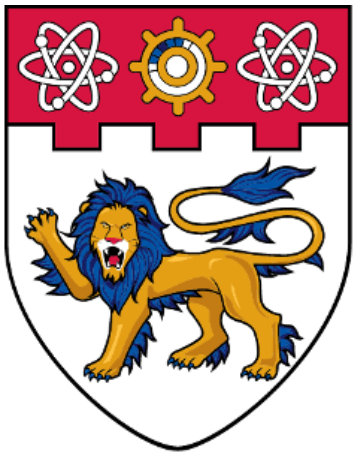


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

SC4001 Neural Networks & Deep Learning

Name of Group Members:
DANIEL LI RUNZE (U2121848H)
SINGH DHRUV (U2121340G)
VINH KHAI TRUONG(U2020331A)

Page Count: 10 (excluding table of contents, references, appendix)
Submitted on: 10 November 2023

Table of Contents

Table of Contents

Table of Contents 2
Introduction 4
Deconstructing the Problem: 4
Existing Methodologies 4
  Convolutional Neural Network 4
  Recurrent Convolutional Neural Network 5
Experiment Setup and Models 5
  Dataset 5
  Exploratory Data Analysis 6
  Text Preprocessing 6
    Regular Expression Parsing: 7
    Tokenization and Lemmatization: 7
    BERT Sub-word Tokenization: 7
    Embedding Extraction: 7
    Padding to Fixed Size: 7
    Vectorization and Encoding of Sentiment Labels: 7
    Handling Class Imbalance: 7
Modeling 8
  Dual Chanel Convolutional Neural Network - Bidirectional LSTM 8
    Model Overview 9
    Inner Workings 9
    Effectiveness for TER 10
Results and discussion 10
  Metrics Considerations 10
Discussion and future work 12
    Model Results 12
    Alternate Results Methodologies 12
    Color wheel emotions visualization 13
References 14
Appendix A - EDA Images 15

Appendix B – Hypothetical combined model with word-wise embeddings: 19
Data flow during inference and training phase: 20
TER Specific Effectiveness 20
Appendix C – Metrics Used 21
Appendix D - Bayesian Optimization 22

Introduction

Text Emotion Recognition (TER) holds great importance in bridging the relations between expressions and intentions through the analysis of semantic, syntactic, and pragmatic cues expressed in digital formats, constituting a critical pillar in the domain of NLP and NLU,

TER is often hindered by the complexities of linguistic techniques, and classification tasks on multi-class problems are salient examples of the degree of inaccuracy involved with accurately deciphering connotations and denotations of lexes.

To explore and understand the nuances of the domain of NLP and NLU, we wish to utilize deep learning methods to identify local emotions expressed by words, and global emotions represented by complete sentences.

Deconstructing the Problem:

Tasks around TER are aggregates of two major components – local and global emotions. These components require classifying word-emotion associations to understand local emotions, and contextualizing sentences and passages to identify global emotions.

Individually, words can have significant emotional connotations that contribute to the overall sentiment or emotional tone of the sentence or passage, and we can create clear emotional associations by individually evaluating words. For example, words like "happy," "sad," "angry," or "excited" have clear emotional associations that can be classified.

Conversely, global emotions require that we consider the contextual meaning of words to evaluate emotional meanings conveyed by collections of words, as sentences, passages, or quotes. This task evolves beyond simple mappings between words and denotations and elicits the requirement for deep learning to derive syllogisms which define constructs for interpreting interactions between words and contexts.

Existing Methodologies

Text emotion recognition (TER) involves predicting emotions expressed in text and documents. Existing algorithms find emotion by learning the relationships of words using recurrent neural networks (RNN) or convolutional neural networks (CNN). RNN and CNN capture local information (i.e., emotion of words) and ignore the global information (i.e., emotion of sentence).

Convolutional Neural Network

Convolutional neural networks (CNN) utilize layers with convolving filters that are applied to *local* features (LeCun et al., 1998). Popular CNN architectures typically used for computer vision tasks include LeNet, AlexNet, VGG, GoogLeNet, ResNet, and many others (Siddharth Das,

2017). These architectures have varying depth, complexity and performance, and are used for different tasks.

While application to natural language processing is similar to computer vision tasks in that the filters are slided over a word vector instead of pixels in an image, the first layer of learned filters will perform a similar task as a feature capturing n-gram.

In the research paper titled “Convolutional Neural Networks for Sentence Classification” by Yoon Kim, the model architecture used involved a word vectors representation that is mapped into multiple convolutional layers and feed through max-over-time pooling before a final dropout and softmax output. Interesting variations analysed in the paper included performing a standard convolution on a single random, static and learning word vector channels, as well as a depthwise convolution on a combination of static and non-static channels. The results were promising despite a simple architecture and little hyperparameter tuning (Kim, 2014). However, the limitations of CNN is evident in its localisation of features.

Recurrent Convolutional Neural Network

Recurrent Neural Networks (RNN) perform sequential analysis, relying on a hidden layer with a fixed size to store previous information. This memory capability, behaving as a sliding context window, allows the model to utilize the global context and semantics of entire sentences.

However, this model is prone to biasedness where more emphasis is placed on later words, and thus the research paper titled “Recurrent Convolutional Neural Networks for Text Classification” (Lai et al., 2015) aims to combine RNN with an unbiased CNN.

By considering the context of the words before and after and using their information in the convolutional layer together with its own word embeddings, the results is a model that consistently performs better (96.49% accuracy) on global information than CNN across all context window sizes.

Experiment Setup and Models

Dataset

We decided to use a Kaggle dataset (link) containing 40,000 tweets labeled with 1 of 13 sentiments. The reason of choice is because tweets have character limits which will allow us to have constraints on sentence length and thus a denser word vector.

Despite so, in our data analysis, we identified that the tweets had varying lengths, special characters, and writing styles, and in an untreated form could affect our modeling and decision making process. Hence our data preprocessing decisions are explained in greater details.

Exploratory Data Analysis

In order to maximize our page utility, the image outputs of the EDA process are shown in Appendix A, with a brief results summary in the below table:

Analysis	Result	Inference
Word Cloud	High numbers of stop words, links, and other non-character inputs	The data contains a high prevalence of non-informative elements like links and stop words, necessitating thorough cleaning and preprocessing.
Sentiment Distributions	High level of class imbalance between different sentiments.	The significant imbalance across different emotion classes may lead to biases in the model.
Ngram Plots	Frequent occurrence of specific n-grams associated with specific emotions.	Common n-grams indicate expressions or phrases correlated with sentiment trends.
Tweet Content Distributions	Varying lengths of tweets, with an average size of 8 words per tweet.	Varying tweet lengths require that our model should be able to generate inferences from both short and long text inputs effectively.
Latent Dirichlet Allocation (LDA)	Distinct topic clusters related to different emotional expressions.	The presence of clear topic clusters in LDA again signals to words and phrases with a high emotional valence.

Text Preprocessing

Our text preprocessing pipeline was designed with two objectives – first, reconcile the shortcomings in the data highlighted by our EDA, and second, reformat the data to be ready for machine learning models.

Regular Expression Parsing:

The code utilizes regular expressions to clean tweets by removing URLs, stock tickers, usernames, and profile mentions. This is based on the understanding that while certain special characters carry semantic meaning, others, such as those mentioned, may not contribute meaningfully to sentiment analysis and can introduce noise into the data.

Tokenization and Lemmatization:

The text is tokenized using the NLTK's word_tokenize method, and words are then lemmatized to their base or dictionary form. This reduces the variability of the language and helps in normalizing the text for better generalization. The lemmatization process involves reducing words to their lemma or canonical form. For example, "running" would be lemmatized to "run". Stopwords from the NLTK's predefined list are also removed as they generally do not contain important meaning and are quite common across all types of texts.

BERT Sub-word Tokenization:

The BertTokenizer from Hugging Face's transformers library is employed to tokenize the cleaned and lemmatized text into subwords or tokens. BERT's tokenizer breaks words into smaller units (subwords) based on a trained vocabulary, which helps in handling out-of-vocabulary words by breaking them down into known subwords, thus leveraging the benefits of both word-level and character-level representations.

Embedding Extraction:

For each tokenized word, BERT is used to obtain a contextualized embedding. The embeddings for subwords of a single word are summed to get a single representation of the word. This step captures the context around each word in a high-dimensional space, facilitating the model's ability to discern nuanced sentiment.

Padding to Fixed Size:

Since neural networks require inputs of a consistent shape, the variable-length sequences of embeddings are padded to a fixed size. This is accomplished by appending zero vectors to the sequences until they reach the maximum sequence length found in the dataset.

Vectorization and Encoding of Sentiment Labels:

The sentiment labels are one-hot encoded using Scikit-learn's OneHotEncoder, which transforms categorical labels into a binary matrix representation suitable for training neural networks.

Handling Class Imbalance:

Class weights are computed using scikit-learn’s compute weights method and passed into the Cross Entropy Loss criterion along with a label smoothing of 0.5. Additionally, we adopt

stratified-K-folds to maintain the ratios of the sentiment classes, ensuring the model does not become biased towards the majority class.

Modeling

Dual Chanel Convolutional Neural Network - Bidirectional LSTM

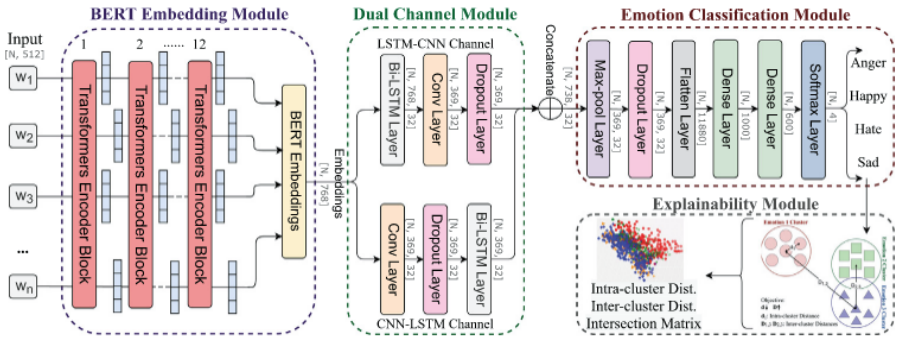


Fig. 1. Schematic description of the proposed system's architecture. Here, w_i denotes the i^{th} word of the input sentence S and N denotes the no. of training examples.

Architecture of P. Kumar and B. Raman's original model

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
DualCnnBiLsmtModel                       [32, 13]                  --
├─LSTM: 1-1                              [32, 128]                 427,008
├─Conv2d: 1-2                            [32, 1, 1]                131,104
├─ReLU: 1-3                              [32, 1, 1]                --
├─Dropout: 1-4                           [32, 1, 1]                --
├─Conv2d: 1-5                            [32, 1, 737]              32,800
├─ReLU: 1-6                              [32, 1, 737]              --
├─Dropout: 1-7                           [32, 1, 737]              --
├─Linear: 1-8                            [32, 128]                 94,592
├─ReLU: 1-9                              [32, 128]                 --
├─Linear: 1-10                           [32, 13]                  1,677
==========================================================================================
Total params: 687,181
Trainable params: 687,181
Non-trainable params: 0
Total mult-adds (Units.GIGABYTES): 2.53
==========================================================================================
Input size (MB): 0.10
Forward/backward pass size (MB): 0.26
Params size (MB): 2.75
Estimated Total Size (MB): 3.10
==========================================================================================
```

Our implementation of P. Kumar and B. Raman's original model

The DualCnnBiLsmtModel we have implemented is derived from P. Kumar and B. Raman's 2021 research paper "A BERT based dual-channel explainable text emotion recognition system". The model in the original paper exemplified the contemporary trend towards hybrid neural architectures that leverage both convolutional and recurrent elements to capture a wide spectrum of linguistic features for the complex task of Text Emotion Recognition (TER).

In an implementation matching the original model, shown in **Appendix B,** we identified that the model was massive, complex, and heavy. It meant that the operations and deployment of such a model, as well as tasks in explaining and fine-tuning would be very computationally expensive.

As such, we implemented a lighter variant of the original model with our own implementations of tokenization, embedding generations, and network design.

Model Overview

The architecture begins with BERT sentence embeddings as inputs. BERT, or Bidirectional Encoder Representations from Transformers, has revolutionized the field of NLP since its introduction by Devlin et al. (2018). It captures deep bidirectional context by pre-training on a large corpus of text and then fine-tuning for specific tasks, setting a new standard in a wide array of NLP benchmarks.

The Bi-LSTM layer, a variant of the original LSTM architecture introduced by Hochreiter & Schmidhuber (1997), enables the model to contextualize information both forwards and backwards through a sequence, capturing temporal dependencies effectively. The dual-channel convolutional approach allows the model to extract both high-level features processed by the LSTM and direct features from the raw embeddings, which may include local contextual cues that are often pivotal in understanding emotional nuance.

Inner Workings

The LSTM's output and the original sentence embeddings undergo convolution and pooling operations via CNN layers. The CNN's ability to capture translational invariances and local patterns within data is well-documented (LeCun et al., 1998), and their application in NLP has shown considerable success in semantic analysis (Kim, 2014).

Following feature extraction, the flattened outputs are concatenated and processed by fully connected layers. The dimensionality reduction and integration of features at this stage is crucial for classification tasks, as it aligns with the universal approximation theorem which posits that a feedforward network with a linear output layer and at least one hidden layer with any "squashing" activation function (such as ReLU) can approximate any Borel measurable function from one finite-dimensional space to another with any desired non-zero amount of error (Cybenko, 1989; Hornik, 1991).

Effectiveness for TER

The effectiveness of this hybrid architecture for TER is rooted in its capacity to leverage both the global, sequential context provided by the LSTM and the nuanced, local patterns that CNNs excel at extracting. This two-pronged approach is supported by empirical findings in the field where hybrid models often outperform those that rely on a single type of neural network architecture (Zhou et al., 2015) (Kumar & Raman, 2022).

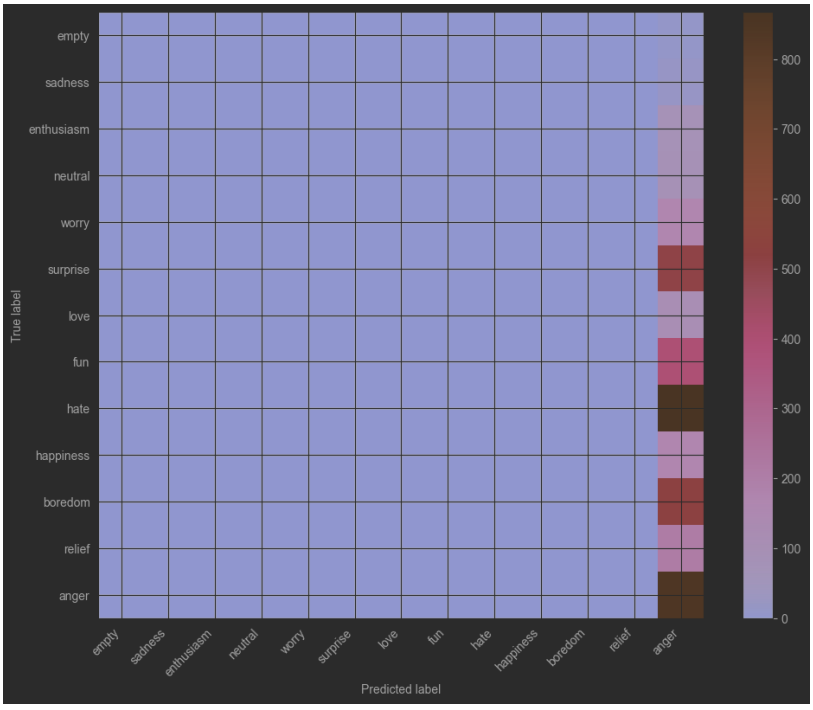
Results and discussion

Metrics Considerations

Average validation accuracy across our k folds is used as our primary metric. In our case where our tweets only have a single dominant emotion label, accuracy is a simple benchmark for model evaluation. Furthermore, performance can be compared to existing models which have used similar metrics.

We had 3 major runs whose results we discuss and evaluate in this section:

1. A naive attempt to apply the model without consideration for class weights or regularization



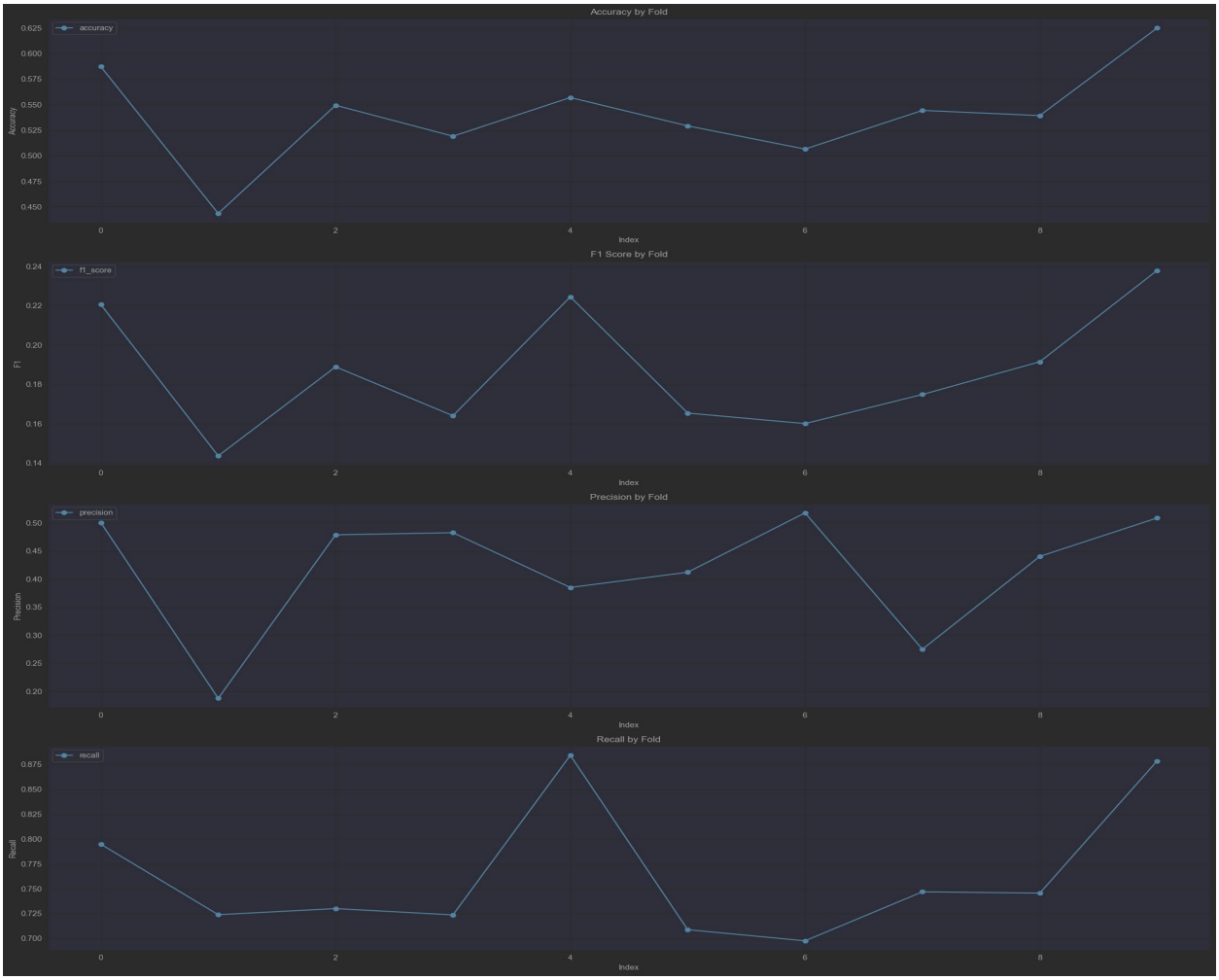
accuracy	precision	recall	f1
0.21125	0.044627	0.21125	0.073687

2. A modification to the naive attempt with early stopping
    a. The results match the above results with a small margin of difference, unshown here to save space.

3. A modified model balancing class weights, implemented l1 and l2 regularization, and using sentence rather than word embeddings. This model was architecturally similar to the above 2 models except that it used full sentence embeddings instead of word embeddings.

The results on our naïve data and modelling showed that there was a significant impact of class imbalances on our models which led to very poor scores. To combat this, we came up with our third strategy which was to perform the same k-folds validation but with L1 and L2 regularization and criterion balancing using class weights.

The results are below. Our defined metrics are shown in **Appendix C.**



Discussion and future work

Model Results

Analyzing the trends and dynamics of the results, we see that:

1. The accuracy metric, indicative of the overall classification performance, exhibits variability across the folds with a notable trough at fold three and a peak at fold ten. Such variability implies an inconsistent model performance, which could be attributed to fold-specific data characteristics or a lack of model convergence.
2. The F1 score, a harmonic mean of precision and recall, serves as a critical metric in imbalanced class distributions. The observed oscillations in the F1 score, particularly the nadir at fold five, suggest disparities in the precision-recall equilibrium. This warrants further investigation into the model's classification thresholds and decision boundaries.
3. Precision, the proportion of true positive predictions to total positive predictions, experiences a marked decline in folds two and five. This decrease may point to an increase in false positives or a reduction in true positives, signaling potential overfitting to specific data features within those folds.
4. Recall, or the sensitivity of the model, highlights the model's capability to identify true positives. The recall metric displays significant variation, with a prominent dip in fold five, indicating missed true positives and suggesting potential underfitting or class imbalance.

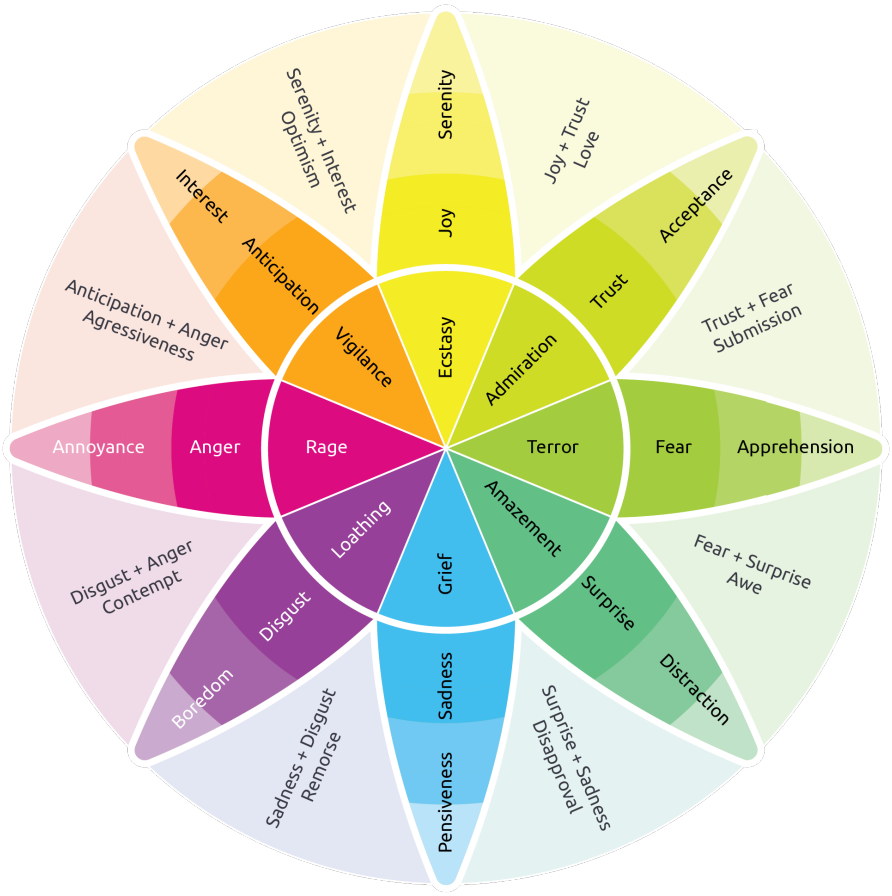
Considering these results, we hypothesize that there lies potential for model improvement beyond the scope of 50 epochs per fold. This is owed to the observed fluctuations across metrics which denote that the model's learning trajectory is not at an asymptote, holding room for improvement to scores in the 90^{th} percentile as in the original paper.

The notable enhancement in the final fold for all metrics, except recall, suggests an evolving model generalization capability; however, the diminished recall intimates a possible shift towards a conservative prediction bias.

The divergent performance across folds intimates the necessity for extended training epochs to achieve model stabilization. Furthermore, the performance metrics underscore the need for a detailed evaluation of the data distribution within each fold to mitigate fold-specific performance anomalies. Addressing the observed inconsistencies may involve exploring advanced regularization techniques, refining the model's architecture, or optimizing hyperparameters.

Alternate Results Methodologies

While the dataset is single-labelled, the tweet could possibly contain multiple emotions. As such, a top-k accuracy score could be a better alternative and this is done by evaluating if the correct y label is within the top k predicted labels. Furthermore, the emotions labelled might not be clearly distinct and may be of different severity, hence a color wheel of emotions can be adopted and distance between each level of emotions can be more accurately tracked instead of a binary classification.



Color wheel emotions visualization

For future scope, we aim to implement an extension of the modelling and research into model explainability and computation costs through visualizations of model predictions such as emotion clusters separating and converging within the network, and ablation techniques to understand the complexity.

Additionally, for future scope, we would like to take a broader dataset which contains data beyond just tweets and varies in intent, style, semantics, syntax, and length. This will allow us to train a lightweight model which is well generalizable and adaptive and can be easily explained and further developed and optimized.

References

Ascendik. (2023, July 11). *Unlocking the Palette of Feelings with the Color Wheel of Emotions*. https://www.ascendik.com/unlocking-the-palette-of-feelings-with-the-color-wheel-of-emotions/#page-content

Das, S. (2021, June 8). CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more. . .. *Medium*. https://medium.com/analytics-vidhya/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

Devlin, J. (2018, October 11). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv.org. https://arxiv.org/abs/1810.04805

Gradient-based learning applied to document recognition. (1998, November 1). IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/document/726791

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, *4*(2), 251–257. https://doi.org/10.1016/0893-6080(91)90009-t

Kim, Y. (2014a). Convolutional Neural Networks for Sentence Classification. *ACL Anthology*. https://doi.org/10.3115/v1/d14-1181

Kim, Y. (2014b, August 25). *Convolutional neural networks for sentence classification*. arXiv.org. https://arxiv.org/abs/1408.5882

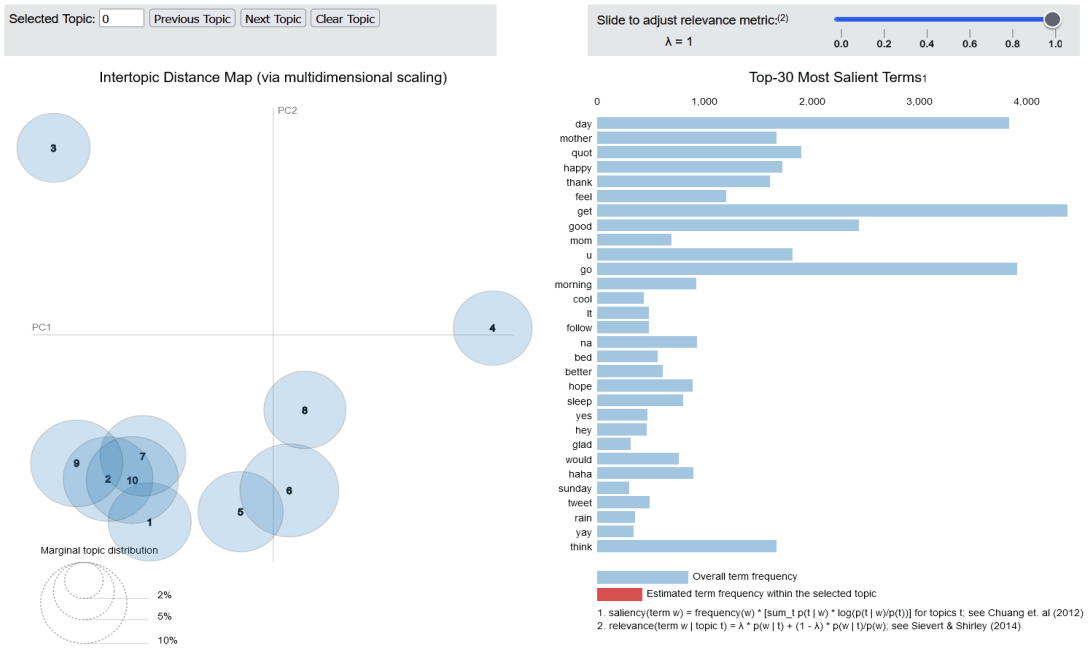
Kumar, P., & Raman, B. (2022). A BERT based dual-channel explainable text emotion recognition system. *Neural Networks*, *150*, 392–407. https://doi.org/10.1016/j.neunet.2022.03.017

Lai, S., Xu, L., Liu, K., & Zhao, J. (2015). Recurrent convolutional neural networks for text classification. *Proceedings of the . . . AAAI Conference on Artificial Intelligence*, *29*(1). https://doi.org/10.1609/aaai.v29i1.9513

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. https://doi.org/10.1109/5.726791

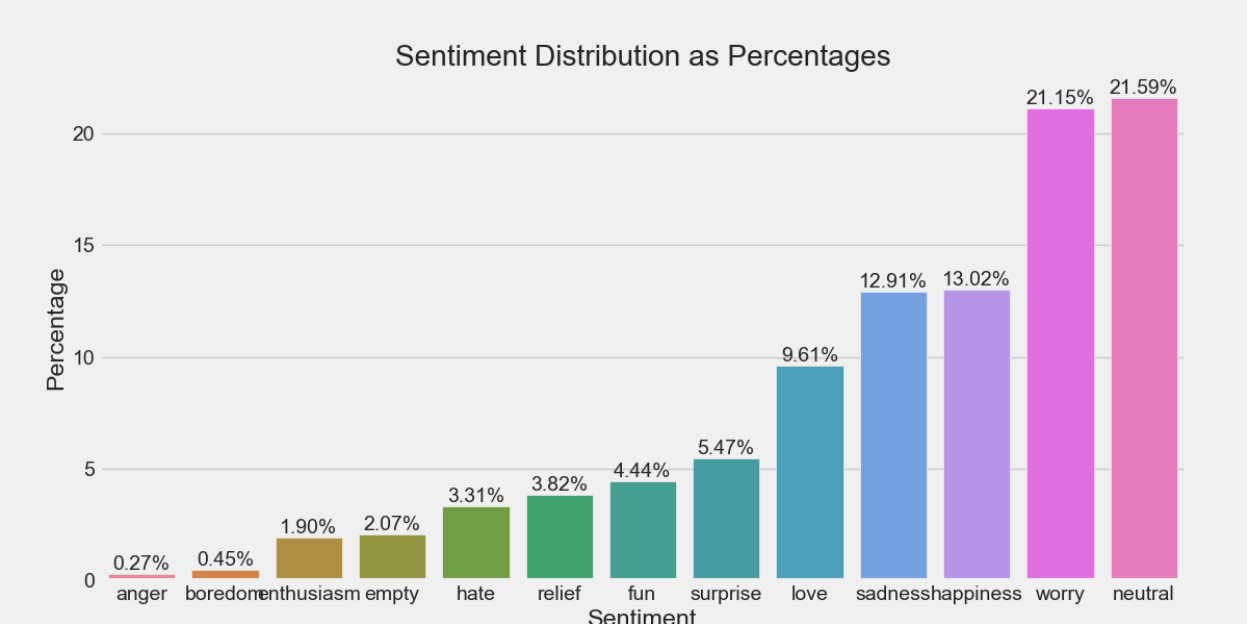
Zhou, P., Wei, S., Tian, J., Qi, Z., Li, B., Hao, H., & Xu, B. (2016). Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification. *ACL Anthology*. https://doi.org/10.18653/v1/p16-2034

Appendix A - EDA Images

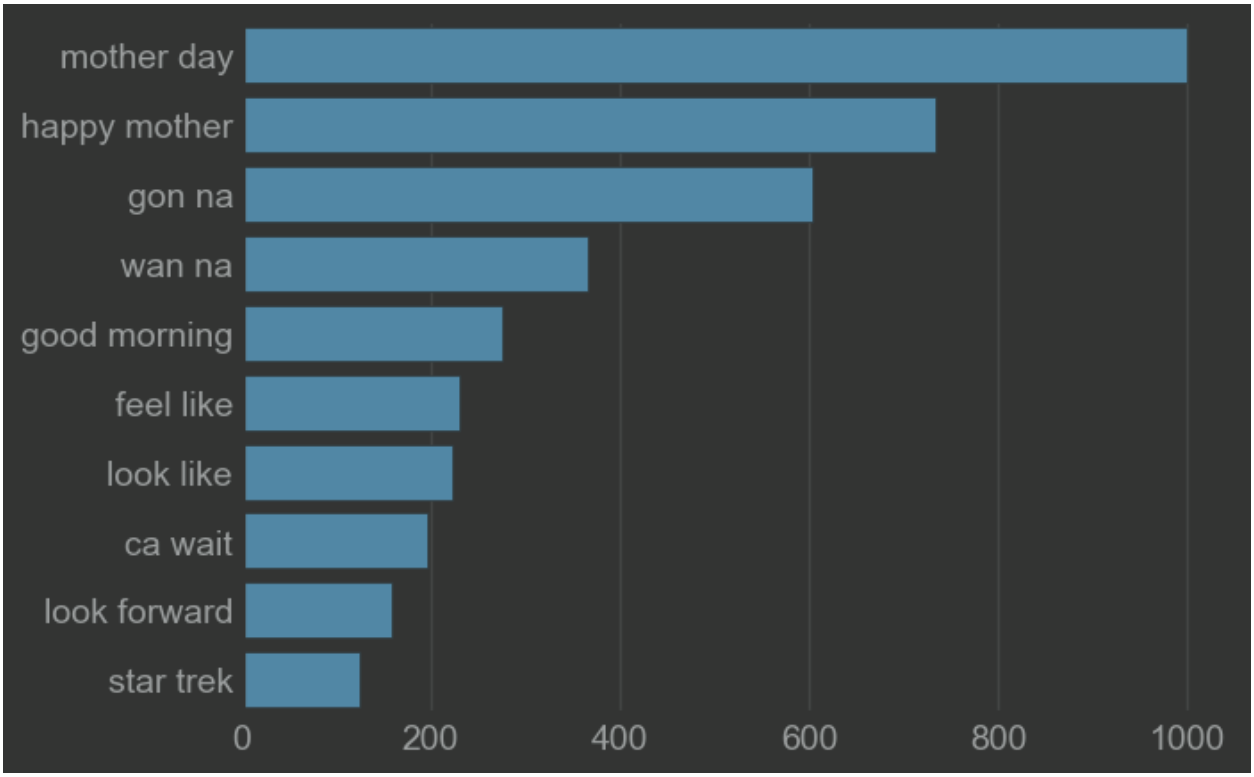


Latent Dirichlet Allocation Visualization (visualisation)

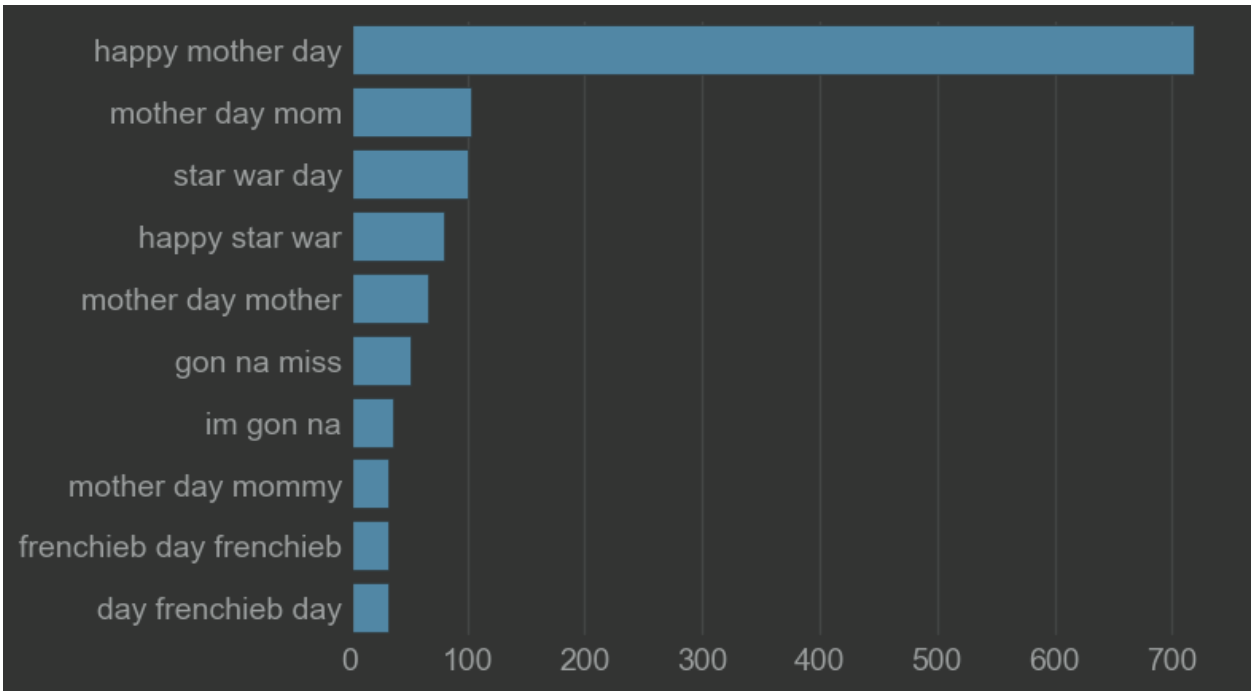
Word Cloud Visualisation



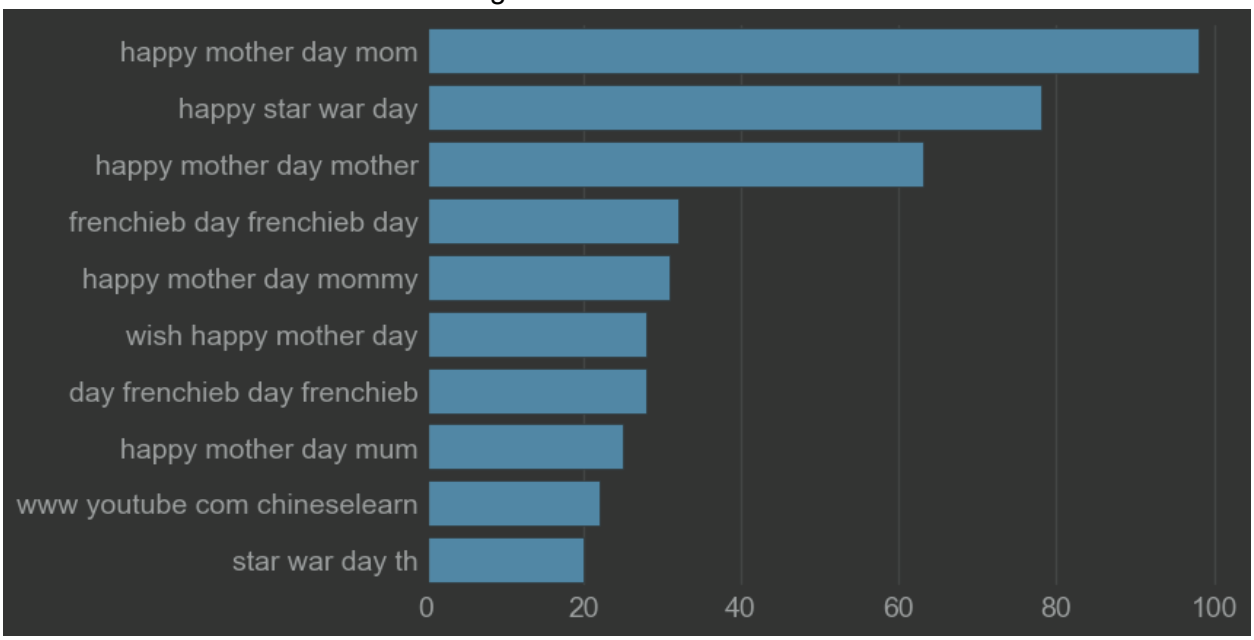
Sentiment distribution visualisation



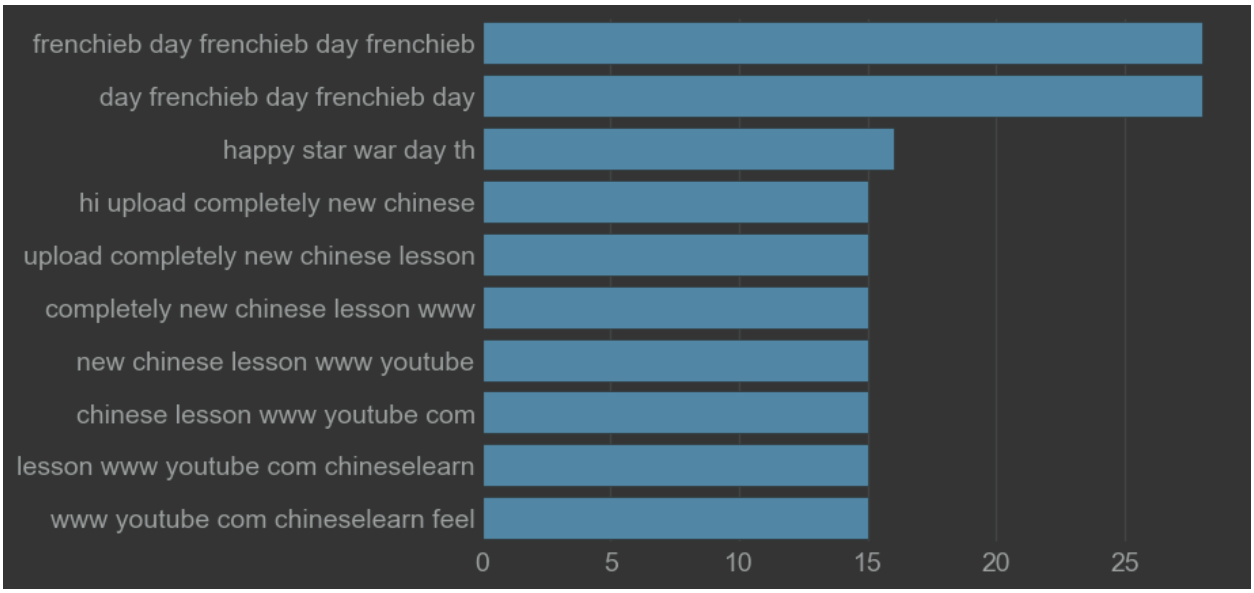
Bigram chart visualisation



Trigram chart visualisation



Quadrigram chart visualisation



Pentagram chart visualisation

Appendix B – Hypothetical combined model with word-wise embeddings:

```
==========================================================================================================================
Layer (type:depth-idx)                                   Output Shape              Param #
==========================================================================================================================
EmotionRecognitionModel                                  [32, 13]                  --
├─DualChannelModule: 1-1                                 [32, 512, 768]            --
│    └─BertModel: 2-1                                    [32, 768]                 --
│    │    └─BertEmbeddings: 3-1                          [32, 512, 768]            23,837,184
│    │    └─BertEncoder: 3-2                             [32, 512, 768]            85,054,464
│    │    └─BertPooler: 3-3                              [32, 768]                 590,592
│    └─LSTM: 2-2                                         [32, 512, 512]            2,101,248
│    └─Conv1d: 2-3                                       [32, 256, 512]            590,080
├─EmotionClassificationModule: 1-2                       [32, 13]                  --
│    └─MaxPool1d: 2-4                                    [32, 768, 256]            --
│    └─Linear: 2-5                                       [32, 128]                 25,165,952
│    └─Dropout: 2-6                                      [32, 128]                 --
│    └─Linear: 2-7                                       [32, 13]                  1,677
==========================================================================================================================
Total params: 137,341,197
Trainable params: 137,341,197
Non-trainable params: 0
Total mult-adds (Units.GIGABYTES): 48.39
==========================================================================================================================
Input size (MB): 0.26
Forward/backward pass size (MB): 13693.59
Params size (MB): 549.36
Estimated Total Size (MB): 14243.21
==========================================================================================================================
```

This model implementation consists of two main modules:

- DualChannelModule: A module that processes input text using two channels:
    - BertModel: This is the backbone for extracting word embeddings. It consists of:
        - BertEmbeddings: Learns contextualized word representations.
        - BertEncoder: Encodes the embeddings using multiple layers (12 in bert-base) of transformer architecture which allows it to learn the context of a word based on all the other words in the sentence.
        - BertPooler: Applies a pooling operation to the output of the transformer layers to get a fixed-size encoding of the input.
    - LSTM: A bidirectional LSTM with an output feature dimension of 256. It processes the sequence data from the BERT model, capturing long-range dependencies in both directions (forward and backward).
    - Conv1d: A one-dimensional convolutional layer with an output feature dimension of 256, designed to capture local patterns in the sequence.
- EmotionClassificationModule: A module for classifying emotions based on the features extracted by the DualChannelModule:
    - MaxPool1d: Applies a max pooling operation over the features, reducing their dimensionality and aggregating important information.

- Linear: Fully connected layers that map the extracted features to the final classification outputs.
- Dropout: Used to prevent overfitting by randomly setting a fraction of the input units to 0 during training.

Data flow during inference and training phase:

1. Input Processing: Tokenized input text is fed into the model. BERT embeddings are generated for each token.
2. Dual Channel Processing: The BERT embeddings are passed through the LSTM and Conv1d layers in parallel, forming a dual-channel architecture.
3. The LSTM captures sequential and contextual information from the text.
4. The Conv1d layer captures local features (like n-gram patterns) within the word embeddings.
5. Feature Aggregation: The outputs from both channels are combined, preserving both local and contextual information.
6. Classification: The combined features are pooled, flattened, and passed through fully connected layers to produce logits for each emotion class.

TER Specific Effectiveness

The effectiveness of this model in TER tasks comes from the synergy of its components:

- BERT: By using pre-trained BERT embeddings, the model leverages rich, pre-learned contextual representations of language, which can capture the nuances and complexities inherent in text-based emotion recognition.
- Bidirectional LSTM: The LSTM can understand the sequence both from past to future and future to past, capturing the context in both directions which is important for understanding the sentiment or emotion conveyed in a sentence.
- CNN: The convolutional layers can pick up on local cues and patterns, such as the presence of specific phrases or combinations of words that are indicative of emotional content.
- Dual Channel: This architecture captures both the benefits of recurrent layers (LSTM) and convolutional layers (CNN), potentially leading to a more robust feature representation for classification tasks.

Together, these components make a powerful model for TER, capable of understanding complex patterns in text data and classifying emotions with a high degree of accuracy.

Appendix C – Metrics Used

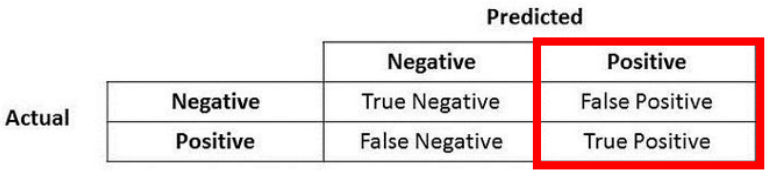


Fig 4. Illustration of confusion matrix components to calculate accuracy

Accuracy: Ratio of correctly identified positive (1) and negative labels (1) to the total test size.

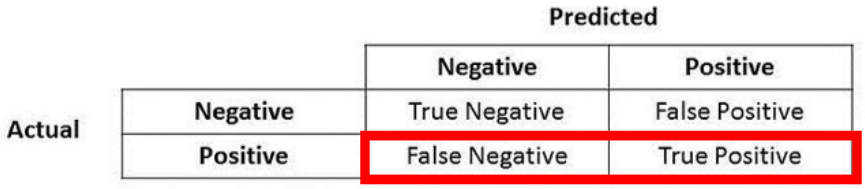


Fig 5. Illustration of confusion matrix components to calculate precision

Precision: Ratio of correctly identified positive labels (1) over all true positive cases.

Recall = Ratio of correctly identified positive labels (1) over all true positive cases and falsely identified negative cases.

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

F1 Score = Ratio to balance precision and recall; F2 Score: Weightage of 2 applied to Recall

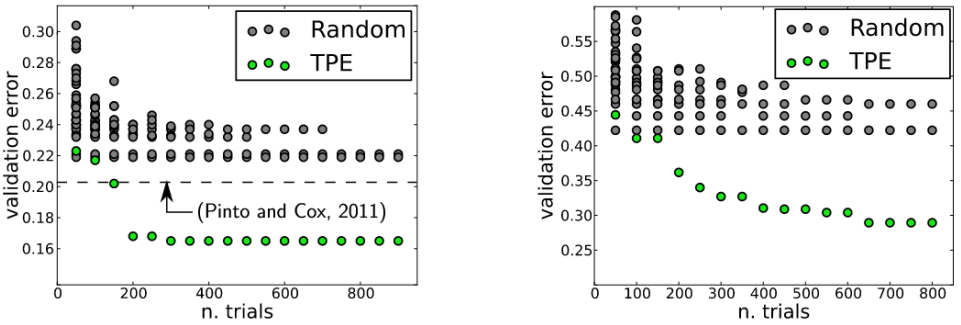
Appendix D - Bayesian Optimization

Bayesian optimization is an alternative to existing hyperparameter optimization techniques such as grid search or random search. Rather than iterating over a set of given values in a range, or randomly selecting points to use to maximise model performance without learning from the results of the previous sets of evaluated hyperparameters, Bayesian optimization estimates a posterior distribution of the estimated error scores from each trial, then fits the values expected to minimise the estimated error.

By doing so, it can more effectively determine the most optimal set of hyperparameters for a given task. In their paper, Bergstra et. al evaluate the performance of models with hyperparameters tuned using automatic optimization techniques of Tree of Parzen Estimators, and random search.

Their evaluation found that TPE optimization had a considerable improvement over a simple random search:

Making a Science of Model Search



Method (# configurations)	LFW View 2 Error (%)	PubFig83 View 2 Error (%)
TPE-optimized (750)	**15.5** ± .7	**13.50** ± .7
High-throughput (15K)	**15.9** ± .7 (Pinto & Cox, 2011)	14.78 ± .45 (Pinto et al., 2011)
Random search (2K)	20.8 ± .8	19.0 ± .8
Chance	50.0	98.8

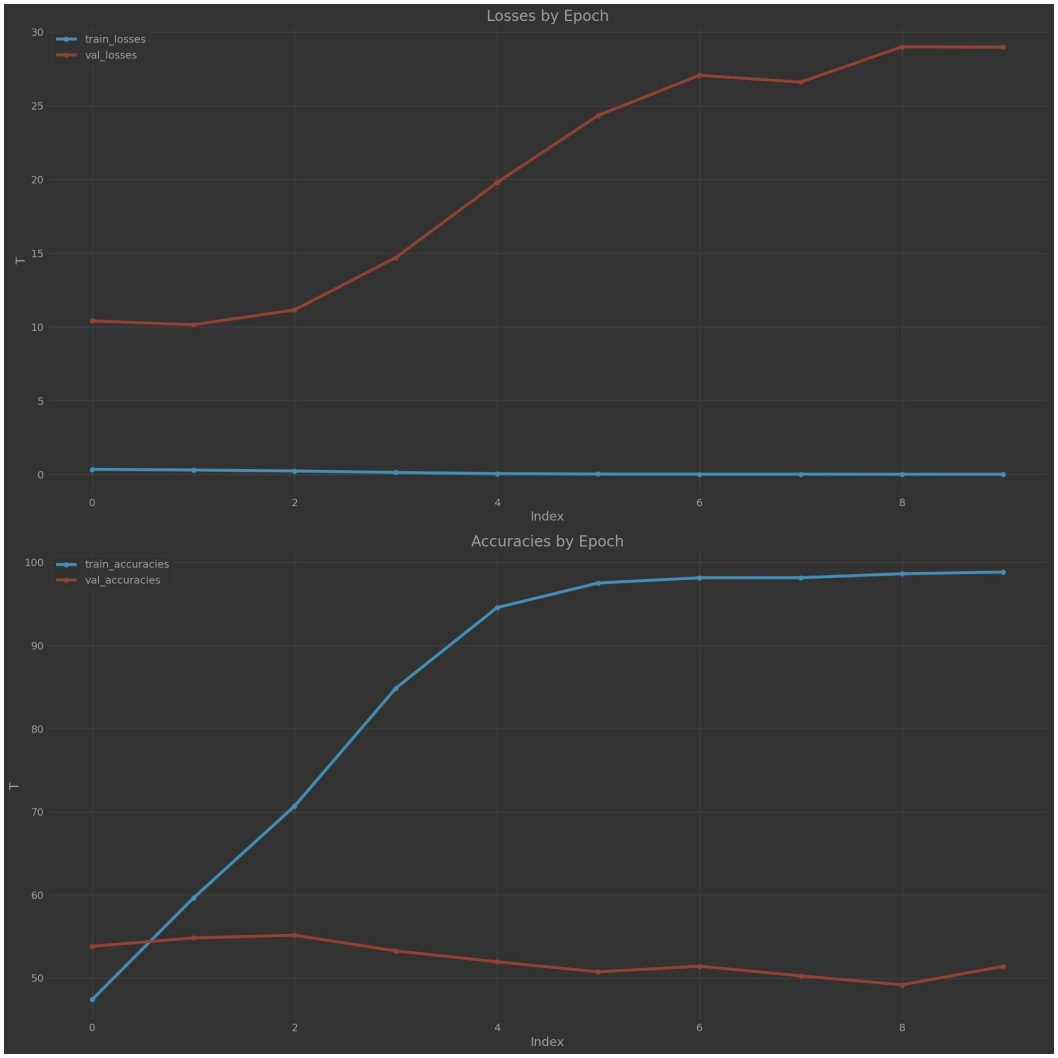
Random Search vs TPE

Appendix E – Gated Recurrent Unit

We attempted to use a GRU on a binned version of the data, and obtained significantly improved results. This further reinforces our belief that there is value in pursuing the data and modelling strategies adopted, and that with more time, we can achieve significant improvements.

```
EmoGRU(
  (embedding): Embedding(83294, 256)
  (dropout): Dropout(p=0.5, inplace=False)
  (gru): GRU(256, 1024)
  (fc): Linear(in_features=1024, out_features=3, bias=True)
)
```

Simple GRU model to Classify Emotions



Model performance showing improvement across training data with potential for improvement in validation data