Big Data Analytics: Hate Detection in Memes

Team 4: Amisha Jindal, Edward Carlson, Pascal Bakker

ABSTRACT

Identification of hate in memes poses a challenge because the text or image in isolation may not be offensive on their own, but the combination of the image and text may make it offensive. This makes it difficult for unimodal models, those that just take image or text, to properly classify which memes are offensive, and which are not. Our project idea is to be able to classify offensive and hateful memes from those that are not hateful in nature.

KEYWORDS

bert, alexnet, resnet, vgg, lstm

1. Introduction

In the past few years, multimodal situations have seen a rise owing to the fact that many real-world problems are multimodal in nature. Memes belong to a very interesting subset of multimodal problems. While it is relatively easy to crack down on an emotion automatically when it is purely text through methods such as word filters, it can be very hard to deduce the context when it is in the form of a text picture. When there's just text, it becomes a unimodal scenario and is relatively easy to identify. But when combined with a picture, the meaning of the text can change considerably. While humans can look at a meme and understand the combined meaning of it, it's not easy for machines to do the same since they look at the image and text independently [1].

Facebook AI has created the first dataset to aid researchers in understanding multimodal hate speech and build better systems. The Hateful Memes dataset proposes a challenge that focuses on fine-tuning existing pre-trained large scale multimodal models as opposed to training new models from scratch.

For the scope of the paper, hateful speech, as utilized by Facebook, will defined as:

"A direct or indirect attack on people based on characteristics, including ethnicity, race, nationality, immigration status, religion, caste, sex, gender identity, sexual orientation, and disability or disease. We define attack as violent or dehumanizing (comparing people to non-human things, e.g. animals) speech, statements of inferiority, and calls for exclusion or segregation. Mocking hate crime is also considered hate speech. "

An Exception to the rule is a meme attacking a famous person so long as it does not violate any of the characteristics defined above. Additionally, attacking other hateful groups, such as terrorist groups, is not considered hate. The complications of hate speech memes detecting such will require some world knowledge.

2. Related Work

A lot of prior work has been about detection of hate speech and natural language processing. Many Twitter based text-old hate speech datasets have been used for classification. Hate speech hasn't been as simple owing to varying terminologies that can be used to identify hate. To make it easier, our focus has been on hate speech in a narrowly defined context [2].

While hate speech identification has been prominent, there has been little work on multimodal hate speech with only some works having both images and text. Augmenting text with image embedding information has been said to boost performance in the detection. Some works collect on Instagram images and their associated comments and experiment with various features and classifiers to detect bullying.

The Hateful Memes dataset is larger and explicitly formed to increase difficulty for unimodal architectures; specific examples with high-confidence ratings are included and a balance is created to include a variety of multimodal fusion problems. Earlier methods propose interpreting multimodal speech through modalities consisting of text and socio-cultural information instead of images. A larger and noisier Twitter-based dataset of memes has also been created for the purpose.

Multimodal hate speech detection involves vision and language both. A prominent number of such tasks focus on (autoregressive) text generation or retrieval objectives. Even though such situations are important to the interest of the community, they are not in sync with the real-world problems that are encountered in industry by companies like Facebook or Twitter that focus on classification of multimodal posts, comments, etc. for a huge variety of classes. Such problems often require large scale multimodal classifiers akin to the one proposed by Facebook AI.

Another problem arises due to the lack of a standard dataset or benchmark task. Despite having multimodal classification tasks, issues arise due to differences in the qualities of datasets which vary substantially and unavailability of data to different organizations. This gap is filled by the Hateful Memes dataset in the space of multimodal classification datasets.

In the paper that we've used as our baseline, Facebook AI has worked on the evaluation of various models belonging to one of the three classes: unimodal models, multimodal models that were unimodally pre trained and multimodal models that were multimodally pretrained. Image encoders such as ResNet-152 and Faster-RCNN were used along with BERT for textual modality. Transfer learning from pre-trained text and image models on multimodal tasks also formed a category for choice of models. As the final step, these methods were compared to models trained on a multimodal objective as an intermediary step before fine tuning on the current task.

3. Dataset

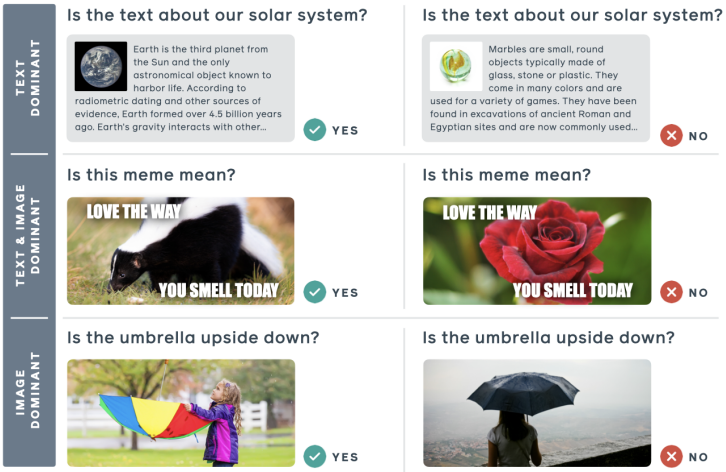
The dataset is about four gigabytes in size, containing about 10,000 memes. For the competition, the primary loss metric will be AUROC(otherwise known as AUC) as submission guidelines for the competition [2].

$$\int_{inf}^{-inf} TRP(T)FPR'(T)dT$$

The potential max AUC will be 1 and the minimum will be 0. Additionally, the model will try to maximize classification accuracy.

$$Accuracy = \frac{1}{N}\sum_{i=0}^{N} I(y_i - \widehat{y_i})$$

The dataset focuses on hate speech detected in multimodal memes. To be clear there are three kinds of image data in the scope of this project: text dominant, text and image dominant, and image dominant (illustrated in the figure below). The project will be handling text and image dominant memes.



4. Background

4.1. BERT

One model that we tried was the BERT model, or more specifically the BERT base uncased model. The BERT model works by using transformers, a system of encoders and decoders in order to avoid the problems that recurrent architectures have. The encoder for the transformer takes in all the words in the sentence or sequence at once, generating word embeddings and placement embeddings, and it implicitly learns attributes of the language such as sentiment or grammar through semi-supervised learning. One of the advantages that transformers have over recurrent architectures is that they do not suffer from vanishing gradients as rnns do, as they do not process the words sequentially, but all at once. This passing of all the words at once also can allow for some performance increases due to the parallelization of the task. These encodings are then passed into the transformer's decoder, generating the desired output from the encoded language data. One common use for transformers is using them to translate text from one language to another. By first putting an English sentence into the encoder, you are able to generate encodings for that sentence. If these encodings are then passed into a decoder that decodes to French, the encoder and decoder pair are capable of translating English text to French. BERT works by stacking these transformers, along with various dense, dropout, and normalizing

layers in order to produce desired output that can be used to classify textual data. While we chose a specific version of the BERT model, the BERT base uncased model, there are many different versions of BERT, such as those built around other languages, others that handle upper and lower case text. There are still other types of BERT that are the same except for that they have more layers and so can capture more complexity, at the cost of taking significantly longer to train due to the increase in weights that are needed to train.

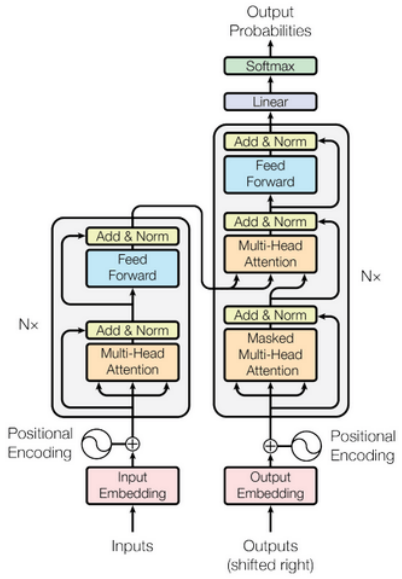


Figure 1: The architecture for a transformer [3]

4.2. **Alex**

AlexNet is a unimodal image processing architecture that we also tried to apply to the dataset in order to tell the difference between hateful and non hateful memes. The architecture used in AlexNet is a series of convolutional layers followed by a series of dense layers. Convolutional layers work by creating filters that pass over an image, making activation maps that represent certain features. While at the start these features are very simple, such as simple straight edges, as the layers of convolutions increase the filters start to represent more and more high level concepts such as geometrical shapes. What made AlexNet unique at the time is that it was much deeper than many other architectures, making it difficult to train without access to GPU's due to the high number of parameters that are required to tune. The reason we chose to try the AlexNet architecture instead of another visual architecture is because it had very good performance, winning many competitions when it first came out and it had a drastically better accuracy than most of the competition, and so we believed it would perform well for our task too.

Since we can change output size from AlexNet and BERT to any size we want to, we later made a new ensemble architecture based off of using AlexNet and BERT to make embeddings of their input data. Using these embeddings, we could pass them through subsequent layers in order to have a final output that is based on both the visual and textual data, making it so that it is able to learn from both parts of the meme, hopefully outperforming either architecture on their own.

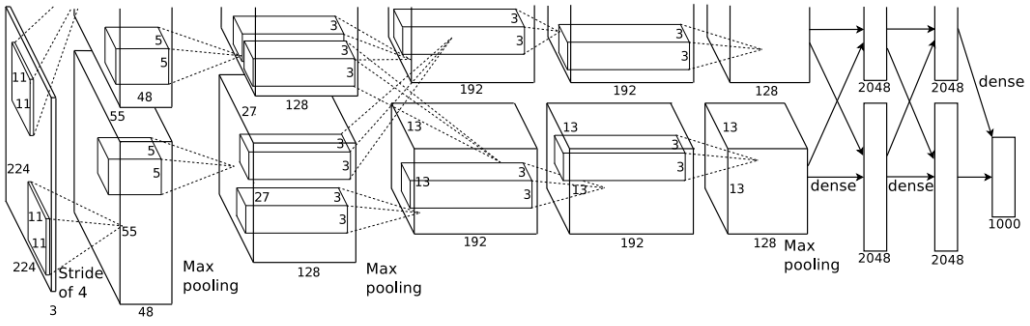


Figure 2: Architecture for AlexNet [4]

4.3. **ResNet**

ResNet-50, a pre trained deep learning model for image classification of the Convolutional Neural Network (CNN), was also used for the classification of images in our project [5].

ResNet or Residual Network is a traditional neural network that serves as a backbone for various computer vision tasks. Skip connection was first introduced by ResNet. The figure below explains the concept of skip connection. The left figure is stacking convolution layers together one after the other. The right figure does the same but also adds the original input to the output of the convolution block. Skip connections help in mitigating the problem of vanishing gradient and allow the model to learn an identify function which ensures that the layer at higher end will perform at least as good as the layer at the lower end, and not worse.

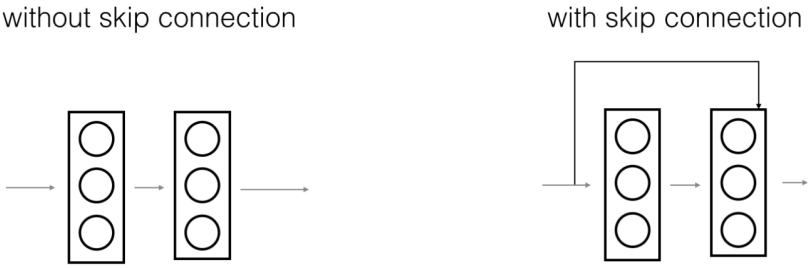


Figure 3: Skip Connection in ResNet Architecture

ResNet-50 is 50 layers deep and has been trained on a million images of 1000 categories from the ImageNet database. This model comprises 5 stages, each formed by a convolution and an identity block. Every convolution block and each identity block further has 3 convolution layers. With over 23 million trainable parameters available, we only used some basic parameters for our experiments. We used the pre-trained ResNet-50 in Keras [6].

This model was tried on the images first and then it was merged with VGG16 and LSTM for the multimodal classification.

4.4. VGG

VGG16 (Very Deep Convolutional Network for Large-Scale Image Recognition) is one of the most popular pre-trained models for image classification. It is a modified version of AlexNet created by replacing large kernel-sized filters with multiple 3 x 3 kernel-sized filters one after another [7]. The model is sequential and consists of a lot of filters. At each stage, the number of parameters are reduced using small 3 x 3 filters. Further, all hidden layers use the ReLU activation function. There are about 139 billion parameters for this model which makes it slower and harder to train than others [8].

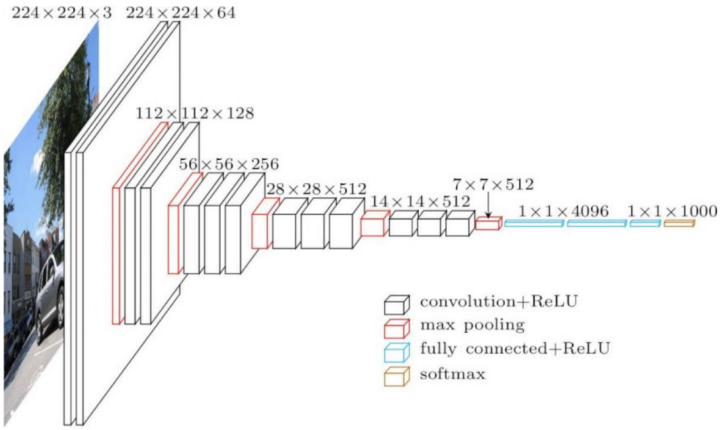


Figure 4: VGG16 Architecture [8]

4.5. LSTM

Long Short Term Memory networks, or LSTMs, form a significant category in RNNs and are capable of learning long-term dependencies. They are immensely popular nowadays owing to their exceptional performances on a huge variety of problems. They're specifically designed to avoid long-term dependency problems.

Like all RNNs, LSTMs also have a chain-like structure with a different repeating module. Instead of a single neural network layer, these networks have four layers which interact in a unique way as shown below.

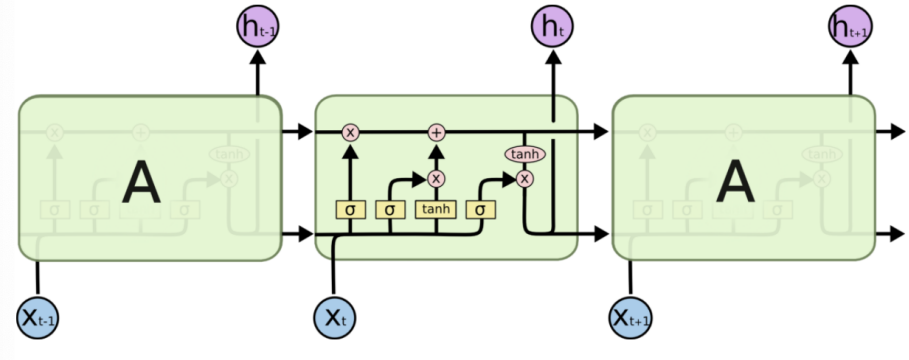


Figure 5: LSTM Architecture

The cell state, or the horizontal line running through the top of the diagram, is the key to LSTMs. This key runs the entire chain with a few minor linear interactions making the flow of information very easy. The LSTM regulates the information to the cell state through structures called gates. These gates are formed of a sigmoid neural net layer and a pointwise multiplication operation and are used to optionally let information through. The output from the sigmoid layer is between zero and one and details how much of each module should be passed. An LSTM has three gates to protect and control the cell state [9].

5. Methodology

We first trained unimodal models on the text and images individually and then combined different models to classify the meme as a whole. The models that we used are given below.

5.1. BERT

We were able to find the weights for this off the shelf, and so in order to speed up the training process we used these weights as a starting point to train our own version of BERT to detect hateful vs non-hateful memes. This resulted in a testing accuracy of about 55% after some amount of hyperparameter tuning, which was similar to what the paper was able to achieve with a unimodal text only BERT architecture. One of the downsides with the bert model is that it is unimodal, and so as is it can only process the text of the meme, and not the text in combination with the image. This causes it to run into some of the problems mentioned by the paper, such as the fact that a meme's context can change wildly when the image is changed, even if the text remains the same.

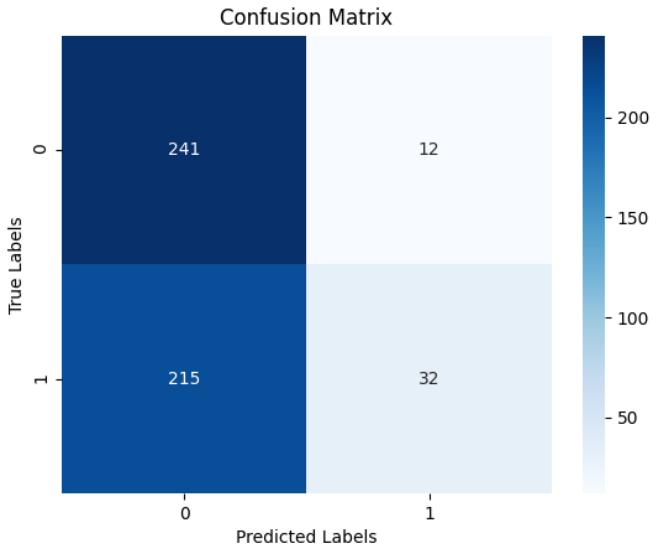


Figure 6: Predicted vs actual labels for BERT

5.2. **Alex**

Due to problems with the ace server initially, we were not able to use an off the shelf version of AlexNet with the weights pre made, and so we used the same architecture but trained the weights from scratch. Since the dataset had images of variable sizes and ratios, we first had to preprocess the images. In our preprocessing of the images, we first would scale down any images that were too big, in order to make sure that the images we passed into the Alexnet architecture had enough information in them to be understood and classified. We would then pad the dimensions of any image so that the minimum dimension was 227, which is the size of our Alexnet input size. We then would center crop the images in order to make sure that the image's height and width were a multiple of 227 so we could divide it evenly. If the image was not 227 x 227 at this point, we would then crop the image into multiple smaller 227 x 227 images that we would then pass through the Alexnet model. The reason we chose the 227 x 227 dimension size is that it is what the original Alexnet architecture had, and so we decided to stick with this architecture. This was able to achieve a slightly higher accuracy of 63%, beating the BERT model by approximately 4%. We also combined the BERT model and AlexNet model in order to create a multimodal modal made from the two of them. By first making it so that the AlexNet architecture and the BERT architecture output an encoding of shape 1x10, we were then able to concatenate these embeddings together, and then sending them through a final dense layer in order to arrive at our classification output. This combination of the AlexNet architecture and the BERT architecture was now multimodal in nature, allowing it to better understand and classify the memes, both the image and text together. This allowed it to achieve a higher accuracy of 77%, beating out both the AlexNet architecture and the BERT architecture.

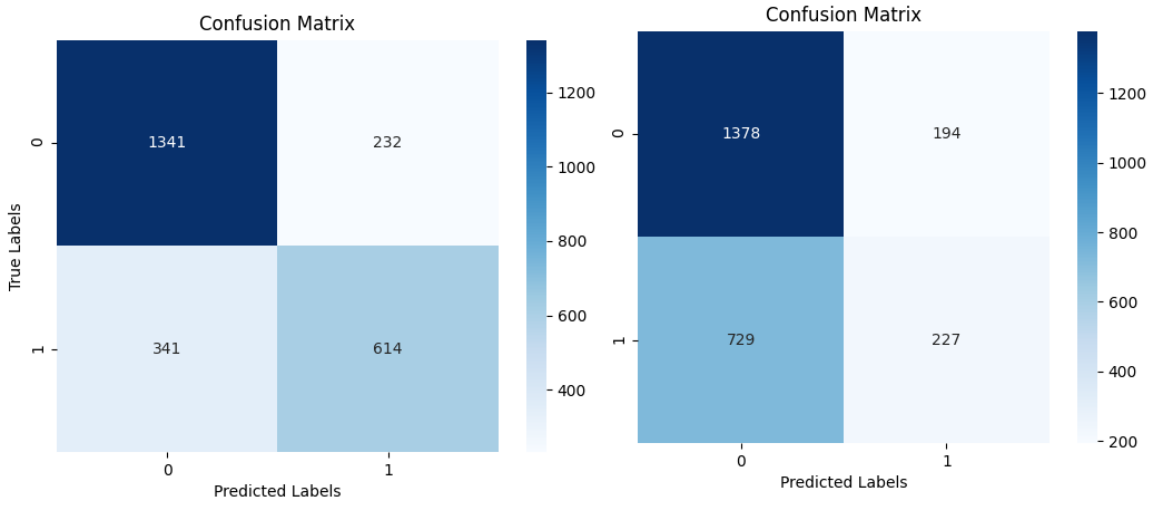


Figure 7: Confusion matrix for Alex (Left) Confusion matrix for Alex and BERT ensemble (Right)

5.3. **ResNet-50**

We started the process by reading the images and then resizing them to 224 x 224 before storing them in an array. The size was changed for consistency and can later be modified depending on

the problem. The preprocessing involved scaling the images between -1 and 1 and one-hot encoding the target variable. Finally, a ResNet model was defined with weights = None so that the model could be initialized with random weights. We didn't proceed with the ImageNet pretrained weights for our scenario. The include_top parameter was set to False to uninclude the final pooling and fully connected layer in the original model. Instead, we added a global average pooling and a dense output layer to our model. A validation accuracy of 52.9% was achieved after training for 30 epochs on a batch size of 64 and this is close to what was achieved in the paper. No hyperparameter tuning was done so the results may change as we try to find best hyperparameters.
Since this model only analyses images and not the text over the images, we combined it with VGG16 and LSTM for our purpose.

5.4. **LSTM**

To analyse text alone, we trained a LSTM model on it. The text was first cleaned and standardized. For the model, we started with the text vectorization layer, then an embedding layer with embedding_dim = 16, bidirectional LSTM layers, batch normalization layers, global max pooling layer, a dense layer, and lastly, a dropout layer. After training for 30 epochs on a batch size of 32, an accuracy of 89% was achieved.
Since this just works on the text, it was combined with other models for the classification as a whole.

5.5. **ResNet-50, VGG16, LSTM**

In order to create a combined model, we created baseline models for images and text and then put them together.
First, the images were read and normalized. The image layers were composed of data augmentation layer, preprocessing layer where images were rescaled, the concatenated layer which had base layer 1 as ResNet-50 and base layer 2 as VGG16 (both pretrained from Keras). After this the global average pooling layer was added. Lastly, a dropout layer was added to the model.
For the text, we first cleaned and standardized the text. This was followed by a text vectorization layer. After this, we added an embedding layer with embedding_dim = 16, Keras bidirectional LSTM layers, batch normalization layers, global max pooling layer, a dense layer, and a dropout layer at the end.

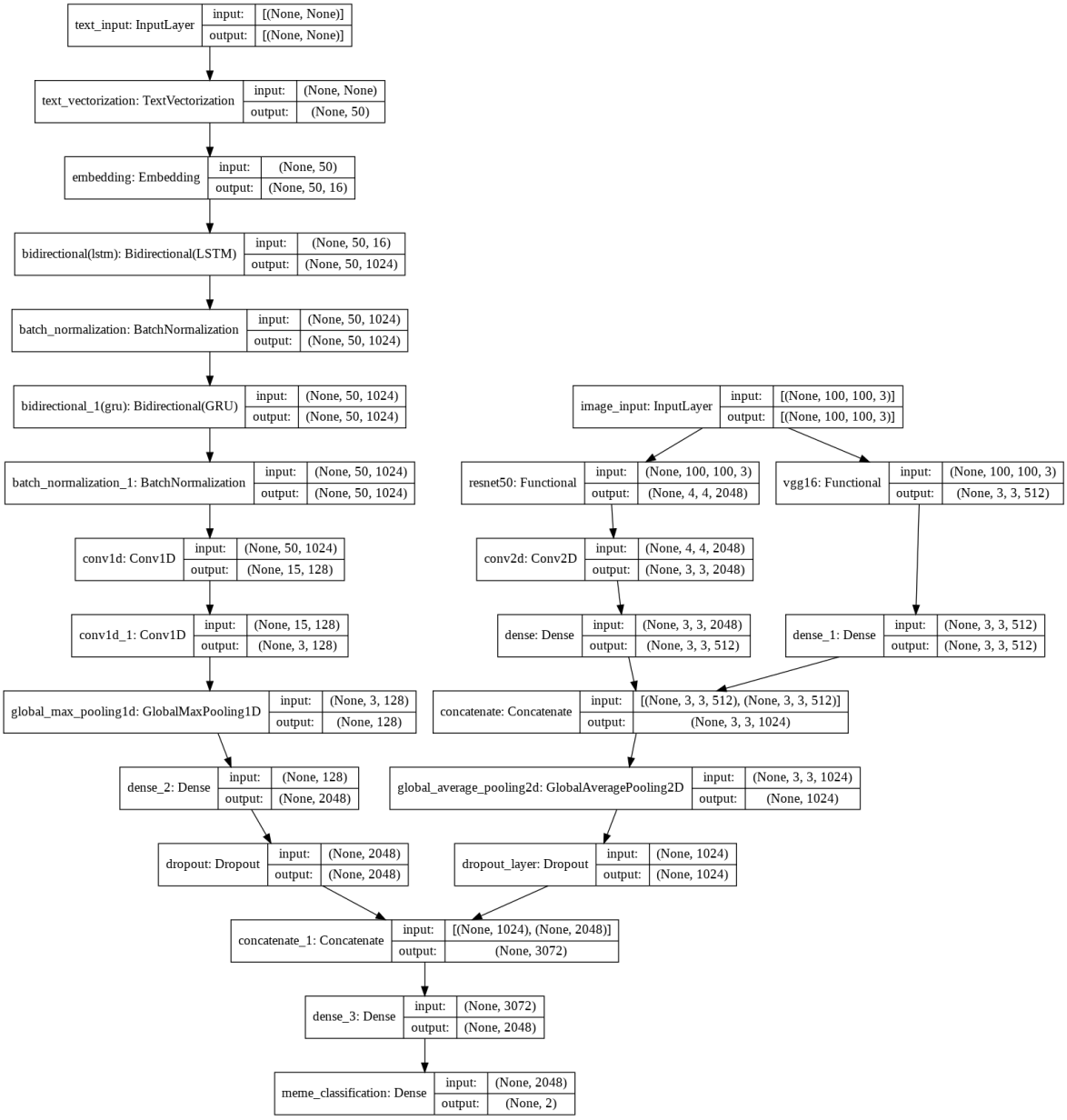


Figure 8: Model Architecture

Then, a model was created with the concatenated text and image layers followed by a softmax layer and a sigmoid layer at the end for prediction where 0 represented non-hateful memes and 1 represented hateful memes. The decaying learning rate was used for the model. After training for 25 epochs on a batch size of 256, an accuracy of 64.47% was achieved on the test data with a 0.355 micro-F1 score and 0.262 macro-F1 score. This is similar to what was achieved in the paper originally.

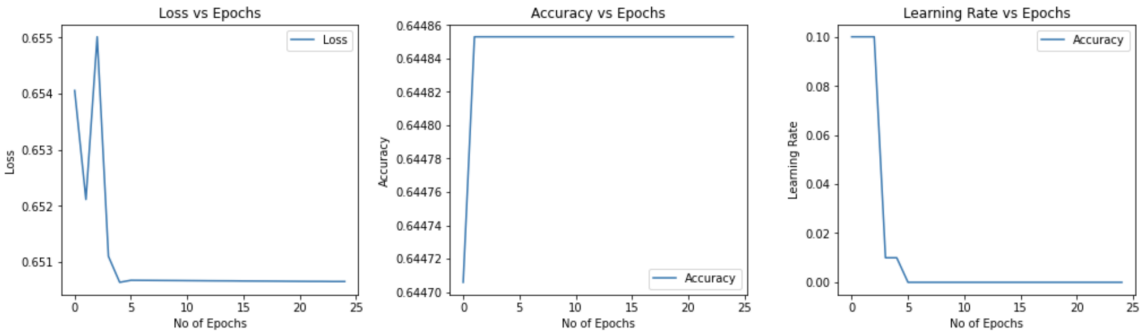


Figure 9: Results from the classification

6. Discussion and Future Scope

We trained a combination of BERT + AlexNet and ResNet + VGG + LSTM for our classification and achieved significant results as mentioned above. In our experiments, we quickly realized that it was very important to have visual and textual data in order to see if a meme was offensive or not, as a change in one of these parts could wildly change the context.

While we were not able to try it out due to a lack of necessary data, one way we believe we could see if a meme is offensive is to see how related the original poster of the meme was to another user we know is offensive already. Since a user that is not offensive is not likely to follow or like the same posts that an offensive user does, this could be an important datapoint our models could use to determine if a post is offensive.

7. Code Documentation

All the trained models and codes can be found on Google drive: Code

REFERENCES

[1] Hateful Memes Challenge and Dataset: https://ai.facebook.com/tools/hatefulmemes/

[2] Douwe Kiela, Hamed Firooz, Aravind Mohan, Vedanuj Goswami, Amanpreet Singh, Pratik Ringshia, Davide Testuggine: The Hateful Memes Challenge: Detecting Hate Speech in Multimodal Memes. CoRR abs/2005.04790 (2020)

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 6000–6010.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet classification with deep convolutional neural networks. In Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1 (NIPS'12). Curran Associates Inc., Red Hook, NY, USA, 1097–1105.

[5] Simple Image Classification with ResNet-50: https://medium.com/@nina95dan/simple-image-classification-with-resnet-50-334366e7311a

[6] Understanding and Coding a ResNet in Keras: https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33

[7] VGG16 - Convolutional Network for Classification and Detection:
https://neurohive.io/en/popular-networks/vgg16/
[8] Top 4 Pre-Trained Models for Image Classification with Python Code:
https://www.analyticsvidhya.com/blog/2020/08/top-4-pre-trained-models-for-image-classification-with-python-code/
[9] Understanding LSTM Networks: https://colah.github.io/posts/2015-08-Understanding-LSTMs/