EMOTION DETECTION FOR SOCIAL MEDIA DATA USING DEEP LEARNING

Samyuktha Wuyyuru[1]
swuyyur1@umbc.edu
CB06298

Geeta Sanjetha Pericherla[2]
geetasp1@umbc.edu
TF00971

University of Maryland, Baltimore County

1. ABSTRACT

The objective of our project is to create a robust model for Multi label text classification using deep learning algorithms applied to the GoEmotions dataset. The dataset is a large-scale emotional analysis dataset that contains more than 58,000 Reddit comments labeled with 27 different emotions. The goal is to develop a model that can accurately identify and categorize a given text into one or more emotions present in the dataset. To achieve this objective, we evaluated the data using deep learning algorithms, including ANNs (Artificial neural networks) and BERT (Bidirectional Encoder Representations from Transformers). These models are trained and tested. The performance of both models is evaluated on various metrics such as accuracy, precision, recall, and F1 score. The results of this project will provide insights into the effectiveness of ANN and BERT models for Multi label text classification.

Keywords: *BERT, GoEmotions, Accuracy, Embeddings, Text Preprocessing*

2. INTRODUCTION

Our daily lives are significantly influenced by our emotions, and social media platforms have emerged as a popular medium for people to communicate these emotions. Businesses can use the emotions shared on social media platforms to enhance their goods and services, create more specialized marketing campaigns, and better comprehend the demands of their target audiences. Yet, as it requires evaluating and interpreting emotions from social media data is a difficult undertaking. This is a challenging problem in natural language processing (NLP), as emotions are subjective and context dependent. However, with the increasing amount of text data available on social media platforms, and other online sources, there is a growing need to automatically classify emotions from text to extract valuable insights and improve decision-making processes.

Deep learning techniques like ANN (Artificial neural networks) and BERT (Bidirectional Encoder Representations from Transformers) are used in solving various NLP tasks like Multi label text classification [5]. ANN (Artificial neural networks) simulates the human brain's neural network to recognize patterns in data. It has been applied in many NLP tasks, like text classification and sentiment analysis [4]. BERT (Bidirectional Encoder Representations from Transformers), on the other hand, is a pre-trained deep learning model that uses a transformer architecture to process natural language. It has achieved state-of-the-art results in tasks, like sentiment analysis, natural language inference, and question-answering [1]. The effectiveness of these models in sentiment analysis and emotion recognition tasks highlights the potential of deep learning to enable machines to understand human language and provide valuable insights for various applications [2][3].

3. DATASET AND PREPARATION:

GoEmotions is a large dataset consisting of 58k Reddit comments in a duration between 2005-2019 from a variety of popular posts. This dataset tries to cover all the possible emotions within the positive, negative, and neutral categories creating a new taxonomy for classifying emotions in NLP. Offensive comments are removed from the data but some of the vulgar comments are still preserved to make sure there is no imbalance of the negative emotions. This dataset provides a vast coverage of all sorts of emotions from Reddit and in general which is what makes this dataset most suitable for Multi label text classification for identifying emotions.

3.1 DATA ANALYSIS

It is important to analyze the data accurately to determine what kind of techniques and pipeline should be used in the text preprocessing stage or check if the data is imbalanced. Since the data is collected from social media, no one would bother writing in a proper and understandable language. On observing the data, we can see that the text consists of a lot of slang language, vulgar terms, various languages' words, numerical values, and lots of random characters. Most of the vulgar and offensive text is removed by the authors of the dataset but some of it is preserved to maintain a balance of negative emotions in the dataset.

Preparing textual data cannot be completed in a single step as there are a multitude of possibilities for cleaning text. We can get started with removing all the unnecessary special characters since they are not useful in identifying emotions in any context. Users tend to write shortened or slang versions of words but the model will not be able to understand it, so we need to modify the data present in this form. There could be words which are not spelled correctly which can be corrected and remove unnecessary spaces, turn all the text into lowercase, and remove any unnecessary special characters.

A word graph in Figure 1 is a representation of the overall data which displays the most repetitive words. This gives an overall picture of most words present in the dataset and the intensity of their repetition.

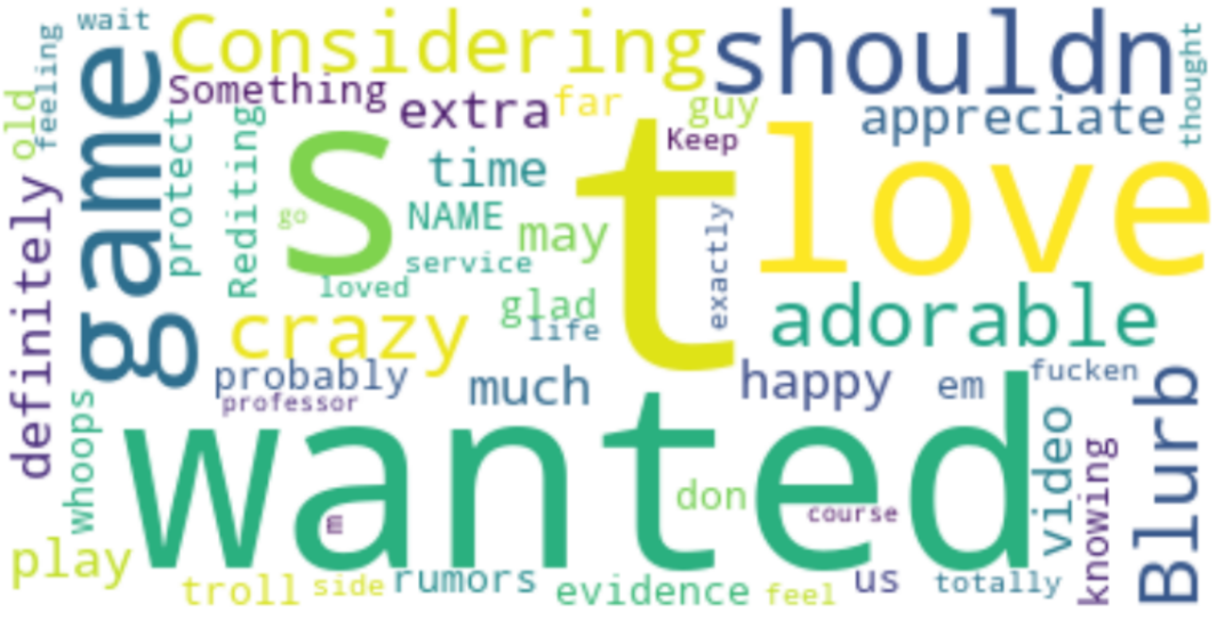


Figure 1: Word graph displaying the most repeated words in the dataset.

We implemented a bar plot to observe the distribution of each emotion in the data and check if there is any significant imbalance present between them. As we can see from Fig.2 , 'joy' and 'neutral' are represented better in the data whereas 'disgust' and 'fear' show the least representations. This leads to a possibility where the model will be able to efficiently categorize the highest repeating labels while the less represented ones tend to be misclassified more often.

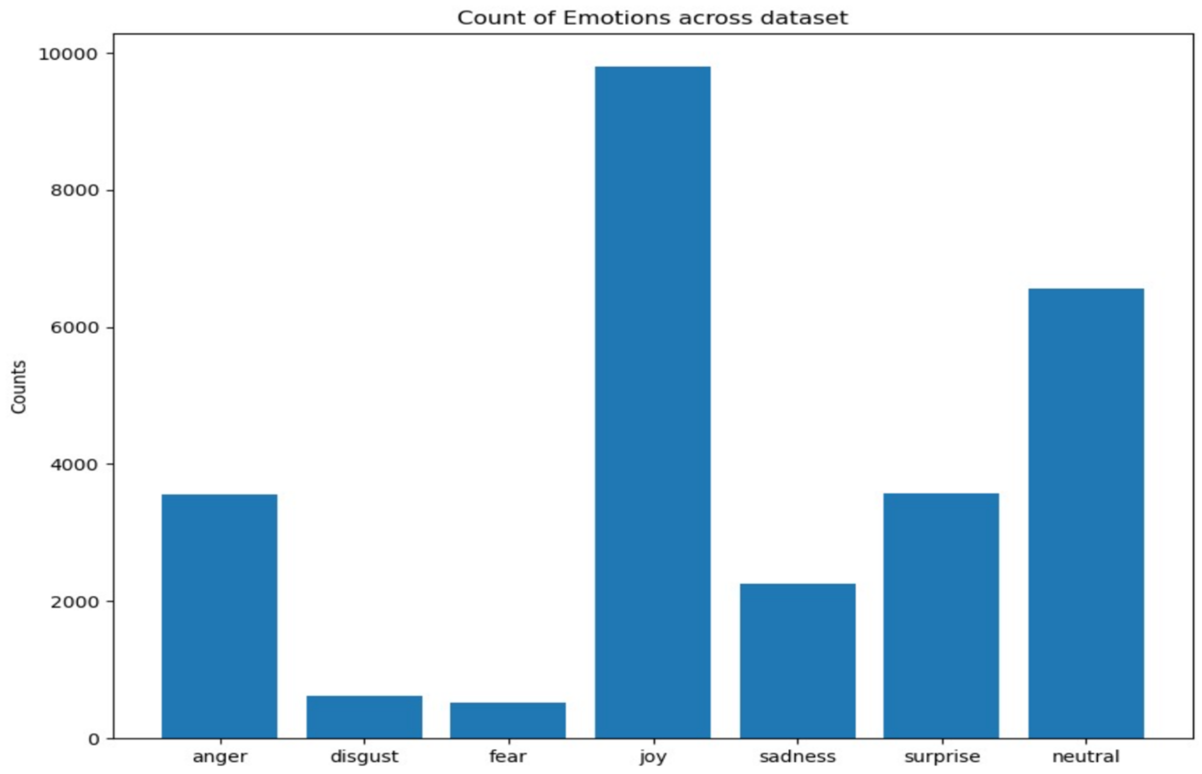


Figure 2: Distribution of emotion categories in the data.

3.2 DATA PREPARATION

Data preparation is performed to transform erroneous data and remove any data which is not necessary from the dataset. This is an essential process required for the machine learning model to process the given data in an efficient manner. In this paper, we will be using the Ekman taxonomy for classification of emotions based on Reddit comments present in the data which are in textual format. The original dataset has around 28 categories of emotions and one neutral category which is an advanced taxonomy. So, the categorization from the original dataset is compressed to 7 categories of emotions which will fit the Ekman categorization.

3.3.1 TEXT PREPROCESSING PIPELINE

Text preprocessing is a critical step to make the data clean and perceivable before building the model. Since this process contains multiple steps and techniques, we create a custom-built pipeline which includes all these steps in the necessary order and the data goes through each of them and gets modified accordingly.

For each row in the text column, the pipeline will go through the following steps to improve the data quality:

1. Text cleaning: This is a custom function which includes inbuilt libraries like emoji, BeautifulSoup, python functions and code to turn all text into lower case, demojize, remove URLS, numerical values in words and any unnecessary symbols from the text except alphabets and some basic punctuations. In the above function, converting all text to lowercase ensures that the model does not treat the same word in different sentences as different words. URLs are removed as they do not carry any meaningful information and might add noise to the data. Removing punctuation and special characters to ensure that the model does not treat them as separate words and to make the text more consistent.
2. Lemmatization: The data we have is collected from a social media website and most of the comments have slang and incomplete sentences and words. To rectify such data, we build a dictionary which contains mappings of the incomprehensible words observed from the data to words which correctly

represent them in an understandable language. When the text goes through this function, it removes words which are defined in the dictionary as incomprehensible and replaces them with the mapped words such that the model will be able to clearly understand the data. This makes the text more consistent.

3. Spellings: There is a high probability that text present in this data can be incorrectly spelled. So it is best to maintain a dictionary for such data through observation of the data. So, once the text goes through this function, it will replace the wrongly spelled words with the correct words.
4. Removing unnecessary spaces, punctuations, and miscellaneous data: Most of the data is cleaned by the time we get to this part, so the remaining data will have some punctuations which are unnecessary since they don't contribute to any emotional comprehension and extra spaces can be removed. Social media is a place of many languages, and many people feel comfortable communicating in their own language. Such parts of the data can be ignored since we are processing the data in only one main language.

Once the data goes through this pipeline, the quality of the text is improved significantly, and it will have a direct impact in improving the performance of the model for classifying emotions. Even if the models have inbuilt text preprocessing modules, it is not enough to deal with data that comes from social media. To build algorithms which are highly efficient and give the best performance would not mean anything if the data were fed into the models is inconsistent and incomprehensible. Hence, it is always a best practice to prepare the data by creating a tailored pipeline.

3.3.2 EMBEDDINGS:

The next step is to generate embeddings, which convert the pre-processed text into numerical representations expressed in the form of a vector to be processed by machine learning algorithms. This vector representation makes it easy for the model to understand the importance of a sentence semantically and classify the sentence into categories accordingly.

Embeddings can be generated using different techniques among which Word2Vec and BERT are most suitable for the dataset we have. The disadvantage of Word2Vec is that it might not be able to generate different embeddings for the same words which have different contexts. Instead, we can use BERT to generate embeddings as it is pretrained on a huge corpus and transfer learning can be further used to generate embeddings for the dataset we have. The advantage of BERT is that it produces embeddings based on the words/sentences around it so it introduces semantic meaning into the vector. This is highly useful for scenarios like text classification as these embeddings can identify similar sentences based on their vectors.

4. MODELING:

To perform multi-label text classification on the chosen dataset, we consider 1,00,000 rows of data. The data is split into train, validation, and test data in a ratio of 81, 9 and 10. We propose two models to process the data which are a basic neural network and BERT model. Once the data goes through the text preprocessing pipelines and the embeddings are generated, the data is fed as input to the models. We build a basic version for each of the models to train the data and evaluate how it performs on the validation data. To improve the results for these models, we experiment with the hyper parameters of the model inorder to build an optimized version of the basic model. Once the hyper parameter optimization is done and the best parameters are chosen, we create a new model with fine-tuned parameters to build the optimized versions of each model. After the models are constructed, we feed the training data to these models to get improved results.

4.1 BASELINE MODEL: ARTIFICIAL NEURAL NETWORK ALGORITHM

To evaluate the performance of Artificial Neural Network on the dataset, we build a baseline model to classify the text into the seven categories. Initially, we built a container-like object using the "Sequential" function from the TensorFlow package which is useful in adding more layers to build a Neural network model. While adding each layer, we define the number of neurons it can have, the parameters required to define it and the dropout to remove unnecessary noise produced by each layer.

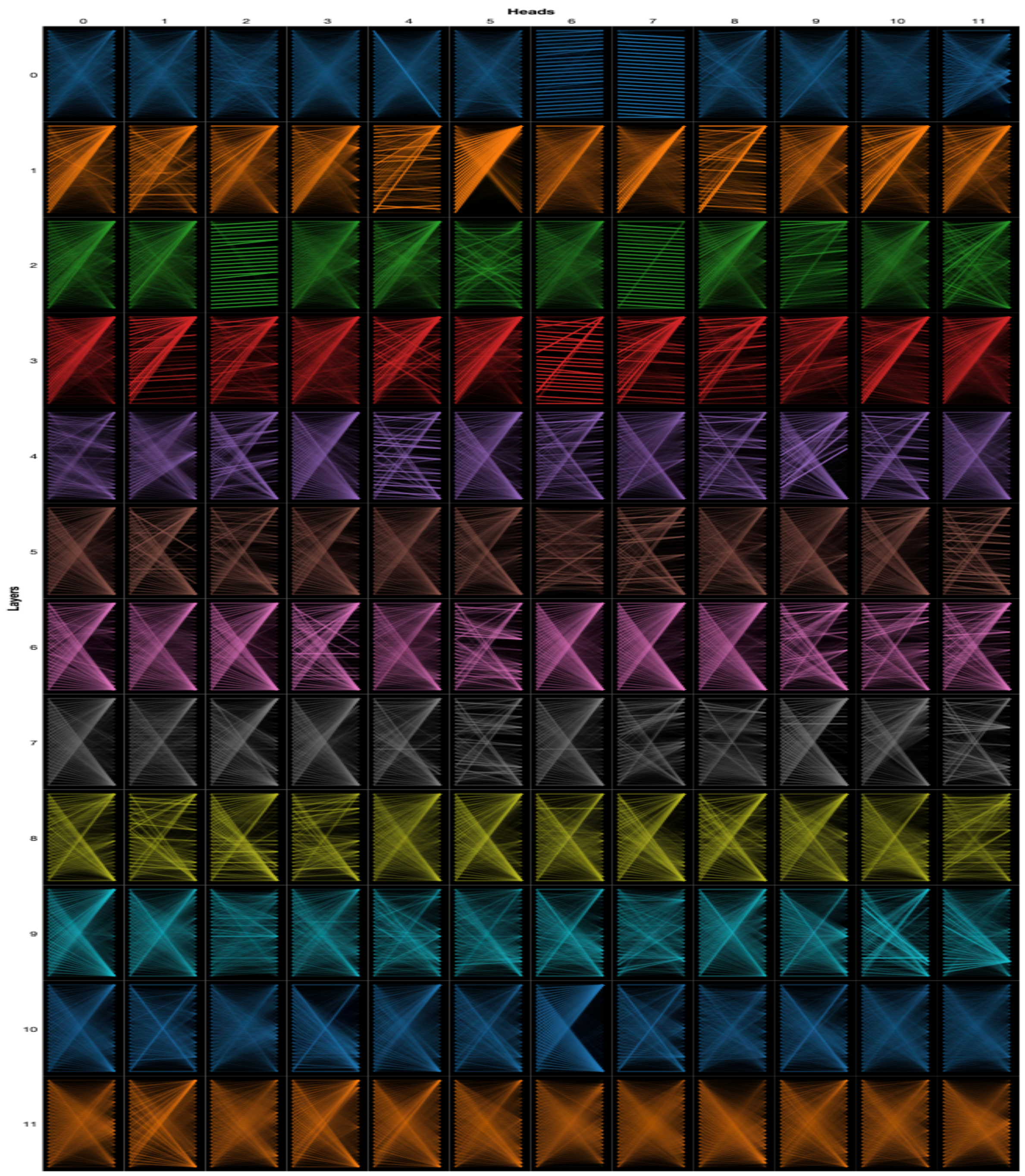


Figure 3: This is a representation of the embeddings generated by the BERT model in each layer.

The ANN model will be defined with an input layer, multiple hidden layers, and the output layer. The first layer, which is the input layer, will consist of 512 neurons, a kernel initializer ‘he_uniform’, and an activation function RELU. The second hidden layer consists of 256 neurons followed by another hidden layer of 64 neurons with a RELU activation function for each layer. The final output layer has 7 neurons identifying each of the 7 emotions with Sigmoid activation function. Here, the sigmoid activation function is used because it provides an individual probability of classifying the record for every category. The batch size is set to 250 with 50 epochs with a validation split of 20%. The loss function specified for model compilation is binary cross entropy, optimizer is the Adam optimizer and accuracy being the evaluation metric. The basic model gives us a training accuracy of 48.6 % and a testing accuracy of 46.8%. We perform hyper parameter optimization to fine tune its performance which can be referred to in section 4.2.

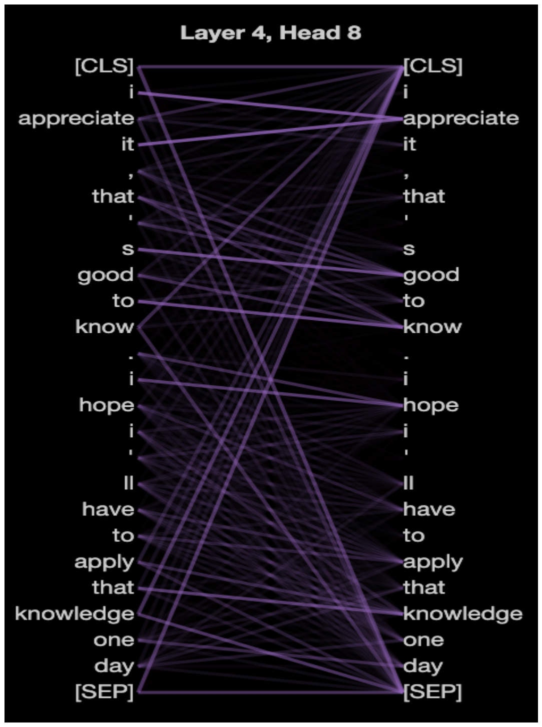


Figure 4: The BERT model semantically maps each input in all the layers.

Once the experimental parameters are considered, a custom learning rate function is passed on to the model. The evaluation metric is the categorical cross entropy, epochs are set to 200 and early stopping criteria is provided. Now, we experiment with a batch size range of 8- 2024. All the parameters which are supposed to be tuned are made into an objective function and are optimized using Bayesian Hyper parameter optimization which finds the optimal parameters for the model by finding the local minima in each iteration by using optuna package.

4.2 HYPER PARAMETER OPTIMIZATION:

We try to optimize the loss function by working with hyper parameters of the model on a trial-and-error basis. For the ANN/BERT model, we experimented with the following parameters:

- Number of hidden layers for the model within a range of 1-10.
- The parameters of each individual layer which includes the following:
    1. Number of neurons for each layer within a range of 32-768
    2. Activation functions like: RELU, swish and Linear
    3. Kernel initializer: 'LecunUniform','he_uniform','glorot_uniform'
    4. Dropout layer: 0 -70 %

4.3 BIDIRECTIONAL ENCODER REPRESENTATIONS OF TRANSFORMERS

For the BERT model, the initial layer accepts a column of strings as the input. The next layer is a preprocessing layer in which we use the BERT preprocessor functionality by downloading the pretrained bert_en_uncased_preprocess. The preprocess model takes the input text from the first layer and preprocesses it. The second hidden layer is the encoder layer where the preprocessed text from the first hidden layer is the input, and all this text is converted into a 768-dimensional embedding. The next layer is a dropout layer with a 20% dropout ratio. The final layer is the output layer with 7 neurons representing 7 classes of emotions with a sigmoid activation function. This is a blueprint of the basic BERT model. Number of epochs is 20 with a batch size of 250 and early stopping criteria is specified. The loss function defined for this model is binary cross entropy, optimizer is the Adam optimizer and accuracy is the metric used for evaluating the model. With this basic model, we achieved a training accuracy of 42.5% and a validation accuracy of 43.6%.

Since there is scope for improvement, a better version of the model should be created by performing Hyper parameter optimization. To fine-tune the model, a custom learning rate function is passed to the model along with early stopping criteria with accuracy, precision, and recall as evaluation metrics. Optimizer is Adam and loss is binary entropy. Number of epochs are 10 and batch size is experimented between the range of 8-2048. A custom objective function is used to finetune the model with Bayesian optimization. The optimization is done for the number of hidden neurons within a range of 32 - 768 in each layer. Now, the best model parameters are chosen by identifying the least loss in the optimization process. A new BERT model is constructed using the auto construct custom function with these parameters as reference and the training data is fit into the model.

5. RESULTS:

In the process of creating a better version of the Artificial Neural Network model, for each combination of the parameters considered the hyper parameter optimization is repeated 50 times and the best model with least loss of 0.32 is selected. The hyper parameter optimization plot is generated to explain the objective value and the best value for every trial. Once the optimized version of the ANN model is constructed using the fine-tuned hyper parameters, it's fitted for the training data once again.

Similarly for BERT, we construct a new model which is the optimized version based on the hyper parameters selected during the hyper optimization process. Once this is created, we can train this new model again on the training data and test the model to see how it performs which is illustrated in Figure 6.

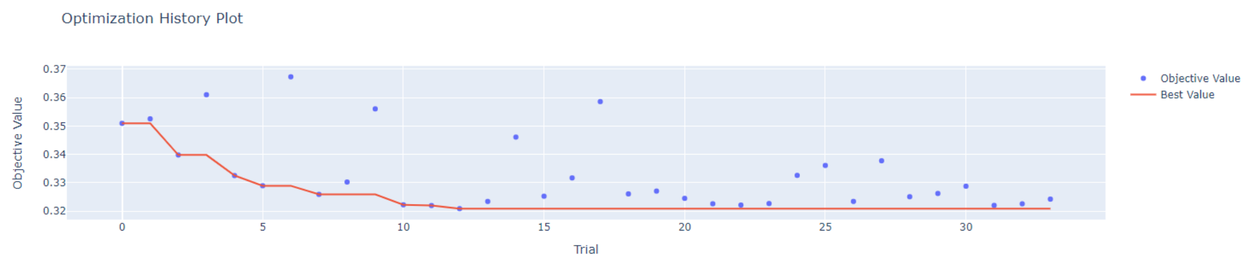


Figure 5: Optimization Plot for ANN model

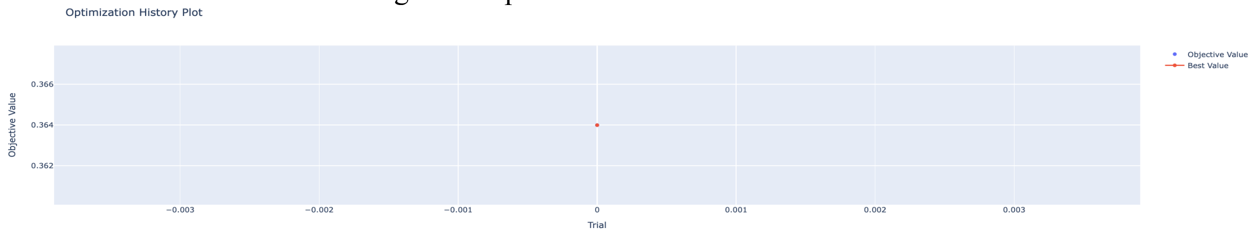


Figure 6: Optimization Plot for BERT model

From the below table, the fine-tuned ANN model gives a training accuracy of 0.4494, training loss of 0.3285, Validation loss of 0.3304, and a Validation accuracy of 0.4435. The fine-tuned ANN model gives the least loss in comparison to the other models. The fine-tuned BERT model has the best accuracy in training, validation and testing data compared to the remaining models.

	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy	Testing Loss	Testing Accuracy
BASIC ANN	0.3285	0.4494	0.3304	0.4435	-	-
FINE-TUNED ANN	0.3185	0.4736	0.32125	0.4691	0.31887	0.47029
BASIC BERT	0.34417	0.42321	0.3368	0.4333	-	-
FINE-TUNED BERT	0.36527	0.84642	0.35853	0.8498	0.3692	0.8439

Table 1: Results of the trained ANN/BERT models

We choose the better version of the ANN and BERT models because they are fine-tuned using Hyper parameter Optimization to conclude our analysis. With respect to accuracy, fine-tuned BERT model gives the optimal performance compared to any other model. From figure 7, accuracy and precision are the most significant parameters which give us the most important inferences from the fine-tuned ANN model. Precision from fig.7 gives a good representation of how well the emotions can be classified. We can see that the precision is extremely high for some of the emotions because of their high distribution in the data and extremely low for emotions like disgust and fear, which means the data regarding these emotions is insufficient for the model to train. For analysis, the exact matching ratio is generated for the fine-tuned ANN model and gives a training accuracy of 0.36169 and validation accuracy of 0.36358.

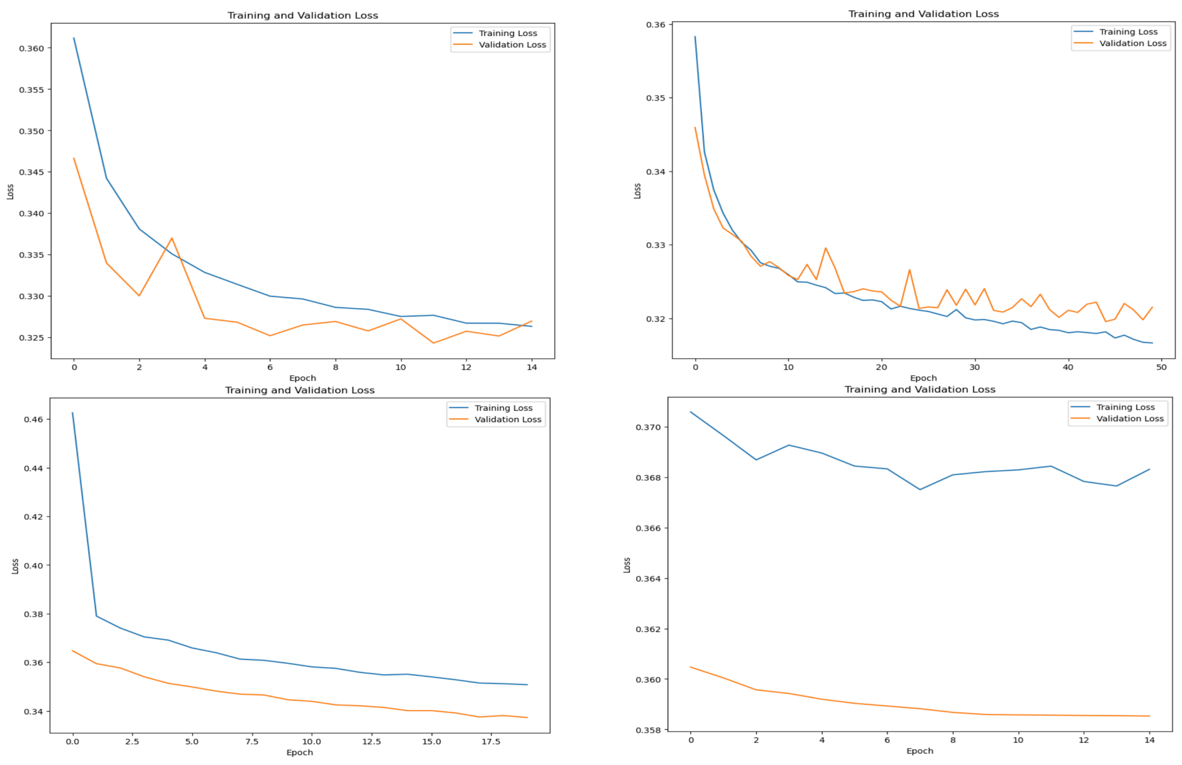
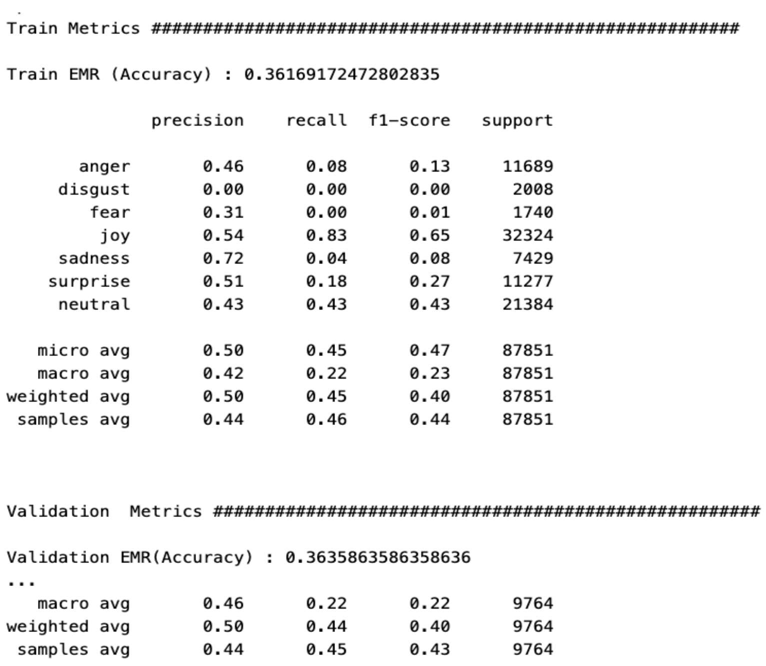


Figure 7: (a), (b), (c) & (d)Validation Loss curves for basic and fine-tuned version of Artificial Neural Networks & BERT models



```
Train Metrics ##########################################################

Train EMR (Accuracy) : 0.36169172472802835

              precision    recall  f1-score   support

       anger       0.46      0.08      0.13     11689
     disgust       0.00      0.00      0.00      2008
        fear       0.31      0.00      0.01      1740
         joy       0.54      0.83      0.65     32324
     sadness       0.72      0.04      0.08      7429
    surprise       0.51      0.18      0.27     11277
     neutral       0.43      0.43      0.43     21384

   micro avg       0.50      0.45      0.47     87851
   macro avg       0.42      0.22      0.23     87851
weighted avg       0.50      0.45      0.40     87851
 samples avg       0.44      0.46      0.44     87851


Validation  Metrics ######################################################

Validation EMR(Accuracy) : 0.3635863586358636
...
   macro avg       0.46      0.22      0.22      9764
weighted avg       0.50      0.44      0.40      9764
 samples avg       0.44      0.45      0.43      9764
```

Figure 6: Exact Matching Ratio for the best performing model

6. CONCLUSION & FUTURE SCOPE:

From the results, we can conclude that the fine-tuned BERT model performs better than the fine-tuned ANN model. We prefer a model which gives the least difference between the train, validation and test accuracies, as the least difference between the three sets indicates the model produced consistent results. Comparing the two models. The ANN model we have constructed is a simple neural network compared to BERT which is one of the most advanced models in NLP. In machine learning, as per the occum's razor if a simple model gives better performance compared to a complex model, we choose the simple model. The reasoning behind an advanced model like BERT giving such poor performance could be the less computational power and limited data we are considering. For improving the performance of the BERT, we need to train the model with a large set of data which also requires a significant amount of computational power. This project can be implemented in many areas which involves predicting multiple classes for a given sentence, including social media monitoring, and analysis of consumer feedback. Businesses may better understand their consumers' emotions and take the required steps to improve their goods and services by using accurate Multi label text classification models.

7. REFERENCES:

1. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.
2. Guo, W., Zhang, J., Liu, Y., & Liu, Y. (2019). An effective approach for emotion classification in Chinese microblogs with BERT. Neurocomputing, 357, 94-102.
3. Kim, Y. (2014). Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882.
4. Kouloumpis, E., Wilson, T., & Moore, J. D. (2011). Twitter sentiment analysis: The good the bad and the OMG!. ICWSM, 11(538-541), 164-167.
5. Li, X., Wang, X., Li, Y., & Huang, J. (2019). Multi-label text classification with deep learning models. Information Sciences, 486, 198-211.
6. Link: https://www.featureform.com/post/the-definitive-guide-to-embeddings

7. Link: https://catalog.workshops.aws/semantic-search/en-US/module-1-understand-internal/semantic-search-technology/2embedding#:~:text=BERT%20offers%20an%20advantage%20over,by%20the%20words%20around%20them