Emotional Tweets: a Convolutional-BiLSTM Approach to Emotion Classification

Abstract

We present a neural-network model based on a convolutional and bidirectional long short-term memory architecture that performs the task of emotion classification. We work with a dataset of over 200 000 English-language tweets collected from the Twitter API dated from January 2020 to September 2021 and pertaining to the novel COVID-19 pandemic. With short-form text data vectorization using pre-trained 50-dimensional GloVe word embeddings, our model performs to a degree of accuracy significantly greater than the accuracy that would be expected from a random classification process, achieving a 65.39% accuracy in classifying each tweet as one of five primary emotions, performing significantly better than the random baseline of 41.5%. In addition to reviewing relevant prior works in the field that have informed our model design decisions, we outline the data collection and processing pipeline necessary to generate the data used as well as the process of developing, training, and tuning this model.

1 | Introduction

Evaluating emotional expression in textual data can be a difficult task for human agents but to do so on the scale of hundreds of thousands or millions of words of text without software tools can be entirely intractable. In this report we present our work in designing, developing, and training a neural-network model based on a convolutional and bidirectional long short-term memory architecture and applying that model to the task of classifying the primary emotion expressed in short text data samples in the form of tweets. We train this model using recent tweet data from the past two years collected from the Twitter API pertaining to the COVID-19 pandemic in order to ensure the novelty of the dataset and avoid analyzing an over-examined or established canonical dataset. The motivation for such a model is immediate: with the availability of an accurate and reliable model that can analyze emotional expression in publicly accessible data from a ubiquitous platform such as Twitter, interested parties could evaluate public response to e.g. major global events. As an example use case, policymakers could evaluate public emotive responses in order to gauge the impact of the policies they enact and use that knowledge to inform and guide future policy. We outline our project progress from initial literature review to constructing the data collection and processing pipeline through to the design and training of our model and the assessment of its performance. Additionally, we holistically examine some of the potential socio-ethical impacts and effects of work we have carried out as well as its possible broader applications. All the relevant datasets collected and compiled to support this project in addition to the scripts and code written in its development can be accessed and viewed in our repository at https://github.com/g-x-w/tweet_emoclass. We will be referencing the contents of that repository throughout the report, so we encourage the concurrent viewing of that repository with this report.

2 | Prior Work & Research

This section explores and summarizes prior work that addresses problems similar to the one we have identified above. There are many methods that exist for classifying the polarity of tweets where each tweet is assigned a positive or a negative sentiment label. It is much more difficult to determine the specific emotion associated with a given tweet such as happy, sad, fear, anger, surprise, denial, etc. However, there are existing techniques that tackle this issue using different types of methods. Existing methods can be categorized into two different classes; psychological principles and machine learning techniques. The psychological [1] background required for this

application involves defining methods for determining what emotion is represented in a tweet. These principles are useful for manually labeling twitter datasets which can then be fed into machine learning models. In terms of the machine learning techniques, there are different approaches that have been utilized such as logistic regression, naive bayes emotion classifier, transformers such as BERT, CNNs, RNNs and more. We focus on three specific prior works and explore them in detail. These papers use the following techniques; RNN architecture, CNN architecture and utilizing BERT.

Prior Work #1: "EMOCOV: Machine Learning for emotion detection, analysis and visualization using COVID-19 tweets" [2]

The authors of this paper aim to identify societal response to COVID-19 by determining the sentiment of COVID-19 related posts on twitter. Specifically, they have developed a model that gives one of the following labels to tweets; neutral, optimistic, happy, sad, surprise, fear, anger, denial, joking and pessimistic. Furthermore, they have also built a model that determines which segment of each tweet is most responsible for the overall emotion of a given tweet. Since our focus is more aligned with the paper's first goal of determining overall sentiment of a tweet, we focus on exploring its model. For the data collection process, the authors decided to create their own dataset as they determined there is no suitable extant dataset. They collected tweets through the Twitter Streaming API using Ptyhon's Tweepy Library and then three students manually labeled a portion of the data. The model structured used in this paper is outlined below:

- *Input*: A vector representing a preprocessed tweet is fed into the network.
- *Embedding*: By using the GloVE algorithm, each word in the input (tweet) is mapped to an embedding vector.
- *Bidirectional Long-Short Term Memory (BiLSTM):* Useful features are extracted using the BiLSTM.
- *Attention:* Determines how semantically close words are through an attention score.
- *Auxiliary Features:* Using Python's Natural Language Toolkit, features such as emoticons and punctuation are extracted.
- *Another BiLSTM Layer:* Incorporates output from the auxiliary layer.
- *Output:* Sigmoid activation function is used in a fully connected layer to output the emotion.

Training, validating and testing was performed on the manually labeled portion of the dataset and on an augmented dataset with existing labeled datasets. The model was evaluated using four different metrics as follows. For the manually labeled set, the results were as follows; accuracy: 0.8647, Jaccard: 0.5366, F1-Micro: 0.5514, F1-Macro: 0.5392. For the augmented data; accuracy: 0.8951, Jaccard: 0.6475, F1-Micro: 0.6893, F1-Macro: 0.6342.

Prior Work #2: "Multi-Channel Convolutional Neural Network for Twitter Emotion and Sentiment Recognition"[3]

Similar to prior work #1, the goal of this work is to detect emotions in tweets. However, the method being this model involves a multi-channel convolutional neural network (CNN), categorizing tweets into the following labels: joy, sadness, anger, love, thankfulness, fear, surprise, guilt and disgust. A CNN model was chosen due to its fast computation and ability to achieve high performance with noisy data which is applicable to tweets since they may contain many grammatical mistakes. In terms of data collection, multiple existing labeled datasets were used. The final model was run on each dataset separately and the performance on each dataset was compared. The data preprocessing involved the use of the TweetTokenizer package, lowercasing of tweets, formatting user mentions, unabbreviating words and the use of the emoji python library. The classification model includes the following layers;

- *Embedding:* Two GloVe vectorizations are used. The first one embeds the tweets themselves and the other one uses auxiliary features such as emoticons, emoticons and hashtags.
- *Convolutional:* Two CNN layers are applied to each of the embedded matrices along with a RelU activation function.
- *Pooling:* A max-over pooling strategy is used for both channels to determine which features are most important.
- *Hidden:* The feature vectors created from the pooling layer along with a few external lexicon feature vectors are concatenated into one feature vector. Then it is passed into another ReLU activation function.
- *Output:* A fully connected layer with a softmax activation function is used to output the predication.

To evaluate the model, it was tested on three different variations of the model. The first one only used the tweet embeddings, the second one used the tweets along with the auxiliary features and the last variation included external lexicon features. The highest accuracy was from the third variant which achieved around 87%.

Prior Work #3: Emotion and Sentiment Analysis of Tweets Using BERT[4]

The goal of this prior work is to perform emotion detection by categorizing tweets into one of the following classes; sadness, fear, anger or happiness. The approach used in this paper uses the BERT architecture. The paper uses an existing labeled dataset. To ensure a balanced dataset, the authors have decided to sample the same number of tweets from each emotion category. The model used is as follows:

- Preprocess tweets: Removing mentions, urls and retweets.
- Tokenize tweets.
- Feed into BERT pretrained model. This paper used the BERT-Base version which has twelve encoders with eight layers in each encoder, four layers of multi-head self attention followed by four forward layers.
- Apply a dense classification layer followed by a softmax to determine the predicted sentiment.

Furthermore, the authors experimented with the BERT architecture by trying the uncased BERT base model followed by the cased one. The uncased model accuracy was 89% whereas the cased model achieved an accuracy of 90%.

Relevance to our work:

After exploring different prior works, we have decided to use a combination of techniques suggested by the prior work #1 and #2, specifically using both CNN and recurrent neural network (RNN) techniques. As mentioned previously, CNN's work well for dealing with noisy data and are useful for extracting features independent of their spatial position in the data. However, RNNs are well-suited for dealing with sequential data which is applicable to tweets. Therefore, as both of these methods are applicable to tweet analysis and both perform well in isolation, we have decided to implement a model that will use both techniques. Furthermore, we will be using a BiLSTM to avoid the issue of vanishing gradients that occurs with RNNs. A BiLSTM is preferred over an LSTM as it is able to have access to information in both directions. Moreover, we will also be using GloVE to vectorize the data, as the prior work suggests this to be an effective strategy for data vectorization. An overview of the model that we have implemented is given below and will be discussed in greater details in the following sections.

1. Processing and cleaning the data: Removing punctuation and emoticons.
2. Tokenizing the data and turning the labels into a one-hot vectorization.
3. Vectorizing the data: Using GloVe embeddings.

4. Splitting data into training, validation and test set.
5. Convolution Layer: Generate features.
6. Batch Normalization: Allows training to be faster.
7. ReLU activation function: Applying non-linearity.
8. Pooling Layer: Extract most prominent features.
9. BiLSTM Layer: RNN layer.
10. Linear Layers and apply softmax to generate outputs.

3 | Data Collection & Processing

The first step of the data collection process was to collect the actual contents of the tweets used. To do so, we first parse the tweet ID numbers from the prior research dataset and using those IDs along with the Tweepy python library for Twitter's API, we query the tweet that each ID is associated with, download it and write it out along with its labels and other pertinent information to another csv file, found in the data_from_api directory within the github repository. The source code used to carry out and automate this initial data collection and API interaction can be found in data_intake.py script. There were a number of issues faced during this collection process, the most significant of which being the rate-limiting restrictions imposed by Twitter on the volume of API interactions allowed per given time frame. On a free/educational level of access we were using, the API allowed only 900 queries each 15 minutes, or 1 query per second. As such, in order to accumulate a dataset that was sufficiently large, it was necessary to automate the process of querying the Twitter API over the course of roughly 58 hours of operation. Another goal of the data processing was to work within relevant space constraints; it would be untenable to work with monolithic data files that would be too large to efficiently load into the model, so we partition our data collection into a number of manageably sized portions. Lastly, while runtime complexity was not a major priority due to the API rate-limiting being the bottleneck in the pipeline, we abide by best practices to ensure that the program's performance is not hindered by inefficient no-ops or the like. A total of 207863 tweets were collected. After the tweets were collected, they were further processed to include just the contents of the tweet along with the primary emotion label. This processing was done for ease of use in training batches with the model.

In order to ensure accuracy of labeling and to avoid too many non-informative "no specific emotion" samples, we label each tweet with the emotion scored the highest in the research set but apply a different filter, using a threshold value such that if the emotion scores for any of the tweets differs by 0.01 (with scores ranging from [0, 1]) or less, it is given a label of 'no emotion'. The complete dataset after this process includes the contents of the tweets and their labels as one of: 'fear', 'sadness', 'happiness', 'anger' or 'no emotion'. The data can be found in the tweet_labels directory and the source code for producing the labels is located in actual_code/label_data.py. In the process of data collection for the testing set, our team went with the second approach in the original proposal. By leveraging Twitter's search API in conjunction with the Tweepy python library, a total of 400 tweets were collected and compiled, filtered by English language and keywords linked to strong primary emotion expression (e.g. "amazing" or "fantastic" imply happiness). This approach enables a generalized evaluation accuracy unaffected by possible skews within the COVID-19 centered dataset. It also allows for manual labeling to ensure the integrity of the test set i.e. the labels for the test examples are human-verified. A query consists of a list of 6 keywords from one specific label joined by or operators excluding retweets. A query is used to pull 100 tweets per labeled category (fear, happiness, anger, sadness). The no emotion set is excluded in the pulling process since no emotion could not be easily extracted using keywords; tweets devoid of emotion tend to be

factual/news tweets that can vary widely by coverage and typically fail to share easily identified common words. The context of each pulled tweet is verified manually to be matched to its label. The data can be found in the manually_labeled directory and the source code used to carry out the queries is included in manual_label_tweets_gen.py.

During data processing, the actual tweets pulled from API are fed into the tokenizer function where punctuations and emoticons are removed and the sentences are then broken down into lists of word tokens with each list representing one tweet using text_to_word_sequence for Keras. The corresponding emotion label of the tweet is one hot encoded into a list simultaneously. An example of a pair of input-output for one specific tweet instant could be "I hate birthdays!", anger → ['I', 'hate', 'birthdays'], ['1', '0', '0', '0', '0']. The function tokenize_csv and tokenize_manual in main_model.ipynb tokenizes our main data set and the manually labeled set respectively. The output of the tokenizer function is embedded using the 6B tokens, 400k vocab, 50-dimension pre-trained GloVe embedding. The choice of embedding is to satisfy the minimum viable design; the 6B tokens package takes up significantly less memory space as compared to 42B and 840B embeddings, and we believe that 50-dimensional vector representations seems to provide a good balance between enough abstraction through dimensionality and fair computational intensity. Through this process, each tweet is turned into a 50x50 matrix representation, with tweets consisting of fewer than 50 words being padded by zeros. The load_pretrained_embeddings function loads GloVe pre-trained embeddings into a dictionary with word tokens as keys and embeddings as values. The tweet_vectorize function embeds its input using the embedding dictionary. The output of the GloVe embedding is shuffled along with their labels within the entire set. It is then split into 0.8:0.1:0.1 for training validation and testing set. The test set obtained is combined with the manually labeled set generated using keywords to provide more comprehensive testing criteria. The source code is included in the split_train_val_test function which can be viewed in our repository. Figures 1 and 2 below provide a comparison for the data distribution from the prior research team and after our preprocessing.

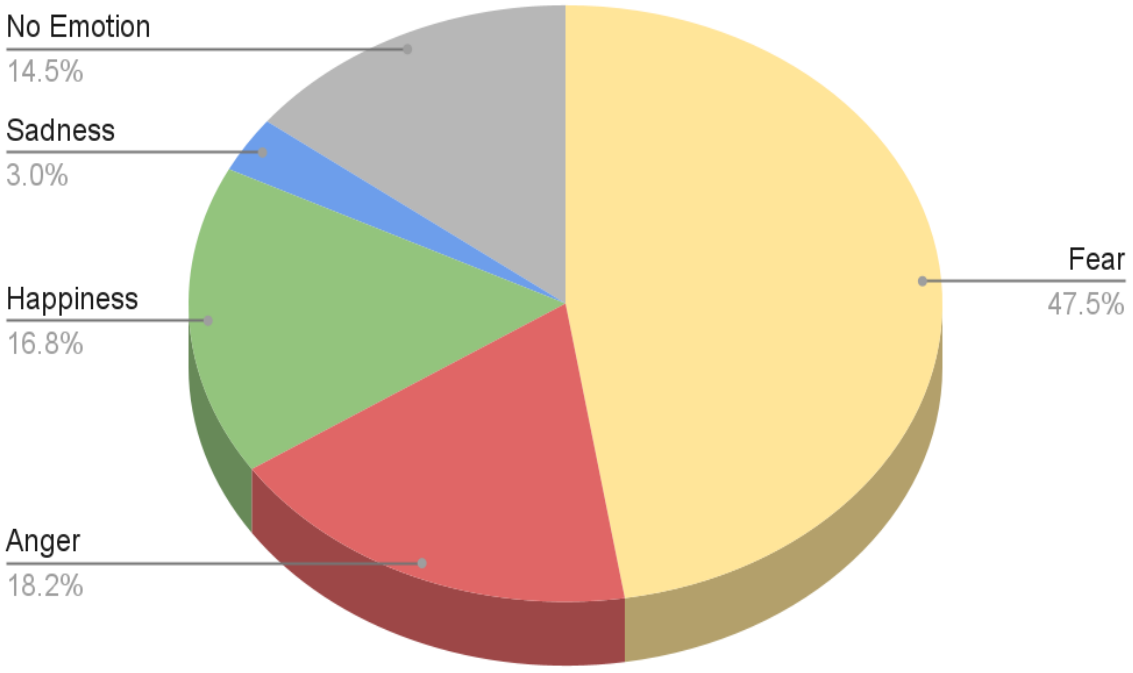

Fig. 1: Tweet Emotion Distribution, as per Gupta et al.

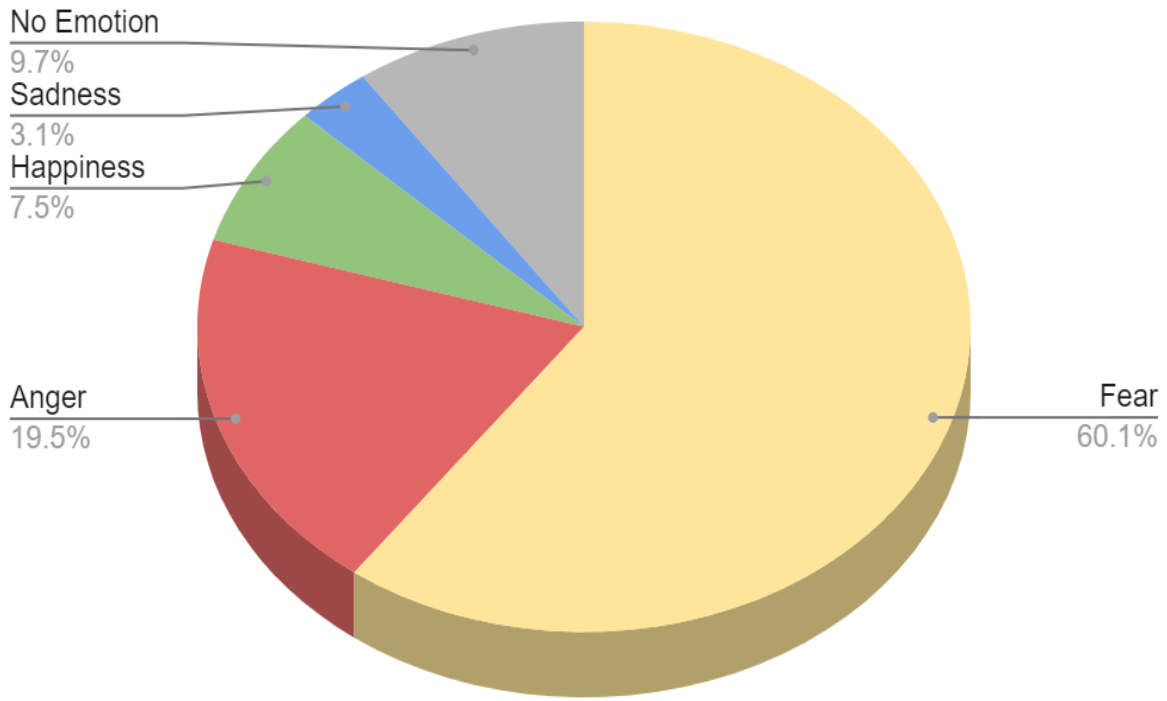

Fig. 2: Tweet Emotion Distribution with 0.01 Score Thresholding

4 | Model Implementation & Training

To implement and train our model we transform the processed and vectorized data into tensor objects of dimensionality 50 x 50 x N for the data tensor and 1 x 5 x N for the labels tensor, where N = 207863 as the number of examples we have. Each layer of the data tensor corresponds to the 50 x 50 matrix representation of a tweet in the dataset, vectorized using pretrained GloVe embeddings as mentioned prior. Accordingly, each layer of the label tensor is a 1 x 5 one-hot encoding for the primary emotion expressed by the tweet in the corresponding data tensor layer. Figure 3 below provides a representation of the tensor dimensionality of the data we work with.

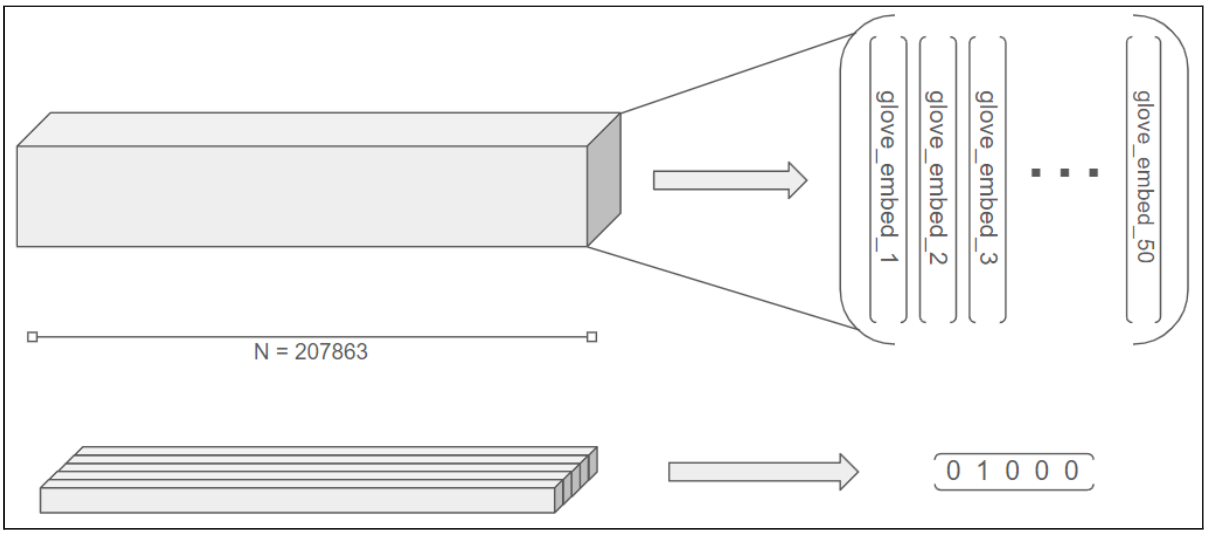

Fig. 3: Dimensionality Visualization of Data Tensors

With this data dimensionality in mind we can construct our convolutional bi-LSTM model. We begin with a convolutional layer with an initial kernel size of 3 and default stride of 1. These

parameters provide a convolution window of size 50 x 3 that slides over the matrix representation of each tweet moving one token vector at a time. Thus, the convolution allows for extraction of information and relational features that may exist in a given tweet between adjacent words. The outputs of this convolution are then batch normalized to standardize input ranges between mini-batches during training. Ideally, this normalization assists in stabilizing the training and learning process and reducing the training time necessary. After this batch normalization, the inputs are then passed to a rectified linear unit (ReLU) activation function, and lastly max-pooled in order to highlight the most relevant information. After this sequence of convolution, normalization, activation, and pooling, the data then goes into the bi-LSTM layer. In order to ensure dimension compatibility between the convolution output and the expected inputs to the bi-LSTM layer, we carry out a minor computation of the expected output shape based on the shape of input data and the chosen convolution kernel size. Using that expected input shape along with a single-layer bi-LSTM of size 4, we pass our data into the bi-LSTM. The output of that bi-LSTM layer is then fed into two consecutive fully-connected, dense layers which serve to transform the output shape into a 1 x 5 prediction vector which a softmax function can be applied to. Ultimately that prediction is then one-hot encoded to allow for comparison with the label for the corresponding tweet to assess accuracy. Figures 4a and 4b below provide visualizations of the layers used in the model. Specific details and information about our model implementation can be viewed in our GitHub repository.

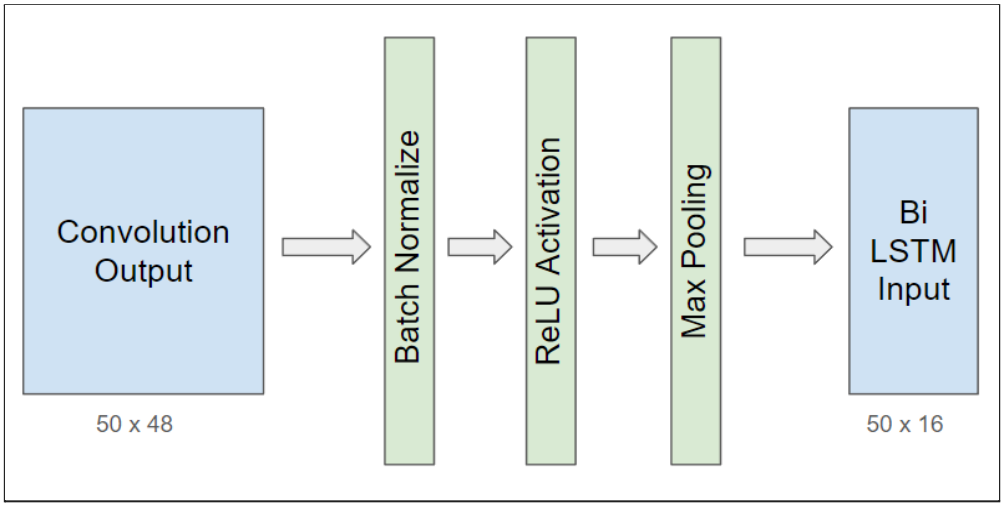

Fig. 4a: Visualization of Convolution through Maximum Pooling Model Layers

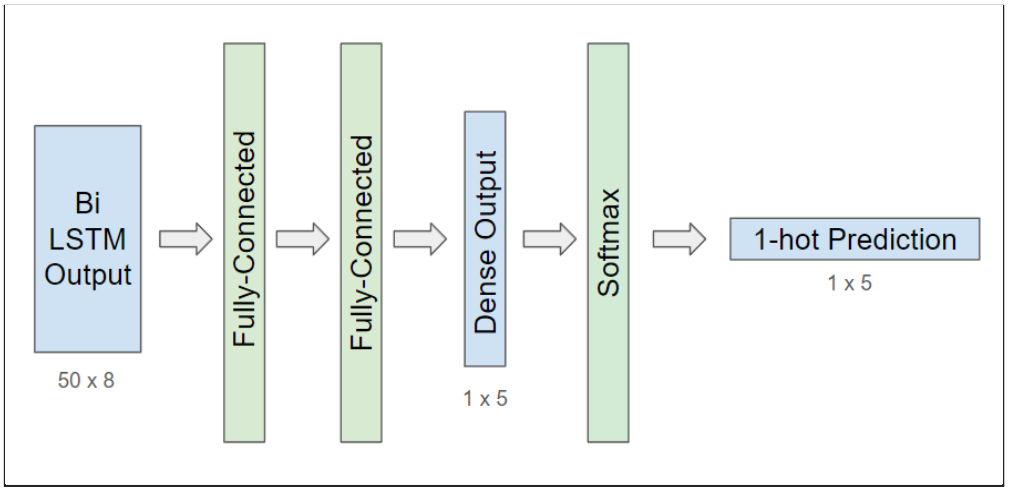

Fig. 4b: Visualization of Bi-LSTM through Output Prediction Model Layers

For the training process, we made use of batching primarily due to the size of the dataset; while we initially attempted to run training cycles on the entire training set of 166000 tweets, the tensor object proved too large for the model to parse due to GPU cache size limitations. As such, we train instead in batches with batch size being a tunable hyperparameter associated with the training process. During tuning of hyperparameters, we explored a range of parameters for both the model such as convolution kernel size and stride size or bi-LSTM hidden layer size as well as parameters related to the process of training such as the learning rate, number of epochs trained over, or the batch size. Due to the project's time constraints we were unable to implement a rigorous hyperparameter exploration method such as grid searches but we were able to achieve significant improvement in model accuracies through manual parameter searching and tuning.

5 | Results

With an initial parameter set of {epochs = 10, learning rate = 0.001, batch size = 10000, loss = BCELoss, output channels = 30, kernel size = 3, stride = 1, hidden size = 4} we achieve accuracies of 60.49% on the training set and 60.34% on the validation set. After some manual parameter tuning, we find that a parameter set of {epochs = 50, batch size = 500, loss = CELoss, output channels = 50} with other parameters remaining unchanged yielded significantly improved accuracies of 69.2% on the training set and 65.77% on the validation set with a final test set accuracy of 65.39%. Notably, the parameters that seemed to have the most immediate and significant impact on improving accuracy were training on a smaller batch size over more epochs. All the combinations of various different parameter values that were manually tuned and tested can be found in the jupyter notebook containing our model which can be viewed on our GitHub repository. Figure 5a and 5b below show the model's loss and accuracy on the training set over 50 epochs of training with the final hyperparameter set. For a random baseline classifier choosing classes with uniform random distribution we would expect an accuracy of 41.55%; in comparison, our model's performance of just over 65% accuracy is significantly greater..

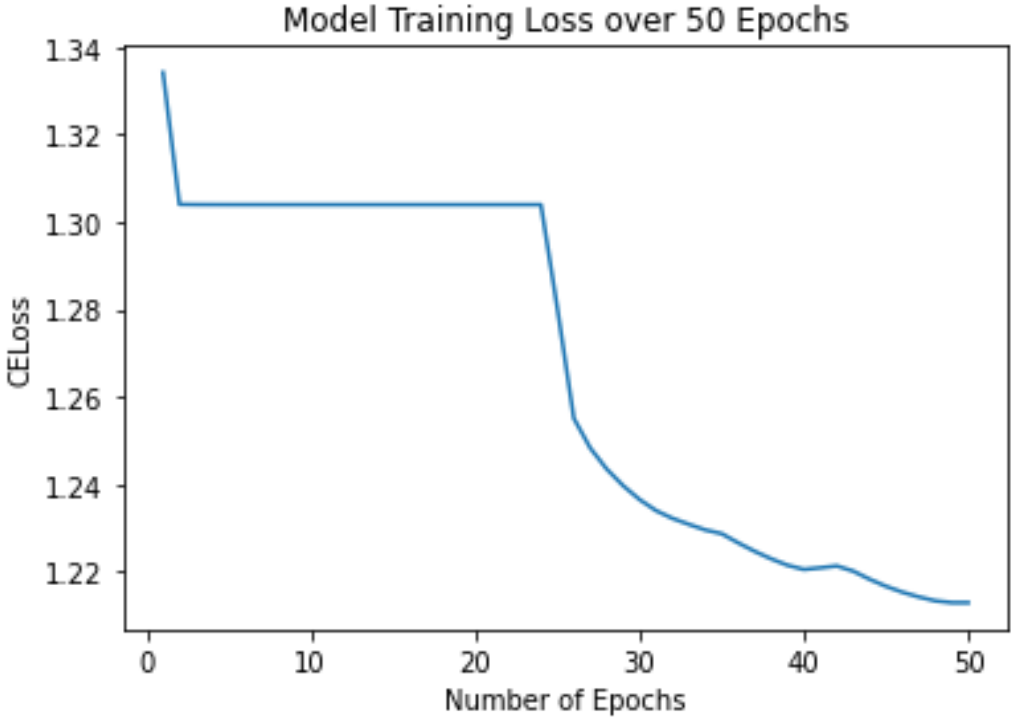

Fig. 5a: Model Training Loss with Final Parameter Set

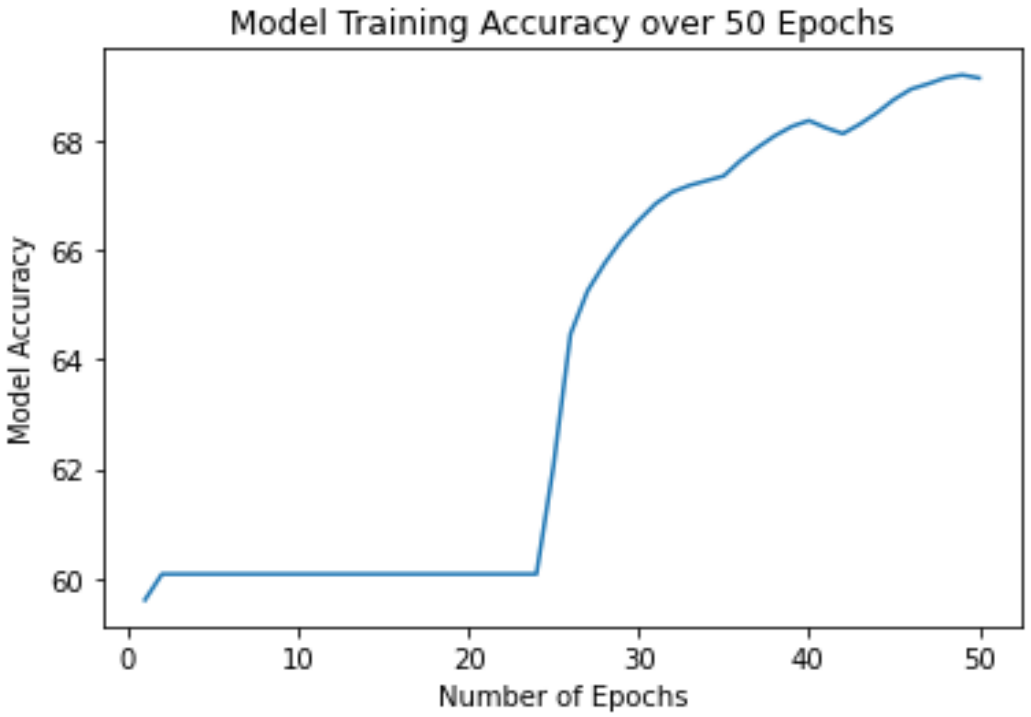

Fig. 5b: Model Training Accuracy with Final Parameter Set

6 | Broader & Ethical Impact Statement

The motivating principle for our model is to detect sentiment elements in an informal writing platform to aid various decision-making processes. With minor modifications, our model could be applied to other informal interactive platforms such as Facebook, and Quora to analyze sentiments in opinions. At a theoretical level, we can generalize to the broader process of algorithmically extracting implicit information from human writing. As a result, it can be a useful tool in areas such as marketing, governance, and policymaking. For product marketing, the sentiment analysis of customer reviews provides direction for product development and research. Due to the use of the COVID-19 dataset by design, our model can also be used to better understand public response to major events and assess the effectiveness of the incumbent policy as well as inform future decision making. However, concerns arise because of the scope of its applications: do general citizens necessarily want governments and other third-party entities to be making use of technology built with their data? The topic of such technological grey areas are complex socioethical issues that must be considered in various areas of machine learning and artificial intelligence research in general. In the context of our model, some significant concerns include privacy, model design details, and more broadly the possible inadvertent endangerment to the preservation of less common languages through exclusion.

The first ethical aspect to look at relates to the topic of privacy. Our model relies on collecting tweets posted by members of the general public. A potential issue that follows from this collection is that users are not explicitly agreeing to having their tweets analyzed during model training and may not be aware that this is a possible use case for their data. Although all twitter users agree to the privacy terms and conditions when making an account, the vast majority do not read these conditions and are therefore not aware of the possible uses of the content they generate and share. Furthermore, it is difficult to directly indicate what purposes user content will be used for in the terms and conditions because tweets are publicly available and can be used for many types of models and other complex applications. Therefore, the official terms and

conditions often fail to accurately describe what the tweets may be used for and thus be a concern for users. A possible solution to this issue, would be to give the option for users to entirely disallow their tweets from being made available on twitter API's.

In addition to the privacy concerns related specifically to the model we have built, it is also important to look at how our model can be extended and the potential issues that arise. For example, to improve the accuracy of the model, one possible approach would be to look at each user's history to classify new tweets. Specifically, looking for patterns among tweets that a specific user has posted could understandably make future predictions more accurate. This extension brings up further privacy concerns as models could then keep a record of the users in the dataset. Furthermore, as mentioned above, our model can be useful for applications such as analyzing public response to new policies or global events making users even more concerned with the usage of their posts. As such, different avenues by which our work can be extended or built upon will inevitably bring accompanying ethical and privacy concerns.

Another point of ethical impact relating to our project is the model's ability to generalize to different languages. The vast majority of digital content generated and shared on the modern internet is contained within a subset of the languages that are most popular such as English, Chinese, Spanish, etc [5]. Due to this, languages that are less common are not able to participate in a heavily technology-driven modern society or in advancing areas of technological development such as NLP applications due to their lack of digital presence. The amount of data available for representing languages that do not have as much digital presence is significantly smaller. Consequently, the lack of data makes it more difficult to train models well that will achieve high performance. As such, a model such as ours built on the existing corpus of data unintentionally contributes to the repression of minority languages through lack of representation severely restricting the accessibility of the technology to groups who speak these under-represented languages. Therefore, it is important to prioritize the model's ability to generalize well to as many different languages as possible to make the model more accessible.

7 | **Conclusion**

Overall, we observe that a convolutional bi-LSTM based architecture is a viable foundation for the task of emotion classification in text. While we note that our achieved accuracies are below that of prior works in the field, our model does perform significantly better than the expected baseline and we have identified multiple avenues for potential improvement as well. Possible future work to be done or next steps that could be taken to improve the model's performance include more thorough and rigorous hyperparameter tuning during training such as through algorithmic approaches like a parameter grid search. Additionally, the prior works referenced suggest that the use of multi-headed attention as well as auxiliary features such as emoticons, URL hyperlinks, or hashtags could significantly increase prediction accuracies as well. As such, incorporations of different layers in our model as well as different additional features in the dataset or the use of higher dimensionality embeddings of that dataset could be potential next steps to improve model performance. Moreover, we believe the model could be generalized and applied to other forms of textual data aside from tweets to provide it additional use-cases and versatility.

References

[1] A. Bandhakavi, N. Wiratunga, S. Massie, and D. P., "Emotion-aware polarity lexicons for Twitter sentiment analysis," 09-Dec-2017. [Online]. Available: https://onlinelibrary-wiley-com.myaccess.library.utoronto.ca/doi/pdfdirect/10.1111/exsy.12332. [Accessed: 30-Jan-2022].

[2] Y. Kabir and S. Madria, "EMOCOV: Machine learning for emotion detection, analysis and visualization using COVID-19 tweets," *Online Social Networks and Media*, 16-May-2021. [Online]. Available: https://reader.elsevier.com/reader/sd/pii/S2468696421000197?token=8CF8B1C1797BC134400A3D6383E7F6185BF52029E035B9B611C4105072C35C060B74F65F49B2C0F0A3A2C4ECB55E7BDE&originRegion=us-east-1&originCreation=20220124035631. [Accessed: 30-Jan-2022].

[3] J. Islam, R. E. Mercer, and L. Xiao, "Multi-Channel Convolutional Neural Network for Twitter Emotion and Sentiment Recognition," 02-Jun-2019. [Online]. Available: https://aclanthology.org/N19-1137.pdf. [Accessed: 30-Jan-2022].

[4] A. Chiorrini, C. Diamantini, A. Mircoli, and D. Potena, "Emotion and sentiment analysis of tweets using BERT." [Online]. Available: http://ceur-ws.org/Vol-2841/DARLI-AP_17.pdf

[5] J. Johnson, "Internet: Most common languages online 2020," Statista, 26-Jan-2022. [Online]. Available: https://www.statista.com/statistics/262946/share-of-the-most-common-languages-on-the-internet/. [Accessed: 09-Apr-2022].