Explaining the Decisions of a Multi-Modal Hate Meme Classifier

Sydney Nguyen
Project Thesis 2
Center for Artificial Intelligence
Zurich University of Applied Sciences
sydneynguye@gmail.com

***Abstract*—This paper presents an in-depth study on enhancing and interpreting the GITforHatefulMemes model, a multimodal system designed for detecting hateful content in memes. Given the vast volume and complexity of internet memes, which combine images and text, manual detection poses notable challenges. To address this, an AI-based approach is utilized, employing a deep neural network for automatic classification. The research focuses on various strategies such as enriching text data with image captions, removing text from images to improve image captioning, selecting pre-trained models to enhance the accuracy of image captioning, increasing the dataset size, and balancing classes in the dataset. The model's performance is evaluated using various configurations, with the "Mix + 10 epochs MAMI, then Hateful Memes" model exhibiting the highest performance, achieving an AUROC score of 72.26% and accuracy of 67.1%. This represents a notable improvement over the baseline model. To enhance interpretability, the Integrated Gradients method is employed, providing insights into the decision-making process of the model for both text and visual components. The findings underscore the importance of thoughtful data selection, effective data augmentation, and the utilization of pre-trained models to achieve enhanced performance in hate speech detection. The insights gained from this study contribute to the ongoing efforts to build more robust, transparent, and trustworthy AI models for hate speech detection and moderation, fostering a safer and more inclusive digital environment.**

***Index Terms*—Hate Speech Detection, Multimodal System, Integrated Gradients Method, Data Augmentation, Model Interpretability**

I. Introduction

In the era of digital platforms, the spread of hateful content has become a pressing issue, necessitating the development of robust and transparent AI models capable of detecting hate speech. This challenge has raised interest among researchers and practitioners alike, as they battle with the complexities of multimodal data. The background of this research lies in the domain of machine learning, specifically deep learning models that have been employed to detect hate speech. However, these models often lack interpretability, making it difficult to understand their decision-making process. This opacity can lead to potential biases, unfairness, and a lack of trust in these AI systems.

This paper positions its approach at the intersection of performance enhancement and interpretability. The focus is on the GITforHatefulMemes model, a multimodal system designed to identify hate speech across text and visual data in memes. The approach is to investigate a range of strategies, such as enriching text data with image captions, removing text from images to bolster image captioning, selecting pre-trained models to augment the accuracy of image captioning, expanding the dataset size, and balancing classes within the dataset. The specific research problem is the improvement of both the performance and interpretability of the GITforHatefulMemes model. This paper aims to answer the question: How can the model's performance in detecting hate speech in memes be enhanced while also making its decision-making process more transparent and understandable?

The paper is structured as follows: After this introduction, a detailed review of related work in the field of hate speech detection and interpretability is presented. Then the methodology, detailing the various strategies employed, is described. This is followed by an evaluation of the models and a discussion of the results. The paper is then concluded with a summary of the findings, their implications, and directions for future research. This study contributes to the ongoing efforts to construct more robust and transparent AI models to combat hate speech on online platforms, fostering a safer and more inclusive digital environment.

II. Related Work

To provide an understanding of the project's foundation, this chapter examines related research and advancements in multimodal transformers, hate speech detection networks, and explainability methods used to interpret the predictions of these systems.

A. Multimodality

Multimodality in AI involves the integration of various types of data (e.g., text, images, audio) to enhance predictions and identifications [1]. This process, known as multimodal fusion, can be performed through early, late, or hybrid fusion methods. Early fusion integrates data before analysis, either by removing correlations or combining data at its lower-dimensional latent subspace. However, it can be challenging to synchronize data sources with variable sampling rates and convert them into a fixed representation [2]. Early fusion is depicted in Figure 1. Late fusion as seen in Figure 2 uses individual modality sources for fusion during decision-making, resembling human cognitive abilities, and can be integrated to generate a single

common decision. Hybrid fusion, uses deep neural networks

Fig. 1. Early fusion.

Fig. 2. Late fusion.

for intermediate fusion, changing input data to a higher-level abstraction and learning a joint representation of different modalities [2]. This is illustrated in Figure 3.

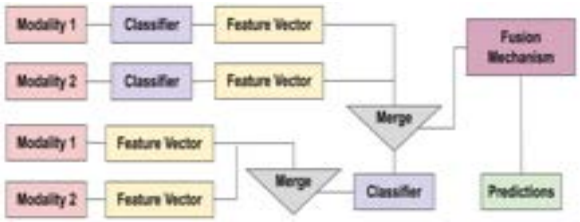

Fig. 3. Hybrid Fusion

B. Hate Speech in Networks

Research in recent years has focused on detecting hate speech in network science [3] and natural language processing [4] [5] [6]. Several hate speech datasets, primarily from Twitter [7], have been released, and various classifier architectures [8] [9] have been proposed. However, hate speech detection remains challenging and is subject to unwanted bias [10]. One reason for this is the lack of consensus on the definition of hate speech, with different terminology used in different studies.

This paper focuses on hate speech as defined in the Hateful Memes challenge report [11] which refers to speech that involves attacking individuals based on specific characteristics such as ethnicity, race, nationality, immigration status, religion, caste, sex, gender identity, sexual orientation, disability, or disease. Attacks can manifest as violent or dehumanizing speech, statements of inferiority, and calls for exclusion or segregation. Additionally, hate speech includes mocking hate crimes. The definition draws inspiration from community standards on hate speech employed by platforms like Facebook. There are certain subtle exceptions within the hate speech definition. For instance, attacking individuals or famous people is allowed as long as the attack is not based on any of the protected characteristics mentioned in the definition. Furthermore, attacking groups perpetrating hate, such as terrorist groups, is not considered hate speech. These exceptions provide necessary context for evaluating and classifying memes accurately.

There has been limited research on multimodal hate speech [11], with only a few papers considering both images and text. Yang et al. [12] found that augmenting text with image embeddings improves hate speech detection performance. Similarly, other studies [13] collected datasets of Instagram images and comments, using image features to enhance classification. The authors of [11] stand apart from these studies as their dataset is larger and deliberately designed to challenge unimodal architectures. The focus is specifically on hate speech and the authors include high-confidence ratings from trained annotators while carefully balancing the dataset for various multimodal fusion problems. Vijayaraghavan et al. [14] proposed methods for interpreting multimodal hate speech detection models using text and socio-cultural information, while Gomez et al. [15] introduced a larger Twitter-based dataset, including memes, which could serve as pretraining data for this task.

Multimodal hate speech detection includes visual question answering, image caption generation, visual reasoning, and multimodal machine translation, among others. However, many existing tasks [11] focus on autoregressive text generation or retrieval objectives, relying on bounding boxes or similar features for performance. Real-world multimodal classification problems faced by companies like Facebook or Twitter often involve large-scale, text-dominant multimodal classification. While there are related multimodal classification tasks, such as multimodal sentiment analysis [16] and various datasets using internet data, there is no agreed-upon standard dataset or benchmark task for multimodal hate speech detection. The Hateful Memes dataset [17] addresses this gap, providing a valuable resource for the research community to develop and evaluate multimodal hate speech detection models.

C. Transformer

Transformers, especially the Vanilla Transformer, can be understood from a geometrically topological perspective. The self-attention mechanism allows modeling tokenized inputs as fully-connected graphs in the topological geometry space. This flexibility sets Transformers apart from other deep networks like CNNs [18], which are limited to aligned grid spaces.

1) Vanilla Transformer: The Vanilla Transformer serves as the foundation for Transformer-based research, employing an encoder-decoder structure. It takes tokenized inputs and uses Transformer layers/blocks for both encoding and decoding. Each block contains two sub-layers: multi-head self-attention (MHSA) and position-wise fully-connected feed-forward network (FFN). Residual connections with normalization layers aid gradient backpropagation. The output of the MHSA and FFN sub-layers can be represented as $Z \leftarrow N(\text{sublayer}(Z) + Z)$, where sublayer$(\cdot)$ is the sub-layer's mapping and N$(\cdot)$ denotes normalization [19].

Input Tokenization: Vanilla Transformer utilizes tokenized sequences as input, treating each token as a node in a graph. Tokenization offers advantages such as geometrically topological flexibility, flexible information organization, compatibility with task-specific tokens, and inherent support for multimodal data processing.

Position Embedding: Vanilla Transformer employs sine and cosine functions for position embedding. Position embeddings provide temporal or spatial information to the Transformer. Their necessity depends on the input type, and they can be seen as a form of additional information.

Self-Attention (SA) and MHSA: SA is a core component of the Vanilla Transformer, enabling each element of an input sequence to attend to all other elements. SA models the input as a fully-connected graph, providing a global perception similar to Non-Local Networks. Masked Self-Attention (MSA) modifies SA to incorporate contextual dependencies and prevent attending to future positions. MHSA stacks multiple SA sub-layers in parallel, allowing the model to jointly attend to information from multiple representation subspaces. MHSA acts as an ensemble mechanism, enhancing the Transformer's ability to process diverse information.

Feed-Forward Network (FFN): The output of the multi-head attention sub-layer in the Transformer passes through a position-wise FFN. The FFN consists of successive linear layers with non-linear activation. For example, a two-layer FFN can be represented as:

$$FFN(Z) = \sigma(ZW1 + b1)W2 + b2,$$

Here, W1, b1, W2, and b2 denote the weights and biases of the linear transformations, while $\sigma()$ represents a non-linear activation function such as ReLU(·) [20] or GELU(·) [21].

2) Multimodal Transformers: The Transformer architecture can process each input as a fully-connected graph through self-attention. This allows Transformers to work with various modalities by treating the embedding of each token as a graph node. Users only need to tokenize the input and select an embedding space before inputting the data into Transformers. Tokenization and embedding approaches are highly flexible, offering alternatives such as using ROIs and CNN features, patches and linear projection, or object detection and graph features [19]. From a geometric topology perspective, each modality can be seen as a graph. For example, an RGB image represents a neat grid graph, while video and audio are clip/segment-based graphs with temporal and semantic patterns. Both uni-modal and multimodal Transformers utilize special/customized tokens as placeholders in token sequences. Common special tokens, such as [CLS] for classification and [SEP] as a separator, are defined to add semantic meaning to the token sequences. Token embedding fusion is a technique used in Transformers to combine multiple embeddings for each token position which allows for early fusion of embeddings. Token-wise summing is a common method of fusion, providing flexibility in various Transformer models, including multimodal surveillance AI. in In multimodal Transformer applications, this approach combines different embeddings using token-wise operators like addition. Examples include VisualBERT [22], Unicoder-VL [23], VL-BERT [24], InterBERT [25], and ImageBERT [26], which leverage token embedding fusion for improved performance in multimodal tasks.

In multimodal Transformers, self-attention and its variants are used for processing cross-modal interactions such as fusion and alignment [19].

D. Generative Image-to-text Transformer

The Generative Image-to-text Transformer (GIT) [27] is a model architecture specifically designed for multimodal data within the broader framework of transformers. It aims to unify vision and language tasks, including image/video captioning and question answering. This project uses GIT as the base model for the hate speech detection task. The network architecture as seen in Figure 4 is composed of an image encoder and a text decoder.

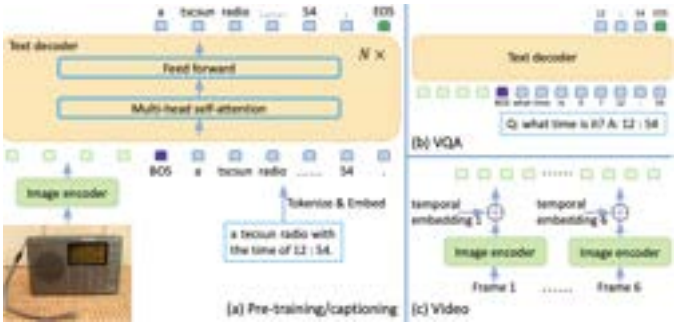

Fig. 4. Network architecture of GIT, composed of one image encoder and one text decoder. (a): The training task in both pre-training and captioning is the language modeling task to predict the associated description. (b): In VQA, the question is placed as the text prefix. (c): For video, multiple frames are sampled and encoded independently. The features are added with an extra learnable temporal embedding (initialized as 0) before concatenation. [27]

The image encoder uses a contrastive pre-trained model. It takes a raw image as input and generates a 2D feature map. This map is flattened into a list of features, which are then projected into D dimensions using a linear layer and a layernorm layer. This image encoder is used because recent studies show that contrastive tasks perform well in this context [27]. The authors also note that a stronger image encoder boosts Vision-Language (VL) performance. The text decoder is a transformer module that generates the text description. The text is tokenized, embedded into D dimensions, and then positionally encoded. The image features and the text embeddings are then concatenated and input to the transformer module, which starts decoding from the [BOS] token in an auto-regressive manner until the [EOS] token or the maximum step limit is reached. The text token only depends on preceding tokens and all image tokens, while image tokens can attend to each other. This is different from a unidirectional attention mask. Instead of initializing the image encoder, the text decoder is randomly initialized. This is because random initialization has shown similar performance compared to BERT [28] initialization in previous studies. Additionally, the BERT initialization cannot comprehend the image signal, which is important for VL tasks. This approach makes it easier to explore different design choices without being dependent on initialization.

E. Explainable AI

Explainable AI (XAI) involves understanding the working mechanism and decision-making process of AI systems. It aims to answer questions like why the system made a particular prediction (interpretability) and how the system came to a specific decision (explainability) [29].

Model-specific explanations apply to a specific model, while model-agnostic methods are independent and irrespective of

the model. Feature attribution-based methods highlight image regions that are contributors to decision-making. Distillation methods build an approximate local model or a surrogate model on top of the original model for interpretation. Intrinsic methods are explainable and self-explain using models' attention mechanisms to focus on important visual and textual regions. This category includes joint training approaches that combine predictions and explanations. Figure 5 shows the taxonomy of various deep explainability methods for both unimodal and multimodal scenarios [2].

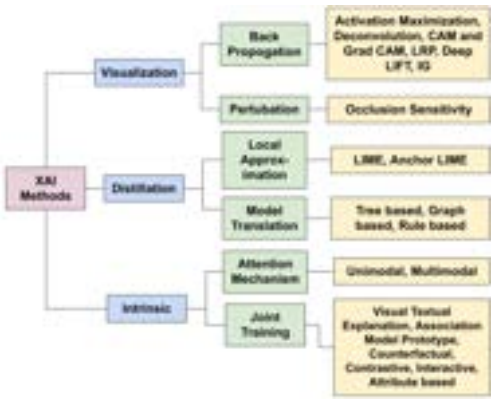

Fig. 5. XAI methods.

1) Explainability Methods for Multimodal Systems: The following methods [2] aim to provide interpretable explanations for tasks like visual question answering, image captioning, and common sense reasoning.

Attention-based methods assign more weight and importance to specific factors in multimodal data. They are commonly used in tasks like visual captioning and visual question answering, where attention mechanisms align and fuse information from different modalities. These approaches generate explanations based on attention features and improve interpretability. However, challenges exist in evaluating and ensuring consistent explanations.

Counterfactual explanations focus on contrasting decisions and causal understanding. They recommend actionable insights and minimal changes to achieve desired outcomes. Counterfactual approaches have been applied in visual question answering, visual captioning, and image description tasks to analyze model behavior and improve predictions.

Interactive explanations involve user feedback by combining model explanations, user annotations, and active learning to rectify incorrect predictions and enhance user trust. They have been applied to tasks like VQA and customer relationship management.

Graph-based methods leverage scene graphs and knowledge graphs to improve explanation quality. Scene graphs represent relationships between objects in an image, while knowledge graphs incorporate semantic information. These approaches enhance interpretability in tasks like visual question answering and neuro symbolic AI.

Attribute-based methods focus on the importance of attributes in generating explanations. They associate visual features with attribute information to provide class-discriminative and concept-explaining explanations. Attribute maps, counter attributes, and spatiotemporal attention mechanisms are utilized to improve interpretability.

2) Integrated Gradients: Integrated Gradients (IG) is an attribute-based model interpretability technique that assigns importance scores to input features. It does so by approximating the integral of gradients of the model's output concerning the inputs along a straight-line path from given baselines to inputs. IG relies on two fundamental axioms, namely Sensitivity and Implementation Invariance, as stated in Definition 1 and Definition 2 of the paper "Axiomatic Attribution for Deep Networks" [30]. These axioms hold significance as they are believed to be essential traits of all attribution methods.

Definition 1 (Axiom: Sensitivity) states that an attribution method satisfies Sensitivity if, when presented with two inputs and baselines that differ in only one feature, resulting in different predictions, the differing feature is attributed a non-zero value. In cases where the deep network's mathematical implementation does not rely on a particular variable, the attribution to that variable should always be zero.

Definition 2 (Axiom: Implementation Invariance) defines functional equivalence between two networks, indicating that their outputs are identical for all inputs despite varying implementations. An attribution method satisfying Implementation Invariance would ensure that attributions remain the same for functionally equivalent networks.

The sensitivity axiom requires a baseline which is defined as an absence of a feature in an input. It can be understood as an "input from the input space that produces a neutral prediction." By treating the baseline as an input, counterfactual explanations can be generated, exploring how the model behaves while transitioning from the baseline to the original image.

The authors contend that gradient-based methods violate Sensitivity (Def. 1). To illustrate this, they present a simple function, $f(x) = 1 - ReLU(1 - x)$, as shown in Figure 6. When attempting to generate attribution for $x = 2$, the function's output changes from 0 to 1, but after $x = 1$, it becomes flat, resulting in a gradient of zero. Although x contributes to the result, the flatness of the function at the input being tested leads to invalid attribution, breaking Sensitivity. Breaking Sensitivity causes gradients to focus on irrelevant features [30].

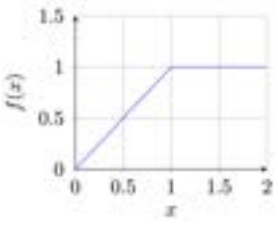

Fig. 6. $f(x) = 1 - ReLU(1x)$ where $x \in [0, 2]$

a) Computing Integrated Gradients: In the definition of IG, function F represents the model, where the input $x \in R^n$ (with n denoting the number of dimensions) and the baseline $x' \in R^n$. The method involves computing gradients along a straight-line path between x and x'. The integrated gradient along the i^{th} dimension is formally defined as shown in equation 1.

$$IG_i(x) := (x_i - x_i') * \int_{\alpha=0}^{1} \frac{\delta F(x' + \alpha * (x - x'))}{\delta x_i} d\alpha \quad (1)$$

However, since the original definition involves an integral, it is infeasible to calculate directly. Therefore, the practical implementation of IG utilizes an approximation by replacing the integral with a summation, as in equation 2.

$$IG_i^{approx}(x) := (x_i - x_i') * \sum_{k=1}^{m} \frac{\delta F(x' + \frac{k}{m} * (x - x'))}{\delta x_i} * \frac{1}{m} \quad (2)$$

To obtain the approximated calculation (equation 2), parameter m is used to define the number of interpolation steps. For example, when visualizing the interpolations with m equals five (see Figure 7), the process can be better understood. In practice, the number of interpolation steps typically ranges from 20 to 300, with the most common value being 50. The results of applying IG can be observed in Figure 8.

Fig. 7. Five-step interpolation between the baseline x' and the input image x. The first image on the left (alpha:0.0) is not a part of the interpolation process [31].

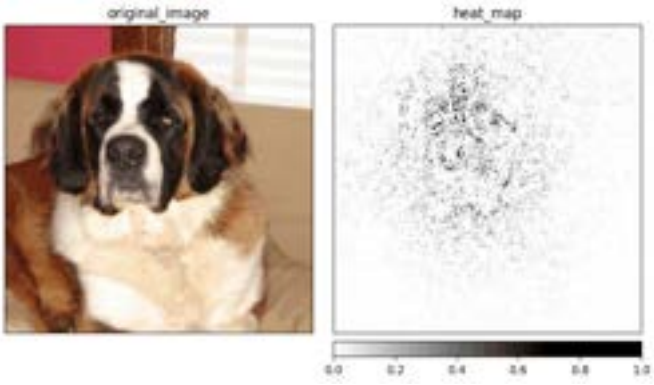

Fig. 8. Visualization of the saliency map by the IG generated for the class saint-bernard. The result is averaged over 50 interpolation steps [31].

III. Methodology

A. Problem Statement

The task is to classify memes consisting of an image and pre-extracted text, into two categories: hateful or non-hateful. The classification is based on whether the meme aligns with the definition of hate speech provided in paper [11]. The model must analyze both the image and text inputs, conditioning on their joint information to generate the binary classification output.

B. Dataset Description

1) Hatefulness Dataset: The Hateful Memes dataset is a multimodal dataset consisting of images and text designed for hate speech detection [17].

a) Qualitative Analysis: Memes are known for their subtle nature, where their true underlying meaning may be easy for humans to detect but challenging for AI systems. The Hateful Memes dataset is designed to measure true multimodal understanding and reasoning. By including so-called "benign confounders", where alternative images or captions flip the label from hateful to not-hateful, the dataset challenges models to exhibit sophisticated multimodal reasoning. This requirement ensures that models must consider both image and text modalities to accurately classify memes. Examples are seen in Figure 9.

Fig. 9. Multimodal hateful memes and benign confounders, for illustrative purposes. Hateful memes (left), benign image confounders (middle) and benign text confounders (right) [17].

b) Meme Type Distribution: The Hateful Memes dataset comprises 10'000 memes, divided into test and training sets. It is balanced, covering multimodal hate, unimodal hate, benign text confounders, benign image confounders, and random non-hateful examples. The labels indicate whether each meme is hateful or not, with the following distribution as plotted in Figure 10:

- Multimodal Hate: Memes exhibiting hate in both the image and text modalities. 40%
- Unimodal Hate: Memes demonstrating hate in either the image or text modality. 10%
- Benign Text Confounders: Memes in which the original text is replaced with alternative text, leading to a change in label from hateful to not-hateful. 20%
- Benign Image Confounders: Memes in which the original image is substituted with an alternative image, resulting in a change in label from hateful to not-hateful. 20%
- Random Non-Hateful: Memes randomly selected and classified as non-hateful. 10%

2) Additional Dataset: To address model overfitting, additional data from external sources is sought. A search on Kaggle

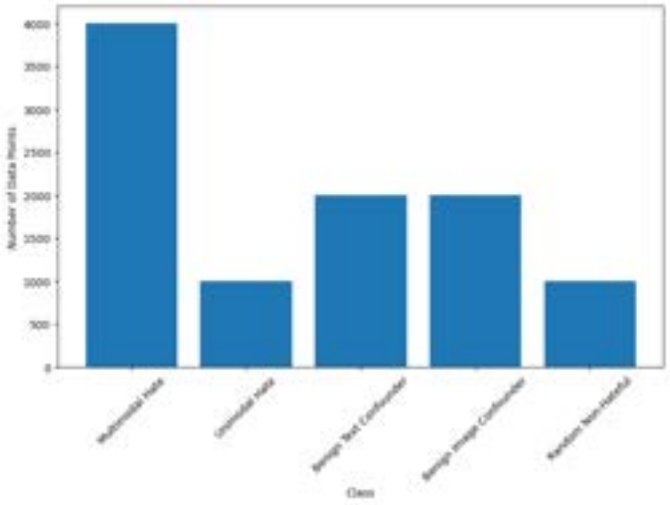

Fig. 10. Hateful Memes dataset meme type distribution.

led to the discovery of several datasets. Three potential datasets are identified:

- Memotion dataset [32], comprising 7'000 labeled data points based on offensiveness levels. However, upon closer examination, it is found that the nature of offensiveness in this dataset differs from that of the hateful memes dataset as their definition of hatefulness is not aligned with the definition used in this project, making it unsuitable for this purposes.
- MultiOFF dataset [33] is considered, however, it is not publicly available, preventing its use in this study.
- The Multimedia Automatic Misogyny Identification (MAMI) dataset [34] is also evaluated. Due to a partial overlap in the hateful speech definition, the MAMI dataset is selected for further work.

The search for additional data results in the selection of the MAMI dataset to supplement the existing data in tackling model overfitting.

C. GITforHatefulMemes Model Architecture

The model, named GITforHatefulMemes, is built upon the GIT pre-trained model, implemented using the PyTorch [35] framework. This architecture includes several key components, each serving a specific role in the model's operation:

1) Base Model: The GIT model, pre-trained with "git-base-coco" [27] checkpoint, forms the core of the GITforHatefulMemes model. This base model is loaded with the transformers library [36], enabling efficient usage of the pre-trained weights.
2) Output Layer Modification: To adapt the model for classification tasks, a fully connected (fc) layer is introduced. This modification adjusts the output layer to align with the desired number of classes for classification. The task involves binary classification, hence the output size is set to 2.
3) Loss Function: For training, the Cross-Entropy loss [37] is utilized as the criterion. This loss function efficiently computes the discrepancy between predicted logits and the provided labels, driving the model toward accurate predictions.
4) Forward Pass: The forward method orchestrates the model's forward pass. It takes text and image embeddings as inputs, representing the tokenized input and corresponding image pixel values, respectively. The method then processes these inputs through the base model, generating logits, which are the raw predicted scores for each class. Subsequently, the logits pass through the fc layer to produce the final predicted probabilities.
5) Loss Calculation: If label information is provided during training, the loss using the criterion and the predicted logits are computed in the forward pass. This loss value guides the model's learning process. If no labels are provided (e.g., during inference or evaluation), the loss is set to None.
6) Prediction Generation: The forward method identifies the predicted class with the highest probability from the logits. Furthermore, the softmax function [38] is applied to obtain normalized probabilities, representing the model's confidence in its predictions.

A workflow visualization is depicted in 11.

Fig. 11. GITforHatefulMemes model architecture.

D. Pre-trained Model Selection

The Pre-trained Model selection phase involves evaluating different pre-trained models available on HuggingFace [39] to achieve optimal model performance. The focus is on identifying the most suitable model based on its ability to generate captions that align with the hatefulness definition.

Several pre-trained models are assessed, including GIT-base trained on 10 million image-text pairs and GIT-base-coco, which was additionally fine-tuned on 10 million image-text pairs from the Coco dataset [40]. The evaluation also considered GIT-large, trained on 20 million image-text pairs, and GIT-large-coco, fine-tuned on 20 million image-text pairs from the Coco dataset.

E. Data Preprocessing and Augmentation Techniques

Two main methods are implemented to enhance model performance:

1) Generating Image Captions: To provide the model with more contextual information, image captions are generated for each image. These captions are then concatenated with the original text data, using the [SEP] token.
2) Removing Meme Text from Images: Captioning processes often include the meme text along with the caption, even though the meme text may not be directly related

to the actual content depicted in the image. An example can be found in the appendix in Table III. With the ”inpainted” method, the meme text is ”blurred out” or removed from the image. This ensures that the text no longer influences the image captioning process and allows the model to focus solely on the visual content of the image.

IV. Experimental Setup

A. Train and Test Set

The dataset used in the experiments is divided into two main subsets: the train set and the test set. The official Hateful Memes Challenge provided the initial split for these sets. In this project, the initial train set is used in the first few experiments. In a later iteration, the train set is augmented by incorporating additional data from the MAMI dataset. However, the test set remains unchanged, following the same composition as the original challenge. The mixed train set comprises two main datasets: the MAMI dataset and the Hateful Memes dataset. The MAMI dataset contains 1’547 instances labeled as 0 (non-hateful) and 1688 instances labeled as 1 (hateful). Similarly, the Hateful Memes dataset consists of 5’874 instances labeled as 0 and 3266 instances labeled as 1: Collectively, the train set contains 7’421 non-hateful and 4’954 hateful labels totally as plotted in Figure 12.

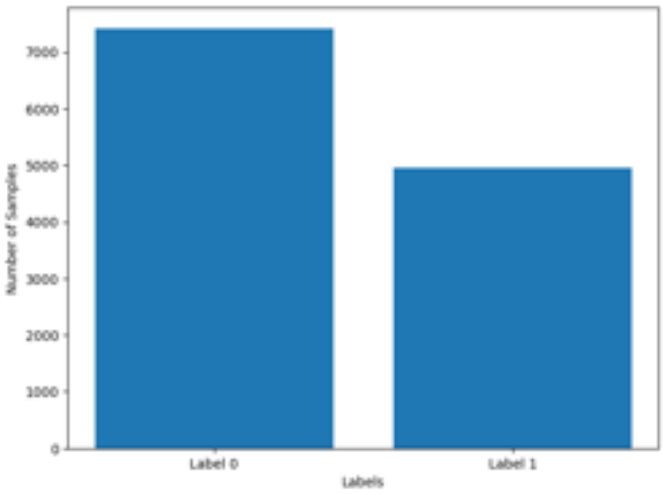

Fig. 12. Train set label distribution.

The test set contains only Hateful Memes data with 510 non-hateful and 490 hateful labels as seen in Figure 13 for all experiments.

B. Evaluation Protocol

The evaluation of model performance in the experiments relies on the Area Under the Receiver Operating Characteristic curve (AUROC) [41] as the primary performance metric. AUROC provides an assessment of the model’s ability to discriminate between hateful and non-hateful memes by considering the trade-off between true positive and false positive rates.

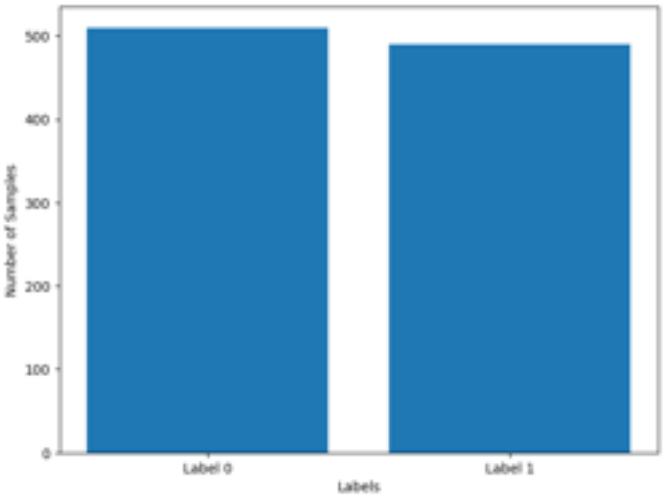

Fig. 13. Test set label distribution.

C. Implementation Details

The experimental setup for all conducted experiments in this study follows the configuration described in the paper GIT [27]. The image encoder is initialized using a pre-trained contrastive model. The hidden dimension (D) for the network’s hidden layers is set to 768, and the text decoder is composed of 6 transformer blocks, which are randomly initialized. The model contains a total of 0.7 billion parameters. During the inference stage, a beam size of 4 is used, along with a length penalty of 0.6. For this study’s specific experiments, the learning rate is set to 5e-5, and the Adam optimizer is used during training. Due to hardware constraints, the batch size is set to 8 for the data loader for each training and evaluation. To avoid overfitting and optimize training efficiency, early stopping is implemented to halt training when no performance improvement is observed. The model is evaluated on the test set every 5 epochs, and a checkpoint of the model is saved to enable loading of model weights for future iterations.

V. Results

The results are obtained through a series of experiments as described in each corresponding subsection. Seven distinct models are trained and their respective performance metrics are compared. Hereby, the best AUROC score in any epoch of each model is compared. The configuration for each model is determined based on insights gained from the preceding experiments, with the objective of enhancing overall performance. The baseline model, denoted as the ”Base Model,” serves as the starting point for comparison. The Base Model is the GITforHatefulMemes model trained on the original train set containing solely the Hateful Memes dataset. Table I summarizes each model’s best performance. All plots showcasing model performance on the test set are generated using spline interpolation [42] with a smoothing parameter (s) set to 1 to ensure a more visually refined representation for easier analysis. Among all models evaluated, the ”Mix +

Model	AUROC [%]	Accuracy [%]
Base Model	0.5946	58.1
Base Model + Caption	0.6586	59.5
Base Model + Inpainted Caption	0.6675	62.7
Coco Model + Inpainted Caption	0.6956	64.6
Mix	0.7193	65.8
Mix + 10 epochs MAMI, then Hateful Memes	0.7226	67.1
Mix + Balanced Classes	0.6804	64.0

TABLE I
MODEL PERFORMANCE

10 epochs MAMI, then Hateful Memes" model achieves the highest performance, with an AUROC score of 72.26% and an accuracy of 67.1%. This represents a notable improvement over the baseline model's AUROC score of 21.53% and accuracy of 15.49%. In Figure 14, the AUROC scores are presented for all the evaluated models.

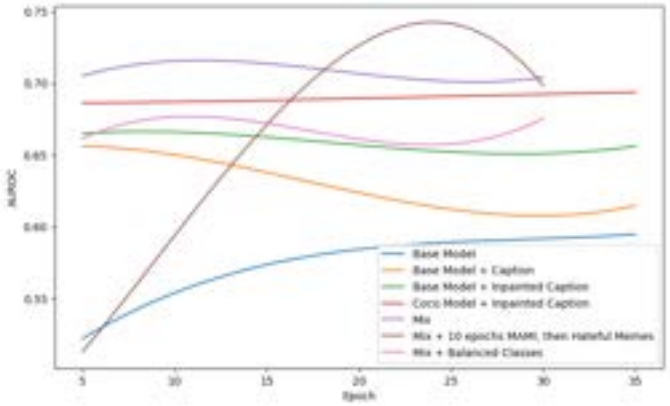

Fig. 14. AUROC [%] all models.

Figure 15 presents the evaluation results of the "Base Model", including training accuracy, test AUROC, and text accuracy. A visible gap between training accuracy and test accuracy is observed, indicating strong overfitting in the model. The subsequent subsections detail the strategies employed to address overfitting and enhance model performance. Comprehensive metric plots for all models are provided in the appendix for reference.

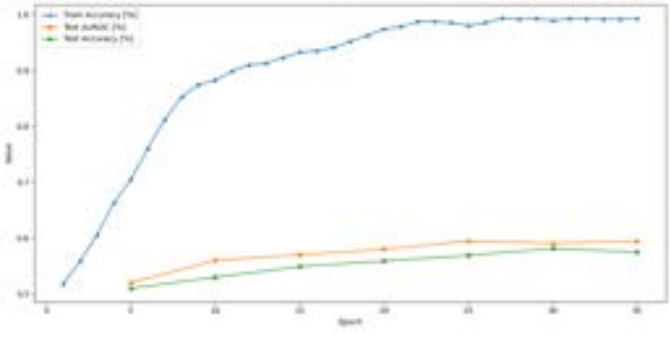

Fig. 15. Metrics for Base Model.

A. Benefits of enriching text data with image captions

By generating and concatenating captions to the sample text, the input data was enhanced with additional image descriptions, leading to an improvement in the model's performance ("Base Model + Caption"). The results seen in Figure 16 demonstrate an increase of 10.76% in AUROC compared to the base model.

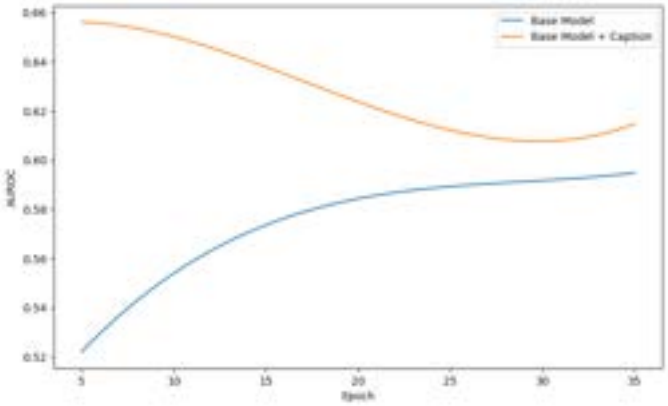

Fig. 16. AUROC [%] of baseline model and model enriched with captions, highlighting the efficacy of this data enrichment technique.

B. Benefit of Text Removal on Image Captioning

The removal of text from images improves image captioning, thereby enhancing model performance. By applying the inpainted method to remove text from the images, a more reliable image captioning has resulted, leading to improved model performance ("Base Model + Inpainted Caption"). By eliminating irrelevant text from the images, the model's focus on visual content is enhanced, resulting in a slight increase of 1.35% in AUROC compared to the model based on the original captions. This is plotted in Figure 17.

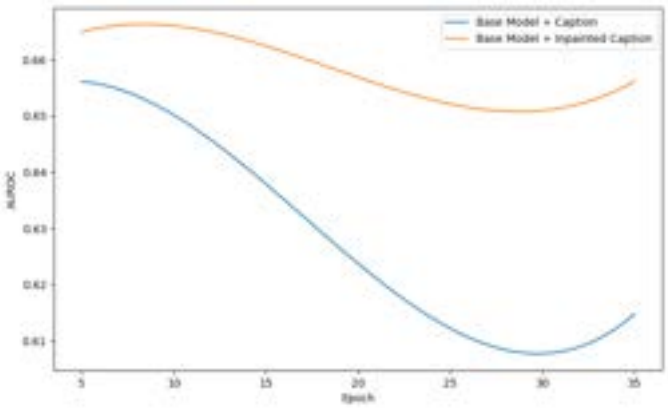

Fig. 17. AUROC [%] of training on original images and inpainted images.

C. Benefit of Pre-Trained Model Selection

The selection of pre-trained models positively affects the accuracy of image captioning for the hate speech detection

task. This experiment employs a qualitative analysis to evaluate the quality of generated captions by different pre-trained models. Captions are compared to the corresponding images to assess their informativeness and alignment with the hatefulness definition. Table II shows the results which indicate that "GIT-base-coco" provides more informative attributes related to ethnicity and gender, which are essential for certain hate speech classifications. Interestingly, minimal differences are observed between "GIT-base-coco" and "GIT-large-coco". As a result of its improved performance and relative simplicity, GIT-base-coco is selected for all subsequent experiments ("Coco Model + Inpainted Caption"). Detailed examples are presented in the appendix in Table III for reference. Table II summarizes the findings:

Pre-trained Model	Dataset Variant	Observations
git-base	10M Image-Text	–
git-base-coco	10M Image-Text	Better attributes recognition
git-large	20M Image-Text	–
git-large-coco	20M Image-Text	Similar to git-base-coco

TABLE II
SUMMARY OF PRE-TRAINED MODEL SELECTION

The model "Coco Model + Inpainted Caption" enables improved model performance compared to model "Base Model + Inpainted Caption" with GIT-base as pre-trained model. To assess the performance of the two models, both are trained with captions generated on inpainted images. The results as seen in Figure 18 show that GIT-base-coco outperforms GIT-base by an increase of 4.21% in AUROC.

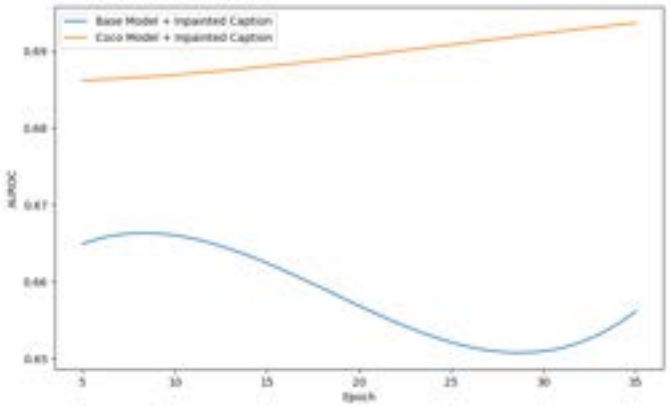

Fig. 18. AUROC [%] of different pre-trained models.

D. Benefit of Data Augmentation on Model Performance

The impact of data augmentation on model performance is investigated by comparing three different models. The baseline model used for comparison is the Coco model enriched with captions and trained on inpainted images ("Coco Model + Inpainted Caption"), which exhibited the best performance in previous experiments.

The second model involves the addition of the MAMI dataset to the training set, which is randomly mixed with the existing data ("Mix"). The third model undergoes initial training for 10 epochs on the mixed dataset and then undergoes fine-tuning solely on the original Hateful Memes dataset ("Mix + 10 epochs MAMI, then Hateful Memes"). This approach is chosen to leverage data augmentation benefits, addressing overfitting, while ensuring fine-tuning on the original dataset for optimal results, given that the test set comprises only the hateful memes dataset.

Figure 19 demonstrates an improvement in model performance when augmenting the dataset. The second model enhances the baseline by 3.41%, while the third model surpasses the baseline by 3.88%. These results validate the hypothesis that increasing the dataset size positively impacts model performance.

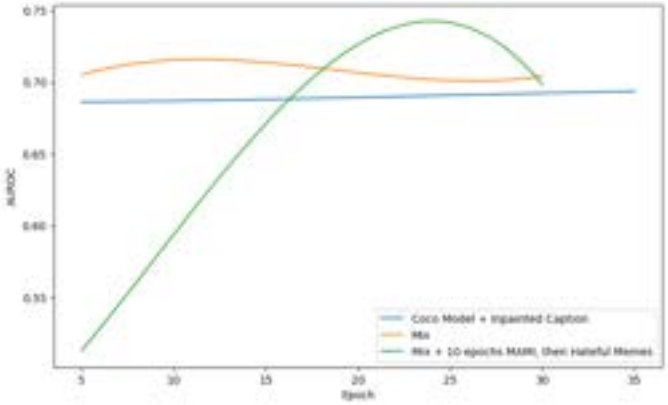

Fig. 19. AUROC [%] of models with augmented datasets.

E. No Direct Impact of Class Balance on Model Performance

The hypothesis tested in this experiment is whether better model performance can be achieved through class balance. Class balance is implemented by augmenting the train set with MAMI data labeled as hateful and simultaneously reducing the Hateful Memes data labeled as non-hateful to attain a perfect label balance. However, contrary to the initial assumption, the model's performance as seen in Figure 20 shows a decline after this adjustment. This decrease is attributed to the reduced number of total data instances.

VI. INTERPRETATION OF THE GITFORHATEFULMEMES MODEL

In this chapter, the outputs of several memes data samples are interpreted using the IG method. The attribution scores of both the text and visual components of the GITforHatefulMemes model ("Mix + 10 epochs MAMI, then Hateful Memes"), are analyzed. By employing IG, insights are sought into how the model makes its predictions and the relative importance of different data modalities is understood.

The following sections contain visualizations and discussions of potentially sensitive content related to hateful memes and their interpretation. The content may include offensive language, images, or themes that could be distressing or triggering to some readers.

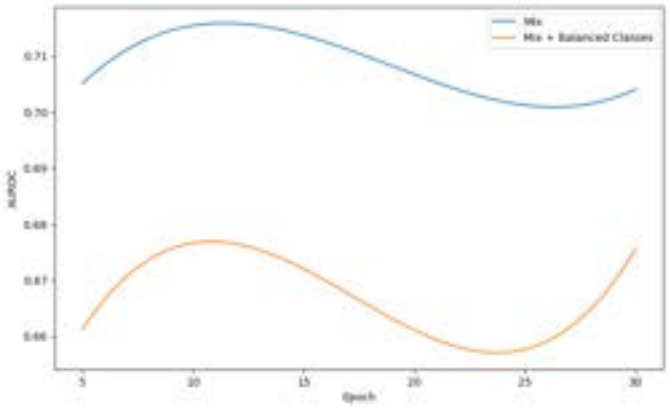

Fig. 20. AUROC [%] of models with and without class balancing method.

A. Implementation

The model architecture can be visualized as a multi-layered structure comprising two main components, namely the GIT model which is responsible for encoding both text and image inputs, and the fc layer which is responsible for the final classification of the model's output. The GIT model consists of three sub-models: the Embeddings layer, the Image-encoder, and the Encoder. The Embeddings layer embeds the text and image inputs into a vector space. The Image-encoder utilizes a transformer architecture to encode the image input, while the Encoder employs a self-attention mechanism to encode the text input. For the interpretability implementation, the Embeddings layer is chosen, as it handles the model's multimodal characteristics. The IG method is implemented following the guidelines in [30], using the PyTorch framework and the Captum library. To start summing the gradients, a black image serves as the baseline for the image, and a zero baseline is used for the text. These baselines represent the absence of signal, allowing for a clearer interpretation of the attributions.

B. Visualization of Examples

The legend denotes red for negative, green for positive, and white for neutral attributions for the text. The "True Label" represents the data label and the "Predicted Label" denotes the label predicted by the model with a probability score above 0 and below 0.5 for non-hateful classification and a probability score between 0.5 and 1 for hateful classification. The "Attribution Label" indicates the label for which the attribution is made. The attribution score [30] reflects that a positive value means the input in that particular position positively contributed to the final prediction, while a negative value means the opposite. The magnitude of the attribution score indicates the strength of the contribution, and a zero attribution score signifies no contribution from that particular feature. Word importance showcases the visualization of attributions for text tokens, which consist of text concatenated with captions and separated by the [SEP] token. An attribution map is displayed for the image visualization, where black pixels correspond to absolute attribution values. The choice of absolute attribution values is to highlight the importance of each image pixel. The intensity of the black pixel color represents the magnitude of the integrated gradients at that specific pixel, while colorless pixels signify the absence of gradients at those locations. Both text and image contributions align with their attribution scores and collectively sum up to the total contribution, representing their combined impact on the prediction.

In Figure 21, positive text attributions are observed for the words "spinner," "woman," "seen in," and "washing," indicating their importance in the model's prediction. Conversely, a negative text attribution is found for the word "machine," which is considered redundant as the word "washing" already conveys the necessary information. The image visualization highlights the attribution for the woman in the picture, which aligns with how a human evaluator would interpret the content. This particular data sample is correctly classified as "hateful." The text contribution has a magnitude of 1.16, which dominates the overall prediction, while the image contribution is -0.16. This dominance of text contribution is expected since the model likely relies heavily on the text to make accurate predictions, as supported by the hypothesis conducted in previous evaluation.

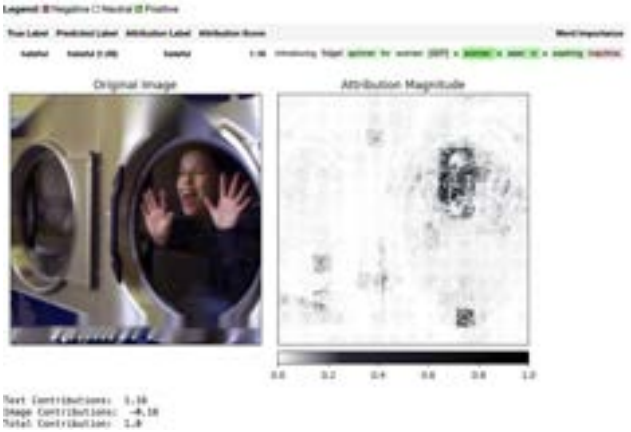

Fig. 21. Example of a correctly classified hateful data sample.

In Figure 22, a goat image is displayed along with positive text attributions for the words "im" and "sky." The image attribution is focused on the goat's head. However, these contributions are challenging to interpret, especially given the model's prediction of "hateful" for a data sample that is actually non-hateful. One possible explanation for this misclassification is that the training set contains several hateful examples featuring goats. As a result, the model might have learned to associate the appearance of goats with hateful classifications, leading to misattributions in this particular case.

Figure 23 presents another incorrectly classified non-hateful example. In this case, it is suspected that the misclassification is primarily attributed to the text part. The positive attributions are found in words such as "retarded" and "people," among

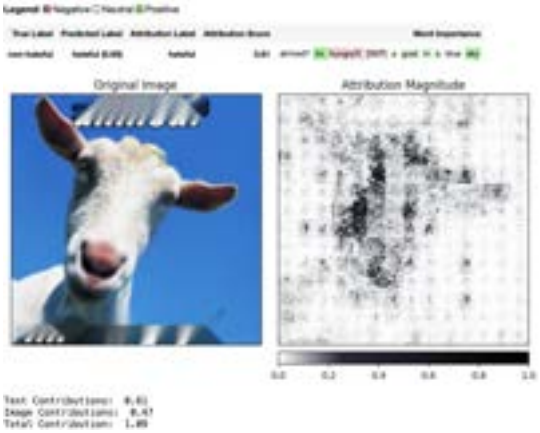

Fig. 22. Example of an incorrectly classified non-hateful data sample with a focus on the visual feature.

others. On the other hand, the image attributions do not provide a clear attribution map, resulting in a negative image contribution. The text tokens with the highest attribution values might be considered hateful by a human evaluator, and this could be influenced by the model's pretraining data. However, it's important to note that the specific definition of hatefulness used for this classification task does not classify these tokens as hateful.

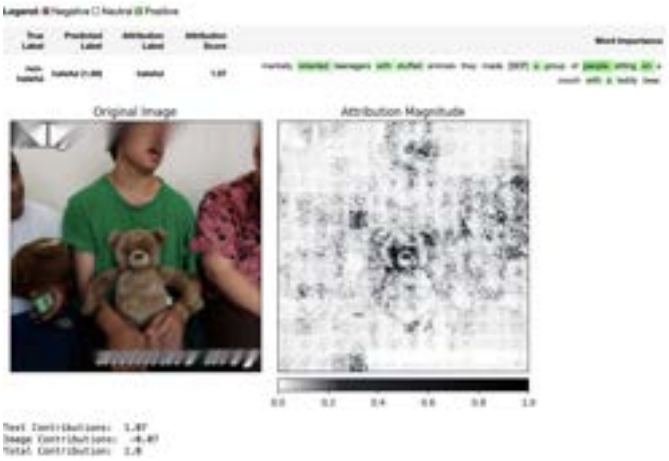

Fig. 23. Example of an incorrectly classified non-hateful data sample with a focus on the text feature.

Figures 24 and 25 showcase data samples representing benign confounders. These samples come with the same original text but different images, leading to different labels. Figure 24 is correctly classified as non-hateful, while Figure 25 is incorrectly classified as non-hateful. The differences between the two are their generated captions and images, with identical text. The misclassification in Figure 25 is presumed to occur because the model fails to recognize the ethnicity of the two individuals depicted in the image. Consequently, the hatefulness definition does not apply in this context.

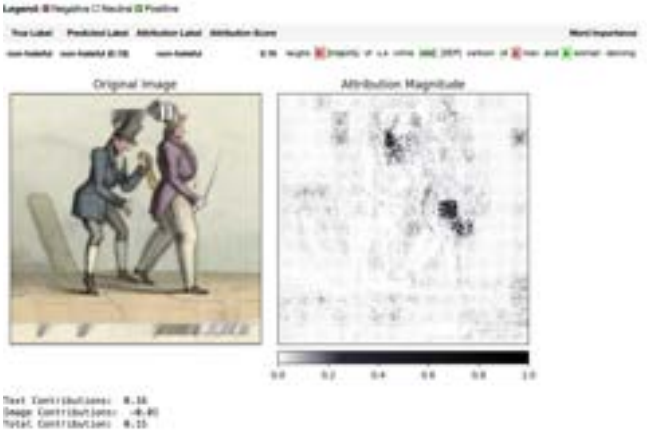

Fig. 24. Benign confounder example of a correctly classified non-hateful data sample.

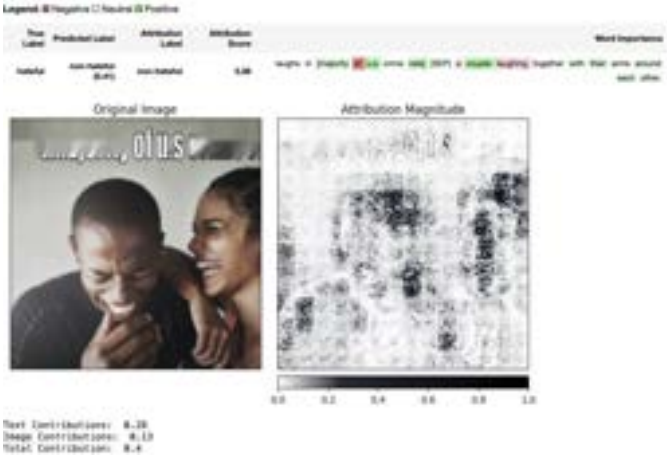

Fig. 25. Benign confounder example of an incorrectly classified hateful data sample.

VII. Conclusion and Outlook

This study has contributed to research in hate speech detection in multimodal data by employing various strategies such as enriching text data with image captions, improving image captioning by removing text from images, using pre-trained models, increasing dataset size, and balancing classes. A key focus was on interpretability, using the Integrated Gradients method to understand the model's decision-making process and the relative importance of different data modalities. The best-performing GITforHatefulMemes model achieved an AUROC score of 72.26%, improving the baseline model by 21.53%. Despite the presence of overfitting, the study demonstrated that the used strategies improved hate speech detection. However, the complexity of interpreting deep learning models and some misclassifications highlight the need for careful application of such models. Future research can explore additional interpretability methods, advanced data augmentation techniques, and incorporating contextual information to refine the model's performance and address challenges in hate speech classification.

VIII. Acknowledgment

I'd like to express my sincere gratitude to my supervisor, Dr. Jasmina Bogojeska. Her support, especially in providing key literature and valuable feedback, significantly contributed to the successful completion of my project. Thank you, Dr. Bogojeska. Your guidance was instrumental.

REFERENCES

[1] P. Xu, X. Zhu, and D. A. Clifton, "Multimodal learning with transformers: A survey," 2023.

[2] G. Joshi, R. Walambe, and K. Kotecha, "A review on explainability in multimodal deep neural nets," *CoRR*, vol. abs/2105.07878, 2021.

[3] M. H. Ribeiro, P. H. Calais, Y. A. Santos, V. A. F. Almeida, and W. M. J. au2, "Characterizing and detecting hateful users on twitter," 2018.

[4] Z. Waseem, T. Davidson, D. Warmsley, and I. Weber, "Understanding abuse: A typology of abusive language detection subtasks," in *Proceedings of the First Workshop on Abusive Language Online*, (Vancouver, BC, Canada), pp. 78–84, Association for Computational Linguistics, Aug. 2017.

[5] 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*, (Valencia, Spain), pp. 1–10, Association for Computational Linguistics, Apr. 2017.

[6] P. Fortuna and S. Nunes, "A survey on automatic detection of hate speech in text," *ACM Comput. Surv.*, vol. 51, jul 2018.

[7] Z. Waseem, "Are you a racist or am I seeing things? annotator influence on hate speech detection on Twitter," in *Proceedings of the First Workshop on NLP and Computational Social Science*, (Austin, Texas), pp. 138–142, Association for Computational Linguistics, Nov. 2016.

[8] R. Kumar, A. K. Ojha, S. Malmasi, and M. Zampieri, "Benchmarking aggression identification in social media," in *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, (Santa Fe, New Mexico, USA), pp. 1–11, Association for Computational Linguistics, Aug. 2018.

[9] S. Malmasi and M. Zampieri, "Challenges in discriminating profanity from hate speech," 2018.

[10] T. Davidson, D. Bhattacharya, and I. Weber, "Racial bias in hate speech and abusive language detection datasets," 2019.

[11] D. Kiela, H. Firooz, A. Mohan, V. Goswami, A. Singh, C. A. Fitzpatrick, P. Bull, G. Lipstein, T. Nelli, R. Zhu, N. Muennighoff, R. Velioglu, J. Rose, P. Lippe, N. Holla, S. Chandra, S. Rajamanickam, G. Antoniou, E. Shutova, H. Yannakoudakis, V. Sandulescu, U. Ozertem, P. Pantel, L. Specia, and D. Parikh, "The hateful memes challenge: Competition report," in *Proceedings of the NeurIPS 2020 Competition and Demonstration Track* (H. J. Escalante and K. Hofmann, eds.), vol. 133 of *Proceedings of Machine Learning Research*, pp. 344–360, PMLR, 06–12 Dec 2021.

[12] F. Yang, X. Peng, G. Ghosh, R. Shilon, H. Ma, E. Moore, and G. Predovic, "Exploring deep multimodal fusion of text and photo for hate speech classification," in *Proceedings of the Third Workshop on Abusive Language Online*, (Florence, Italy), pp. 11–18, Association for Computational Linguistics, Aug. 2019.

[13] H. Hosseinmardi, S. A. Mattson, R. I. Rafiq, R. Han, Q. Lv, and S. Mishra, "Detection of cyberbullying incidents on the instagram social network," 2015.

[14] P. Vijayaraghavan, H. Larochelle, and D. Roy, "Interpretable multimodal hate speech detection," 2021.

[15] R. Gomez, J. Gibert, L. Gomez, and D. Karatzas, "Exploring hate speech detection in multimodal publications," 2019.

[16] S. Lai, X. Hu, H. Xu, Z. Ren, and Z. Liu, "Multimodal sentiment analysis: A survey," 2023.

[17] 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," 2021.

[18] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," 2015.

[19] P. Xu, X. Zhu, and D. A. Clifton, "Multimodal learning with transformers: A survey," 2023.

[20] A. F. Agarap, "Deep learning using rectified linear units (relu)," 2019.

[21] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," 2023.

[22] L. H. Li, M. Yatskar, D. Yin, C.-J. Hsieh, and K.-W. Chang, "Visualbert: A simple and performant baseline for vision and language," 2019.

[23] G. Li, N. Duan, Y. Fang, M. Gong, D. Jiang, and M. Zhou, "Unicoder-vl: A universal encoder for vision and language by cross-modal pre-training," 2019.

[24] W. Su, X. Zhu, Y. Cao, B. Li, L. Lu, F. Wei, and J. Dai, "Vl-bert: Pre-training of generic visual-linguistic representations," 2020.

[25] J. Lin, A. Yang, Y. Zhang, J. Liu, J. Zhou, and H. Yang, "Interbert: Vision-and-language interaction for multi-modal pretraining," 2021.

[26] D. Qi, L. Su, J. Song, E. Cui, T. Bharti, and A. Sacheti, "Imagebert: Cross-modal pre-training with large-scale weak-supervised image-text data," 2020.

[27] J. Wang, Z. Yang, X. Hu, L. Li, K. Lin, Z. Gan, Z. Liu, C. Liu, and L. Wang, "Git: A generative image-to-text transformer for vision and language," 2022.

[28] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[29] *XxAI - Beyond Explainable AI: International Workshop, Held in Conjunction with ICML 2020, July 18, 2020, Vienna, Austria, Revised and Extended Papers*, (Berlin, Heidelberg), Springer-Verlag, 2020.

[30] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," 2017.

[31] A. Khosla, N. Jayadevaprakash, B. Yao, and L. Fei-Fei, "Novel dataset for fine-grained image categorization," in *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*, (Colorado Springs, CO), June 2011.

[32] S. Mishra, S. Suryavardan, P. Patwa, M. Chakraborty, A. Rani, A. Reganti, A. Chadha, A. Das, A. Sheth, M. Chinnakotla, A. Ekbal, and S. Kumar, "Memotion 3: Dataset on sentiment and emotion analysis of codemixed hindi-english memes," 2023.

[33] S. Suryawanshi, B. R. Chakravarthi, M. Arcan, and P. Buitelaar, "Multimodal meme dataset (MultiOFF) for identifying offensive content in image and text," in *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*, (Marseille, France), pp. 32–41, European Language Resources Association (ELRA), May 2020.

[34] E. Fersini, F. Gasparini, G. Rizzi, A. Saibene, B. Chulvi, P. Rosso, A. Lees, and J. Sorensen, "SemEval-2022 task 5: Multimedia automatic misogyny identification," in *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*, (Seattle, United States), pp. 533–549, Association for Computational Linguistics, July 2022.

[35] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," 2019.

[36] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Huggingface's transformers: State-of-the-art natural language processing," 2020.

[37] A. Mao, M. Mohri, and Y. Zhong, "Cross-entropy loss functions: Theoretical analysis and applications," 2023.

[38] T. Pearce, A. Brintrup, and J. Zhu, "Understanding softmax confidence and uncertainty," 2021.

[39] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Huggingface's transformers: State-of-the-art natural language processing," 2020.

[40] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," 2015.

[41] D. Zhu, X. Wu, and T. Yang, "Benchmarking deep auroc optimization: Loss functions and algorithmic choices," 2022.

[42] H. Hornbeck, "Fast cubic spline interpolation," 2020.

APPENDIX

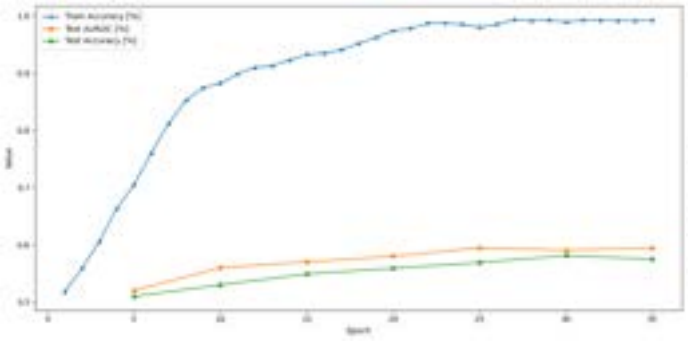

Fig. 26. Metrics for Base Model.

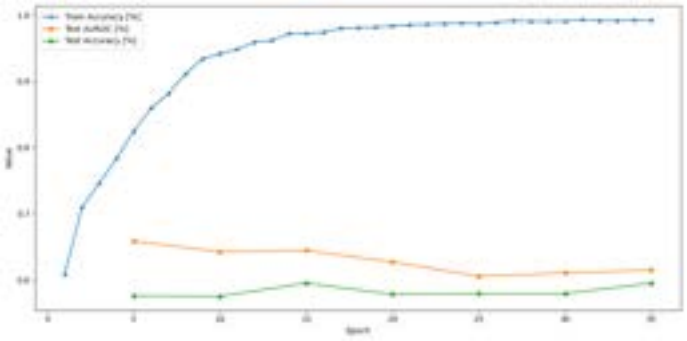

Fig. 27. Metrics for Base Model + Caption.

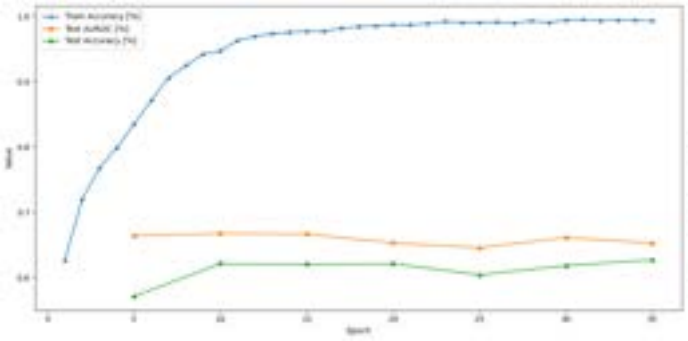

Fig. 28. Metrics for Base Model + Inpainted Caption.

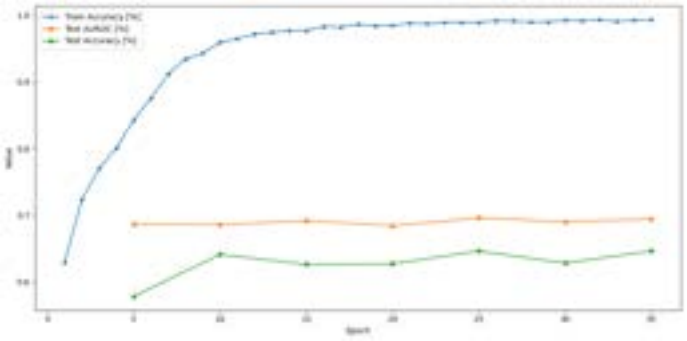

Fig. 29. Metrics for Coco Model + Inpainted Caption.

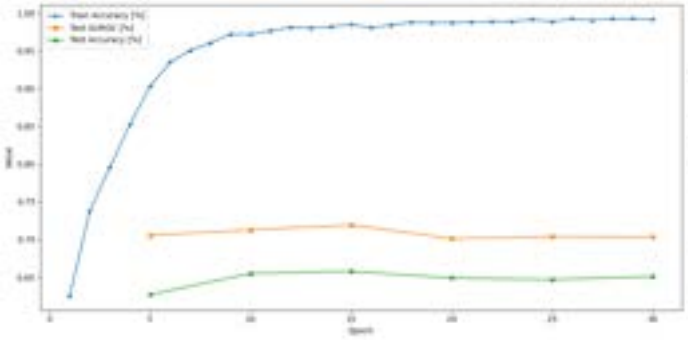

Fig. 30. Metrics for Mix.

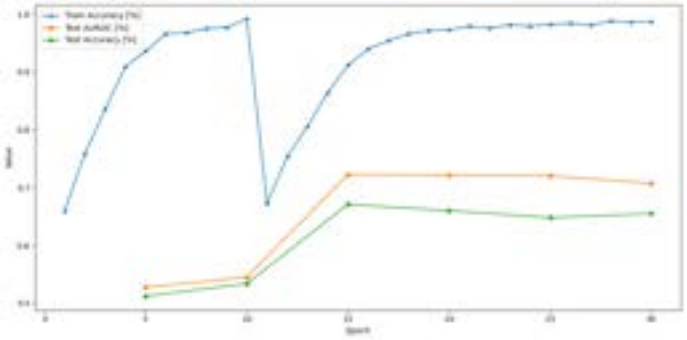

Fig. 31. Metrics for Mix + 10 epochs MAMI, then Hateful Memes.

	(I) git-base-coco	II) git-base-coco inpainted	(III) git-large-coco inpainted
	cartoon of a man and a woman talking to each other	cartoon of a man and a woman dancing	a man in a top hat and a top hat is walking away from a man in a top hat.
{"label": 0, "text": "laughs in [majority of U.S. crime rate]"} (train set)			
	this is the US state of the US state	a couple laughing together with their arms around each other.	a man and a woman laughing.
{"label": 1, "text": "laughs in [majority of U.S. crime rate]"} (test set)			
	a young boy dressed as a police officer	a young boy in uniform giving a thumbs up.	a boy in a blue uniform giving the thumbs up.
{"label": 0, "text": "good guy police officer capturing them young"} (train set)			
	a group of young people shaking hands with a man shaking hands.	a group of children shaking hands.	a teacher shakes hands with a student.
{"label": 1, "text": "good guy police officer capturing them young"} (test set)			
	a young boy with a serious look on his face	a young black boy with his hand on his chin.	a boy with his hand on his chin.
{"label": 1, "text": "a new target for entry-level police academy"} (train set)			

TABLE III
CAPTION TEXT GENERATED BY DIFFERENT PRE-TRAINED MODELS.