Detecting Hate Speech in Memes Using Multimodal Deep Learning Approaches

Chao Wang cwang339@gatech.edu
Kai Ni knimr3@gatech.edu
Jianan Song jsong404@gatech.edu
Zhiyun Chen zchen763@gatech.edu

Abstract

Hateful meme detection is a recent research domain which needs both visual, language context from the meme itself plus additional background knowledge to understand it well. This report summarizes our work of the group project implementation which employs wide-and-deep model and LSTM plus attention model on pre-trained embeddings from both image and text. We list out the experiments and compare our results between our models and with state-of-the-art models as well. With LSTM plus attention model, we are able to achieve 65% test accuracy.

1. Introduction

In Hateful Memes challenge, we use image and text features to detect multimodal classification. Specifically we have used three different models to do the detection. As mentioned in the challenge, the state-of-the-art method performs with 63.0% accuracy[5]. Our goal is to exceed the accuracy of state-of-the-art result.

We cover the related work in Section 1.2. There are also several other papers that combine image and text together for various machine learning use cases. For example, Rex[14] and William[20] combine image and text embedding and feed into graph convolutional neural networks to generate high quality embeddings for all items. Paul [12] combines image, text and other user background information together for YouTube recommendations.

The limits nowadays are that these tasks of combining multiple resources have not achieved the accuracy comparable to human levels. For example, the top 5 winners in Hateful Memes competition achieved about 80-85$\%$ of accuracy only. The top performance models usually have complex deep learning models which are difficult to train for computers with limited GPU resource. Given our intermediate levels of understanding of deep learning and limited time of the project, it is hard to completely understand the winning models and replicate their results. For this reason, we try to get a balance between performance and computing time.

1.1. Problem Descriptions

Hateful memes detection is a new research area that requires visual and text understanding of the meme to perform classification task. To detect hateful content combined with text and pictures, the computer must be able to understand the explicit and inexplicit meaning of the content. The words and the photos are not independent and we have to understand the combined meaning together. To implement this with an algorithm, the program needs to analyze the text and the image separately. In our project, we will leverage several approaches from pre-trained vision and language embeddings to build neural networks models to detect hateful memes. The objective of this project is to apply multimodal to detect real-world hate contents with satisfactory results.

1.2. Related Work

The current multimodal classification established baselines by using several well-known model architectures [8]. Top 5 winning submissions involve using OCR and inpaint model to find and remove the text from the image [21]. and use NLP transformers to images [11]. There are also approaches with growing training sets by finding similar datasets on the web [19]. Other winners also incorporate common sense knowledge into the network. One of the winners also used simple ensembles to smooth out the predictions and tried LXMERT and VLP [16]. Another winner picks out the mislabeled samples and augments them accordingly [10]. In general, all winners involve extensive hyperparameters tuning and trying out different models.

1.3. Why This is Important

Internet memes are often harmless and sometimes hilarious. However, by using certain types of images, text, or combinations of each of these data modalities, the seemingly non-hateful meme becomes a multimodal type of hate speech, a hateful meme [4]. Detecting hateful content can be quite challenging, since memes convey information through both text and image components. The challenges of harmful content [8] affect the entire tech industry and society at large scale.

Many people care about hateful memes to build a healthy

Internet community. They include software engineers, e-commerce companies, social network communities, government officials and entrepreneurs. Currently big tech companies are doing human effort annotation for hateful intents spreading online. If this project can be successful, people can use the model to detect hateful intents in a much more scalable way to save time of annotators, reduce the latency of detection and improve the accuracy as well. The success could potentially reduce the crime and hate rate on Internet, save people from terrorist attacks and so on.

1.4. Data and Model Used for the project

1.4.1 Dataset

We use the dataset published by The hateful memes challenge from `https://hatefulmemeschallenge.com/#download`. This challenge is created by Facebook AI to drive and measure progress on multimodal reasoning and understanding. There are 5 jsonl files in the original dataset and we use train.jsonl, dev_unseen.jsonl, and test_unseen.jsonl as our training set, validation set and test set respectively. In total, our dataset includes 12,140 images. Each line in the .jsonl file is a JSON-encoded example. It contains three fields: 1. the text occurring in the meme; 2. the image that is a path to the image in the image directory; 3. the label for the meme with 0 for not-hateful and 1 for hateful, provided for train and dev .jsonl files.

1.4.2 Existing code and models used

We use the pretrained models in torchvision for Resnet-50, Vgg-19 and Googlenet and they provide us $1,000$-d image features from last layer. For text embedding, we use Glove[15] to generate word2vec embeddings from https://github.com/stanfordnlp/GloVe.

1.5. Deep Learning Framework

In this project, we use Pytorch 1.90 deep learning framework. Wide-and-deep model is trained locally with CPU only and LSTM FC and attention models are trained on Google Colab with GPU.

2. Approach

2.1. Work Overview

Other than replicating state-of-the-art models from web resources, we implement less complicated models using the knowledge we learned from this course to balance performance and training time. Our implementation consists feature extraction and deep-learning models. In extraction of the image features, we use CNN with pre-trained weights such as VGG-19[7], Resnet-50[6], AlexNet[9] and GoogleNet[2]. These well known models are trained on ImageNet with 1,000 classes. We have the images feed into the models and extract the values from the last layer and use them as our deep learning model inputs. On the text side, we extract Word2Vec[18] embeddings from the texts and average them over the whole phrase as our text embeddings fed into the deep learning model. We also train a LSTM model on word embeddings and use the last hidden values as inputs of the model.

2.2. Solution 1: Wide-and-Deep Model

We choose two types of deep learning models, The first one is based on wide-and-deep [1]. Refer Figure 1 for architecture. The wide-and-deep model is originally used in recommendation system for item recommendations in big retails such as Amazon and Alibaba. We innovatively adapt the model here for multi-modal classification problem. The logic behind wide-and-deep is that one set of input features is much more explicit to the label while the other set is implicit. In Cheng's work[1], a deep FC model is used to inference the implicit features and a wide shallow FC model to memorize the explicit features. We choose this model and believe its success in the task because we also have two resources of features in Hateful Memes. The image contains a lot more implicit information than the text. Thus we can build separate FC models on top of the two sets of embeddings, with a deep model on image and a wide(shallow) model on text. We will concatenate the outputs from two models and feed them into one layer for binary classification. Through this way, we achieve 62.5% accuracy in testing which is close to the state-of-the-art methods.

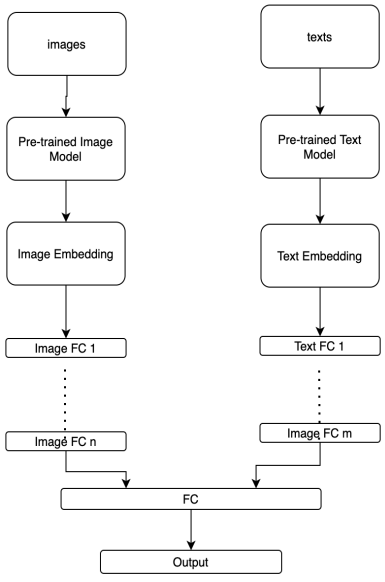

Figure 1. Architecture of Wide-and-Deep model

2.3. Solution 2: LSTM+FC and Attention

Our second model is based on our innovative thoughts on attention inspired by "what color is Greg" example in Lesson 14 Attention class. We first compose all the images into a 224X224X3 matrices, then use a 11X11 filters to generate a set of vector $\{u_1, u_2, \ldots, u_N\}$, we let our text embeddings to the query vector q. We use the attention model based on q and u for the final label of the binary classification. In this way, we achieved 64.25% accuracy in test data which exceeds 63.0% of state-of-the-art results as stated in project statement.

2.4. Pitfalls and Lessons Learned

In doing this project, we met several pitfalls and learned lessons from them. Our first pitfall is to train an combination of image embedding and text embedding. This is a too heavy task for the project given the time and every group member's bandwidth. We instead use pre-trained models such as VGG, Resnet for image and word2vec for text to extract embeddings. Secondly, as to the feature extraction, we were struggling on the choice of layers for embedding outputs. Outputs from first few layers of these will have bigger potential to adapt to our use case but these features might be too rough to use. Choose the outputs from deeper layers could be much more representative, however, it might be difficult to adapt to our use case. Because of the relatively small amount of training data, using the features from shallow layers may not be a good idea for the models to learn the embeddings sufficiently. Therefore we extract image and text embeddings from the full model of VGG, Resnet and etc. Last but not least, for beginners in deep learning, it is so inclined to create a large neural networks to improve the accuracy which may become susceptible to over-fitting problem. For this project, the testing accuracy will be relatively low and the training accuracy is very high with more complex models. For this pitfall, we try to prim the complexity of models and implement a medium size models to train and also put a large probability for dropout rate. In that way, we could increase the test accuracy by around 15%.

2.4.1 Combination of Models

Before we started the project, we expected the combination of image and text embeddings with fully connected layers will not achieve good performance. We achieved 56.6% accuracy in our first trial and proved the feasibility.

2.4.2 Overfitting

However, we found our training accuracy is about 99% which suggests significantly over-fitting in our experiment. We tuned hyper-parameters in layer size and dropout rate in order to improve the accuracy on test data and achieved 63% later.

2.4.3 Choice of Models

We met another problem that there are so many choices of pre-trained models and not sure which one will be the best. Given the limited time of the project, we tried several well know pre-trained embeddings such as Resnet, VGG and GoogleNet. This is definitely not an exhaustive list, but sufficient to achieve a reasonable accuracy.

3. Experiments

We explored different models to find the best way to solve the hateful memes problem. Generally speaking, we use pre-trained models to extract image embeddings and text embeddings. Using different methods(simple concatenating and attention mechanism) to combine the information from both parts, we build our own neural network to make judgement. Different models are built to do contrast and comparison.

3.1. Success Metrics

Various metrics are used from different perspectives to measure the performance of our modeling estimation. We plot the confusion matrix for the actual labels and predicted classification in the test dataset. For the binary classification, confusion matrix describes an output of negative and positive hateful ones by actual and predicted results. It provides an intuitive summary of the inputs to the calculations of metrics to measure modeling success.

Based on the confusion matrix, the first metric used in our project is accuracy. Accuracy measures how often our model is correct overall. The equation is defined as follows:

$$Accuracy = \frac{true\ positive + true\ negative}{sample\ size} \quad (1)$$

However, the positive classification here is to detect the hateful memes. An actual hateful memes is not correctly detected and assigned as not hateful is malicious for user engagement in the Facebook platform and thus potentially affect the platform success. Thus, the cost of having a misclassified false negative is very high in our case and accuracy is not the be-all and end-all model metric to use when selecting the best model.

Another two metrics are precision and recall. The calculation equations are as follows:

$$Precision = \frac{true\ positive}{true\ positive + false\ positive} \quad (2)$$

$$Recall = \frac{true\ positive}{true\ positive + false\ negative} \quad (3)$$

The denominator of precision is the total predicted positive, thus precision measures how precise/accurate the model is out of those predicted positive, how many of them are actual positive. Precision is a good measure when the costs of false positive is high. In our case, the cost of classifying a unhateful memes as hateful is an unpleasant experience for users, but the cost is not as high as the false negative. The denominator of recall is the total actual positive, thus recall calculates how many of the actual positives our model capture through labeling it as positive. Recall should be a good measure to select the best model when there is a high cost associated with false negative. Therefore, recall fits our project since if a hateful meme is predicted as unhateful, the consequence can be very bad for the user experience and engagement.

F1 score is a function of precision and recall. The formula is as follows:

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{4}$$

F1 score is to seek a seek a balance between precision and recall. While accuracy is based on true positive and true negative and does not focus on false negative and false positive which usually has business costs, either tangible or intangible, F1 score might be a better measure to use if we need to seek a balance between precision and recall and there is an uneven class distribution over the confusion matrix.

We also use AUC (Area Under The Curve) ROC (Receiver Operating Characteristics) curve to check or visualize the performance of hateful classification model. AUC - ROC curve is a performance measurement for the classification problems at various thresholds. ROC is a probability curve, plotted with TPR (same as recall) against the FPR where TPR is on the y-axis and FPR is on the x-axis. The equation for FPR is as follows:

$$FPR = \frac{true\ negative}{true\ negative + false\ positive} \tag{5}$$

AUC represents the degree or measure of separability. It tells how much the model is capable of distinguishing between different classes. Area Under the Receiver Operating Characteristics (AUROC) is one of the most important evaluation metrics for checking the classification model's performance. The higher the AUROC is, the better the model is at distinguishing different classes.

3.2. Loss Function Definition

The hateful memes problem is an binary classification problem. Pytorch provides standard binary crossentropy function. To give higher weights to the mistaken classified examples, we also consider the focal loss. When we do statistics about the training examples, we found only 65% of the training examples are positive. Based on that, we also reweight the positive and negative examples with the alpha parameter in focal loss.

3.3. Experiment 1: Wide-and-Deep Model

After we extract features from image mode land text model, We get $1,000$ dimension raw embedding of images and 50-dimension raw embedding of texts. In order to combine them together, We tried different layers on the two sets of embeddings and then concatenate them together for the last layer with the binary label. In order to put the dimension from text and dimension from image on the same level, we use a much deeper model to bring down the dimension of image side gradually to 50-dimension.

3.4. Experiment 2: LSTM-FC model

We believe simply averaging the text embedding [13] from pre-trained models might not reflect the text sequential information. We want to train our our own recurrent neural network LSTM. The output of the hidden units of the LSTM is then combined with image embedding to build upper layer Neural Network. This model better leverage the text sequential information, but how to handle the different length of the text and unknown tokens are tricky. To process the text part of the memes in batches, truncating and padding are used. From statistics, the average length of text is 11 and standard deviation is 8. Based on those information, we truncate or pad all the text to length of 30. For the unknown words, we first take a random vector as its embeddings, but it seems working slightly better when we just take zero vector as its embedding.

Intuitively, simple concatenation of text embedding and image embedding might not be the way that we human judge a hateful memes. We usually first see the text information of a hateful meme. Then we see the image information of the hateful meme.For example, in the hateful meme in Figure 2, when we read "Look how many people love you", we know the content of the text,which is talking about the number of people that love you,but we do not know exactly the number "how many". We need two query in the image to see exactly how many people are there. This is the way we want our algorithm to imitate.

3.5. Experiment 3: LSTM-Attention model

At the first step, we employ the LSTM model to encode the text input information. In parallel, we also use CNN to encode the image information. We believe that different channels in the CNN output captures different features of the image. After LSTM and CNN finish their encoding respectively, we use LSTM hidden layer output as Query, CNN outputs as Keys and Values to do attention mechanism. The query result is then fed into a shallow Neural

Network.

Besides, in this model, we also tried focal loss on the dataset. On one hand, we want to give more weights on the mis-classfied examples. On the other hand, the training dataset is in fact unbalanced. We want to employ focal loss so that the learning process could reweight the positive and negative examples.

4. Results

4.1. Wide-and-Deep Model

Table 1 shows the accuracy of combining images models and the text model together before feeding into wide-and-deep models. Here the text model is averaging over pre-trained word2vec embeddings. And image embeddings are extracted from Resnet-50, Googlenet and VGG-19. The training accuracy are 88.35%, 98.61% and 84.88% and the testing accuracy are 62.5%, 60.75% and 62.5% from Resnet-50, Googlenet and Vgg-19. We tuned various combinations of hyper-parameters such as learning rate, dropout rate, and epochs. The accuracy hits the ceiling of 62.5% in wide-and-deep methodology and can hardly go beyond. Resnet-50 and VGG-19 show the best result in testing because these two models are invented later than Googlenet and have been improved based on the old model architecture [3].

Figure 3 shows the confusion matrix for the text dataset using the Wide-and-Deep model. This trained model has an accuracy of 0.5765 on the test date. Since the cost for false negative is high in our project, accuracy is not the most desirable metric to measure the performance. Besides, testing results show that precision equals to 0.4290 while recall equals to 0.3907. A low recall value implies a high false negative ratio, which is not desirable for our project. Our corresponding F1 score is 0.4090, which balances and integrates the precision and recall values. Figure 4 shows our AUC ROC plot results for the testing data and our AUC value is 0.5471. This model only show a degree of separability at 0.5471, which is not desirable and LSTM-FC and LSTM-ATT models are introduced to improve the prediction results. Overall, the wide-and-deep model does not fully achieve our expectation and we will talk about our improvements in the next two subsections.

4.2. LSTM-FC Model

Figure 5 shows the confusion matrix for the text dataset using the LSTM-FC model. Based on the matrix, the trained model has an accuracy of 0.63 on the test date, which proves to improve our prediction results. Because of the associated cost in false negative and false positive, more metrics are calculated. For this model, precision equals to 0.5103 and recall equals to 0.3307. A lower recall value implies a high false negative ratio, which is not desirable for our project. Our corresponding F1 score is 0.4015, which implies that this model improves on the Wide-and-Deep model when balancing the precision and recall values. Figure 6 shows our AUC ROC plot results for the testing data and our AUC value is 0.6132. Our model still shows a comparable degree of separability and is capable of distinguish between hateful and unhateful memes at an effectiveness rate of 0.6132.

Figure 7 and Figure 8 shows the learning curve for LSTM_FC model.Definitely, the model learns some patterns of the memes dataset. The average loss of the model decrease steadily as epochs increase. However, the training accuracy and dev accuracy diverges. Obviously, the reason behind is that the model starts to overfit. We think there might be three ways to solve problem.The first one is to employ Dropout layer or regularization,which has already been added into the model.The second way is to increase the dataset or do data wrangling,which is not practical at the moment. The third way is to build a concise model biased on human level to judge a hateful meme example. In the LSTM_ATT moel, we are trying on this approach.

4.3. LSTM-Attention Model

Figure 9 shows the confusion matrix for the text dataset using the LSTM-Attention model. Based on the matrix, the accuracy of the LSTM-Attention model is 0.6425 which is slightly higher than the accuracy of the LSTM-FC model. However, accuracy is not the unique metric we are measure the performance. The LSTM-Attention model has a slightly higher precision value at 0.5499 and much lower recall value at 0.2573. Thus, the LSTM-Attention model improves on the final accuracy by having higher true positive rate but at the cost of higher false negative rate. This is desirable for our project since the comparative cost is higher to fail to classify a actual hateful meme as not hateful. The F1 score is 0.3506, lower than the LSTM-FC model. It implies that LSTM-Attention does not improve on the LSTM-FC model when considering the true positive and false negative rates. Figure 10 shows our AUC ROC plot results for the testing data and our AUC value is 0.6132, which is very close to the results in LSTM-FC model. Overall, LSTM-Attention model still shows a comparable degree of separability and is capable of distinguish between hateful and unhateful memes at an effectiveness rate of 0.6132.

Figure 11 and Figure 12 shows the learning curve for the model. Even though we employ Dropout and regularization, the over-fitting problem is still severe. The model is still rather complex and we need more data to generalize well. From the testing accuracy perspective, it goes slightly better than previous model, hitting the accuracy of 63.4%. We believe this model has the potential to get much better result. We could try transformer to encode text information, multi-heads attention and do data wrangling if time permits.

Image Embedding Model	Learning Rate	Dropout	Epochs	Training Accuracy (%)	Test Accuracy (%)
RESNET	0.02	0.7	300	88.35%	62.5%
GoogleNet	0.001	0.7	300	98.61%	60.75%
VGG	0.01	0.7	300	84.88%	62.5%

Table 1. Hyperparameter Tuning Results

4.4. Hyper-parameter Tuning

To achieve better result, we did extensive hyper-parameter tuning. In our deep and wide model, to counteract the over-fitting problem, we generally try to adjust the output units of the hidden layers. We also tried different learning rate. Higher learning rate makes the model learn fast, but the learning curve oscillates severely as epochs go on. We choose relatively small learning rate to achieve stability. We also tried different pretrained models to get image embedding, including GoogleNet,VGG and Resnet. For the text embedding, we tried 50-dimension and 300-dimension pretrained word embeddings. It seems 50-dimension embeddings works better.

For LSTMFC model, we first adjust the hidden units of the FC layers. We also tune the structure of the LSTM part of the model. For example, the number of the units in hidden layer,the number of the layers of the LSTM model. We also explored 50-dimension and 300-dimension embeddings. In this model, 300d embedding works better.

In the LSTMATT model, LSTM model structure and hidden units number are tuned. Because in this model we used Focal Loss instead of binary cross entropy, we tuned the Focal Loss parameter alpha and gamma. They take value of 0.5 and 1.0 working best. We also tried regularization in this model,but the effect is trivia.

4.5. Learned Parameters

Our inputs into the fully connected layers are extracted from pre-trained models without learning. In wide and deep model, our learning starts from the fully connected layers after we get image and text embeddings. Our LSTM + FC and LSTM + attention models, we learn parameters in LSTM, Fully connected layers, and attention models through Convolutional neural network.

5. Future Work

In our future work, we will use OCR algorithm to anchor the position of texts and fill in with similar colors. We will enhance training data samples by adding variants of the images. For example, flipping the images horizontally or adding some masks in images. We need to crawl background information related to Hateful Memes to enhance the background knowledge. For example, we can add some contexts like skunk is smelly and rose smells good. We can use more advanced models such as VL-Bert [17], multi-modal transformer [21] and etc to further improve the accuracy. We can also train the model on platforms such as AWS and GCP with bigger and faster computation capacity.

6. Work Division

Every member has contributed to this project equally. Table 2 shows the work contribution and details for each team member.

References

[1] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 7–10, 2016.

[2] Yangqing Jia Pierre Sermanet Scott Reed Dragomir Anguelov Dumitru Erhan Vincent Vanhoucke Andrew Rabinovich Christian Szegedy, Wei Liu. Going deeper with convolutions. *arXiv:1409.4842v1*, 2014.

[3] Siddharth Das. Cnn architectures: Lenet, alexnet, vgg, googlenet, resnet and more.... *Analytics Vidhya*, 2017.

[4] CASEY FITZPATRICK. Meet the winners of the hateful memes challenge.

[5] Praveen Jalaja. Hateful meme detection. *medium.com*, 2016.

[6] Shaoqing Ren Jian Sun Kaiming He, Xiangyu Zhang. Deep residual learning for image recognition. *arXiv:1512.03385v1*, 2015.

[7] Andrew Zisserman Karen Simonyan. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015.

[8] 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. *arXiv preprint arXiv:2005.04790*, 2020.

[9] Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv:1404.5997v2*, 2014.

[10] Phillip Lippe, Nithin Holla, Shantanu Chandra, Santhosh Rajamanickam, Georgios Antoniou, Ekaterina Shutova, and Helen Yannakoudakis. A multimodal framework for the detection of hateful memes. *arXiv preprint arXiv:2012.12871*, 2020.

[11] Niklas Muennighoff. Vilio: State-of-the-art visio-linguistic models applied to hateful memes. *arXiv preprint arXiv:2012.07788*, 2020.

Student Name	Contributed Aspects	Details
Zhiyu Chen	Text Embedding and Modeling Implementation	Apply pre-trained image model (GloVe) to extract the text embedding, build multiple models to train; analyze and write up the results.
Kai Ni	Image Embedding and Modeling Implementation	Apply pre-trained image model (Convolution) to extract the image embedding, combine text and image embedding together to train; analyze and write up the results.
Jianan Song	Coding Environment Setup and Modeling Implementation	Setting up environment for team work; organizing team meeting; help to build up image embedding and modeling implementation; analyze and write up the results.
Chao Wang	Text Embedding and Metrics Buildup	Building up metrics to measure modeling success; help to build up image embedding; analyze and write up the results.

Table 2. Contributions of team members.

[12] Emre Sargin Paul Covington, Jay Adams. Deep neural networks for youtube recommendations. *Proceedings of the 10th ACM Conference on Recommender Systems*, 2016.

[13] Martín Pellarolo. How to use pre-trained word embeddings in pytorch. *https://medium.com/@martinpella/how-to-use-pre-trained-word-embeddings-in-pytorch-71ca59249f76*, 2019.

[14] Kaifeng Chen Pong Eksombatchai Rex Ying, Ruining He. Graph convolutional neural networks for web-scale recommender systems. *Association for Computing Machinery*, 2018.

[15] Julien Velcin Robin Brochier, Adrien Guille. Global vectors for node representations. *arXiv:1902.11004v1*, 2019.

[16] Vlad Sandulescu. Detecting hateful memes using a multimodal deep ensemble. *arXiv preprint arXiv:2012.13235*, 2020.

[17] Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. Vl-bert: Pre-training of generic visual-linguistic representations. *arXiv preprint arXiv:1908.08530*, 2019.

[18] Greg Corrado Jeffrey Dean Tomas Mikolov, Kai Chen. Efficient estimation of word representations in vector space. *arXiv:1301.3781v3*, 2013.

[19] Riza Velioglu and Jewgeni Rose. Detecting hate speech in memes using multimodal deep learning approaches: Prize-winning solution to hateful memes challenge. *arXiv preprint arXiv:2012.12975*, 2020.

[20] Jure Leskovec William L. Hamilton, Rex Ying. Inductive representation learning on large graphs. *31st Conference on Neural Information Processing Systems*, 2017.

[21] Ron Zhu. Enhance multimodal transformer with external label and in-domain pretrain: Hateful meme challenge winning solution. *arXiv preprint arXiv:2012.08290*, 2020.

Appendix

Figure 2. An example hateful memes, *Image is a compilation of assets, including ©Getty Image.*

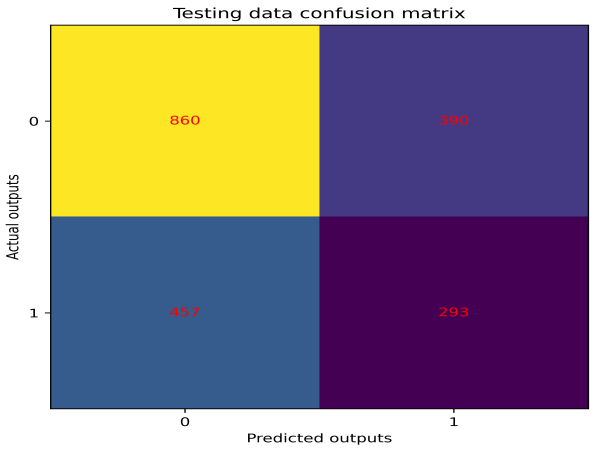

Figure 3. Confusion Matrix for Wide and Deep model

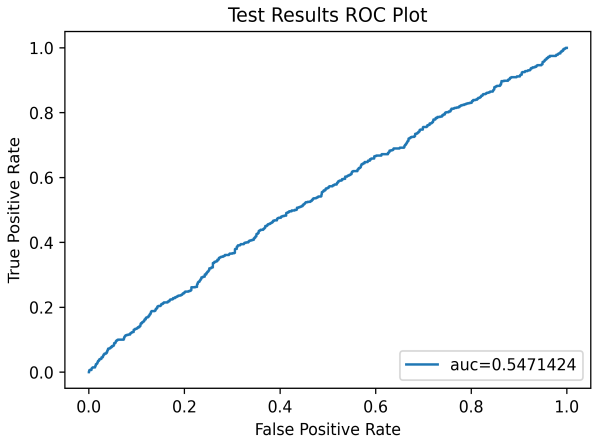

Figure 4. AUC ROC plot for Wide and Deep model

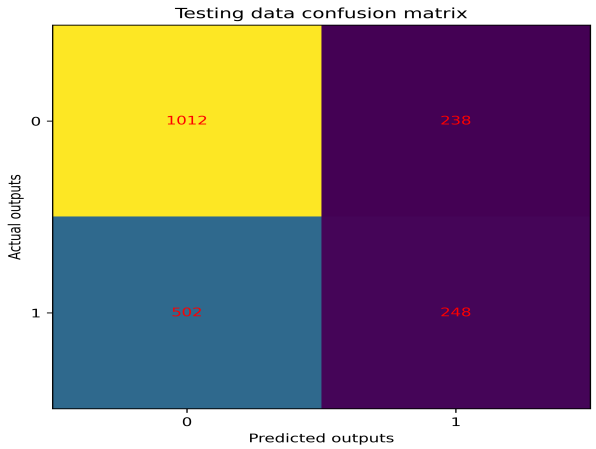

Figure 5. Confusion Matrix for LSTM_FC model

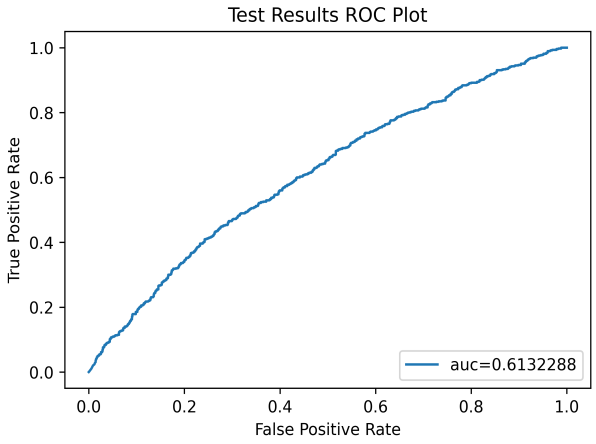

Figure 6. AUC ROC plot for LSTM_FC model

Figure 7. Loss for LSTM_FC model

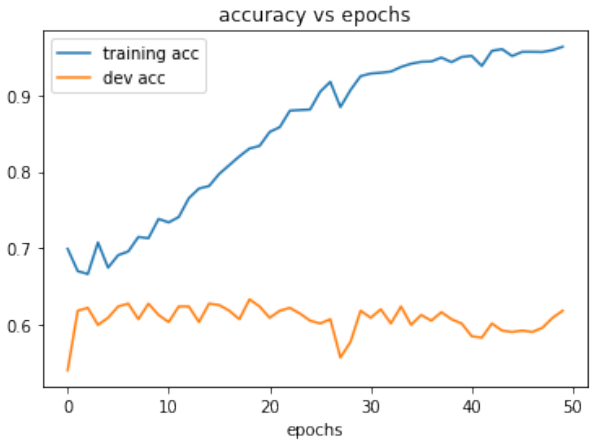

Figure 8. Accuracy for LSTM_FC model

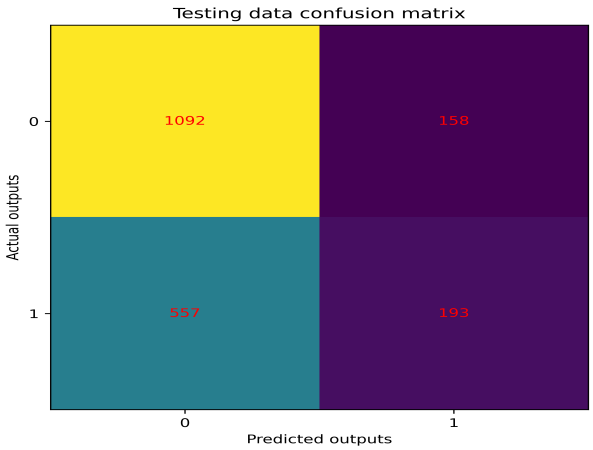

Figure 9. Confusion Matrix for LSTM_ATT model

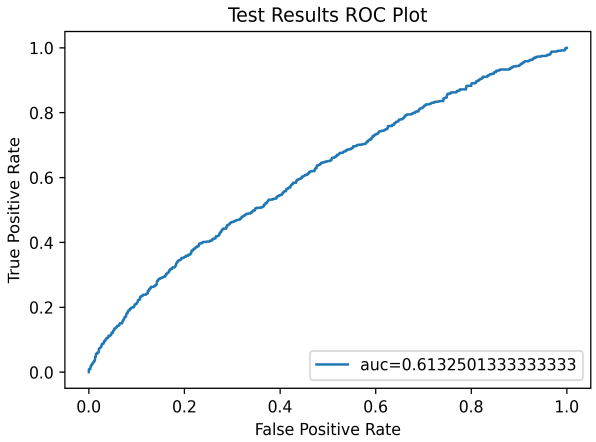

Figure 10. AUC ROC plot for LSTM_ATT model

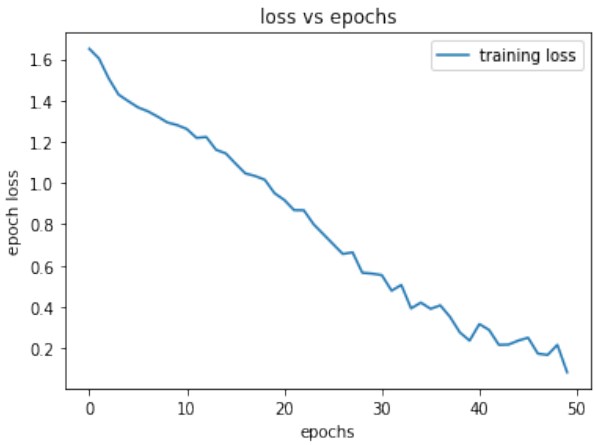

Figure 11. Loss for LSTM_ATT

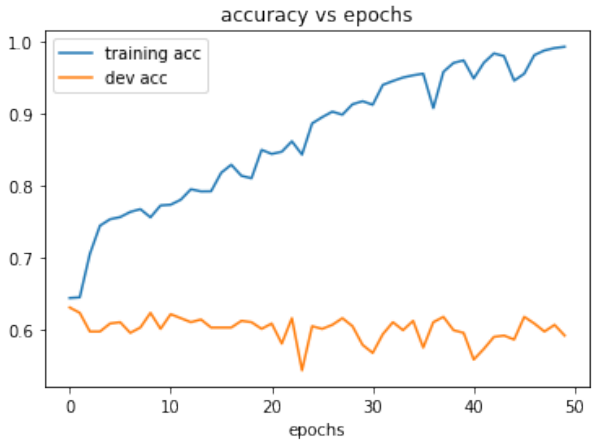

Figure 12. Accuracy for LSTM_ATT

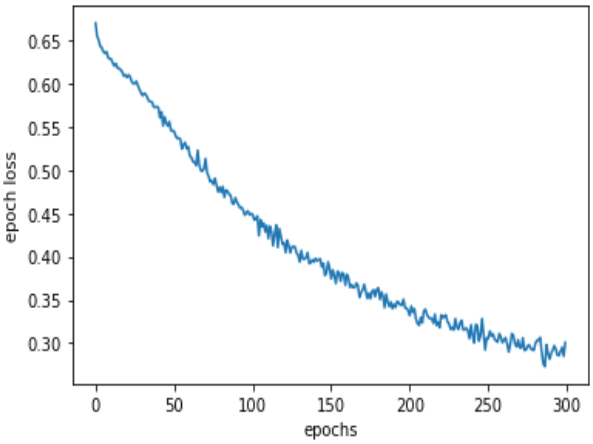

Figure 13. Loss for GoogleNet across epochs

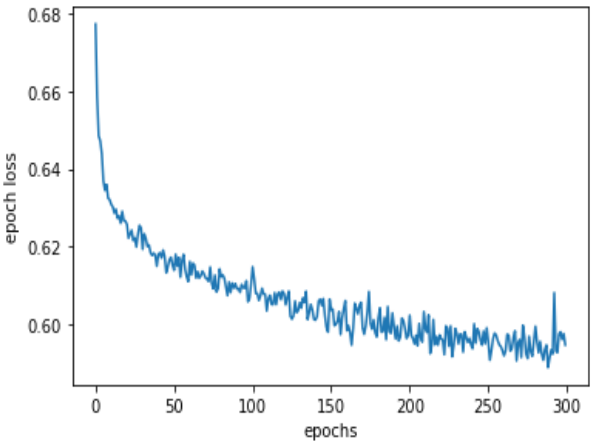

Figure 14. Loss for VGG across epochs

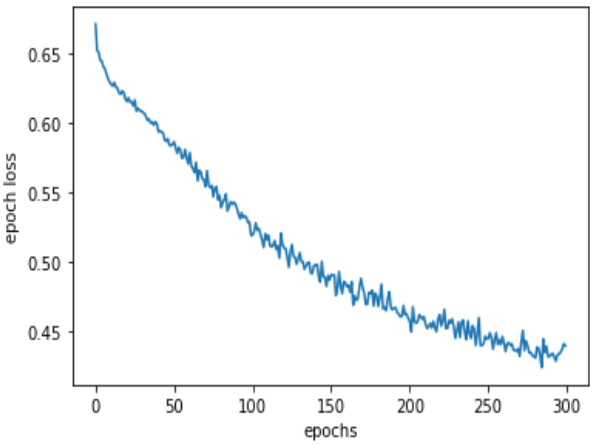

Figure 15. Loss for Resnet across epochs