Deep Learning

Understanding Human Emotions: The DistilBERT-CNN Way

Team 12

Manas Ranjan Sahoo & Chelsea Salyards

School of Graduate Professional Studies

Data Analytics

AI 570 – Deep Learning

Summer Semester, 2025

Document Control

Work carried out by:

Name	Email Address	Exhaustive list of Tasks
Chelsea Salyards	cns5070@psu.edu	• Supporting proposal research, collecting/reviewing datasets • Review original GoEmotions source code and evaluating API, module version, and OS compatibility issues • Convert original Tensorflow 1 code to Tensorflow 2 • Convert Tensorflow Estimator API code to Keras • Initial conversion of code to use TFDistilBert from HuggingFace • Refactor multifile Python script/module project into Jupyter notebook • Experimented with configuration flags of original project to find best configuration (uncased, multilabel, not freezing BERT layers, using sentiment and correlation weighting) • Implemented functional correlation weighting code • Validation graphs • Coalesce research and existing information from proposal documents, collected research papers, and scientific readings into project write-up document, analysis of research • Supported model analysis
Manas Ranjan Sahoo	mzs7206@psu.edu	• Initial proposal document creation • Initial draft network architecture creation • Convert TextCNN model architecture into Tensorflow from pyTorch • Incorporate combined model architecture using TFDistilBert and TextCNN with custom Classifier head and other custom classes from GoEmotions original work. • Add serialization to custom classes to create a downloadable model and tokenizer • Add visualizations to the predictions over test dataset

		• Added weighted-loss function to the custom loss function to handle class imbalance • Early stoppage mechanism • Added different metrics for performance measurement • Recorded analysis for different performance results • Document write up, recording additional analysis on document • Overall formatting, citation and bibliography

Revision Sheet

Date	Revision Description
8/3	Initial template for Project Write-up Initial Challenges section Initial Data Collection section Initial Data Preprocessing section Initial Methodology section
8/4	Initial Introduction Initial Problem Statement Initial Importance and Impacts
8/5	Data Collection Revisions Data Preprocessing Revisions
8/7	Initial Related Works
8/8	Related Works revisions Initial References Initial citations Initial References
8/10	Introduction update Problem statement update Challenges update Importance update
8/13	Updated References General document cleanup Initial Feedback section
8/14	Initial Discussion of Results Data Preprocessing and Methodology revisions

8/15	Overall document updates across various sections Addition of finalized graphs and figures Results and Interpretation
8/16	Final formatting, citation and font checks

TABLE OF CONTENTS

1 INTRODUCTION

Sentiment analysis has been a challenging field for machine learning and AI models for multiple decades. A finer understanding of human emotions is important in applications like customer service chatbots, recommendation engines and social media monitoring, as business critical functions rely on machines understanding user intent accurately and providing relevant information. Human emotions are layered, and while languages keep on evolving to grasp all the emotions that humans try to emulate, ambiguity and complexity behind human sentiments, expressed through language, still prevails. The primary reasons for the ambiguity in understanding human emotions resides in the inherent complexities of human communication, including:

- Nuances and ambiguity in written languages like sarcasm and irony through which one can use a positive word to express a negative sentiment, and vice-versa (e.g., "You've come up with a truly brilliant idea there" can mean both positive and negative).
- Lack of context – a sentence could be construed in a different way due to the context (e.g., "That's just great" lack of context doesn't tell us if it is said positively or not).
- Cultural differences – sentence could bring out different sentiments in different cultures (e.g., the word 'gift' is positive in English but means married or poison in Norwegian)
- Perhaps the most significant limitation has been the lack of large-scale, high-quality datasets for training and validation, particularly for fine grained emotion recognition.

The critical gap in data has over the years limited the performance of sentiment analysis models. This is where the Google Emotions dataset presents a unique opportunity as the largest human annotated fine-grained emotions dataset to date. We decided to use it to train a novel deep network to better understand human emotions and sentiments. This paper details our methodology and demonstrates how our approach strives to improve upon the performance benchmarks set by the original study.

2 PROBLEM STATEMENT

While the GoEmotions dataset was no doubt a valuable resource for us, it was accompanied by a baseline BERT-based model that showed significant limitations for practical application. Through our exploration of sentiment analysis related work done by other researchers, we found multiple of them having proposed hybrid architectures are the solution to improving the sentiment analysis performance over models that just use BERT alone. We propose creating a multi-input model with one input being BERT, the other being a CNN model, specifically TextCNN. BERT for sentiment analysis but could only muster an F1 score of 0.46. (Demszky et al. 2020). A good F1 score for social media context tagging is considered 0.6 or higher (F1 Score in Machine Learning 2025), and with this project, we aim to breach that benchmark by utilizing other models and architectures.

Additionally, due to BERT being a large, highly process-intensive model to train and additional challenges posed by our device limitations, we proposed swapping from the original

GoEmotions BERT implementation to a smaller, faster BERT variation called DistilBERT (Sanh et al. 2019). The GoEmotions team fine-tuned an existing BERT model used by a different Google Research team, providing their model code with customizations to the public (GoEmotions-Fine-grained emotion classification 2021). Due to our limited processing resources, we are looking to replace their original underlying BERT model with a different variation of BERT that we expect will work better on our machines, but still replicate their customizations of additional classification layers, a loss function, metrics, and learning rate schedule, while giving targeted results. DistilBERT appears to be a suitable option for our purposes. Although architecturally similar, DistilBERT is optimized for performance by reducing the number of layers required. The result is that DistilBERT retains 97% of the F1-score of the normal BERT implementation while being significantly faster and less process-intensive to run (Sanh et al. 2019).

As we plan to improve upon the neural network used by the teams behind the GoEmotions dataset, we intend to achieve an expected F1 score of 0.6 to meet the benchmark of sentiment analysis for social media posts. The different metrics that we plan to evaluate include precision, recall, and F1 scores.

3 CHALLENGES

We ran into several difficulties while working on this project, most of them being of a larger scope than we initially predicted.

One of the challenges we predicted in our proposal was that in the process of switching to a different BERT model and incorporating TextCNN, we would have to do some additional refactoring to handle the incompatibilities between the APIs. This issue turned out to be much more complicated and time-consuming than we expected.

When first reviewing the GoEmotions codebase, we noted that the code was written in TensorFlow already, so we expected working with the codebase would be simple. The existing code incorporated code for its own BERT model, and we figured that we would run into some differences between the GoEmotions BERT implementation and the DistilBERT implementation when we plugged it in. We also noted that the TextCNN library was written in PyTorch, not TensorFlow, but it looked straightforward, so we did not expect converting it to be an issue.

It turns out the biggest issue was trying to utilize the original GoEmotions codebase at all. What we did not realize at first was that GoEmotions was written using TensorFlow 1, using an old API pattern called Estimators, that is no longer supported. While TensorFlow 2 still has some compatibility libraries to support TensorFlow 1 code, the estimators API was completely removed – nor did the patterns of the estimator API make any sense for us to use in this project because we had planned to use Keras Sequential API, which is not compatible with Estimators. (Keras, n.d.)

Ultimately, we determined that we needed to refactor the relevant parts of the GoEmotions codebase into TensorFlow 2 and Keras Sequential API compatible code. We primarily focused on the bert_classifier.py code from the original repo since it contained most of the code related to processing data and training the models (GoEmotions-Fine-grained emotion classification 2021).

While we attempted to use AI tools such as ChatGPT and Gemini to help assist with the code refactor, it soon became clear the AI tools alone were not capable of making intelligent decisions about what parts of the code needed to change and how and still be able to compile, so we had to do a lot of research into the API docs and trial and error to get something functional. While the result is not identical to the original implementation, having to pull it apart like this provided a lot of education into how the original model was designed and why various pieces of were included.

To save time, we did not convert the custom BERT implementation within GoEmotions since it also depended on a lot of TensorFlow 1/Estimators behavior. Instead, we migrated to DistilBERT directly. We tried two versions of DistilBERT. First, we tried using the default TFDistilBertModel provided by HuggingFace. Because this is a basic model, we had to reimplement a lot of the extra processing layers used in the original GoEmotions project to support the existing regularization and classification features. We also attempted to use TFDistilBertForSequenceClassification from HuggingFace. Initially, we thought this would provide a shortcut for us because it is a version of DistilBERT already trained to handle the classification tasks we were attempting to do. Unfortunately, we struggled to get the TFDistilBertForSequenceClassification implementation to work within the scope of our existing architecture, so we ultimately reverted to using the TFDistilBertModel. An additional issue caused by switching to DistilBERT is that we lost some of the context provided in the dataset from the original study. The original dataset contained segment ids, which are used for training entailment, or semantic relationships between two texts. Unfortunately, the base DistilBERT model was not trained on entailment, so it cannot use that information the way the original GoEmotions BERT implementation could (HuggingFace, n.d.). We ultimately decided that it was not useful enough to keep because the benefits of a smaller, faster model were much more attractive, especially considering our technology limitations, and the study itself does not mention any particular benefit from utilizing that information.

One additional challenge that we observed when we tried to keep all our codebases in TensorFlow was compatibility issues with using HuggingFace transformers and its configurations. HuggingFace recently changed the way they store model weights from Pickle-based .bin files to .safetensors files to make them safer, faster and portable across PyTorch and TF/JAX removing the need for any format conversions. Due to this change, some versions of Transformers still tried fetching the whole dictionary as they would try to do with .bin files and that caused our model compilation to fail as the new approach uses a SafeOpen object instead. Finding a solution to this problem, took multiple hours for us as this was a recent change but finally, we figured out that we needed to upgrade both the transformers and safetensors versions and specifically mention in the model calls to convert it from PyTorch to TensorFlow.

Implementing TextCNN as another input layer for the combined novel deep learning architecture was something that we expected to be less challenging as compared to other parts of work as the steps to process text inputs into the CNN model was quite a standard process. CNN models need a fixed length input and as we ensured that we were padding our input text to the fixed sequence length, we did not expect much of a challenge with the complete integration.

Another difficulty we predicted was long running times for the model because of a lack of a GPU on our machines. Originally, we expected we could mitigate this by using DistilBERT instead of the GoEmotions BERT; DistilBERT was specifically designed to be a smaller, faster version of BERT. Unfortunately, even with DistilBERT, it took approximately an hour for a single epoch to finish, resulting in 4+ hours to complete one training run every time we wanted to make a change. We eventually migrated to running it in Google Colab, which generally decreased the model training code to only taking about a half hour.

But our most critical challenge was the data imbalance prevalent in the original GoEmotions dataset. We could see that the data had an overwhelmingly large number of 'neutral' class records within it which would create model bias towards the majority class. This possibly has been another reason why prior works on the GoEmotions dataset do tend to show poorer performance. While some of the usual approaches to solving for data imbalances are random oversampling of minority classes, random undersampling of majority classes, and synthetic minority oversampling, we instead opted on using weighted loss function to update our custom loss function to account for the class imbalance.

3.1 RELATED WORKS

The foundation of our project relies on three studies. The first is, of course, the GoEmotions project conducted by the Google Research Team. In (Demszky et al. 2020)'s project "GoEmotions: A Dataset of Fine-Grained Emotions", the team describes the process behind the collection and optimization of the GoEmotions dataset that we used as a basis for our project, as well as the reasoning behind their choices in the development of the BERT-based model they used to conduct sentiment analysis on the texts they collected. Their mechanisms and our incorporation thereof with be expanded on throughout our discussion of our project.

The second foundational study for our project is Sanh, et al.'s "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter". One of our first tasks prior to completing the project proposal was identifying a version of BERT that was smaller and faster to run than base BERT but still had decent metrics. For this, we chose DistilBERT, which was developed and described by Sanh et al in their research project. DistilBERT was designed to be a compressed model, where it is trained to replicate the results of a larger model but with less resources and training time involved. DistilBERT was evaluated in that study to have 97% of the metrics performance of base BERT, while having 40% of the size and running 60% faster, making it a good choice to aid in the resolution of our challenge with processing resources for our project. (Sanh et al. 2019)

The original GoEmotions study used BERT alone to conduct sentiment analysis, but there have been a variety of other studies conducted that propose and evaluate other architectures. The results of these studies helped us refine the path we chose to go down in implementing our own model to perform sentiment analysis on the GoEmotions dataset.

The first study we referenced was (Abas et al. 2022)'s "BERT-CNN: A Deep Learning Model for Detecting Emotions from Text". In this paper, the team proposes a hybrid BERT-CNN

architecture, and how they would specifically design such a model. In their paper, they recommended a model using the below layout, where the output of the BERT model subsequently gets passed into a CNN model, where the CNN model is primarily used as a classifier.

Figure 1: Architecture of the BERT-CNN model

Credits: (Abas et al. 2022)

They had theorized that if the BERT and CNN models were combined, accuracy scores greater than using either model independently could be achieved. They evaluated several variations with promising results. Their hybrid BERT-CNN architecture received the highest scores on the datasets they used - for the ISEAR dataset - a decent 0.76 F1 score, and for the semeval dataset - a whooping 0.94. (Abas et al. 2022)

This study inspired us to also implement a combined hybrid BERT-CNN model, but the primary difference between our implementation and theirs is that we decided to pass the dataset into the BERT and CNN models separately, and then use those outputs as a dual inputs; this will be described further in our Methodology section. In our case, both our BERT and CNN sections of the model have implementations of a classifier layer; it is not the CNN part of the model alone working as the classification head.

This leads to the third foundational study of our project which was Kim's "Convolutional Neural Networks for Sentence Classification". This study describes a variety of text classification experiments the author conducted using a CNN model, which resulted in the TextCNN project. Although CNN was originally invented for computer vision, it has proven useful in natural language processing tasks, particularly in tasks involved in capturing patterns in localized contexts, which Kim's study demonstrated. We used TextCNN as our CNN model implementation. (Kim Y, n.d.)

The idea of using a model with multiple inputs to perform sentiment analysis is supported and evaluated by Duong, Lebret, and Abererécole in "Multimodal Classification for Analysing Social Media." Like our project, they evaluate using a multi-input architecture to train sentiment analysis on a social media dataset. In theirs, they primarily chose a multimodal architecture because they were evaluating multiple types of data – not just text, but also images used in social media posts, and combining the analysis of the text with the associated image to extract the overall sentiment. Since to their knowledge there was no existing dataset that combined text and images together, they ultimately did their sentiment training on two separate datasets. Like GoEmotions, they used Reddit as a basis for their text training, but on the image side, they pulled their data from Flickr. (Duong et al. 2017) While they are using multimodal inputs to evaluate different types of data and correlate the results, in our model, we are evaluating the same data but using the two inputs to extract and balance the strengths of each particular methodology. BERT and CNN pair well together because they can both be used to perform sentiment analysis but do so by different

mechanisms. CNN excels at pattern-matching in small local context windows and can match simple words to emotions, while BERT uses a larger frame of reference, with less focus.

A very recent study published this year (2025) by (Guo and Li 2024), "Research on Sentiment Analysis of Online Course Evaluation Based on EN- BERT-CNN" further evaluates the usage of a hybrid BERT-CNN model, specifically over a dataset of reviews of an online course platform. They propose, like in our project, that the strengths of using a transformer like BERT can be combined with the strengths of CNN, overall resulting in a stronger model. The researchers believed that by combining the two, the result will be better able to detect the correlation between words, resulting in an improved understanding of the meaning. Different from ours, it is performing this analysis of reviews of online courses versus a social media site like Reddit, so it is interesting to compare the results of such an architecture on different datasets. They also do an examination of the BERT-CNN architecture to others, comparing the results of using a BERT-CNN hybrid model to other architectures like BERT-RNN, which they say suffers from a disappearing gradient problem, and BERT-LSTM, which looks promising, but time-consuming to perform.

In their results, they discover that even though they chose CNN over LSTM in the hopes of better speed performance, their model still takes a long time to train. They also discovered that their combined model was very susceptible to bad training data, so poor quality data would result in inferior results. (Guo and Li 2024)

We referred to some additional studies that referenced alternative architectures from the one we ultimately chose.

One of these was (Tan et al. 2022) "RoBERTa-LSTM: A Hybrid Model for Sentiment Analysis With Transformer and Recurrent Neural Network", which evaluated a hybrid BERT-LSTM architecture similar to the one mentioned in Guo and Lei's research, but ultimately rejected due to concerns about length of time spent training. In Tan, et al., they proved that such a model was indeed very effective; the model uses RoBERTa for contextual embeddings and LSTM for capturing long-term dependencies while also solving for lexical diversity and imbalanced datasets using GloVe (Global Vectors for Word Representation, a tool used for embeddings) to create a robust sentiment analysis model. Their model shows significant improvement in performance metrics of accuracy, precision, recall, and F1-score over previous research models. (Tan et al. 2022)

This paper shows how an improved version of BERT, i.e. RoBERTa could be used to improve the speed and efficiency of word embeddings as well as the usage of LSTMs which helps retain long-term dependencies in sentences. It helps us understand that using a variation of BERT would bring additional value to the performance of the model and that is the reason why we chose to go with DistilBERT for our model. DistilBERT has shown great performance without being computationally expensive. Initially we had thought of using RoBERTa for our model, but due to the computationally expensive nature of the model, we decided to go with DistilBERT instead. While DistilBERT lacks temporal modeling which is available with RNN or LSTM models, we

believe its self-attention mechanism would be able to provide a suitable alternative while the CNN model helps us perform efficient feature extraction.

Another research team, Kokab, et al., utilized a different approach to perform sentiment analysis in "Transformer-based deep learning models for the sentiment analysis of social media data". In theirs, they used a different small BERT model, BERT-Mini model, and combined it with CBRNN. CBRNN is an architecture that combines the strengths of CNN and bi-directional LSTM processing. In theirs, BERT was used to conduct contextual embedding by understanding words based on their surrounding context. Their diluted CNN layers were used to extract both local and global contextual semantic features without losing information due to excessive pooling, which is a problem observed in classical CNNs. Finally, the Bi-LSTM layers were used to process the pooled features by learning long-term dependencies in sequential data through processing of information in both forward and backward directions. Their results suggested that by integrating BERT, dilated CNN, and Bi-LSTM features into a single model, it addresses the challenges of noise, out-of-vocabulary words, and contextual information loss experienced in traditional sentiment analysis models. (Tabinda Kokab et al. 2022)

This paper shows how a BERT based CBRNN model can effectively address the persistent problems of traditional sentiment analysis methods. It also helps show that a combination of BERT, CNN, and LSTM models is doable and can yield superior performance over diverse datasets. This helps validate our approach of using DistilBERT and CNN for fine-grained sentiment analysis as the right approach. We opted for DistilBERT for our model, as we expect it to be less computationally intensive as compared to BERT or BERT-Mini and we want to use CNN for better feature extraction as well. The usage of Dilated-CNN does give us some more insight into the weaknesses of traditional CNN models, which gives us insight on how we can modify our approach to address those problems based on this research paper.

The final research paper we reviewed was (Wang et al. 2025)'s "A BERT-based sentiment analysis model for depressive text". This article was of interest to us because it performed sentiment analysis using a similar BERT-only architecture to GoEmotions, but they did so with the goal of medical based research, being able to identify depression in written text. In our proposal, we mentioned the AI therapy market was an important use case for the sentiment analysis models generated by projects like GoEmotions and our implementation. Seeing another group truly evaluate this use case was encouraging. In their study, they also acknowledge other researchers' suggestions of hybrid models like the one we implemented but ultimately decided not to pursue that path themselves.

One challenge they had and tried to solve in their dataset was the issue of uneven data, particularly not having enough data demonstrating positive depression cases versus data where the text author was not. In their project, they resolved this using data augmentation, artificially creating new sample data to train on by rearranging words or sentences in the existing data. They had also suggested that this problem could be mitigated by adding weights to certain categories.

Like the dataset used in Wang, et al., GoEmotions also suffers from uneven categories, but the original researchers of GoEmotions went down the path of trying to select their data collection

in areas that might result in more data in rarer categories and using the weighting methodology instead of the data augmentation one to generate artificial samples. This is elaborated in our Data Collection and Data Preprocessing sections. However, the data augmentation technique could be a useful optimization for the GoEmotions dataset as well.

3.2 IMPORTANCE AND IMPACTS

A deep learning network with a better understanding of human emotions is crucial for advancing artificial intelligence from a transactional tool to a genuinely empathetic and effective partner. The problem of accurately identifying fine-grained emotions in text holds significant social, economic, and scientific importance.

Social Impact: Improvement of online safety and mental health support are some of the social impact areas that an improved sentiment analysis AI model can achieve. Additionally, improved content moderation systems through identification of harmful or violent content, even in nuanced language (e.g., sarcasm or coded communication) can be achieved through a robust model. On an individual level, creation of advanced AI therapists and mental health support systems are some of the possible implementation areas for such networks which can help them better gauge a user's emotional state and provide tailored, empathetic, and life-saving responses.

Economic and Business Impact: Understanding human emotions aids companies retrieve a deeper analysis of customer feedback, helping distinguish general dissatisfaction, frustration or brand loyalty. Improved business decisions, improved customer service and proactive brand reputation protection are some of the outcomes that companies can utilize these models for. On an individual level, the creation of intuitive and engaging personal assistants and chatbots based on robust networks will lead to a higher adoption, more positive experience and increased brand loyalty.

Scientific Impact: From a scientific perspective, focusing on the complex problem of fine-grained emotion recognition in a large, real-world dataset helps contribute to the understanding of how to build more robust and generalizable models. Our findings on hybrid architectures provide a new benchmark and a valuable resource for future research, helping create more sophisticated sentiment analysis and human-computer interaction in the future across various domains.

4 DATA COLLECTION

For our dataset, we simply used the existing dataset collected during the GoEmotions research study curated by the teams of Stanford Linguistics and Google Research. This publicly available dataset consists of 58,000 carefully curated Reddit comments.

The dataset uses the following format:

Field	Description
id	Unique identifier for the data row
text	Reddit comment text

example_very_unclear	Boolean value (TRUE/FALSE) flagging if the example text was difficult to derive emotion
admiration	Does text demonstrate admiration? 0 or 1.
amusement	Does text demonstrate amusement? 0 or 1.
anger	Does text demonstrate anger? 0 or 1.
...	

The comments are curated from popular English subreddits and are then labeled for 27 different emotion categories and 1 neutral category. The complete list of emotions classified includes admiration, amusement, anger, annoyance, approval, caring, confusion, curiosity, desire, disappointment, disapproval, disgust, embarrassment, excitement, fear, gratitude, grief, joy, love, nervousness, optimism, pride, realization, relief, remorse, sadness, surprise, and neutral. Multiple emotions can be assigned to a single example text. Additionally, as shown in the data structure, the appearance of an emotion in the text is signaled using 1 as a flag for presence and 0 for absence. If the *emotions_very_unclear* Boolean is TRUE, that means that no emotions could be assigned.

Therefore, the data included is categorical, and each Reddit comment could be assigned more than one category if required, making this multilabel text classification problem. In addition to assigning comments to individual emotions categories, the emotions themselves were categorized into higher level categories representing overall tone, such as whether the emotion was positive, negative, or neutral. (Bansal 2021)

The original research team did a lot of work in cleaning the dataset and writing the preprocessing code to handle typical data issues, so we did not have to do much work in that respect. The original research team summarized the data as follows.

Number of examples	58,009
Number of emotions	27 + neutral
Number of unique raters	82
Number of raters / example	3 or 5
Marked unclear or difficult to label	1.6%
Number of labels per example	1: 83% 2: 15% 3: 2% 4+: .2%
Number of examples w/ 2+ raters agreeing on at least 1 label	54,263 (94%)
Number of examples w/ 3+ raters agreeing on at least 1 label	17,763 (31%)

Credits: (Demszky et al. 2020)The original research team did a lot of work in cleaning the dataset and writing the preprocessing code to handle typical data issues, so we did not have to do much work in that respect. Discussion of the data processing tasks are included in the Data

Preprocessing section.The original research team summarized the data as follows.

Number of examples	58,009
Number of emotions	27 + neutral
Number of unique raters	82
Number of raters / example	3 or 5
Marked unclear or difficult to label	1.6%
Number of labels per example	1: 83% 2: 15% 3: 2% 4+: .2%
Number of examples w/ 2+ raters agreeing on at least 1 label	54,263 (94%)
Number of examples w/ 3+ raters agreeing on at least 1 label	17,763 (31%)

Outliers that remain in the dataset could be defined as texts marked as unclear or difficult to label (meaning it is hard to use to train sentiment), texts that are associated neutral sentiment (meaning no particular emotion stood out), or data where raters did not generally agree on the labels assigned to the texts. (Demszky et al. 2020)

5 DATA PREPROCESSING

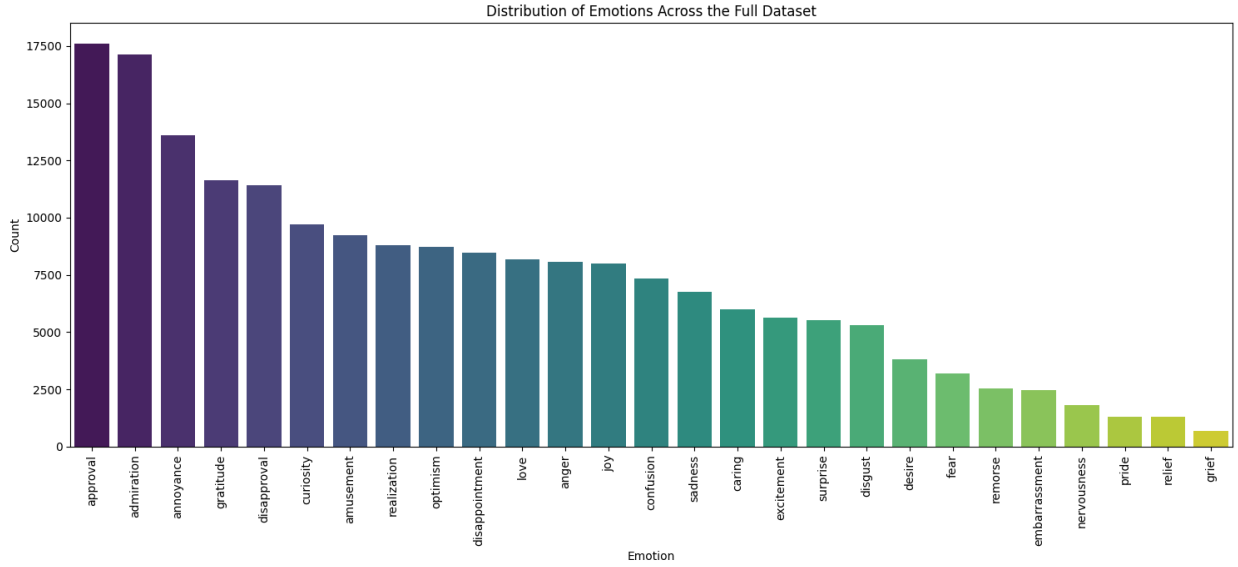
Because our project used an existing Git repository and dataset as a basis, data preprocessing tasks were completed in the research study prior to sharing the actual dataset, and several programmatic preprocessing tasks preexisted in the code.

As described in the original study itself, there are some issues with the GoEmotions dataset that the original research team acknowledge and attempted to account for where possible. The first issue is bias created by sampling all the data from a single platform like Reddit. Reddit itself is not representative of the total population; the users tend to be younger and male. However, having a large, pre-categorized dataset such as this is extremely useful in developing sentiment analysis tools like the ones we described in our proposal, so this data is valuable, data collection biases aside. (Demszky et al. 2020)

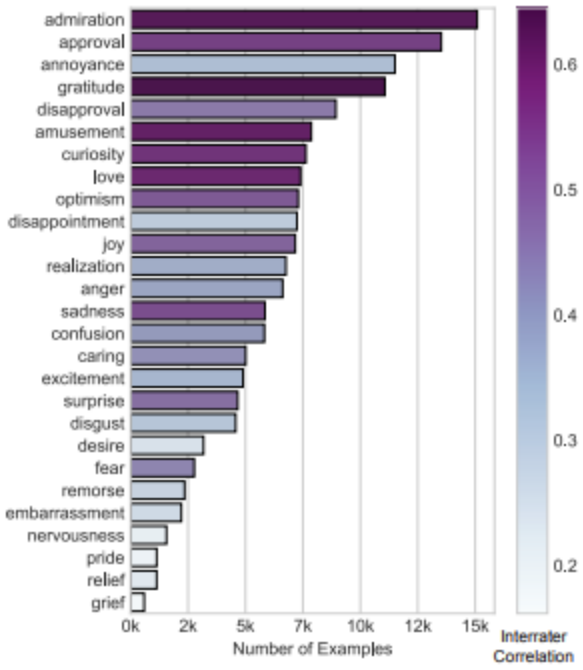
Additionally, there are certain norms on Reddit that differ from those of other communities in general, such as a tendency to use harsh, explicit language, that may not be present in communication in other contexts where such a model might be used. One of the preprocessing tasks completed by the original study was filtering out comments from subreddits that were particularly offensive. As for individual comments, they ran those through a predefined lists of terms to try to remove terms that were particularly inappropriate or harmful and tried to remove content that was bigoted or hateful. They did keep a lot of the swearing since that was important context for negative emotion learning. (Demszky et al. 2020)

Another issue with the data is that it is not reasonable to get an even number of examples of each emotion in each category because emotions themselves do not occur in an equally distributed fashion. This is also complicated by how a single text can be classified into multiple emotions. One mechanism the original study used to help with this was they tried to not include subreddits

that did not show a variety of emotions in their content. (Demszky et al. 2020) However, the categories for representation are still uneven. In this diagram below we generated, it shows certain emotions like admiration and approval have over 17k examples, while rarer emotions like pride, relief, and grief have very few representations in comparison.



There are different types of outliers and inconsistent data that can occur in a dataset such as this. Firstly, the data they collected obviously did not come prelabeled with emotions categories; the researchers themselves had to assign emotions to each comment without input from the comment author on what they meant. The researchers could have different opinions on what emotions were being expressed, or how strong that emotion was. To mitigate this, they had three raters rate each comment, and if they disagreed, additional raters were assigned to further evaluate. If a rater said a comment was too difficult to rate, they removed it from the dataset, taking out one form of outliers. (Demszky et al. 2020)



Credits: (Demszky et al. 2020)

The above chart also includes a distribution representing how much the raters agreed with each other when assigning emotions in the final dataset. The higher interrater correlation demonstrates general agreement, while the lower ones struggled to achieve consistency. This seems to suggest that certain emotions are harder to categorize than others, even for humans.

In addition to categorizing within the specific set of 27 emotions they defined, they also allowed raters to categorize emotions as neutral. In the GoEmotions study, they ultimately excluded neutral from their data analysis since neutral comments were not showing strong emotions.

When dealing with categories with such uneven content or with disagreement on labeling, one way to still gain insight from the lack of data is correlating and weighting the training based on similar data that does have more information.

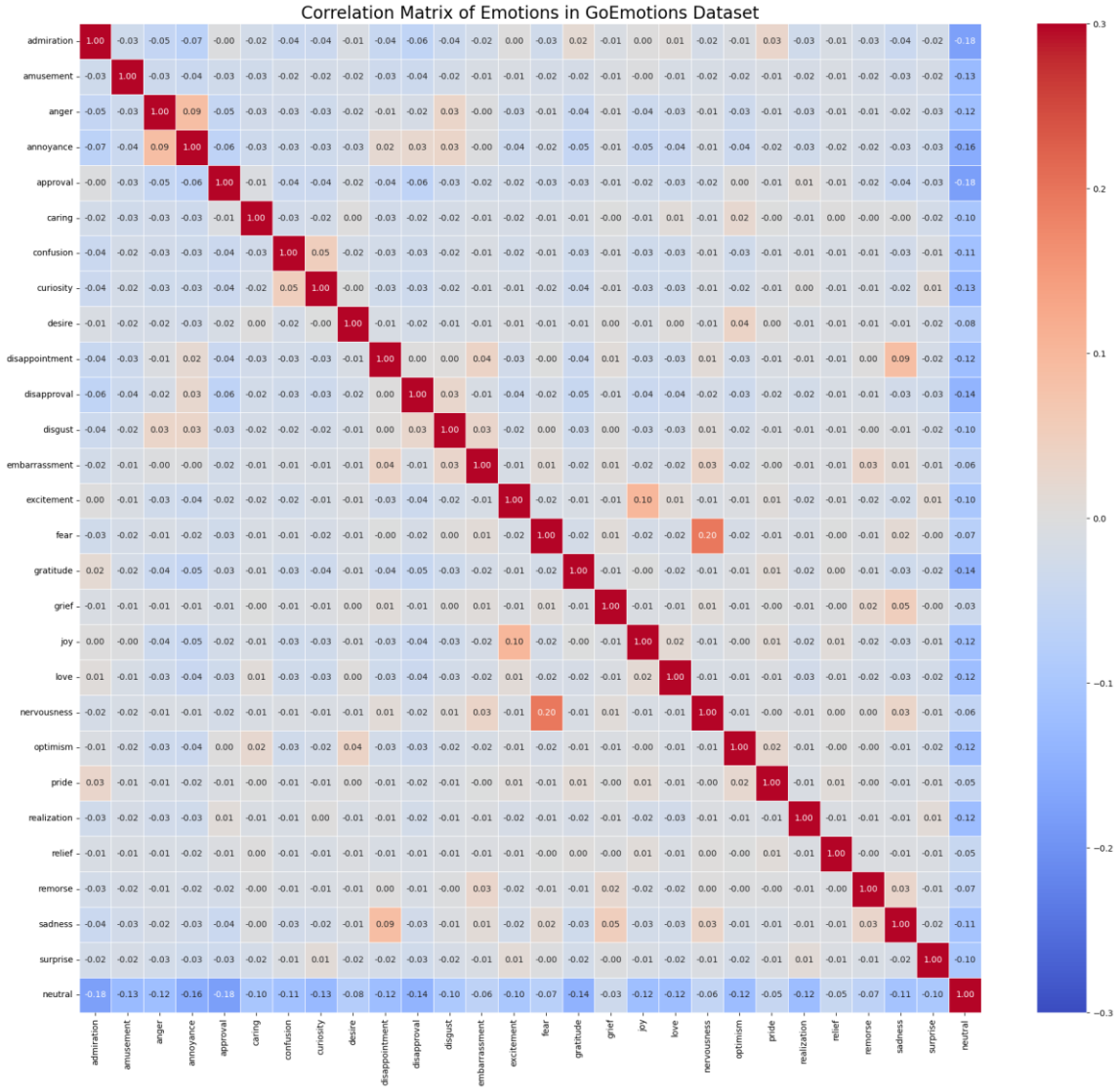
For example, one way the original team handled this was to assign each of the 27 emotions to a higher-level sentiment category. Their classification was as follows:

positive	amusement, excitement, joy, love, desire, optimism, caring, pride, admiration, gratitude, relief, approval
negative	fear, nervousness, remorse, embarrassment, disappointment, sadness, grief, disgust, anger, annoyance, disapproval
ambiguous	realization, surprise, curiosity, confusion

(GoEmotions-Fine-grained emotion classification 2021)

In their model, they included a weighting and bias algorithm that took in this relationship grouping and penalized predictions done for unrelated sentiment groups.

Similarly, they performed a correlation analysis that looked for emotions that seem to be related or occurred together, and that too was information utilized in their weighting and bias algorithm. However, their code did not seem to provide a way to replicate the correlation they loaded into their algorithm, so we reimplemented the correlation analysis using Pandas and demonstrated the output with the following chart, which includes the specific numerical correlation values associated with the correlation.



Finally, some complications in data processing can occur because comments can differ wildly in length, so they performed some data regularization to down-sample to a more consistent token length to aid in learning. (Demszky et al. 2020)

However, we additionally considered exploring the overall distribution of data across different emotion classes and then accounted for the class imbalance through a weighted-loss function based on the inverse of the count of records under each emotion class. This provided us with a better result than one without this weighted-loss function.

6 METHODOLOGY

For the purposes of our project, we took the original GoEmotions project and made some architecture changes in the hopes of improving the metrics.

The original project used a pretrained large BERT model that was further optimized with additional layers (GoEmotions-Fine-grained emotion classification 2021). BERT is a widely used transformer and encoder-based model that excels in natural language processing applications. However, regular BERT is a large, long-running, and very processive-intensive model, and we did not have the resources to run it successfully. Since they were already using BERT, we decided to use a newer, but smaller and more efficient variation of DistilBERT. However, since the DistilBERT model used to be fairly basic, we had to recreate most of the layers that existed in the original model.

First, we imported the pretrained DistilBERT model from HuggingFace, and as in the original code, we allowed the option to use the cased or uncased model as a basis. For our analysis, we used the uncased model like the original GoEmotions project. To work to improve the metrics, we combined the results of the DistilBERT model with that of TextCNN.

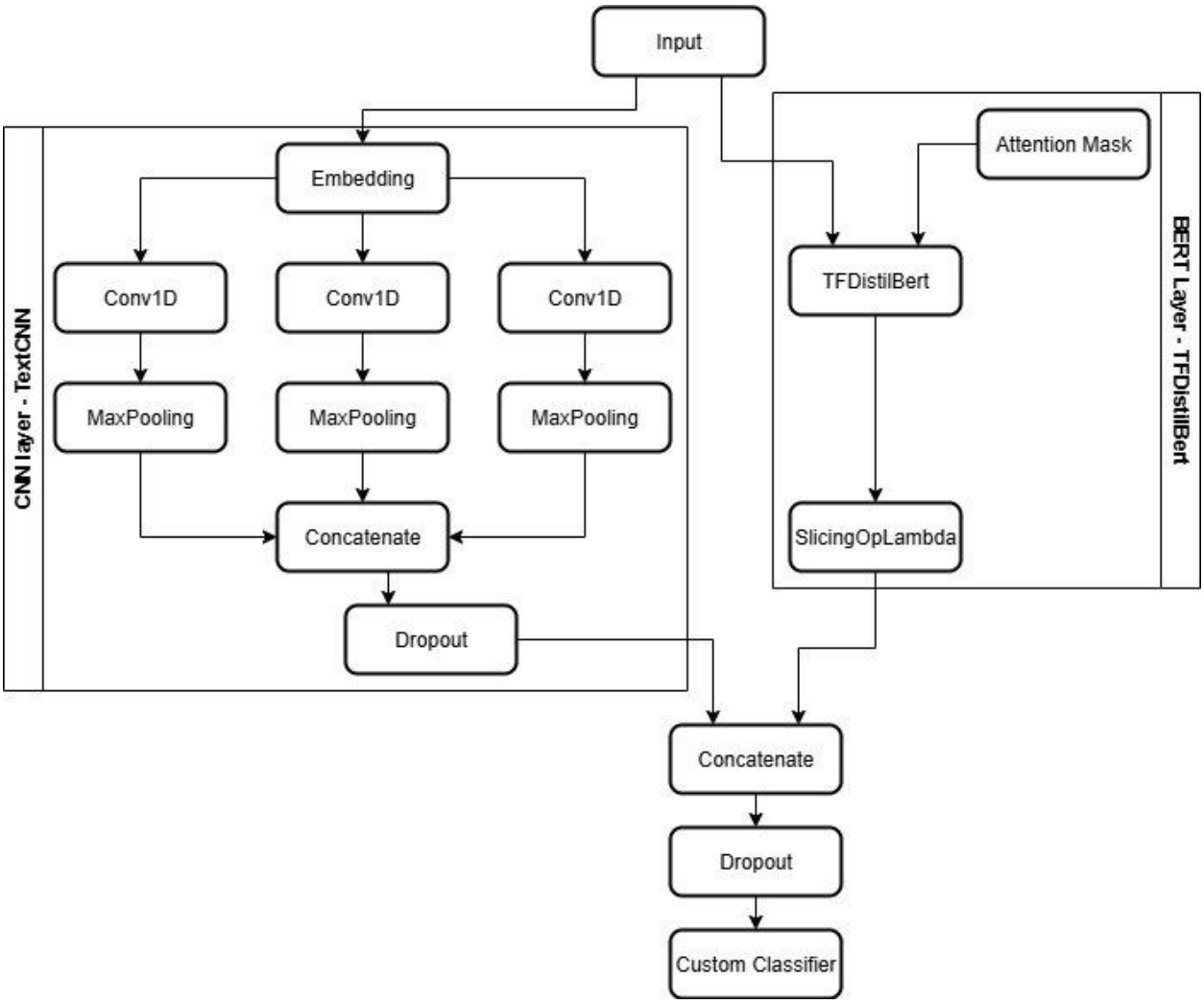
As described in our Related Works section, other studies discuss the benefits of combining BERT with other model types like CNN to improve the learning results. However, those discuss training using one sequential model, where the BERT layers get trained first and are then immediately passed to the next model layer. One such study, Abas et al, illustrates that architecture as follows:

Figure 1: Architecture of the BERT-CNN model

(Abas et al. 2022)

In the interest of trying something more novel, we passed the GoEmotions dataset to the DistilBERT and CNN models separately, then use the outputs of those models as inputs to another model that will combine the data, creating a multi-input one-output architecture. We will use a concatenate layer to merge the feature data from the two sources, pass it through a dropout layer, and pass the results through a custom classifier function.

The architecture of the TextCNN model version we used is inspired by the works of Kim Y where the researchers used CNN models to classify sentiments in texts.(Kim 2014) As we were supposed to create a multi-input model, we passed the tokenized input from the DistilBERT layer even as an input to the TextCNN layer so that they are both tokenized using the same tokenizer and both also have the same sequence lengths within them. The tokenized sentence sequence is then passed through the embedding layer and then through a custom set of filters to fetch local contextual information using CNN. The convoluted sequence is then passed through a MaxPool layer to extract the most important feature as predicted by each filter type. The vectors are then concatenated to extract the most important overall features before being combined with the DistilBERT layer's output as an input to a combined model with a custom classifier.



Final Network Architecture (Simplified)

This combination of the two processing arms of DistilBERT and TextCNN was performed using the Keras Sequential API. Apart from using these two as the independent input arms, we also used some custom functions. They are:

- Custom Classifier Function
- Custom Loss Function
- Custom Learning Rate Function
- Custom Metrics Function

For classification, we used a custom classifier function, CustomClassifierLayer. The original code structured a custom classifier in the appropriate mechanism for the Estimators API, and we had to migrate this to something appropriate for the Keras API while also making the implementation appropriate for our model. In this class, we initialize the output weights to a small number based on the number of labels that exist in the model – the number of emotion categories. We do something similar for bias but initialized to zeroes. It then uses the matmul and biasadd functions to perform the equivalent of a Dense layer functionality, but without utilizing a specific

activation function so that the code has more control over the weights, and we introduce the activation layer. All of this is identical in functionality to the original, we just migrated it to compatible APIs.

Additionally, we perform classification weights, bias, and regularization functionality through our custom loss function, RegularizedClassificationLoss. The original code included the ability to run training for both multi-label and single label training tasks and we preserved that behavior, but for our analysis, we only considered the multilabel results. For multilabel problems, we used sigmoid, as in the original code, since that is the appropriate activation function to use on outputs for multilabel classification. For single label, it would use SoftMax. This is used to calculate the cross-entropy loss, or the difference between actual true labels and what the model predicted. We added a weighted-loss function on top of the base loss to account for the class imbalance within the data set. This weighted-loss function penalizes the model for any wrong predictions for a minority class inversely proportional to the count of the minority class within the dataset.

The loss function also utilizes the relationships between sentiment (emotions and their positive/negative categories) and correlation between emotions, to regularize the output by creating a final loss value. The assumption is that if different emotion categories seem to be similar or related, we would expect the probabilities to be similar, so the results of those calculations should be used to affect the final cross-entropy loss value. The original code also supported using entailment for this, but since DistilBERT was not trained to account for this, reference to entailment were removed from our code.

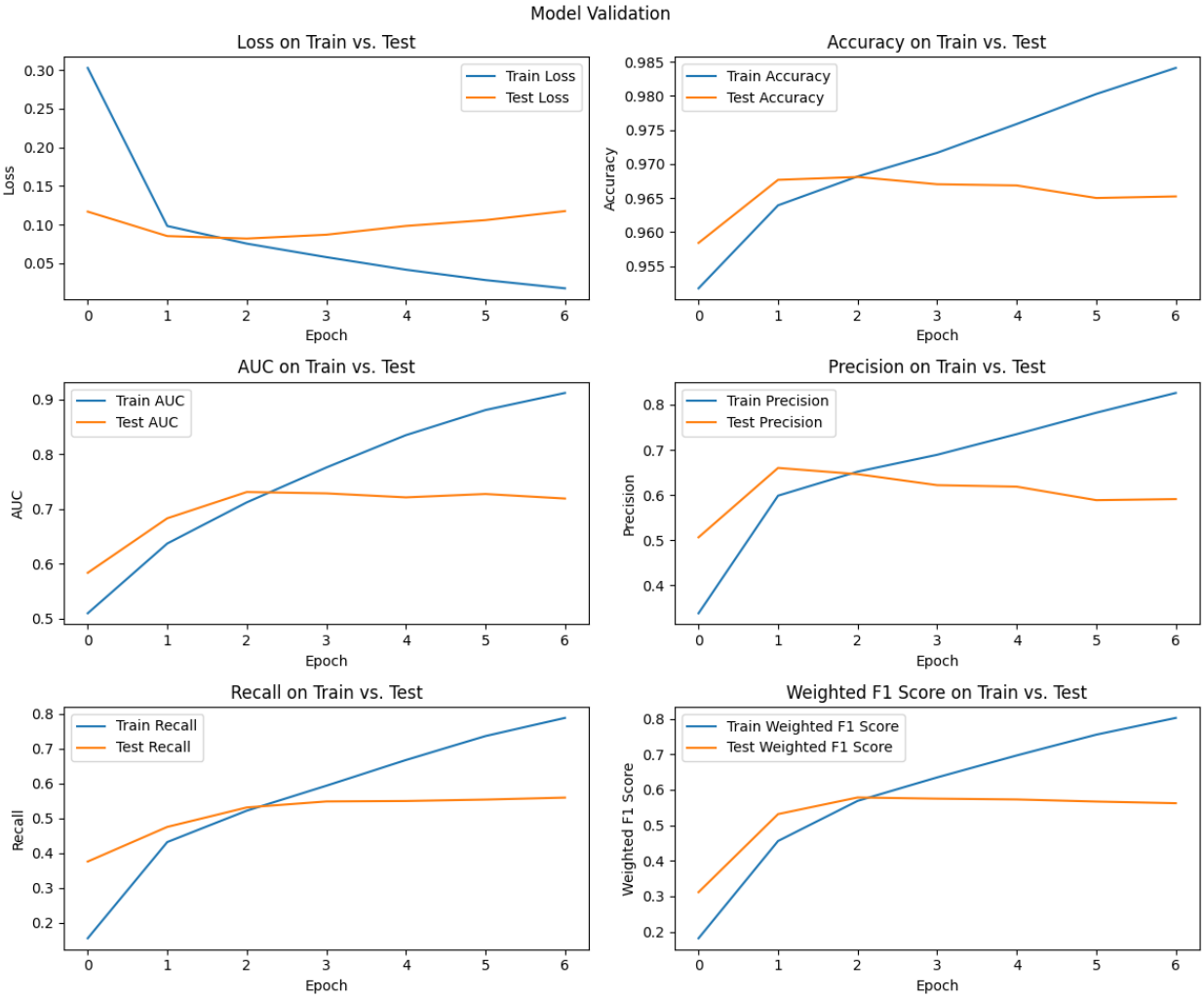
For the optimizer, the original code used a customized version of Adam, but upon examination of their implementation, we realized they were just implementing AdamW, so we swapped to use the AdamW optimizer and set it to use the same parameters the original code was defining. AdamW is particularly useful in models where you want more control over the weight decay and is recommended over Adam for BERT models specifically.

The original model also used a custom learning schedule in their optimizer, which we recreated in the WarmUpAndDecaySchedule class. This implements a linear learning rate schedule for the AdamW optimizer – it first warms up based on the num_warmup_steps parameter passed it, then it decays to zero over the rest of the training process. The goal of this sort of schedule is to enforce learning stability at the start of model training (during the warmup steps), and to encourage the model to actually converge successfully during the decay steps.

Finally, we implemented a custom F1 score metrics function. Standard F1 metrics are better for single-label problems and a weighted F1 score metric is better suited for multi-label class imbalanced cases as a prediction could be positive for multiple classes together. Therefore, a custom metric helps calculate the right F1-score considering the True Positive, True Negative, False Positive, False Negatives across each of the classes. Additionally, having a custom F1-score for an imbalanced dataset allows the measurement of scores across different averaging methods (micro, macro, weighted), accurately providing different information for each averaging method.

The complete labeled dataset was split into 80% for training, 10% for validation and 10% for test data. The model was run with an epoch count of 20 with an early stopping enabled to prevent

overfitting. After the training of the model was completed, it had to predict the emotion classes for a previously unseen test dataset. The results of the model's prediction over test dataset were analyzed to gather better understanding about the model's performance on new data whereas the test and validation loss comparison derived during training was used to understand how well the model learned over epoch.

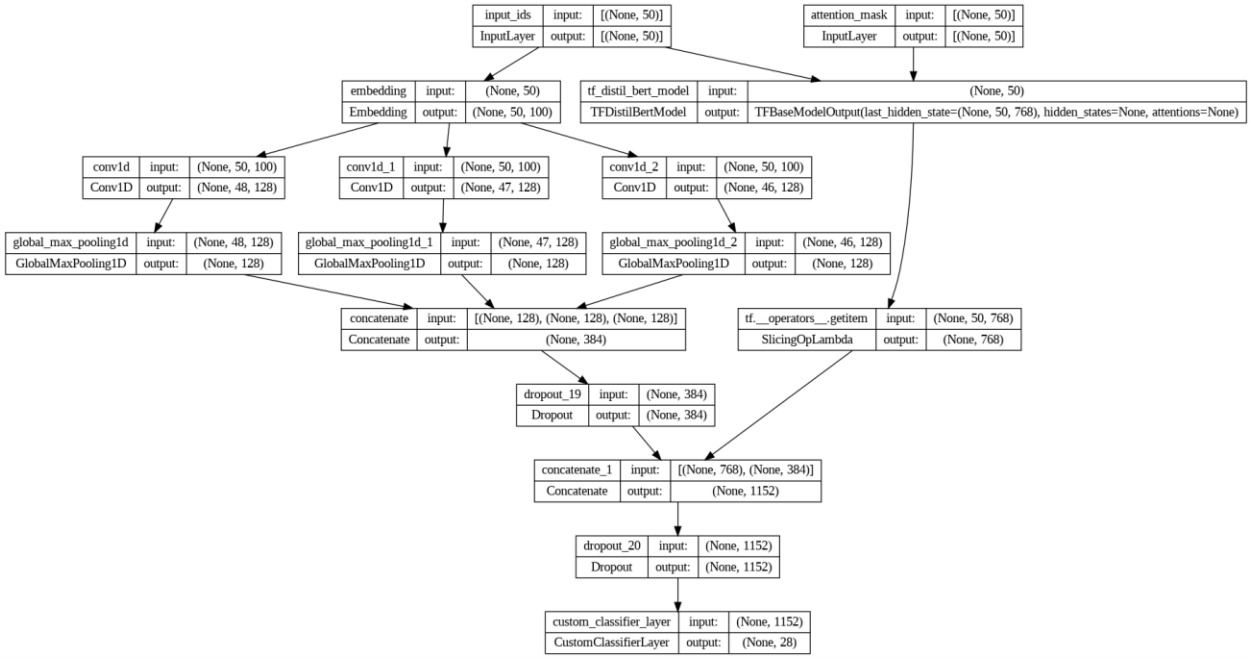


As we can see from the train and validation graphs, the loss was slowly on an increasing trend and that the AUC as well as the weighted F1-scores were plateauing. Due to the early stopping functionality included in the model training steps, the training ends after 7 epochs. While the graphs show a high accuracy, for multi-label models' accuracy is not a good indicator as it cannot measure partial correctness. For example, in this case the high accuracy of 0.965 could be due to the absence of certain classes and not due to the correct prediction of the right class.

The overall reason for choosing a hybrid model using DistilBERT and TextCNN was to fetch the global context through DistilBERT and the local context through TextCNN. This is different

from a standard open-source model, like say, DistilBERTForSentimentAnalysis which cannot be fine-tuned as per the dataset that easily. Considering the complexity of the data that we are using and the data imbalance it has, using a custom model architecture was a better choice.

We used a variety of software to complete these tasks. To pull the original source code for GoEmotions and TextCNN, we used Git, and we also used Git to aid in sharing our code. For writing the Python code we initially used Spyder and Jupyter Notebooks (as part of Anaconda) running locally on our machine but ultimately migrated to running the Jupyter Notebook in Google Colab to take advantage of their cloud resources. To aid in code conversion, understanding, and troubleshooting advice, we used ChatGPT and Gemini, supported by API documentation and other articles found on the internet.



Final Network Architecture

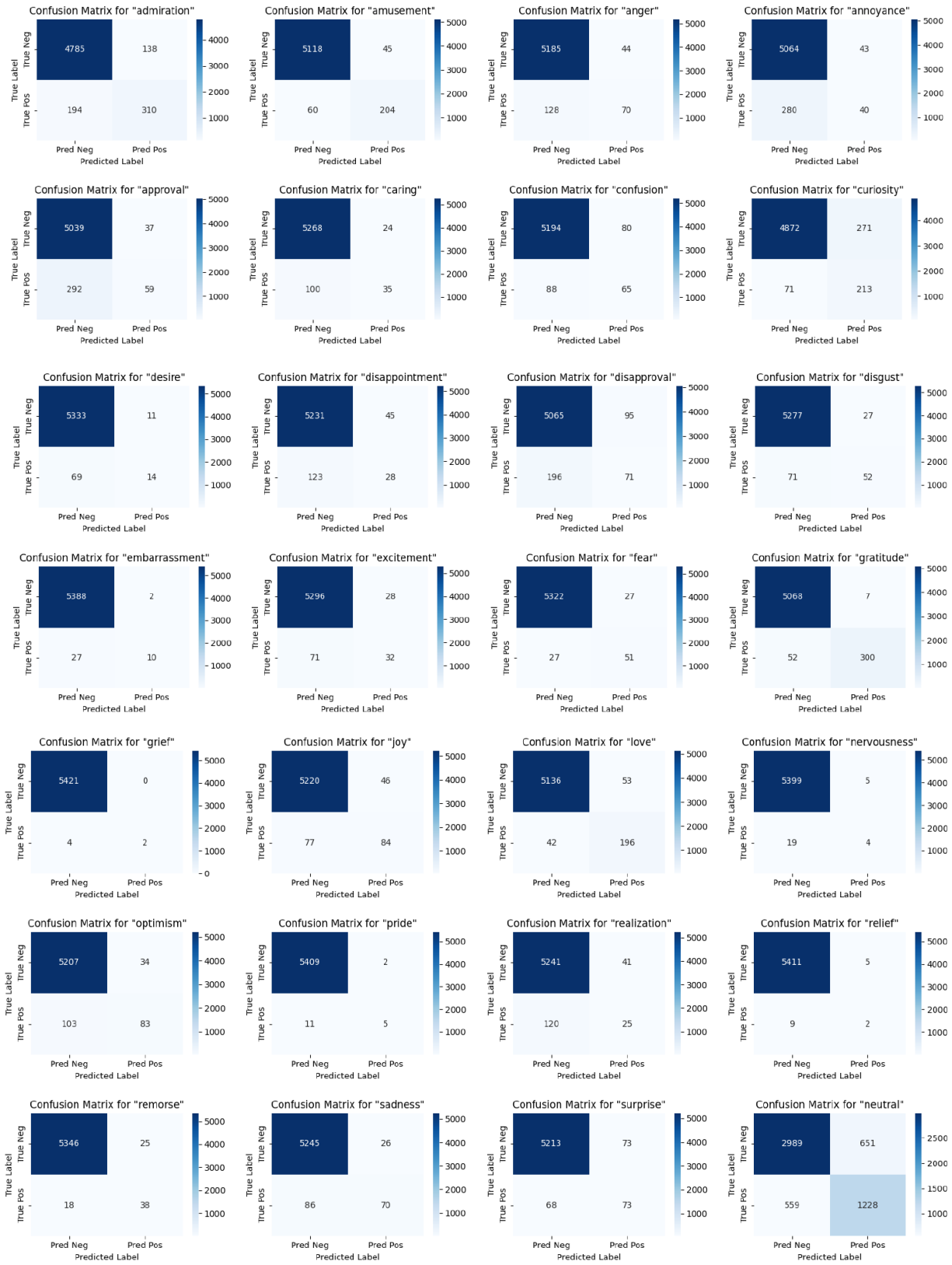
7 RESULTS AND INTERPRETATION

To test the model over an unseen test dataset, we run prediction on the test dataset and measured the following:

- Confusion matrix
- Classification report
- Precision, recall and F1-score on a per emotion class basis
- Jaccard scores
- Model performance summary report

The detailed results for our model over a test dataset are as follows:

- Confusion Matrices for all emotion classes (28):


Confusion Matrix for "admiration"
4785
138
194
310
Confusion Matrix for "amusement"
5118
45
60
204
Confusion Matrix for "anger"
5185
44
128
70
Confusion Matrix for "annoyance"
5064
43
280
40
Confusion Matrix for "approval"
5039
37
292
59
Confusion Matrix for "caring"
5268
24
100
35
Confusion Matrix for "confusion"
5194
80
88
65
Confusion Matrix for "curiosity"
4872
271
71
213
Confusion Matrix for "desire"
5333
11
69
14
Confusion Matrix for "disappointment"
5231
45
123
28
Confusion Matrix for "disapproval"
5065
95
196
71
Confusion Matrix for "disgust"
5277
27
71
52
Confusion Matrix for "embarrassment"
5388
2
27
10
Confusion Matrix for "excitement"
5296
28
71
32
Confusion Matrix for "fear"
5322
27
27
51
Confusion Matrix for "gratitude"
5068
7
52
300
Confusion Matrix for "grief"
5421
0
4
2
Confusion Matrix for "joy"
5220
46
77
84
Confusion Matrix for "love"
5136
53
42
196
Confusion Matrix for "nervousness"
5399
5
19
4
Confusion Matrix for "optimism"
5207
34
103
83
Confusion Matrix for "pride"
5409
2
11
5
Confusion Matrix for "realization"
5241
41
120
25
Confusion Matrix for "relief"
5411
5
9
2
Confusion Matrix for "remorse"
5346
25
18
38
Confusion Matrix for "sadness"
5245
26
86
70
Confusion Matrix for "surprise"
5213
73
68
73
Confusion Matrix for "neutral"
2989
651
559
1228
True Label
True Neg
True Pos
Pred Neg
Pred Pos
Predicted Label

The confusion matrices provide us with detailed information about the model's performance for each emotion class. This helps us visually understand for which emotions the model is a good indicator and for which it isn't. Some of the key findings from the confusion matrices are:

- The model exhibits a strong bias towards 'neutral' class as seen through the high count of False Positive cases whereas many other emotions exhibit a higher volume of False Negative cases.
- Emotion classes like 'annoyance', 'disapproval' and 'caring' show a lot of False Negative cases which shows these were misclassified possibly as 'neutral' instead.
- Emotions that are vastly different in terms of the language patterns used to express them show a strong performance (higher True Positive and True Negative ratio). Emotions like 'gratitude', 'love' and 'amusement' are very different from each other and humans usually use different words to express them. So, there is a possibility that the model could train itself on these emotions due to patterns of words used to express them.
- Many emotions show higher False Negative as compared to False Positive showing instances where the model could not predict them accurately or that the model exhibited conservative prediction.
- A few emotions show higher False Positives as compared to False Negatives and these are the instances where the model preferred to just cast a wide net for these emotions or say predicted them aggressively.

We continued our analysis of the test results through the generation of classification report.

- Classification report:

Classification Report:

	precision	recall	f1-score	support
admiration	0.69	0.62	0.65	504
amusement	0.82	0.77	0.80	264
anger	0.61	0.35	0.45	198
annoyance	0.48	0.12	0.20	320
approval	0.61	0.17	0.26	351
caring	0.59	0.26	0.36	135
confusion	0.45	0.42	0.44	153
curiosity	0.44	0.75	0.55	284
desire	0.56	0.17	0.26	83
disappointment	0.38	0.19	0.25	151
disapproval	0.43	0.27	0.33	267
disgust	0.66	0.42	0.51	123
embarrassment	0.83	0.27	0.41	37
excitement	0.53	0.31	0.39	103
fear	0.65	0.65	0.65	78
gratitude	0.98	0.85	0.91	352
grief	1.00	0.33	0.50	6
joy	0.65	0.52	0.58	161
love	0.79	0.82	0.80	238
nervousness	0.44	0.17	0.25	23
optimism	0.71	0.45	0.55	186
pride	0.71	0.31	0.43	16
realization	0.38	0.17	0.24	145
relief	0.29	0.18	0.22	11
remorse	0.60	0.68	0.64	56
sadness	0.73	0.45	0.56	156
surprise	0.50	0.52	0.51	141
neutral	0.65	0.69	0.67	1787
micro avg	0.64	0.53	0.58	6329
macro avg	0.61	0.42	0.48	6329
weighted avg	0.63	0.53	0.56	6329
samples avg	0.56	0.56	0.55	6329

The classification report provides us with an easier representation of performance across different emotion classes. Through this we can divide the performance of the model into different categories.

F1-Score ≥ 0.80 (Strong Performance)

- Gratitude (0.91)
- Amusement (0.80)
- Love (0.80)

0.60 ≤ F1-Score < 0.80 (Good Performance)

- Neutral (0.67)
- Admiration (0.65)
- Fear (0.65)
- Remorse (0.64)

0.40 ≤ F1-Score < 0.60 (Moderate Performance)

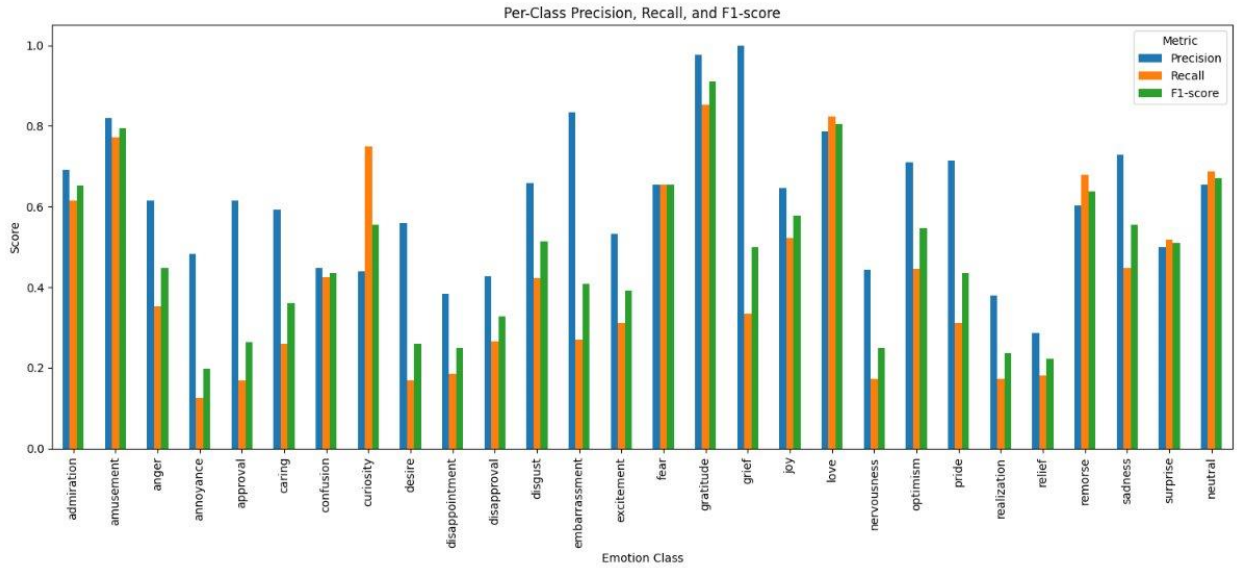
- Joy (0.58)
- Sadness (0.56)
- Optimism (0.55)
- Curiosity (0.55)
- Disgust (0.51)
- Surprise (0.51)
- Grief (0.50)
- Anger (0.45)
- Confusion (0.44)
- Pride (0.43)
- Embarrassment (0.41)

F1-Score < 0.40 (Needs Improvement)

- Excitement (0.39)
- Caring (0.36)
- Disapproval (0.33)
- Approval (0.26)
- Desire (0.26)
- Disappointment (0.25)
- Nervousness (0.25)
- Realization (0.24)
- Relief (0.22)
- Annoyance (0.20)

The performance of the different emotions across the metrics of precision, recall and F1-score can be graphically shown through a per emotion class basis bar graph.

- Precision, recall and F1-score on a per emotion class basis



We also tried to utilize the metrics usually used in CNN models to see if they could help us gather any additional insights. One of the metrics usually used in CNN models is the Jaccard score.

- Jaccard Score:

```
Calculating Jaccard Score:
Jaccard Score (Samples Average): 0.5195
Jaccard Score (Weighted Average): 0.4110
Jaccard Score (Macro Average): 0.3358
```

How we interpret these scores is given below:

- Jaccard Score (Samples Average): This provides us with the average overlap between true and predicted labels. As seen here, a 0.5195 score reflects that 51.95% of the true labels and predicted labels overlapped with each other or in other words, overall, 51.95% of true labels were predicted correctly.
- Jaccard Score (Weighted Average): This takes into consideration the frequency of true labels associated with each emotion to calculate a weighted average of true labels that got predicted. The score of 0.4110 or 41.10% shows that when frequency of true labels is added into the mix, the percentage decreases and shows that there are some class labels which perform poor which is reflected through this weighted average.
- Jaccard Score (Macro Average): This calculates the Jaccard score for each individual label (emotion) and then calculates the unweighted average. This metric treats all labels equally and helps highlight performance in less frequent classes. The Jaccard score of 0.3358 or 33.58% shows that some of the classes fare poorly to bring the overall average down. This is expected in a dataset which has data imbalance and has classes that have performed poorly.

We can now combine all of the analysis done to create an overall performance summary.

- Model performance summary:

```
--- Model Performance Summary ---

Emotions the model understands better (higher F1-score):
            Precision    Recall  F1-score
gratitude    0.977199  0.852273   0.91047
love         0.787149  0.823529  0.804928
amusement    0.819277  0.772727  0.795322
neutral      0.653539  0.687185   0.66994
fear         0.653846  0.653846  0.653846
admiration   0.691964  0.615079  0.651261
remorse      0.603175  0.678571  0.638655
joy          0.646154  0.521739   0.57732
sadness      0.729167  0.448718  0.555556
curiosity    0.440083      0.75  0.554688

Emotions the model struggles with (lower F1-score):
              Precision    Recall  F1-score
excitement     0.533333   0.31068  0.392638
caring          0.59322  0.259259  0.360825
disapproval    0.427711  0.265918  0.327945
approval       0.614583  0.168091  0.263982
desire             0.56  0.168675  0.259259
nervousness    0.444444  0.173913      0.25
disappointment 0.383562   0.18543      0.25
realization    0.378788  0.172414  0.236967
relief         0.285714  0.181818  0.222222
annoyance      0.481928     0.125  0.198511
```

Some of the key findings are:

- Best performance - gratitude, love, and amusement (higher F1-scores).
- Poor performance – realization, relief, and annoyance (lower F1-scores).
- When looked at the performance all low F1-score emotions, we observe that all of them have poor recall.
- When we look at the confusion matrix for such examples, we see that the count of False Negative is higher than True Positive for such classes showing that the model is unable to generalize about the emotion from the limited data available for the give emotion.
- Thus, we can say that a low count of data examples is one of the possible reasons behind lower F1-scores for emotions like nervousness, relief, caring where the model is rarely able to correctly predict the emotion.
- In some emotions there are some trade-offs observed - "curiosity" has high recall but lower precision, while "approval" has high precision but low recall.
- The "neutral" emotion has a moderate F1-score when we look at only F1-score but if we look at the confusion matrix, we find there are a lot of instances of False Positive (compared to False Negatives), showing the model is biased towards ‘neutral’ emotion.

- We also find that just precision, recall and F1-scores are not enough but a true picture comes up when confusion matrix is used along with it. For example, 'grief' has an F1-score of 0.50 and precision of 1.00. But when we look at the confusion matrix, we find that out of 6 examples only 2 got rightly categorized. Thus, the high precision and F1-score numbers are due to small sample size and not an actual reflection of the model's strength in measuring emotions for classes with low example count.

Additionally, we tried to compare the model's findings with the interrater correlation as present in the original research project to measure if the model performs similar to human raters. The interrater correlation values is a measure of the agreement level for each emotion across the different raters who rated texts for those emotions.

- Comparison between interrater correlation vs model performance:

For interrater correlation, please refer to section Data Preprocessing

Emotion	**Sample Size**	**Interrater Correlation**	**Model Performance**
Admiration	Large	High	Good
Annoyance	Large	Low	Low
Fear	Small	Medium	Good
Nervousness	Small	Low	Low
Relief	Small	Low	Low

Overall, we can say that the model is working as per the given dataset and is able to predict emotions based on data frequency and interrater correlation. While interrater correlation is not a value directly used in the dataset, it does represent the spread of how humans struggled to categorized certain samples and caused a spread of labels, so it makes sense the model would pick up on those uncertainties too.

8 DISCUSSION OF RESULTS

The overall goal of our project was to improve upon the work of the existing GoEmotions project and create something that could perform sentiment analysis – specifically multi-label emotion classification – better than the original. In general, we succeeded, but not to the extent that we hoped. Our F1 score of .48 versus theirs of .46 is a minor improvement.

Emotion	Precision	Recall	F1
admiration	0.53	0.83	0.65
amusement	0.70	0.94	0.80
anger	0.36	0.66	0.47
annoyance	0.24	0.63	0.34
approval	0.26	0.57	0.36
caring	0.30	0.56	0.39
confusion	0.24	0.76	0.37
curiosity	0.40	0.84	0.54
desire	0.43	0.59	0.49
disappointment	0.19	0.52	0.28
disapproval	0.29	0.61	0.39
disgust	0.34	0.66	0.45
embarrassment	0.39	0.49	0.43
excitement	0.26	0.52	0.34
fear	0.46	0.85	0.60
gratitude	0.79	0.95	0.86
grief	0.00	0.00	0.00
joy	0.39	0.73	0.51
love	0.68	0.92	0.78
nervousness	0.28	0.48	0.35
neutral	0.56	0.84	0.68
optimism	0.41	0.69	0.51
pride	0.67	0.25	0.36
realization	0.16	0.29	0.21
relief	0.50	0.09	0.15
remorse	0.53	0.88	0.66
sadness	0.38	0.71	0.49
surprise	0.40	0.66	0.50
macro-average	**0.40**	**0.63**	**0.46**
std	**0.18**	**0.24**	**0.19**

Table 4: Results based on GoEmotions taxonomy.

Classification Report:

	precision	recall	f1-score	support
admiration	0.69	0.62	0.65	504
amusement	0.82	0.77	0.80	264
anger	0.61	0.35	0.45	198
annoyance	0.48	0.12	0.20	320
approval	0.61	0.17	0.26	351
caring	0.59	0.26	0.36	135
confusion	0.45	0.42	0.44	153
curiosity	0.44	0.75	0.55	284
desire	0.56	0.17	0.26	83
disappointment	0.38	0.19	0.25	151
disapproval	0.43	0.27	0.33	267
disgust	0.66	0.42	0.51	123
embarrassment	0.83	0.27	0.41	37
excitement	0.53	0.31	0.39	103
fear	0.65	0.65	0.65	78
gratitude	0.98	0.85	0.91	352
grief	1.00	0.33	0.50	6
joy	0.65	0.52	0.58	161
love	0.79	0.82	0.80	238
nervousness	0.44	0.17	0.25	23
optimism	0.71	0.45	0.55	186
pride	0.71	0.31	0.43	16
realization	0.38	0.17	0.24	145
relief	0.29	0.18	0.22	11
remorse	0.60	0.68	0.64	56
sadness	0.73	0.45	0.56	156
surprise	0.50	0.52	0.51	141
neutral	0.65	0.69	0.67	1787
micro avg	0.64	0.53	0.58	6329
macro avg	0.61	0.42	0.48	6329
weighted avg	0.63	0.53	0.56	6329
samples avg	0.56	0.56	0.55	6329

From the comparison above, we can conclude the following:

- A slight improvement in F1-score is observed over the original research project.
- The hybrid architecture results in a model with higher precision and lower recall than the original model.
- The model performs best on emotions like gratitude, love, amusement, fear, and admiration, as indicated by their higher F1-scores.
- The model struggles significantly with emotions such as annoyance, relief, realization, disappointment, nervousness which exhibit lower F1-scores.
- Emotions with very low F1-scores, like nervousness and relief, likely suffer from low True Positives and False Positives due to limited data support and the model's difficulty in predicting these rare instances.

- Conservative prediction (high precision, low recall): Model is conservative is predicting emotions like sadness.
- Aggressive prediction (low precision, high recall): For emotions like curiosity, the model is more aggressive in predicting the emption.
- The neutral emotion has a good F1-score with balanced precision and recall but when we look at the overall confusion matrix for neutral, we can see it has a high False Positives and False Negatives showcasing the confusion the model experiences at times, confusing neutral with some other emotion.

Some of the next steps that we think could improve the model significantly are:

- Address the low-performing emotion classes (e.g., annoyance, relief) by exploring data augmentation techniques or collecting more labeled data for these specific emotions.
- Investigate potential class imbalance issues and consider resampling techniques (oversampling minority classes, undersampling majority classes) or using different loss functions to improve performance on underrepresented or challenging emotions.

We framed our project from the perspective of using it in AI therapy contexts. For sentiment analysis used for AI therapy or customer service chatbots, higher precision scores are preferred over recall. In our model, we improved the precision over the GoEmotions values but decreased the recall. Overall, we consider our model an improvement.

The ability to identify emotion from written text is extremely important in just about any aspect of AI that involves natural language processing, so any advancements in this area would be to the benefit of many applications like the ones mentioned in the Importance section. Fine-grained sentiment analysis by specific emotion versus simple positive versus negative emotion classification, is an extremely difficult problem, and these nuances are important when it comes to AI working with real people, especially in contexts as sensitive as therapy.

There are several limitations present in this project. One of the limitations of this project is that it only identifies emotions in 28 different categories. While that seems a lot, there is still more nuance that could be gained from having more emotion categories. Another is that all the dataset is only in the English language, and for speakers of other languages to benefit, datasets in those languages would have to be collected as appropriately. Additionally, outside of the language, even the context in which the data was collected could be a limitation. This data was pulled from Reddit which has a particular sort of internet culture and demographic, but how well the lessons pulled from there can relate to English speakers in general, particularly English speakers of other demographics or not involved in social media, is hard to say.

The future work recommended for this would be to look for ways to improve identification in the weaker performing classes. We attempted to improve performance entirely by algorithmic means, but perhaps it would have been better to look for ways to improve the data itself. One mechanism is to collect more data for the weaker performing classes. Another would be to manipulate the data we already had, which is discussed in the Your Feedback section.

The biggest limitation to our classifier is its inability to consistently handle different labels; some categories it does better, some it does worse. The important thing would be to identify which categories it does well in and figure out how to improve the metrics on the categories that are weak.

9 YOUR FEEDBACK

Our project was based around the idea of taking the existing work done by the GoEmotions team, optimizing it, and adding specific new architecture pieces. While initially it seemed like a good idea -- building directly from their code seemed a great way to compare apples to apples -- but doing so introduced a lot of complications to our project. GoEmotions has not had any meaningful updates to their codebase in approximately four years, which means it was using very out-of-date modules. Most problematic is that the code depended on TensorFlow 1 deprecated and defunct APIs, requiring a bunch of refactoring to be able to migrate to TensorFlow 2 and Keras. One particular defunct API it used, Estimators, was structured using a pattern completely unlike Keras, which made completing the refactor particularly difficult.

An additional complication is that the GoEmotions code was designed to be run using a Linux Bash command line, and both of us have Windows machines. While there are ways to get Bash running on Windows, both of us going through that set-up and disrupting our machines to that extent did not seem overly reasonable and likely to introduce problems where inconsistent machine configurations could cause the project to not work the same on one of our computers versus the other. We first tried a workaround where we simulated the passing of the command line arguments using a Python file, but the multi-file architecture of the original GoEmotions was still confusing, so we ultimately opted to migrate the whole thing to a Jupyter Notebook. This fortunately made the project a lot easier to run and to develop, but it does introduce a bit of weirdness in the code where configuration is still being set by what should be command line config flags, but the actual command line processing is turned off since it would be impossible to functioning within a Jupyter Notebook. The default values for the flags are the actual values.
In retrospect, it would have been easier to use the GoEmotions dataset and their implementation as inspiration, not a starting point for our code.

Another potential improvement is that we could have done more manipulation of the GoEmotions dataset to help with training, specifically around dealing with the uneven data spread of the emotion categories. Originally, we saw that GoEmotions implemented correlation and sentiment grouping code that we thought would effectively handle the issue, which was supported by (Wang et al. 2025) as a valid option. Unfortunately, that did not seem to be enough. We introduced our own additional weights and bias code, but we may have gotten better results using other strategies. One potential strategy would have been data augmentation techniques, described in (Wang et al. 2025) as rearranging the tokens in the existing samples of the rarer classes to create new ones the model could learn from. Another strategy is we could have performed our own data labeling and rating tasks, filling in classes where the original raters failed to categorize.

10 REFERENCES

- Demszky, Dorottya, Dana Movshovitz-Attias, Jeongwoo Ko, Alan Cowen, Gaurav Nemade, and Sujith Ravi. 'GoEmotions: A Dataset of Fine-Grained Emotions', 5 2020. http://arxiv.org/abs/2005.00547.

- Sanh, Victor, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 'DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter', 2019. https://github.com/huggingface/transformers.

- Abas, Ahmed R., Ibrahim Elhenawy, Mahinda Zidan, and Mahmoud Othman. 'Bert-CNN: A Deep Learning Model for Detecting Emotions from Text'. *Computers, Materials and Continua* 71 (2022): 2943–61. https://doi.org/10.32604/cmc.2022.021671.

- Guo, Yuan, and Hongmei Li. 'Research on Sentiment Analysis of Online Course Evaluation Based on EN- BERT-CNN'. In *Proceedings of the 2024 8th International Conference on Electronic Information Technology and Computer Engineering*, 65–68. ACM, 10 2024. https://doi.org/10.1145/3711129.3711142.

- Duong, Chi Thang, Remi Lebret, and Karl Aberercole. 'Multimodal Classification for Analysing Social Media', 2017.

- Tan, Kian Long, Chin Poo Lee, Kalaiarasi Sonai Muthu Anbananthen, and Kian Ming Lim. 'RoBERTa-LSTM: A Hybrid Model for Sentiment Analysis With Transformer and Recurrent Neural Network'. *IEEE Access* 10 (2022): 21517–25. https://doi.org/10.1109/ACCESS.2022.3152828.

- Kokab, Sayyida Tabinda, Sohail Asghar, and Shehneela Naz. 'Transformer-Based Deep Learning Models for the Sentiment Analysis of Social Media Data'. *Array* 14 (7 2022). https://doi.org/10.1016/j.array.2022.100157.

- Wang, Shuo, Yuqi Lu, Ruijue Luo, Boxuan Li, Yingjie Zhang, and Liang Xiao. 'A BERT-Based Sentiment Analysis Model for Depressive Text'. In *Proceedings of the 2025 International Conference on Generative Artificial Intelligence and Digital Media*, 76–83. ACM, 3 2025. https://doi.org/10.1145/3734921.3734934.

- Kim, Yoon. 'Convolutional Neural Networks for Sentence Classification', 8 2014. http://arxiv.org/abs/1408.5882.

- GoEmotions data set: https://www.kaggle.com/datasets/debarshichanda/goemotions Downloaded June 10, 2025

- GoEmotions source code: https://github.com/google-research/google-research/tree/master/goemotions Downloaded June 10, 2025

- TextCNN: https://github.com/delldu/TextCNN Downloaded July 11, 2025

- HuggingFace documentation: https://huggingface.co/docs/transformers/en/model_doc/distilbert

- Keras documentation: https://www.tensorflow.org/tutorials/estimator/keras_model_to_estimator https://www.tensorflow.org/guide/migrate/migrating_estimator

- F1 Score in Machine Learning. 2025: https://www.futurense.com/Uni-Blog/F1-Score-Machine-Learning.

- Classification Metrics Guide: https://www.evidentlyai.com/classification-metrics/accuracy-precision-recall

- GoEmotions emotions via emojis: https://neurohive.io/en/datasets/go-emotions-google-ai-dataset-for-sentiment-analysis/#:~:text=31%20October%202021,as%20improve%20customer%20support%20services
- Metrics for Sentiment Analysis: https://www.getfocal.co/post/top-7-metrics-to-evaluate-sentiment-analysis-models