80	250	warmup_cosine	0.7647	0.7278	1450
1	0.6	TRUE	0.1	1.50E+03	5.00E-05	80	500	warmup_cosine	0.7641	0.6981	750
11	0.6	FALSE	0.2	1.00E+03	5.00E-05	80	2000	warmup_linear	0.7635	0.7222	
24	0.6	TRUE	0.2	1.00E+03	5.00E-05	80	500	warmup_cosine	0.7635	0.7111	750
5	0.3	TRUE	0.5	1.00E+03	5.00E-05	80	250	warmup_cosine	0.7631	0.713	950
25	0.3	TRUE	0.2	1.00E+03	5.00E-05	80	250	warmup_cosine	0.7626	0.7204	700
26	0.6	TRUE	0.2	1.00E+03	5.00E-05	80	1000	warmup_linear	0.7625	0.7278	1300
14	1	TRUE	0.2	1.00E+03	5.00E-05	80	500	warmup_cosine	0.7624	0.7241	2300
12	0.3	FALSE	0.2	1.00E+03	5.00E-05	80	2000	warmup_linear	0.7622	0.7204	
4	0.3	TRUE	0.3	2.00E+03	5.00E-05	80	250	warmup_cosine	0.7619	0.7093	550
3	0.3	TRUE	0.3	1.50E+03	5.00E-05	80	250	warmup_cosine	0.7608	0.7333	2700
10	0.3	TRUE	0.2	1.00E+03	5.00E-05	64	2000	warmup_linear	0.7605	0.7426	

*Figure 6.1: Results of Hyperparameter Training*

One of the challenges in creating this dataset is the lack of images for cultural and historical references, such as customs, practices, and events that are not well-documented in images. In addition, the interpretation of memes can be difficult for an AI tool, especially when it comes to understanding facial expressions or social behaviors. Furthermore, memes can come in different shapes and sizes, with varying placements of text, making it challenging to create fixed templates for detection.

Type	Unimodal	Multimodal	Benign	Random	Adversarial	Total
Train	1750	1300	3200	2250	-	8500

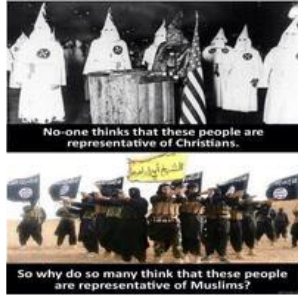
Dev-seen	50	200	200	50	-	500
Test-seen	100	400	400	100	-	1000
Dev-unseen	-	200	200	-	140	540
Test-unseen	-	729	597	-	674	2000

*Table 6.2: Data Distribution in the Hateful Meme Dataset*

Another challenge is accommodating all languages, as existing text corpus datasets are predominantly in English. Additionally, existing datasets do not contain images depicting injury, accidents, self-harm, suicide, or harm to others, making it difficult for the tool to understand memes containing such references. Overall, the compilation of this dataset emphasises the difficulty in identifying nasty memes and the need for more study and advancement in this field.



*Figure 6.1: Interchanging Background Image*



(a) Cultural Reference



(b) Historical Reference

America: Gives rights to minorities in the 60's and gets backlash for being late

South Africa:



(c) Facial Inference



(d) Inside Joke



(e) Different Template

Quando Reino Unido te coloniza y además discrimina a tus locales



(f) Non-English Meme



(g) Meme on Injury



(h) Meme of Accident



(i) Meme on Suicide

*Figure 6.2: Hard to classify Meme Templates*



## CHAPTER 7: SUMMARY

The project work involved developing an algorithm for detecting hateful memes using the Hateful Meme Detection Dataset published by Facebook. The dataset contains images with textual captions that can be classified as hateful or non-hateful. The project work was carried out in two approaches, with both approaches involving the same preprocessing pipeline.

The preprocessing pipeline first involved extracting the textual captions from the meme images using EasyOCR. Then, the images were inpainted using the DeepFillv2 model to fill the patches after removing text with synthetically coherent pixels. The patch extraction and object detection were done using either the Faster R-CNN or Detectron2 model depending on the approach. Auxillary information was gathered from the web entity detection API to aid in improving the accuracy of the test data. This additional information from the Internet results provided more context for the images, and the input to the web entity detection was just the inpainted image.

In Approach-I, a single stream of processing was used for both modalities: text and images. The VL-BERT model, which is pretrained with generic representation for VL tasks, was used to classify the memes as hateful or non-hateful. At each timestep, the textual embedding was tagged with image features, and the image features were tagged with [IMG] tokens. The VL-BERT model was fine-tuned on the Hateful Meme Dataset. The approach resulted in an AUROC of 0.71 with an accuracy of 74%.

In Approach-II, two transformer models were used to train the two modalities, and Detectron2 was used for object detection and feature extraction in images. One transformer was trained using the textual information of the meme captions and the auxillary information from the web entity detection results. The other transformer was trained on the image features extracted from Detectron2. Both transformers ran in parallel, with the key value of Text Transformer connected to the Query of the

Image Transformer and vice versa, forming the co-attention-based learning. The dataset was increased to include more memes from categories such as racism, sexism, apartheid, cyber-bullying, and child abuse, and the Memotion dataset for identifying emotions in memes was also incorporated into the Hateful Meme Detection Dataset. Hyperparameter tuning was performed to find the best configuration, and the best 27 models with an ROC of 0.76 or higher were chosen. Predictions were taken from each of the 27 models, and the class of the data point was determined by the majority voted class by these chosen models. The model resulted in an AUROC of 0.8108 on the unseen test data and 0.7555 on unseen dev data. The accuracy was 0.7352 for the test unseen and 0.7650 for the unseen dev data.

## CHAPTER 8: CONCLUSION

In our second approach, we found that combining multiple weak models generated a strong model, as demonstrated by a 2.5% boost in AUROC when using Majority voting after the Hyperparameter search. The idea behind this approach is that each individual model may have its own strengths and weaknesses, so by combining them through Majority voting, we can benefit from the strengths of each model and improve overall performance. For example, there could be a model that is an expert at detecting sexism, but not an expert in detecting racism, while another model may have the opposite behavior. By using Majority Voting, we can bring together the predictions of the experts and achieve better results. Our proposed methodology resulted in an AUROC of 0.8108 after dataset expansion and using a two-stream processing transformer model for text and image. We also utilized auxiliary information from web entity detection to increase the contextual information of the image.

Furthermore, we could consider incorporating more real-world knowledge into the training data by expanding the dataset to include more diverse examples that are relevant to different cultures and societies. We could also consider incorporating more human input, such as having human annotators label the dataset with additional information about cultural and societal references. This would help the model to learn from a more diverse set of examples and improve its ability to understand and recognize symbolism, references, and cultural context.

Another potential solution is to focus on developing specialized models for specific domains, such as politics, sports, or entertainment. This would allow the models to learn the specific symbolism and references that are relevant to those domains and improve their accuracy in detecting those types of memes.

In conclusion, while our model has shown promising results in detecting hateful

memes, it is important to acknowledge its limitations and work towards improving its ability to understand and recognize real-world knowledge and cultural context. Incorporating diverse training data, human input, and developing specialized models for specific domains are potential ways to address these limitations and improve the performance of the model.

The task of hateful meme detection has only two classes - hateful and not hateful, making it hard to perform one-shot or few-shot classification. To facilitate active learning, we can incorporate more classes or look into finding hateful GIFs. We also need to consider that the fonts and text formats on memes in reality are of varying sizes and shapes. To accommodate varying text formats, we need to integrate OCR and make it trainable end-to-end with the VL model. Referring from the related works, we found that single stream transformer models have outperformed dual-stream transformer models except for ERNIE-Vil. In the future, we could try making the ERNIE-VisualBERT single stream model to outperform the dual-stream ERNIE-Vil model.

# **BIBLIOGRAPHY**

- [1] Wikipedia contributors, "Meme - Wikipedia, the free encyclopedia," <https://en.wikipedia.org/w/index.php?title=Meme&oldid=1057164418>, 2021.
- [2] D. Kiela, H. Firooz, A. Mohan, V. Goswami, A. Singh, P. Ringshia, and D. Testuggine, "The hateful memes challenge: Detecting hate speech in multimodal memes," arXiv preprint arXiv:2005.04790, 2020.
- [3] A. Schmidt and M. Wiegand, "A survey on hate speech detection using natural language processing," in Proceedings of the fifth international workshop on natural language processing for social media, 2017, pp. 1-10.
- [4] P. Fortuna and S. Nunes, "A survey on automatic detection of hate speech in text," ACM Computing Surveys (CSUR), vol. 51, no. 4, pp. 1-30, 2018.
- [5] H. Hosseinmardi, S. A. Mattson, R. I. Rafiq, R. Han, Q. Lv, and S. Mishra, "Detection of cyberbullying incidents on the instagram social network," arXiv preprint arXiv:1503.03909, 2015.
- [6] Z. Waseem, T. Davidson, D. Warmusley, and I. Weber, "Understanding abuse: A typology of abusive language detection subtasks," arXiv preprint arXiv:1705.09899, 2017.
- [7] M. H. Ribeiro, P. H. Calais, Y. A. Santos, V. A. Almeida, and W. Meira Jr, "Characterizing and detecting hateful users on twitter," in Twelfth international AAAI conference on web and social media, 2018.
- [8] H. Zhong, H. Li, A. C. Squicciarini, S. M. Rajtmajer, C. Griffin, D. J. Miller, and C. Caragea, "Content-driven detection of cyberbullying on the instagram social network." in IJCAI, vol. 16, 2016, pp. 3952-3958.
- [9] V. Ordonez, G. Kulkarni, and T. L. Berg, "Im2text: Describing images using 1 million captioned photographs," in Neural Information Processing Systems (NIPS), 2011.
- [10] P. Young, A. Lai, M. Hodosh, and J. Hockenmaier, "From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions," Transactions of the Association for Computational Linguistics, vol. 2, pp. 67-78, 2014.

- [11] J. Xu, T. Mei, T. Yao, and Y. Rui, "Msr-vtt: A large video description dataset for bridging video and language." IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), June 2016. [Online]. Available: <https://www.microsoft.com/en-us/research/publication/msr-vtt-a-large-video-description-dataset-for-bridging-video-and-language/>
- [12] X. Chen, H. Fang, T.-Y. Lin, R. Vedantam, S. Gupta, P. Dollár, and C. L. Zitnick, "Microsoft COCO Captions: Data Collection and Evaluation Server," arXiv preprint arXiv:1504.00325, 2015.
- [13] S. Kazemzadeh, V. Ordonez, M. Matten, and T. Berg, "ReferItGame: Referring to Objects in Photographs of Natural Scenes," in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 787-798.
- [14] H. De Vries, F. Strub, S. Chandar, O. Pietquin, H. Larochelle, and A. Courville, "GuessWhat?! Visual Object Discovery through Multi-Modal Dialogue," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 5503-5512.
- [15] M. Soleymani, D. Garcia, B. Jou, B. Schuller, S.-F. Chang, and M. Pantic, "A Survey of Multimodal Sentiment Analysis," Image and Vision Computing, vol. 65, pp. 3-14, 2017.
- [16] O. Sidorov, R. Hu, M. Rohrbach, and A. Singh, "TextCaps: A Dataset for Image Captioning with Reading Comprehension," in European Conference on Computer Vision, 2020, pp. 742-758.
- [17] D. Gurari, Q. Li, A. J. Stangl, A. Guo, C. Lin, K. Grauman, J. Luo, and J. P. Bigham, "VizWiz Grand Challenge: Answering Visual Questions from Blind People," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 3608-3617.
- [18] D. Gurari, Y. Zhao, M. Zhang, and N. Bhattacharya, "Captioning Images Taken by People Who Are Blind," in European Conference on Computer Vision, 2020, pp. 417-434.
- [19] A. Suhr, S. Zhou, A. Zhang, I. Zhang, H. Bai, and Y. Artzi, "A Corpus for Reasoning about Natural Language Grounded in Photographs," arXiv preprint arXiv:1811.00491, 2018.
- [20] D. A. Hudson and C. D. Manning, "GQA: A New Dataset for Real-World Visual Reasoning and Compositional Question Answering," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 6700-6709.
- [21] N. Xie, F. Lai, D. Doran, and A. Kadav, "Visual entailment: A novel task for fine-grained image understanding," arXiv preprint arXiv:1901.06706, 2019.
- [22] A. Mogadala, M. Kalimuthu, and D. Klakow, "Trends in integration of vision and language research: A survey of tasks, datasets, and methods," Journal of Artificial Intelligence Research, vol. 70, pp. 1027-1075, 2021.
- [23] C. C. Park and G. Kim, "Expressing an image stream with a sequence of natural sentences," in Advances in Neural Information Processing Systems, vol. 28, pp. 73-81, 2015.
- [24] L. H. Li, M. Yatskar, D. Yin, C.-J. Hsieh, and K.-W. Chang, "Visualbert: A simple and performant

baseline for vision and language," arXiv preprint arXiv:1908.03557, 2019.

[25] J. Johnson, B. Hariharan, L. Van Der Maaten, L. Fei-Fei, C. Lawrence Zitnick, and R. Girshick, "Clevr: A diagnostic dataset for compositional language and elementary visual reasoning," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 2901–2910.

[26] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2818–2826.

[27] C. Schmid, S. Soatto, and C. Tomasi, "Conference on Computer Vision and Pattern Recognition," IEEE Computer Society, 2005.

[28] F. Alam, F. Ofli, and M. Imran, "Crisismmd: Multimodal twitter datasets from natural disasters," in Proceedings of the International AAI Conference on Web and Social Media, vol. 12, no. 1, 2018.

[29] X. Wang, D. Kumar, N. Thome, M. Cord, and F. Precioso, "Recipe recognition with large multimodal food dataset," in 2015 IEEE International Conference on Multimedia & Expo Workshops (ICMEW). IEEE, 2015, pp. 1–6.

[30] Y. Ma, Z. Xiang, Q. Du, and W. Fan, "Effects of user-provided photos on hotel review helpfulness: An analytical approach with deep learning," International Journal of Hospitality Management, vol. 71, pp. 120–131, 2018.

[31] Y. Li and Y. Xie, "Is a picture worth a thousand words? An empirical study of image content and social media engagement," Journal of Marketing Research, vol. 57, no. 1, pp. 1-19, 2020.

[32] I. Gallo, A. Calefati, and S. Nawaz, "Multimodal classification fusion in real-world scenarios," in 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), vol. 5, IEEE, 2017, pp. 36-41.

[33] J. Kruk, J. Lubin, K. Sikka, X. Lin, D. Jurafsky, and A. Divakaran, "Integrating text and image: Determining multimodal document intent in Instagram posts," arXiv preprint arXiv:1904.09073, 2019.

[34] J. Arevalo, T. Solorio, M. Montes-y Gómez, and F. A. González, "Gated multimodal units for information fusion," arXiv preprint arXiv:1702.01992, 2017.

[35] Z. Hussain, M. Zhang, X. Zhang, K. Ye, C. Thomas, Z. Agha, N. Ong, and A. Kovashka, "Automatic understanding of image and video advertisements," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 1705-1715.

[36] J. Yu and J. Jiang, "Adapting BERT for target-oriented multimodal sentiment classification," in IJCAI, 2019.

[37] K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu, "Fake news detection on social media: A data mining perspective," ACM SIGKDD explorations newsletter, vol. 19, no. 1, pp. 22-36, 2017.

[38] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," arXiv preprint arXiv:1607.01759, 2016.

- [39] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Computational Intelligence and Neuroscience*, vol. 2018, 2018.
- [40] J. Lu, D. Batra, D. Parikh, and S. Lee, "Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks," *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [41] G. Li, N. Duan, Y. Fang, M. Gong, and D. Jiang, "Unicoder-vl: A universal encoder for vision and language by cross-modal pre-training," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 11 336–11 344.
- [42] W. Su, X. Zhu, Y. Cao, B. Li, L. Lu, F. Wei, and J. Dai, "VI-bert: Pre-training of generic visual-linguistic representations," *arXiv preprint arXiv:1908.08530*, 2019.
- [43] Y.-C. Chen, L. Li, L. Yu, A. El Kholy, F. Ahmed, Z. Gan, Y. Cheng, and J. Liu, "Uniter: Universal image-text representation learning," in *European Conference on Computer Vision*. Springer, 2020, pp. 104–120.
- [44] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh, "Vqa: Visual question answering," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2425–2433.
- [45] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
- [46] R. Zellers, Y. Bisk, A. Farhadi, and Y. Choi, "From recognition to cognition: Visual commonsense reasoning," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.
- [47] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.
- [48] R. Zhu, "Enhance multimodal transformer with external label and in-domain pretrain: Hateful meme challenge winning solution," *CoRR*, vol. abs/2012.08290, 2020. [Online]. Available: <https://arxiv.org/abs/2012.08290>
- [49] N. Muennighoff, "Vilio: state-of-the-art visio-linguistic models applied to hateful memes," *arXiv preprint arXiv:2012.07788*, 2020.
- [50] R. Velioglu and J. Rose, "Detecting hate speech in memes using multimodal deep learning approaches: Prize-winning solution to hateful memes challenge," *arXiv preprint arXiv:2012.12975*, 2020.
- [51] P. Lippe, N. Holla, S. Chandra, S. Rajamanickam, G. Antoniou, E. Shutova, and H. Yannakoudakis, "A multimodal framework for the detection of hateful memes," *arXiv preprint arXiv:2012.12871*, 2020.
- [52] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, "Generative image inpainting with contextual attention," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5505–5514.
- [53] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, vol. 28, 2015.



[54] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," 2019, Available: <https://github.com/facebookresearch/detectron2>.

PAPER NAME

**VaishaliSharma\_Thesis.docx**

AUTHOR

**Aruna Bhat**

WORD COUNT

**12543 Words**

CHARACTER COUNT

**68511 Characters**

PAGE COUNT

**60 Pages**

FILE SIZE

**9.1MB**

SUBMISSION DATE

**May 29, 2023 6:21 PM GMT+5:30**

REPORT DATE

**May 29, 2023 6:22 PM GMT+5:30**

### ● 4% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

- 3% Internet database
- Crossref database
- 2% Submitted Works database
- 1% Publications database
- Crossref Posted Content database

### ● Excluded from Similarity Report

- Bibliographic material
- Cited material
- Quoted material
- Small Matches (Less than 10 words)