Project Report: Multi-modal Hate Speech Detection using the MMHS150K Dataset

Team: 404
Janojit Chakraborty
janojitchakraborty8303@gmail.com

May 1, 2025

Abstract

Hate speech on social media has become a pressing issue due to its widespread societal impact. Traditional unimodal approaches often struggle to capture the nuanced and context-rich nature of such content, especially when both textual and visual elements are involved. This study proposes a comprehensive framework for multimodal hate speech detection by combining image and text data from the MMHS150K dataset. We explored six different fusion-based deep learning architectures. To address label ambiguity, majority voting was applied for label aggregation across annotators. A thorough data exploration was conducted to highlight class imbalance, which was then mitigated during training using weighted cross-entropy loss. All models were trained and evaluated using stratified training, validation, and test splits. Experimental results show that Method 3: GloVe + OCR + MobileNetV2 with Residual Fusion achieved the best performance with an accuracy of 48.81% and F1 score of 0.5048. Our results demonstrate the effectiveness of multimodal fusion strategies in improving hate speech detection, particularly in handling diverse and imbalanced real-world data.

1 Introduction

In recent years, the proliferation of online content has led to a rise in harmful and offensive material, particularly hate speech. This poses significant societal and ethical challenges, prompting the need for robust automatic hate speech detection systems. Traditional approaches primarily rely on textual information; however, the increasing prevalence of memes, images, and multimodal content on social media necessitates the development of models capable of processing and understanding multiple modalities.

This project focuses on multi-model hate speech detection, aiming to effectively classify hate speech using both visual and textual information. We utilize the MMHS150K dataset, a large-scale multimodal corpus comprising tweets with accompanying images, text, and annotations. The

presence of multi-annotator labels in the dataset also provides a unique opportunity to explore effective label aggregation strategies such as majority voting.

To tackle the challenges of label ambiguity and class imbalance inherent in the dataset, we incorporate two key techniques: majority voting to consolidate the annotations into a single ground truth label and weighted cross-entropy loss to address skewed class distributions. The goal of this project is to explore and evaluate the effectiveness of these strategies within a multimodal deep learning framework.

2 Literature Review

2.1 Related Works

A notable prior work in the domain of multimodal hate speech detection is presented in the paper *Exploring Hate Speech Detection in Multimodal Publications* (1). This work introduces the Hateful Memes dataset and highlights the limitations of unimodal systems in detecting hate speech embedded in both image and text.

The paper investigates the use of multimodal data (text and images) for detecting hate speech in social media posts. The authors introduce the MMHS150K dataset, which consists of 150,000 Twitter posts, each containing text and associated images, annotated for hate speech detection.

The authors propose three multimodal architectures: the Feature Concatenation Model (FCM), the Spatial Concatenation Model (SCM), and the Textual Kernels Model (TKM). These models combine image features from a pre-trained Inception v3 network with text features extracted from both the tweet and any text embedded in the images.

In terms of results, the study finds that multimodal models do not necessarily outperform text-only models. The image-only model performs poorly, and while FCM and SCM perform reasonably well, they only slightly improve over text-only models.

A notable prior work on multi-modal hate speech detection is presented in the paper *QUARC: Quaternion Multi-Modal Fusion Architecture for Hate Speech Classification* (2). This work introduces a novel quaternion neural network-based model that effectively handles multi-modal inputs—text, image, and text-of-image—for hate speech classification on social media posts. The authors address the challenge of large parameter sizes in traditional fusion models by leveraging quaternion algebra, achieving approximately 75% reduction in parameters, thereby enhancing efficiency without compromising performance.

2.2 Label Aggregation using Majority Voting

In datasets like MMHS150K, where each data point is annotated by multiple annotators, it becomes essential to derive a consensus label. Majority voting is one of the most commonly used aggregation techniques in such scenarios. Sheng et al. (3) and Snow et al. (4) both support majority voting as

an effective method to mitigate annotator bias and improve reliability.

2.3 Handling Class Imbalance with Weighted Cross-Entropy

Hate speech detection datasets are typically imbalanced. To address this, weighted cross-entropy is used. Cui et al. (5) and Lin et al. (6) show that such techniques enhance performance on imbalanced datasets.

A notable prior work on the limitations of the Receiver Operating Characteristic (ROC) curve on imbalanced datasets is presented in the paper *Limitations of ROC on Imbalanced Data: Evaluation of LVAD Mortality Risk Scores* (7). This work critically examines the use of ROC curves for evaluating classifiers trained on imbalanced data, particularly in the context of Left Ventricular Assist Device (LVAD) mortality prediction. The authors demonstrate that ROC curves can present an overly optimistic view of classifier performance, especially when the minority class (e.g., mortality cases) is underrepresented.

2.4 Text Encoders

BERT, introduced by Devlin et al. (8), revolutionized the field of Natural Language Processing (NLP) by utilizing a transformer architecture to generate deeply bidirectional context-aware embeddings. Unlike previous models that processed text unidirectionally, BERT's ability to consider both preceding and succeeding words improves its understanding of the context. In the methodology, BERT is used to encode tweet text into 768-dimensional feature vectors, capturing the rich semantic meaning of the text..

DistilBERT, a smaller and faster variant of BERT, was introduced by Sanh et al. (9) as a method to reduce BERT's size while retaining 97%. of its language understanding capabilities. DistilBERT achieves this by employing knowledge distillation, a technique where a smaller model (the student) learns to mimic the output of a larger model (the teacher). In this methodology, DistilBERT is used for encoding tweets, providing an efficient alternative to BERT while maintaining high performance.

RoBERTa, developed by Liu et al. (10), is a robustly optimized version of BERT that improves upon the pre-training method. By training on more data and using dynamic masking strategies, RoBERTa demonstrates superior performance compared to BERT on many benchmarks. It has become a standard model for text classification tasks, including hate speech detection, due to its ability to capture fine-grained contextual information.

FastText, proposed by Joulin et al. (11), is a lightweight and fast text classification algorithm that improves upon word embeddings by considering subword information. This helps capture the morphology of words, making it more effective for languages with complex word structures. In the context of this methodology, FastText embeddings are used to represent the tweet text by averaging the word vectors of all tokens in the tweet.

2.5 Image Encoders

ResNet, introduced by He et al. (12), is one of the most influential architectures in deep learning due to its use of residual connections, which allow for the training of very deep networks. ResNet18 is a lightweight version of the original ResNet architecture, comprising 18 layers. It is commonly used for tasks such as image classification, including in hate speech detection tasks involving social media images, where it extracts high-level features from the input image.

EfficientNet, proposed by Tan and Le (13), introduces a novel scaling strategy that uniformly scales the depth, width, and resolution of the network. EfficientNet-B0 is the smallest model in this family and achieves excellent performance while being highly parameter-efficient. It has become a popular choice for image feature extraction in multimodal tasks due to its ability to balance accuracy and computational cost.

MobileNetV2, introduced by Sandler et al. (14), is a lightweight convolutional neural network designed for mobile and embedded vision applications. The model uses depthwise separable convolutions to reduce computational cost. It is particularly suitable for real-time applications where resources are limited, making it an ideal choice for multimodal applications that need to balance speed and accuracy.

The Vision Transformer (ViT), proposed by Dosovitskiy et al. (15), introduces a transformer-based approach for processing images, demonstrating state-of-the-art performance in image classification tasks. ViT treats an image as a sequence of patches, and this patch-based processing enables transformers to model long-range dependencies. ViT is used in the methodology for encoding images, providing high-quality image features that are crucial for multimodal tasks.

DenseNet, introduced by Huang et al. (16), is a type of convolutional neural network that connects each layer to every other layer in a feed-forward fashion. This dense connectivity pattern helps alleviate the vanishing gradient problem and improves feature reuse, which is particularly beneficial for tasks involving limited data or fine-grained classification, such as detecting hate speech in multimodal settings.

2.6 Fusion Mechanisms

Attention Mechanism: Attention mechanisms, particularly in transformer-based models, have become crucial in enabling dynamic focus on different parts of the input sequence. This flexibility enhances performance in tasks across natural language processing (NLP) and computer vision. By allowing models to prioritize important features during learning, attention mechanisms are essential in multimodal tasks, where information from multiple modalities (e.g., text and image) must be integrated. The foundational paper by Bahdanau et al. (17), introduced attention in neural machine translation, significantly improving sequence alignment. Later, the groundbreaking paper by Vaswani et al. (18) further advanced the concept by demonstrating the effectiveness of self-attention in transformer architectures, enabling superior performance across various tasks without

relying on recurrence.

Gating Mechanism: Gating mechanisms, such as those found in Gated Recurrent Units (GRUs), allow selective attention to different parts of the input at each time step. This dynamic selection is particularly useful in multimodal fusion, where models must adjust the influence of each modality during training. The GRU was introduced by Cho et al. (19), which detailed how gated mechanisms improve sequence modeling by enabling the model to control information flow effectively.

2.7 Classifiers

The Multilayer Perceptron (MLP) is a core neural network architecture employed in both fusion and classification tasks. It consists of fully connected layers with nonlinear activations, and dropout layers to prevent overfitting. MLPs are particularly effective when combined with advanced feature extraction methods such as BERT for text or ResNet for images. The concept of the perceptron, introduced by Rosenblatt (20), laid the foundation for modern neural networks, providing a basis for many current classification and fusion strategies.

3 Proposed Methodology

In this section, we describe the approach adopted for multimodal hate speech detection using the MMHS150K dataset. The method involves feature extraction from both text and image modalities followed by a fusion through a neural network for classification. The pipeline is divided into three major components: data preprocessing, feature extraction, and model architecture.

3.1 Method 1: BERT + ResNet18 with Simple MLP Fusion (Model 1)

This method leverages a feature-level fusion strategy combining pretrained BERT for textual encoding and ResNet18 for visual encoding. Both models are used in inference mode with CUDA acceleration when available.

3.1.1 Text Feature Extraction

The tweet text is tokenized using the `BertTokenizer` from the `bert-base-uncased` model (8). The tokenized input is fed into the pretrained BERT model, and the mean-pooled last hidden state is extracted to obtain a 768-dimensional embedding for each tweet. This approach captures deep contextual semantics of the language used in hate speech.

3.1.2 Image Feature Extraction

Images are preprocessed and resized to 224×224, then passed through a ResNet18 model (12), pretrained on ImageNet. The final classification layer is removed to extract the penultimate layer's

output—resulting in a 512-dimensional visual embedding.

3.1.3 Fusion & Classification

The 768-dim BERT text features are passed through a linear layer and projected to 512-dim.

The 512-dim ResNet18 features are passed as-is into a similar linear transformation.

The two branches are concatenated ($512 + 512 = 1024$), followed by two fully connected layers with ReLU and dropout regularization.

The final output layer performs multi-class classification over the 6 hate categories.

This method serves as a baseline for multimodal learning, using a simple late fusion (feature concatenation) mechanism without complex attention or interaction modules.

3.2 Method 2: DistilBERT + EfficientNet with BiGRU Fusion (Model 2)

The second method enhances the feature extraction and fusion strategy by leveraging lightweight yet powerful models: DistilBERT for text and EfficientNet-B0 for images. Unlike the MLP used in Method 1, this method uses a Bidirectional GRU to capture sequential dependencies in the text embeddings before combining them with image features.

3.2.1 Text Feature Extraction (DistilBERT)

Textual content is processed using DistilBERT, a distilled version of BERT that is faster and smaller while retaining much of its performance. Similar to Method 1, tokenized text is passed through DistilBERT, and the final hidden states are extracted. However, instead of simple mean pooling, these embeddings are treated as sequences and passed to a BiGRU to retain contextual flow, yielding a final 512-dimensional feature vector.

Input: Tweet text

Model: `distilbert-base-uncased` from HuggingFace Transformers

Output: Sequence embedding (768-dim tokens) $\rightarrow$ BiGRU $\rightarrow$ 512-dim final text feature

3.2.2 Image Feature Extraction (EfficientNet-B0)

For image modality, we utilize EfficientNet-B0, which offers a better accuracy-to-parameter ratio compared to ResNet. Using the pretrained `efficientnet-b0` model, we extract features from the final convolutional layers and apply global average pooling across spatial dimensions to obtain a 1280-dimensional vector. This vector is then linearly transformed to a 512-dimensional embedding.

Input: Tweet image resized to 224×224

Model: Pretrained EfficientNet-B0

Output: Extracted features $\rightarrow$ Global average pooling $\rightarrow$ 1280-dim $\rightarrow$ Linear projection $\rightarrow$ 512-dim

3.2.3 Fusion via BiGRU and Classifier

Text embeddings from the BiGRU and image embeddings from EfficientNet are concatenated to form a 1024-dimensional feature vector. This vector is passed through a fully connected neural network with one hidden layer (256 units), followed by a final output layer for multi-class classification into six hate speech categories.

3.3 Method 3: GloVe + OCR + MobileNetV2 with Residual Fusion (Model 3)

The third method diverges from transformer-based text encoders and instead combines GloVe-based textual embeddings (including OCR-extracted image text) with MobileNetV2 image features. This approach is lighter in terms of model complexity and also explores an innovative residual fusion technique for modality integration.

3.3.1 Text Feature Extraction (OCR + GloVe)

Instead of relying solely on the tweet's written text, this method combines the tweet text with OCR-extracted text from images. The concatenated text is tokenized using nltk's word tokenizer, and each token is mapped to its GloVe 300-dimensional embedding (from `glove.6B.300d.txt`). The final text embedding is computed as the mean of the available token embeddings.

Input: Tweet text + OCR text

Model: GloVe embeddings (pretrained)

Output: 300-dimensional vector $\rightarrow$ Projected to 512-dim

3.3.2 Image Feature Extraction (MobileNetV2)

Visual content is processed using MobileNetV2. The model is used without its classification head, and the output of the final convolutional feature map is globally average pooled, resulting in a 1280-dimensional image feature vector.

Input: Preprocessed tweet image (224×224)

Model: Pretrained MobileNetV2

Output: 1280-dimensional vector $\rightarrow$ Projected to 512-dim

3.3.3 Residual Fusion Strategy

To fuse the two modalities, the method employs a residual fusion mechanism:

$$\text{Fused} = \text{Text}_{\text{proj}} + \text{Image}_{\text{proj}} + \text{FusionLayer}(\text{Text}_{\text{proj}} \odot \text{Image}_{\text{proj}})$$

This strategy captures both additive and multiplicative interactions between modalities.

3.3.4 Classification Layer

The final fused 512-dimensional vector is passed through a single linear layer that maps to 6 output classes.

3.4 Method 4: Sentence-BERT + ViT with Attention-Based Fusion (Model 4)

This method utilizes Sentence-BERT for semantic text embedding and Vision Transformer (ViT) for image understanding, fused via a lightweight attention-based mechanism.

3.4.1 Text Feature Extraction (Sentence-BERT)

We use the pretrained `all-MiniLM-L6-v2` model from SentenceTransformers.

Input: Tweet text

Model: Sentence-BERT (`all-MiniLM-L6-v2`)

Output: 384-dimensional vector $\rightarrow$ Projected to 512-dim

3.4.2 Image Feature Extraction (Vision Transformer)

Using HuggingFace's ViTModel, we extract the [CLS] token embedding from the final transformer block, resulting in a 768-dimensional image representation.

Input: Tweet image (resized)

Model: ViT-base from HuggingFace

Output: 768-dimensional vector $\rightarrow$ Projected to 512-dim

3.4.3 Attention-Based Fusion

The fusion module learns an attention weight α over the concatenated features:

$$\text{Fused} = \alpha \cdot \text{Text}_{\text{proj}} + (1 - \alpha) \cdot \text{Image}_{\text{proj}}$$

3.4.4 Classification

The fused 512-dimensional vector is passed through a ReLU-activated dropout layer followed by a final linear layer.

3.5 Method 5: RoBERTa + CLIP-ViT with Deep Fusion (Model 5)

This final method leverages RoBERTa for language understanding and CLIP-ViT for visual semantic representation, fused via deep concatenation.

3.5.1 Text Feature Extraction (RoBERTa)

Input: Tweet text

Model: `roberta-base` from HuggingFace

Output: 768-dimensional vector $\rightarrow$ Projected to 256-dim

3.5.2 Image Feature Extraction (CLIP-ViT)

Input: Tweet image (resized)

Model: `openai/clip-vit-base-patch32`

Output: 512-dimensional vector $\rightarrow$ Projected to 256-dim

3.5.3 Fusion and Classification

The text and image embeddings are concatenated into a 512-dimensional vector, which is passed through a multi-layer classifier.

3.6 Method 6: FastText + DenseNet with Gated Fusion (Model 6)

This approach combines FastText for textual representation and DenseNet-121 for deep image feature extraction. These are integrated using a gated fusion mechanism.

3.6.1 Text Feature Extraction (FastText + TweetTokenizer)

Input: Raw tweet text

Tokenizer: TweetTokenizer (NLTK)

Embedding: FastText (cc.en.300) $\rightarrow$ 300-dimensional vector via mean pooling

3.6.2 Image Feature Extraction (DenseNet-121)

Input: Resized tweet image

Model: DenseNet-121 (pretrained)

Output: 1024-dimensional feature vector

3.6.3 Fusion and Classification

A Gated Fusion mechanism combines the 512-dim projections from each modality. The resulting vector is passed through a final classifier.

3.7 Final Model Architecture Comparison Table

Method	Text Encoder	Image Encoder	Fusion Strategy	Highlights
1	BERT (768 → 512)	ResNet18 (512)	MLP (Concatenation)	Strong baseline using standard architectures.
2	DistilBERT + BiGRU (512)	EfficientNet-B0 (1280 → 512)	MLP (Concatenation)	Lightweight with sequential text modeling.
3	GloVe + OCR (300 → 512)	MobileNetV2 (1280 → 512)	Residual + Multiplicative Fusion	Integrates textual signals from OCR-enhanced image context.
4	Sentence-BERT (384 → 512)	ViT (768 → 512)	Attention Fusion	Modality-aware attention for dynamic weighting.
5	RoBERTa (768 → 256)	CLIP-ViT (512 → 256)	MLP (Concatenation)	Uses CLIP's cross-modal pretraining for better semantic alignment.
6	FastText (300 → 512)	DenseNet121 (1024 → 512)	Gated Fusion	Lightweight and fast, with dynamic feature weighting via gating.

Table 1: Comparison of different methods

4 Experimental Result

The performance of each model was evaluated using several key metrics: **Accuracy**, **F1 Score**, **Precision**, and **Recall**. These metrics provide a comprehensive understanding of the models' abilities to classify the data correctly.

4.1 Dataset

4.1.1 Dataset Details

We use the MMHS150K dataset, introduced in the paper *Exploring Hate Speech Detection in Multimodal Publications* by Raul Gomez et al. (WACV 2020). This is a multimodal dataset comprising 150,000 tweets, each accompanied by an image and corresponding annotations. The tweets were collected using a set of hate-related keywords, and the dataset contains annotations from three Amazon Mechanical Turk (AMT) annotators per tweet.

More details and official website: `https://gombru.github.io/2019/10/09/MMHS/`

4.1.2 Dataset Contents

The dataset includes the following components:

- `img_resized/`: Contains images resized such that their shortest dimension is 500 pixels. Filenames correspond to tweet IDs.
- `MMHS150K_GT.json`: A JSON file with tweet IDs as keys. Each entry includes:
  - `tweet_url`: URL to the original tweet
  - `tweet_text`: The text content of the tweet

- `img_url`: Link to the associated image
- `labels`: A list of three numeric annotations from AMT annotators:
  - 0: Not Hate
  - 1: Racist
  - 2: Sexist
  - 3: Homophobe
  - 4: Religion-based Hate
  - 5: Other Hate
- `labels_str`: String equivalents of the numeric labels

- `img_txt/`: OCR-extracted text from the tweet images.
- `hatespeech_keywords.txt`: Keywords used during data collection.
- `splits/`: Contains text files for train, validation, and test tweet ID splits.

4.1.3 Data Preprocessing for Label Aggregation

Given that each tweet has three independent annotations, we apply majority voting to aggregate these into a single label. If two or more annotators agree on a label, that label is selected as the ground truth. In the rare event of complete disagreement, we opt for the most frequent class or apply random tie-breaking to ensure label consistency.

4.1.4 Data Exploration: Class Distribution

A thorough exploration of the dataset reveals a significant class imbalance, with the vast majority of samples labeled as Not Hate (Class 0). This imbalance poses challenges for learning, particularly in detecting underrepresented classes such as Religion (Class 4) and Sexist (Class 2). After applying majority vote label aggregation, the class counts are as follows:

Class	Training Set	Validation Set	Test Set
Class 0 (NotHate)	114,214	3,267	6,522
Class 1 (Racist)	9,794	833	1,661
Class 2 (Sexist)	2,939	252	480
Class 3 (Homophobe)	3,100	253	533
Class 4 (Religion)	131	9	24
Class 5 (OtherHate)	4,645	386	780

Table 2: Class distribution across Training, Validation, and Test sets

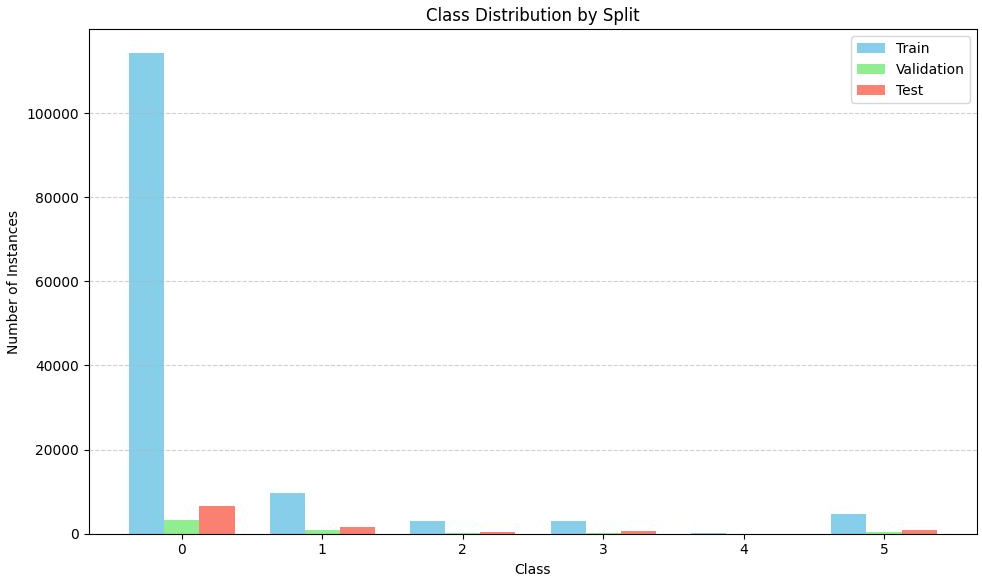

Figure 1: Class distribution across training, validation, and test splits

4.2 Experimental Settings

To ensure fair comparison and reproducibility across all proposed multimodal architectures, we implemented a unified training pipeline in PyTorch that handles data loading, loss balancing, model optimization, and result logging.

4.2.1 Data Preparation

Each of the six models was trained using preprocessed data containing:

- **text:** extracted text features (varied by model),
- **image:** extracted image features,
- **labels:** class labels from majority voting.

The data was split into train, val, and test sets for each method (`preprocessedX_train.npz`, etc.), and loaded via a custom `MMHSDataset` class.

4.2.2 Handling Class Imbalance with Weighted Cross-Entropy Loss

Given the heavy class imbalance in MMHS150K, we used Weighted Cross-Entropy Loss to penalize underrepresented classes more during training.

Cross-Entropy Loss Formula

For a single training example, cross-entropy is given by:

$$L_{CE} = -\sum_{i=1}^{C} y_i \cdot \log(p_i)$$

Where:

- C: number of classes (6 in our case),
- y_i: ground truth label (one-hot encoded),
- p_i: predicted probability for class i.

Weighted Cross-Entropy Loss
To address imbalance, each class i is assigned a weight w_i, leading to:

$$L_{WCE} = -\sum_{i=1}^{C} w_i \cdot y_i \cdot \log(p_i)$$

Where:

$$w_i = \frac{N}{C \cdot n_i}$$

- N: total number of samples,
- n_i: number of samples in class i,
- C: number of classes.

These weights were dynamically computed from the training set and passed to PyTorch's `nn.CrossEntropyLoss(weight=class_weights)`.

4.2.3 Training Configuration

Parameter	**Value**
Optimizer	Adam
Learning Rate	1e-4
Loss Function	Weighted CrossEntropy
Batch Size	64
Epochs	10
Evaluation Metric	Train & Val Loss

Table 3: Training Configuration

4.2.4 Model Training Results

The following table summarizes training and validation loss per epoch for each model:

Model 1 — BERT + ResNet18

Epoch	Train Loss	Val Loss
1	1.3038	1.1621
2	1.0388	1.1085
3	0.9781	1.1214
4	0.9390	1.0538
5	0.9140	1.1270
6	0.8887	1.0962
7	0.8634	1.0387
8	0.8481	1.0723
9	0.8318	1.0802
10	0.8195	1.0869

Table 4: Model 1 Loss: BERT + ResNet18

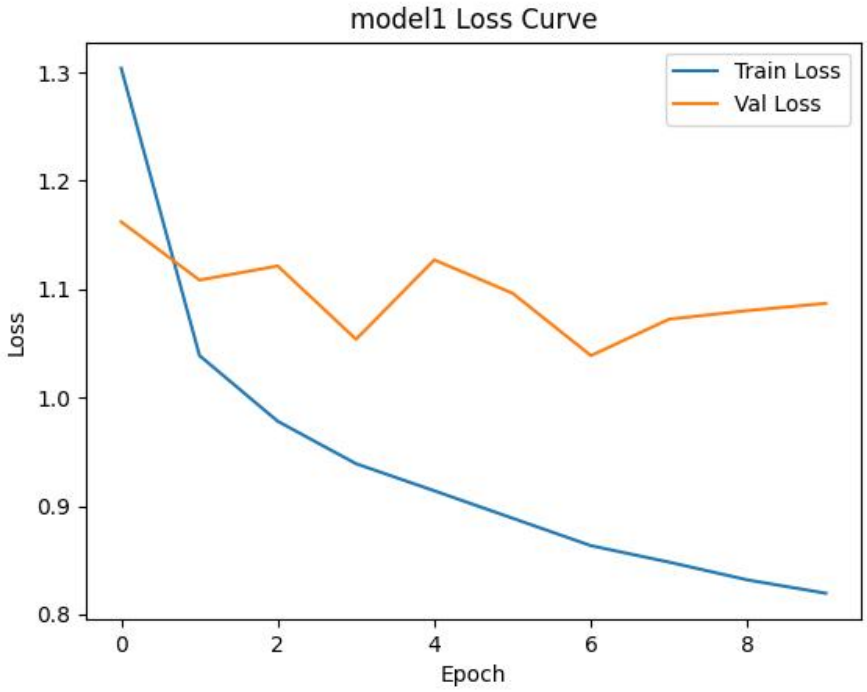

Figure 2: Training and validation loss curves for Model 1

Model 2 — GloVe + EfficientNet

Epoch	Train Loss	Val Loss
1	1.2424	1.1666
2	0.9691	1.0621
3	0.9024	1.0704
4	0.8577	1.1395
5	0.8113	1.0987
6	0.7684	1.0921
7	0.7345	1.1062
8	0.6810	1.1618
9	0.6514	1.1792
10	0.6155	1.2433

Table 5: Model 2 Loss: GloVe + EfficientNet

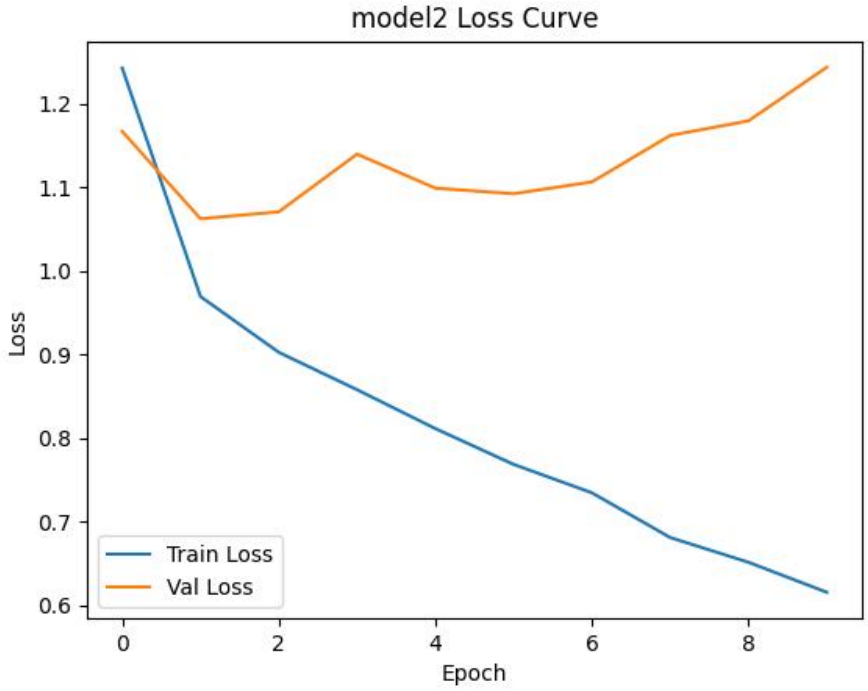

Figure 3: Training and validation loss curves for Model 2

Model 3 — OCR-GloVe + MobileNet

Epoch	Train Loss	Val Loss
1	1.2850	1.1784
2	1.0242	1.1013
3	0.9558	1.1253
4	0.9193	1.0798
5	0.8856	1.1042
6	0.8625	1.0696
7	0.8454	1.0872
8	0.8286	1.0933
9	0.8066	1.0554
10	0.7923	1.0566

Table 6: Model 3 Loss: OCR-GloVe + MobileNet

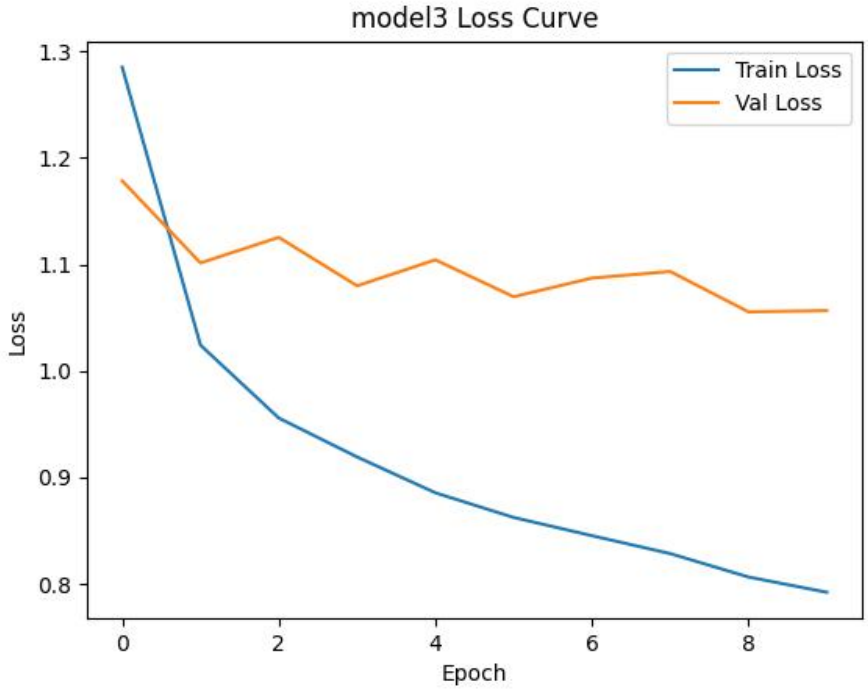

Figure 4: Training and validation loss curves for Model 3

Model 4 — SBERT + ViT (Attention Fusion)

Epoch	Train Loss	Val Loss
1	1.3467	1.4984
2	0.9602	1.3675
3	0.8797	1.3090
4	0.8408	1.2587
5	0.8116	1.2558
6	0.7886	1.2421
7	0.7691	1.2255
8	0.7535	1.2528
9	0.7365	1.2378
10	0.7204	1.2566

Table 7: Model 4 Loss: SBERT + ViT (Attention Fusion)

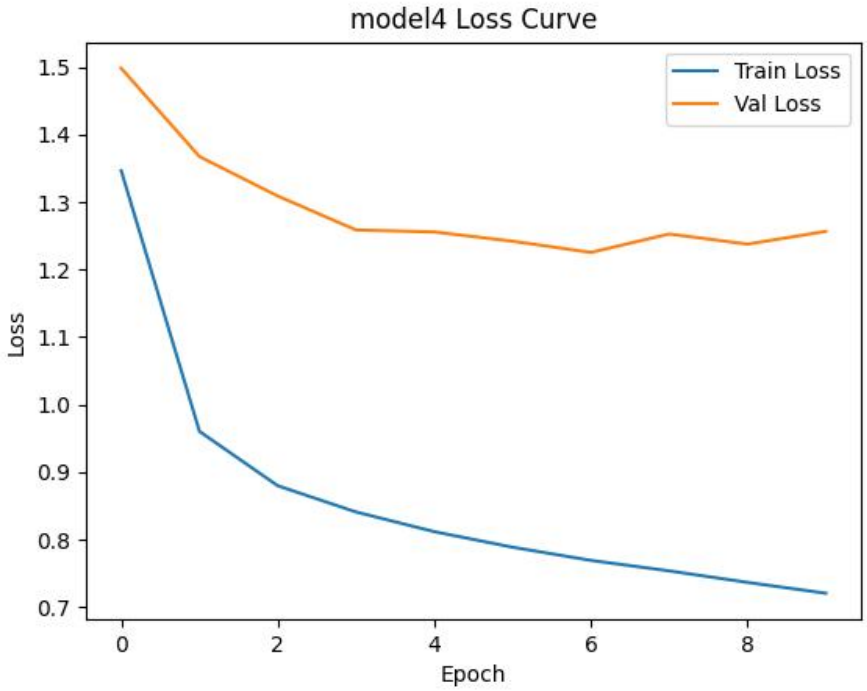

Figure 5: Training and validation loss curves for Model 4

Model 5 — RoBERTa + CLIP-ViT

Epoch	Train Loss	Val Loss
1	1.5153	1.4515
2	1.3916	1.4199
3	1.3468	1.4488
4	1.3255	1.4709
5	1.2869	1.4485
6	1.2570	1.5097
7	1.2158	1.5237
8	1.1754	1.5589
9	1.1388	1.5688
10	1.1002	1.5305

Table 8: Model 5 Loss: RoBERTa + CLIP-ViT

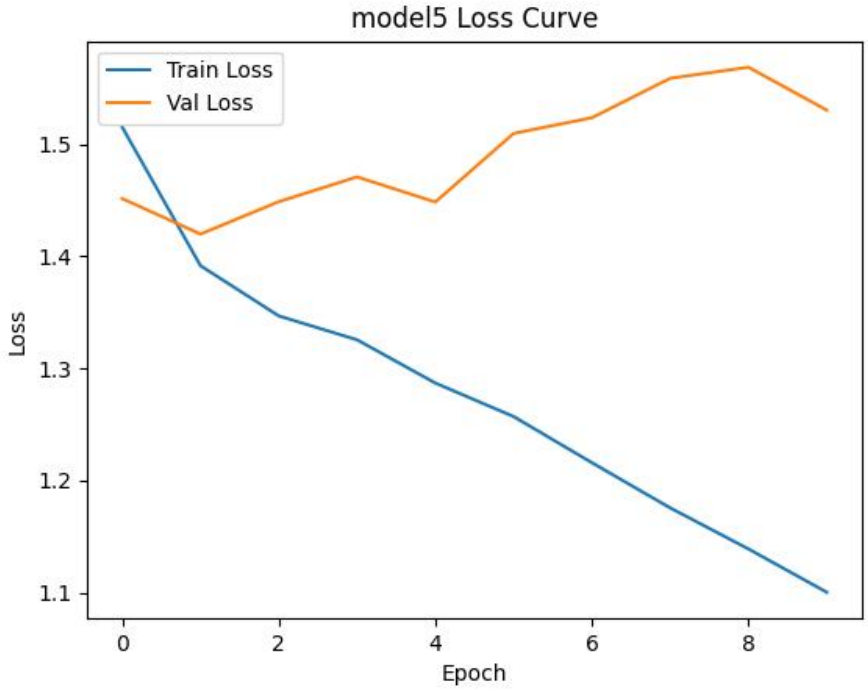

Figure 6: Training and validation loss curves for Model 5

Model 6 — FastText + DenseNet (Gated Fusion)

Epoch	Train Loss	Val Loss
1	1.4923	1.4285
2	1.1712	1.2862
3	1.0670	1.2220
4	1.0121	1.2002
5	0.9716	1.1795
6	0.9334	1.1785
7	0.9125	1.1363
8	0.8869	1.1611
9	0.8606	1.1551
10	0.8392	1.1489

Table 9: Model 6 Loss: FastText + DenseNet (Gated Fusion)

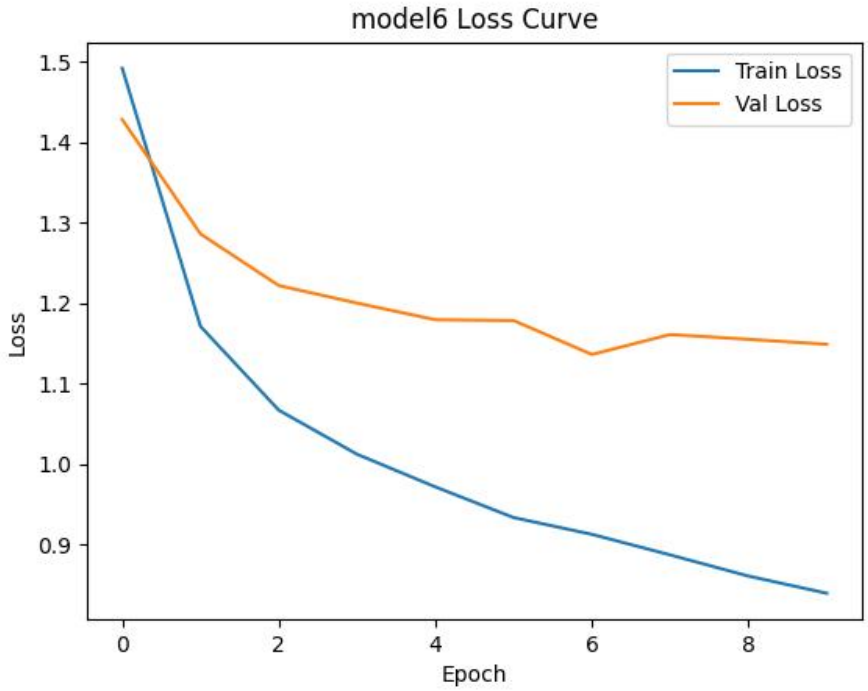

Figure 7: Training and validation loss curves for Model 6

The detailed results are below for each of the six models that were evaluated.

Model 1

- **Accuracy:** 0.4017
  Model 1 achieves an accuracy of 40.17%, indicating that approximately 40% of its predictions match the true labels across the test dataset. While not high, it gives an initial sense of the model's performance.

- **F1 Score:** 0.3968
  The F1 score, which is a harmonic mean of precision and recall, is quite low at 0.3968. This suggests that Model 1 struggles to balance between precision and recall, potentially making errors in both categories.

- **Precision:** 0.6269
  Precision, which measures the percentage of correct positive predictions out of all positive predictions made, is relatively higher at 62.69%. This indicates that when Model 1 does predict a positive class, it is fairly accurate in doing so.

- **Recall:** 0.4017
  The recall score of 0.4017 suggests that Model 1 correctly identifies around 40% of the true positive cases. This means the model misses a significant portion of the positive instances in the dataset.

Model 2

- **Accuracy:** 0.4614
  Model 2 shows a better performance with an accuracy of 46.14%. This is an improvement over Model 1, meaning it correctly classifies a greater portion of the dataset.

- **F1 Score:** 0.4732
  The F1 score for Model 2 is 0.4732, which represents a noticeable improvement compared to Model 1. This suggests that Model 2 strikes a better balance between precision and recall.

- **Precision:** 0.6249
  With a precision score of 62.49%, Model 2 shows that when it classifies an instance as positive, it does so with a high degree of accuracy.

- **Recall:** 0.4614
  Model 2's recall score is 46.14%, meaning it correctly identifies just under half of all the true positive instances. This is a significant improvement over Model 1, but there is still room for improvement in identifying positive cases.

Model 3

- **Accuracy:** 0.4881
  Model 3 outperforms the previous models with an accuracy of 48.81%. This shows that it is better at correctly classifying the instances in the dataset than both Model 1 and Model 2.

- **F1 Score:** 0.5048
  The F1 score for Model 3 reaches 0.5048, which is the highest among all models so far. This indicates a better balance between precision and recall, suggesting that the model is effective at both identifying positive instances and minimizing false positives.

- **Precision:** 0.6348
  Model 3 has a precision score of 63.48%, meaning that it is quite accurate when it predicts a positive class. This is the highest precision score observed so far, indicating that Model 3 is good at predicting positive instances correctly.

- **Recall:** 0.4881
  The recall score for Model 3 is 48.81%, which is also the highest of all models. This suggests that the model is effective at identifying a greater proportion of the true positive instances, though there is still room for improvement.

Model 4

- **Accuracy:** 0.4002
  Model 4 has an accuracy of 40.02%, which is very close to that of Model 1, indicating that it performs similarly in terms of overall classification accuracy. However, its performance is not as strong as the top models.

- **F1 Score:** 0.4012
  The F1 score for Model 4 is 0.4012, indicating that it is slightly better than Model 1 but still struggles with both precision and recall.

- **Precision:** 0.5815
  Model 4 has a precision of 58.15%, which is lower than the precision of Models 2 and 3. This suggests that when it predicts positive instances, it is less reliable than the higher-performing models.

- **Recall:** 0.4002
  The recall score of 0.4002 is comparable to that of Model 1, indicating that Model 4 fails to capture a significant proportion of the positive instances. Like Model 1, it misses a considerable number of true positive cases.

Model 5

- **Accuracy:** 0.2211
  Model 5 has the lowest accuracy at just 22.11%. This indicates that it performs poorly in classifying the instances, making it the least reliable model among all six.

- **F1 Score:** 0.2235
  The F1 score for Model 5 is 0.2235, further reflecting its poor performance. The model struggles with balancing precision and recall, as evidenced by both the low F1 score and accuracy.

- **Precision:** 0.5717
  Despite the low overall accuracy, Model 5's precision is 57.17%. This shows that when it does make a positive prediction, it is relatively accurate. However, this does not compensate for its poor recall and overall classification performance.

- **Recall:** 0.2211
  The recall of Model 5 is 22.11%, meaning it is able to identify only a small proportion of the true positive instances. This is a major factor in the model's low overall accuracy and F1 score.

Model 6

- **Accuracy:** 0.4086
  Model 6 achieves an accuracy of 40.86%, which is similar to Model 1 and Model 4. This places it among the lower-performing models in terms of overall classification accuracy.

- **F1 Score:** 0.3970
  The F1 score for Model 6 is 0.3970, which is comparable to Model 1's score. This suggests that Model 6 faces similar challenges in balancing precision and recall.

- **Precision:** 0.6305
  Model 6 has a precision score of 63.05%, which is higher than that of Model 1, Model 4, and Model 5. This shows that it is fairly good at correctly predicting positive classes when it does make a positive prediction.

- **Recall:** 0.4086
  The recall score of 40.86% indicates that Model 6 identifies a relatively higher proportion of true positive instances compared to Models 1, 4, and 5. However, like the others, it still misses a substantial portion of the positive instances.

The evaluation process reveals that Model 3 performs the best overall, followed by Model 2 and Model 6. Model 5 has the lowest performance across all metrics.

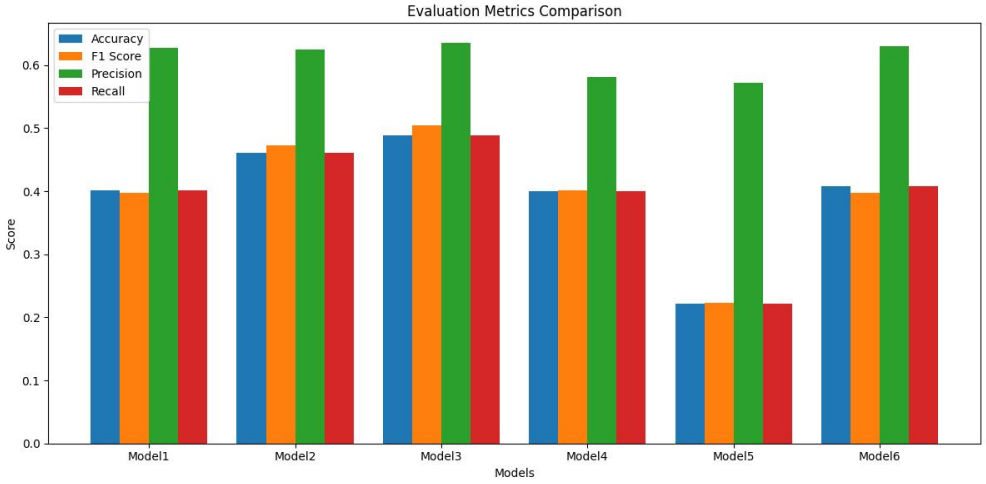

Figure 8: Evaluation Metrics Comparison: Combined bar chart of all model evaluation metrics.

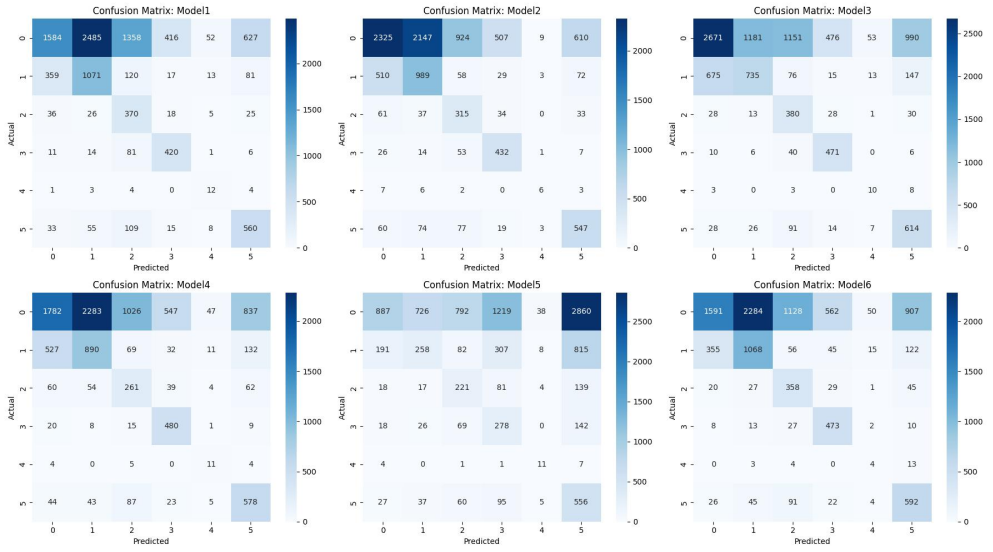

Figure 9: Confusion Matrices: Comparison of confusion matrices for each model.

5 Summary

This project demonstrates the potential of multimodal learning for hate speech detection in online environments where both textual and visual cues are essential for accurate classification. Through the implementation and comparative evaluation of six different multimodal fusion architectures, we found that late fusion techniques with dedicated feature transformations (Method 3) yielded the most effective results, achieving the highest accuracy and F1 score among all models tested. The study also underscores the challenges posed by class imbalance, which was partially alleviated through weighted cross-entropy loss. The use of majority voting for label aggregation proved benefi-

cial in dealing with annotator disagreement, enhancing label reliability. Visual analysis of confusion matrices further emphasized which classes were better handled by different models. Although the overall performance still leaves huge room for improvement, the findings highlight that thoughtfully designed fusion strategies significantly contribute to enhanced model robustness. Future work may involve exploring transformer-based cross-modal attention mechanisms and augmenting the dataset with synthetic samples to further address imbalance and improve minority class detection.

References

[1] Gomez, R., Gibert, J., Gomez, L., & Karatzas, D. (2019). Exploring Hate Speech Detection in Multimodal Publications. *arXiv preprint arXiv:1910.03814*.

[2] Kumar, D., Kumar, N., & Mishra, S. (2020). QUARC: Quaternion Multi-Modal Fusion Architecture for Hate Speech Classification. *arXiv preprint arXiv:2012.08312*.

[3] Sheng, V. S., Provost, F., & Ipeirotis, P. G. (2008). Get Another Label? Improving Data Quality and Data Mining Using Multiple, Noisy Labelers. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 614–622).

[4] Snow, R., O'Connor, B., Jurafsky, D., & Ng, A. Y. (2008). Cheap and Fast – But is it Good? Evaluating Non-Expert Annotations for Natural Language Tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing* (pp. 254–263).

[5] Cui, Y., Jia, M., Lin, T.-Y., Song, Y., & Belongie, S. (2019). Class-Balanced Loss Based on Effective Number of Samples. In *CVPR* (pp. 9268–9277).

[6] Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal Loss for Dense Object Detection. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 2980–2988).

[7] Liu, X., et al. (2020). Limitations of ROC on Imbalanced Data: Evaluation of LVAD Mortality Risk Scores. *arXiv preprint arXiv:2010.16253*.

[8] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*.

[9] Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter. *arXiv preprint arXiv:1910.01108*.

[10] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*.

[11] Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2017). Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers* (pp. 427–431).

[12] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. In *CVPR* (pp. 770–778).

[13] Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *ICML* (pp. 6105–6114).

[14] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *CVPR* (pp. 4510–4520).

[15] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv preprint arXiv:2010.11929*.

[16] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely Connected Convolutional Networks. In *CVPR* (pp. 4700–4708).

[17] Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv preprint arXiv:1409.0473*.

[18] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is All You Need. In *Advances in Neural Information Processing Systems*, 30.

[19] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv preprint arXiv:1406.1078*.

[20] Rosenblatt, F. (1958). The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review*, 65(6), 386–408.