Georgia Institute of Technology, Project Report : CS 7643 Predicting Hateful Memes

Ankit Baraskar
abaraskar6@gatech.edu

Helly Jain
hellyjain@gatech.edu

Lanfen Li
lli456@gatech.edu

Seyhan Emre Gorucu
sgorucu3@gatech.edu

Abstract

Hateful memes present a significant challenge for social media platforms such as Facebook. Machine learning methods can be used to filter the hateful memes. One challenge with hateful memes is that many of these memes do not convey hatred in an explicit form, but through sarcasm and paradoxes. Further, the combination of a benign image and benign text can create hateful content as well. Therefore, multi-modal techniques are suggested. In this research, we replicate, modify and analyze several deep learning implementations to identify hate memes.

1. Introduction/Background/Motivation

There is a lot of unfiltered content being generated on the internet that can be potentially harmful to society. Due to the widespread nature and influence of social media platforms like Facebook, hateful content can quickly propagate for immediate impact. Such impact causes psychological issues and conflict to arise, with content often targeted at minorities or specific identity groups[25]. Filtering hateful content is key for any online platform to help move towards a positive moral use of technology.

It is impossible to handle filtering this manually due to scale, and hence machine learning (ML) techniques are used to help automate this process. Automation does come with its own set of challenges. Most ML algorithms make predictions on either vision or language data alone (uni-modal), while lot of content exists in multi-modal form. Multi-modal is more challenging due to subtleties in meaning shifts when text is combined with vision. Moreover, language data has strong biases due to how language is usually structured and this overpowers any information retrieved from vision data. In this paper we explore both types, with a goal to find the best performing model to help us overcome these challenges.

Current deep learning (DL) state of the art models are transformer based architectures with and pre-training techniques introduced in BERT [5]. These techniques convert the data into vision and language embeddings. Then, they feed embeddings into different encoder architectures to represent semantic information commonalities/contrasts between vision and language data. Common architectures include single-stream from VisualBERT [12], dual-stream from ViLBERT [13] and dual encoders from CLIP [17]. These architectures are pre-trained so that it is learning generic features that help model transferability to various downstream vision-language tasks. The vision to language correlations are learned via cross modal techniques such as Masked Language Modeling (MLM) where the model is predicting masked pieces of text using a region of an image [10]. The objective function to be minimized is a negative log loss variant which includes both modalities. Once this pre-training process completes, the models can be fine tuned on downstream tasks such as detection of hateful memes where performance is benchmarked against human accuracy.

Due to large number of model parameters, the pre-training process can be slow and require lot of data. ALIGN has 675.4 million parameters and uses 1.8 billion image-text pairs [7] to contrast with ResNet-152's 60 million parameters [21]. Other techniques such as knowledge distillation can be used to reduce this process [3].

Facebook's Hateful Memes challenge was published with a multimodal dataset to tackle this problem [9]. It contains 10000 memes with 85% training, 5% dev, 10% test sets. Labels were created based on a specific definition of hatefulness [9]. However, an attack can be considered not hateful if it doesn't use any protected attributes or if against groups that also propagate hate. Words such as color of race may have probabilistic correlation with the output predictions (of being hateful) due to the frequency occurrence. To address this dataset bias and further add variation, benign confounders were added. These were based on existing memes that when modified the text/image flips the la-

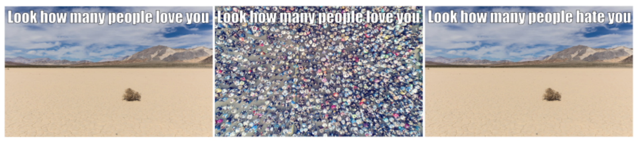
Look how many people love you
Look how many people love you
Look how many people hate you

Figure 1. Mean memes (left), benign image confounder (middle), benign text confounder (right)

bel from hateful to non-hateful (Fig:1). Hence, this dataset should capture some of the subtleties of world semantics and in turn produce performant models that predict reality well.

As of now, the competition is over [8]. The original paper [9] reported around 50-65% accuracy for the unimodal approaches. The baseline multimodal approaches had 60-65% accuracy (65-75% AUROC). Human accuracy is close to 85%. The winners of the competition achieved up to 85% AUROC [24, 15, 23]. However, the accuracy is around 75% –still below human level accuracy–. It is important to note that most of the winners used an ensemble technique, albeit without a different architecture.

2. Approach

We started off with unimodal implementations since they were easier to setup and start experimenting. Eventually needed to use multimodal techniques to match the multimodal structure of the dataset. Moreover, we also experimented pretrained models that were trained on large datasets and would be intuitively easier to optimize for better performance, in terms of compute used and less training needed. Implementation 3 also had additional data added to it which we hoped would not bias the data in a certain direction by adding variation and avoid overfit during training.

Implementation 1, Duplication of results from a Kaggle competition: We only duplicate the results of Placencia [16] by modifying the notebook to be able to run. The model converts images into 224*224 vectors. The words are converted into embedding vectors of size 500. The image data is fed into a convolutional neural network. Word embeddings are fed into a linear layer only. The results are concatenated and binary cross entropy is performed. Pytorch lightning is used to train the model. The dev accuracy and ROC AUC are only 0.52. The reasons for low accuracy may be that the model is initialized randomly and the language processing is done with a dense layer.

Implementation 2, Duplication of results from a Kaggle competition: We only duplicate the results of Moldabek [14] in the notebook. They use the randomly initialized Bertz model of the Keras Library. Images are not used at all. Texts are tokenized and each token is is embedded into a vector of size 160. The model gives 0.63 ROC AUC and 0.59 accuracy on the dev set.

Implementation 3, Multi-Modular Approach by using Driven Data 3rd Winner: We initially work on the notebook provided by Velioglu and Rose [23]. They do a combination of data expansion, image encoding, multimodel transformers such as Visual Bert [12], Vilbert, Transformers, and Ensemble Learning. Their first step is to increase the training dataset size by 428. They notice 100 images are not used at all. Further, they cherry-pick and label 328 images from the Memotion dataset [18]. For image encoding, they take the 6th fully connected layer of a ResNext-152 architecture based on Mask R-CNN trained on Visual Genome data. With further operations, they create a 100*2048 embeddings for each image. Next, they train a VisualBert that was pretrained on the COCO dataset. They do fine tuning on several parameters such as learning rate, learning rate ratio. The best performing 27 models are used in ensemble learning to give the final decision. We skip the ensemble learning part as each run takes 2-3 hours. However, we do fine-tuning and use other models than Visual Bert as well. For example, we implement Vilbert, Transformers, CNN LSTM and Concat Bow.

Implementation 4, Late Fusion Approach using ResNet50 and BERT: In this approach we trained two separate classification models; one on text and other on image using the same label. The prediction probabilities from the two model were multiplied and argmax was taken to calculate the predicted label. This approach will enable some ensemble affect in the model by combining prediction from two different models. For image classification model, we use pre-trained ResNet 50 architecture. This architecture was connected to a set of fully connected layers followed by cross-entropy loss. For training, the parameters from ResNet were frozen and the best performing model was fine tuned using batch size, learning rate, optimizer etc. For text classification we used BERT pre-trained classifier. This BERT model was connected to two fully connected linear layers and RelU. During prediction, the output from both models was fused to predict the final label. This model achieved 65% accuracy on validation data.

Implementation 5, Use CLIP Pre-trained Models and Zero-Shot method to Predict Hateful Meme : CLIP (Contrastive Language-Image Pre-training) builds on a large body of work on zero-shot transfer, natural language supervision, and multi-model learning. The idea of zero-data learning dates back over a decade but until recently was mostly studied in compute vision as a way of generalizing to unseen object categories.[22] "zero-shot" methods generally work by associating observed and non-observed classes through some form of auxiliary information, which encodes observable distinguishing properties of objects.[19] Hence, CLIP is a bridge between computer vision and natural language processing. The architecture of CLIP method can be found in reference [22]. CLIP method has been applied to train multi-model to Hateful Meme data. Kumar, Gokul Karthik and Nandakumar, Karthik published an article [11] and proposed the Hate-CLIPper architecture to model the cross-model interactions between the image and text representations obtained using Contrastive Language-Image Pre-training (CLIP) encoders via a feature interaction matrix (FIM). The classifier based on the FIM representation can achieve state-of-the-art performance on the Hateful Memes Challenge (HMC) dataset with an AUROC of 85.8. Combining arts and science together, several Kaggle projects use the pre-trained CLIP model to classify Hateful Meme image and text to Hateful and Good meme groups.

The problems encountered: The hateful memes challenge [1] is a challenge created by Multi-Model Framework (MMF) website [6] of Facebook AI Research (FAIR). The competition was done at Driven Data [4] platform. Initially, we were not able to retrieve the data. Driven Data informed us the competition is closed, and the data is not available. However, one author of [9] provided us the data with the zip file. The official MMF website is half-complete and seems abandoned. The MMF website provides an api to set up and train the models. However, the api has very limited documentation. The website provides some initial procedures and a Jupyter Notebook. However, the process requires a password from Driven Data, which is not accessible. After several days, we were able to start to understand the mmf api. We also found some codes for the same competition from Kaggle [2]. Further, Driven Data has a list of winners of the competition and their codes are available as well. However, the problem is a lot of the links are broken and the api's are not compatible with each other. We had to try several different paths. With tedious work such as analyzing and modifying mmf source code, and several trial and errors, we were able to get many models to run.

3. Experiments and Results

We use accuracy, F1 score and the competition metric AUROC as metrics. F1 score is the harmonic mean of precision and recall. AUROC means the area under receiving operating characteristics curve. The binary cross entropy returns a probability for a meme to be hateful. The threshold for this probability can be modified. Then, true positive rate and false positive rate at each probability threshold is calculated. True positive rate is plotted against the false positive rate at different probability threshold, which forms a curve. The area under this curve is called AUROC. True positive rate = True positives / (True positive + False negatives), and False positive rate = False positives / (False positives + True negatives).

Implementations 1 and 2 were only duplications of kaggle competitions and they show slightly better performance than random selection. With implementation 3, we tried several models such as Visual Bert, Vilbert, Transformer, CNN+LSTM and Concat Bow.

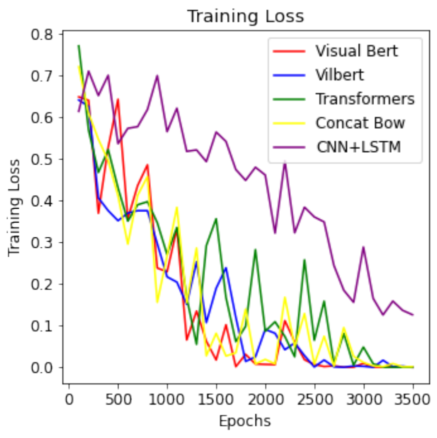

Figure 2. Training Loss for Visual Bert, Vilbert, Transformers, CNN+LSTM and Concat Bow

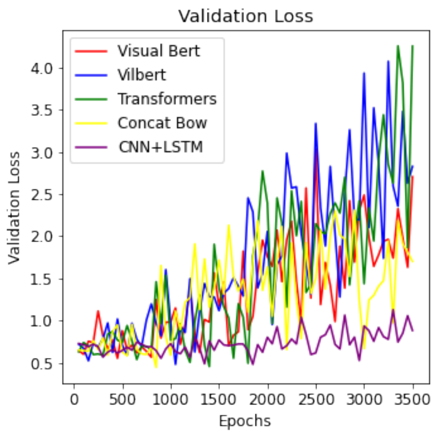

Figure 3. Validation Loss for Visual Bert, Vilbert, Transformers and CNN+LSTM

Figure 2 shows the training loss for five different architectures. While Visual Bert is pretrained on COCO dataset, the other models are randomly initialized. All of the models use binary cross entropy as the loss function, and the train loss goes to zero for all the models with the exception of CNN+LSTM. Perfect fitting is an indication of having more parameters than data. For example, Visual Bert, Vilbert, Transformers and CNN+LSTM have 112044290, 247780354, 171174978 and 2472304 parameters, respec-

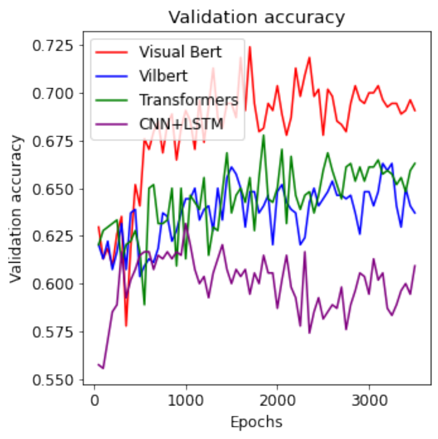

Figure 4. Validation accuracy for Visual Bert, Vilbert, Transformers and CNN+LSTM

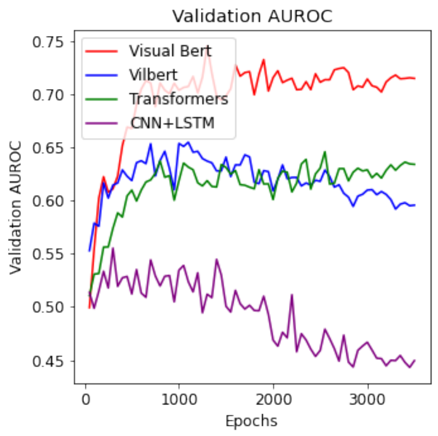

Figure 5. Validation AUROC for Visual Bert, Vilbert, Transformers and CNN+LSTM

tively. When compared to the fact that we only have 10000 images and texts, the number of parameters may be too many. CNN+LSTM is a much more shallow model with only one LSTM layer and several convolutional layers. Vilbert is the most complex model.

Figure 3 shows the loss for the validation set. CNN+LSTM is the only architecture whose loss is not increasing. All the other architectures have increasing validation loss indicating an overfit. Transformers and Vilbert have the largest validation loss. Figure 4 shows the validation accuracy for the four architectures. Visual Bert outperforms the other models. This may be because Visual Bert is pretrained on the COCO dataset which has 328000 images. Moreover, the original authors tuned the Visual Bert by modifying learning rate ratio, warm up, warm up factor, warm up iterations, learning ratio, scheduler warm up steps. We used their best reported hyperparameter values as well as default values, and the validation accuracy increased from 67% to 70%.

We hoped CNN+LSTM would show better accuracy as it doesn't overfit due to its simpler architecture. However, particularly LSTM is too simple to interpret text. While self-attention layers, present in Visual Bert, Vilbert and Transformers, enable communication between different parts of the text, LSTM's are known to have difficulty in carrying the information forward. Therefore, validation accuracy for CNN+LSTM sits around 60%. Another important note is that Visual Bert, Vilbert and Transformer validation loss is increasing as well as the validation accuracy.

Figure 5 shows the AUROC for the four models. Visual Bert shows almost 75% AUROC at the peak, which is a much higher result than all the other models. The original implementation comes from the 3rd winner of the Kaggle and they achieved 81% AUROC on the dev set. This is because they did a grid search over the aforementioned hyperparameters and created 69 models. They kept the best 27 models and did ensemble learning to predict. We were not able to do ensemble learning as each model was taking 2-3 hours and we didn't have the computational resources. For future, it would be nice to freeze many of the layers of the Visual Bert and use the pretrained parameters for those layers, while training the other layers. This may overcome overfitting and give better AUROC values. As we did the runs by using the mmf api, the api does not allow this as an argument. It is still possible to identify the model in the source code and freeze the layers inside the source code. This would be a suggestion for future implementation. Table 2 shows the summary of the validations results on the dev set.

Another cause for overfit could be the smaller batch sizes used for training. Referring to Figure 2 for all architectures (other than CNN+LTSM) we see the loss drop from 0.8 to 0.4 within the first 500 epochs which is a significant drop and could mean early overfitting. At regular intervals and around the same time for all models, the loss goes back up significantly potentially correlating with the batch change. We couldn't experiment with large batch sizes due to CUDA running out of memory. As a future direction, we'd like to use larger batch sizes and adjust the warmup so that we could smooth out the loss curves and address early overfitting issue.

The following is the experiments and results section for implementation 5. Our project starts from the Kaggle competition[1]. In previous sections, we have reported different mmf models by fine-tuning hyper parameters. Hence, it is interesting to learn whether the pre-trained CLIP method performs better than those mmf models or not. This experiment includes 3 steps (Fig:6)(Fig:7), which is a combination of science and art:

- Step1 - Contrastive analysis using pre-trained model:

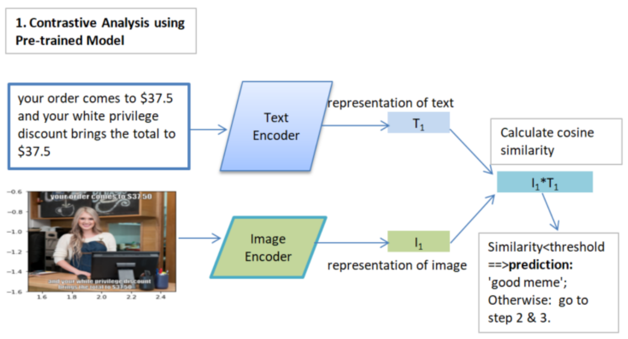
1. Contrastive Analysis using Pre-trained Model
your order comes to $37.5 and your white privilege discount brings the total to $37.5
Text Encoder
representation of text
T1
Calculate cosine similarity
I1*T1
Image Encoder
I1
representation of image
Similarity<threshold ==>prediction: 'good meme'; Otherwise: go to step 2 & 3.

Figure 6. Constrastive Analysis using pre-trained CLIP Model (Step 1)

Among 9 models of CLIP method, two models (RN50x16 and ViT-B/32) are selected for contrastive analysis. The model is used to encode the caption text and image and generate their representations, which are T1 and I1. As we know, the image goes with its caption text. The Cosine similarity [20] (Equation 1 - Cosine Similarity of Non-zero vectors A and B) of I1 and T1 is calculated to evaluate how appropriate text T1 is to that particular image. During the CLIP model pre-train process, this step is designed to maximize the similarity between the image and its caption text, and minimize the similarity between the image and other image's caption text. Hence, the small similarity could suggest that the image is randomly associated with the caption text. It is reasonable to assume that the small similarity indicates 'good meme'. This assumption has been confirmed by the following sensitivity analysis as well. The art and threshold sensitivity analysis help decide the final threshold. If the Cosine similarity is less than this threshold, the hateful image and caption text belongs to 'good meme'. Otherwise, continue to implement step 2 and step 3.

- Step 2 - Classifier from Label Text: The final target label is 'good meme' or 'hateful meme'. The former is interpreted to be a sentence of 'a non-hateful meme that is good', while the latter is interpreted to be a sentence of 'a hateful meme containing racism, sexism, nationality, religion or disability'. The model in step 1 is used to encode these two sentences to T1 and T2 representations.

- Step 3 - Use for zero-shot prediction: I1 is the representation of image in step 1. The Cosine similarity between I1 and [T1, T2] in step 2 is estimated, which will be normalized to probability using softmax. The predicted category of Hateful meme data is the one with larger probability.

$$S_C(A,B) = \frac{A*B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \quad (1)$$

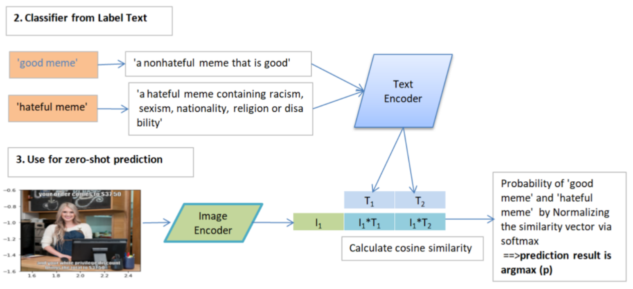
2. Classifier from Label Text
'good meme'
'a nonhateful meme that is good'
'hateful meme'
'a hateful meme containing racism, sexism, nationality, religion or disability'
Text Encoder
3. Use for zero-shot prediction
Image Encoder
I1
T1
T2
I1*T1
I1*T2
Calculate cosine similarity
Probability of 'good meme' and 'hateful meme' by Normalizing the similarity vector via softmax ==>prediction result is argmax (p)

Figure 7. Zero-Shot Prediction (Step 2 and 3)

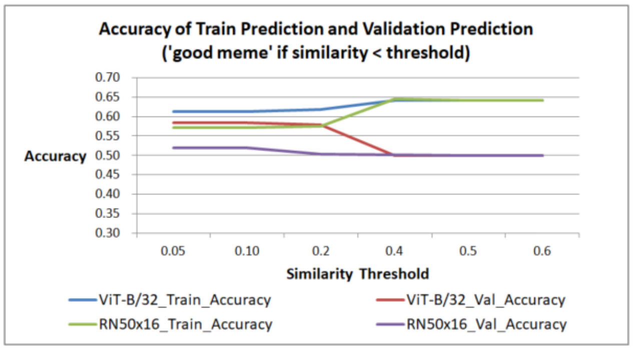

Figure 8. Sensitivity Analysis of Similarity Threshold using Accuracy Metric (Assumption 1)

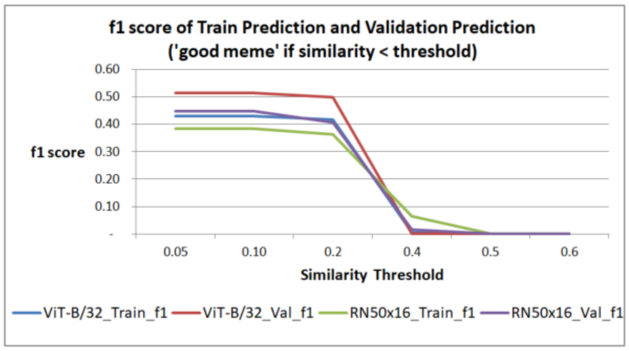

Figure 9. Sensitivity Analysis of Similarity Threshold using f1 score Metric (Assumption 1)

The sensitivity analysis of two models, ViT-B/32 and RN50x16, are performed with two assumptions:

- Assumption 1: Assume the image and caption text belongs to 'good meme' if their similarity is less than threshold. With different thresholds, the accuracy and f1 scores of train data and validation data are estimated for each model. In Figure 8 and Figure 9, it suggests that threshold being 0.2 generates good accuracy and f1 scores.

- Assumption 2: Assume the image and caption text belongs to 'good meme' if their similarity is larger than thresh-

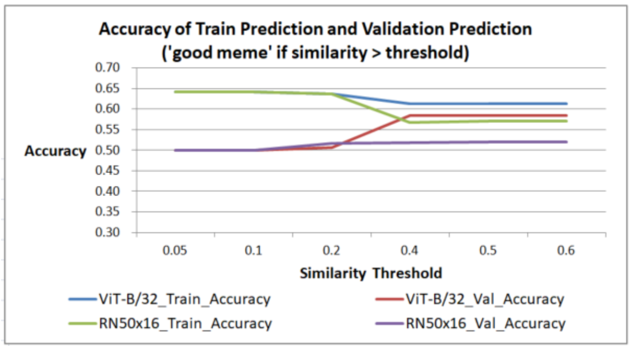

Figure 10. Sensitivity Analysis of Similarity Threshold using Accuracy Metric (Assumption 2)

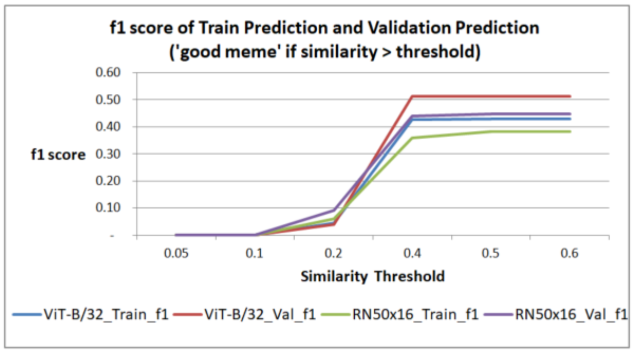

Figure 11. Sensitivity Analysis of Similarity Threshold using Accuracy Metric (Assumption 2)

old. With different thresholds, the accuracy and f1 scores of train data and validation data are estimated for each model. The sensitivity analysis under this assumption is to benchmark with the sensitivity analysis under assumption 1. In Figure 10 and Figure 11, it suggests that threshold being 0.4 generates good accuracy and f1 scores.

• The selected thresholds under assumption 1 and 2 generate the similar accuracy and f1 score. Also, we have assumed that a small similarity between image and caption text indicates a 'good meme'. Hence, we chose Assumption 1 with threshold being 0.2 for further analysis. It is that the Hateful meme is assumed to be 'good meme' when when similarity between image and caption text is less than 0.2.

With the above assumption, 9 pre-trained CLIP models are used to calculate the accuracy and f1 scores of train data and validation data (Table:1). Although train data is not used to train the model at all, the metrics of the train data can play a benchmark role when we select the appropriate pre-trained CLIP model. As we can see, three pre-trained models, ViT-B/32, ViT-B/16 and RN101 have large accuracy and f1 scores from validation data.

CLIP Models	Accuracy		F1 Score	
	Train	Val	Train	Val
RN50	0.55	0.50	0.49	0.55
RN101	0.59	0.57	0.51	0.58
RN50x4	0.50	0.54	0.52	0.61
RN50x16	0.58	0.50	0.36	0.41
RN50x64	0.63	0.53	0.45	0.4
ViT-B/32	0.62	0.58	0.42	0.50
ViT-B/16	0.53	0.55	0.52	0.62
ViT-L/14	0.50	0.52	0.51	0.61
ViT-L/14@336px	0.49	0.52	0.51	0.61

Table 1. Accuracy and f1 scores of Train and Validation Data (Assumption 1: threshold=0.2)

Model	Accuracy	AUROC	F1 Score
Humans	0.85		
Implementation 1	0.52	0.52	
Unimodal	0.59	0.63	
Visual Bert	0.67	0.69	0.50
Visual Bert (tuned)	0.7	0.73	0.6
Vilbert	0.67	0.64	0.38
Transformers	0.67	0.66	0.37
Transformers (tuned)	0.68	0.69	0.44
CNN+LSTM	0.65	0.56	0.25
Resnet50 + BERT	0.65		
MMBT	0.64	0.63	0.31
CLIP ViT-B/32	0.58		0.50
CLIP RN101	0.57		0.58
CLIP Vit-B/16	0.55		0.62

Table 2. Dev Set Metrics

Table 2 shows the accuracy, AUROC, and F1 scores for several implementations from all the team members. Tuned Visual Bert shows the best AUROC probably because it was pretrained on a larger dataset, and because several hyperparameters were tuned.

4. Conclusions

We have implemented several unimodal and multimodal algorithms such as Visual Bert, Vilbert, CLIP, Bert etc. to identify hate memes, a competition initiated by Facebook that is now over. Even though used different architectures and did hyperparameter tuning, the best performance was the tuned Visual Bert from [23]. We obtained 73% AUROC on the dev set while they obtained 81% on the test set by using an ensemble technique. We have identified potential causes for overfit which we were unable to address due to lack of resources. Referring to the previous section, the potential fixes can used as directives for future research and further optimize performance.

Student Name	Contributed Aspects	Details
All team members	Data Retrieval	All team members worked on data retrieval.
Ankit	Literature Review + Implementation 3	Introduction/Background/Motivation, Help with mmf api. Worked with Emre to experiment with VilBert, Transformers and MMBT. Fine tuned Transformers. Identified potential causes for overfitting issue and suggest possible fixes.
Helly	Late Fusion Model	I coded multiple architectures for image and text classification and narrowed down to the best performing architectures and fine tuned them for accuracy.
Lanfen Li	Clip + Implementation 3	I designed a flow chart on how to implement 9 CLIP pre-trained models. By running sensitivity analysis, I am able to select the best pre-trained model to hateful meme data. Also, I worked with Emre to experiment with MMBT model.
Seyhan Emre Gorucu	Implementations 1-3	For implementation 3, I got the reference code running. Further, I implemented Visual Bert, Vilbert, Concat Bow, Concal+LSTM and Transformers. I prepared the plots for these as well.

Table 3. Contributions of team members.

5. Work Division

The contributions of each team member can be found at Table 3.

References

[1] Hateful Memes Challenge. https://ai.facebook.com/blog/hateful-memes-challenge-and-data-set/. 3, 4

[2] Kaggle Hateful Memes Challenge. https://www.kaggle.com/code/atanudahari/hateful-memes-challenge. 3

[3] Wenliang Dai, Lu Hou, Lifeng Shang, Xin Jiang, Qun Liu, and Pascale Fung. Enabling multimodal generation on clip via vision-language knowledge distillation, 2022. 1

[4] Driven Data. https://www.drivendata.org/competitions/64/hateful-memes/. 3

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding, 2018. 1

[6] Facebook AI Research Multimodal Framework. www.mmf.shl. 3

[7] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V. Le, Yunhsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision, 2021. 1

[8] Douwe Kiela, Hamed Firooz, Aravind Mohan, Vedanuj Goswami, Amanpreet Singh, Casey A. Fitzpatrick, Peter Bull, Greg Lipstein, Tony Nelli, Ron Zhu, Niklas Muennighoff, Riza Velioglu, Jewgeni Rose, Phillip Lippe, Nithin Holla, Shantanu Chandra, Santhosh Rajamanickam, Georgios Antoniou, Ekaterina Shutova, Helen Yannakoudakis, Vlad Sandulescu, Umut Ozertem, Patrick Pantel, Lucia Specia, and Devi Parikh. The hateful memes challenge: Competition report. In Hugo Jair Escalante and Katja Hofmann, editors, *Proceedings of the NeurIPS 2020 Competition and Demonstration Track*, volume 133 of *Proceedings of Machine Learning Research*, pages 344–360. PMLR, 06–12 Dec 2021. 2

[9] Douwe Kiela, Hamed Firooz, Aravind Mohan, Vedanuj Goswami, Amanpreet Singh, Pratik Ringshia, and Davide Testuggine. The hateful memes challenge: Detecting hate speech in multimodal memes, 2020. 1, 2, 3

[10] Wonjae Kim, Bokyung Son, and Ildoo Kim. Vilt: Vision-and-language transformer without convolution or region supervision, 2021. 1

[11] Gokul Karthik Kumar and Karthik Nandakumar. Hate-clipper: Multimodal hateful meme classification based on cross-modal interaction of clip features, 2022. 3

[12] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. Visualbert: A simple and performant baseline for vision and language, 2019. 1, 2

[13] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks, 2019. 1

[14] Azamat Moldabek. Kaggle competition codes. https://www.kaggle.com/code/azamatmoldabek/multimodal-modeling-for-hateful-memes-detection. 2

[15] Niklas Muennighoff. Vilio: State-of-the-art visio-linguistic models applied to hateful memes, 2020. 2

[16] Victor Villacorta Plasencia. Kaggle competition codes. https://www.kaggle.com/code/kvdatadragon/learning-multimodal. 2

[17] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry,

Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. 1

[18] Chhavi Sharma, Deepesh Bhageria, William Scott, Srinivas PYKL, Amitava Das, Tanmoy Chakraborty, Viswanath Pulabaigari, and Björn Gambäck. SemEval-2020 task 8: Memotion analysis- the visuo-lingual metaphor! In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 759–773, Barcelona (online), Dec. 2020. International Committee for Computational Linguistics. 2

[19] Zero shot learning. Wikipedia. https://en.wikipedia.org/wiki/Zero-shot_learning. 3

[20] Cosine similarity. Wikipedia. https://en.wikipedia.org/wiki/Cosine_similarity. 5

[21] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2019. 1

[22] CLIP: Connecting Text and Images. https://openai.com/blog/clip. 2, 3

[23] Riza Velioglu and Jewgeni Rose. Detecting hate speech in memes using multimodal deep learning approaches: Prize-winning solution to hateful memes challenge, 2020. 2, 6

[24] Ron Zhu. Enhance multimodal transformer with external label and in-domain pretrain: Hateful meme challenge winning solution, 2020. 2

[25] Oana Ștefăniță and Diana-Maria Buf. Hate speech in social media and its effects on the lgbt community: A review of the current research, 2021. 1