Comparing BERT Transformer Results To Naive Bayes & Random Forest Models for Emotion Detection

By: Richard Hoehn [*]

Middle Tennessee State University

CSCI 6350 - Graduate Project

Prof. Dr. Sal Barbosa

April 15, 2024

Abstract

This graduate project explores the effectiveness of the BERT (Bidirectional Encoder Representations from Transformers) model for emotion detection and compares its performance with traditional machine learning models, specifically Naive Bayes and Random Forest. The study leverages a dataset comprising tweets annotated with emotional labels, aiming to enhance emotion detection capabilities through deep learning. The BERT model is evaluated based on its ability to understand contextual dependencies in text, which is crucial for accurate emotion classification.

The training of the BERT model was completed in a relatively short time, requiring only four epochs to achieve convergence and avoid overfitting, as demonstrated in our results. The validation accuracy reached approximately 39%, with test accuracy close to 38%. These figures represent a significant improvement over the traditional models evaluated, specifically outperforming Naive Bayes and Random Forest by 6-10% in test accuracy, with far fewer epochs and less training time needed. The paper presents a comprehensive methodology including data preparation, model training, and evaluation, employing PyTorch for implementing the BERT model. Results are quantitatively analyzed through accuracy, F1 scores, and a confusion matrix, demonstrating the BERT model's improved ability to capture nuanced emotional expressions compared to traditional models with the same dataset. This research not only underscores the impact of transformers in NLP but also provides future work conclusions for continued research in this field.

[*] Email: rhoehn@mtmail.mtsu.edu; corresponding author

1 Introduction

The motivation for the this graduate paper aims to build upon the research and paper "Improving Emotion Detection through Translation of Text to ML Models Trained in Different Languages"[3] by Hoehn et al. which focused on ML models such as Naive Bayes and Random Forest for ED. By the use of transformers and comparing those to Hoehn's previous work we hope to see if improvements can be made on emotion detection accuracy.

To complete the project, data from Hoehn's[3] paper will serve as the basis for comparison. The dataset[4] itself is, as previously mentioned, from 2021 Twitter, a simple collection of tweets annotated with emotions. There are 9 different labeled emotions and corresponding text with over 1,500 rows in a `csv` file.

Using a Transformer for Emotion Detection the research's focus is centered on four (4) main goals:

- Build, Train, and Validate a Transformer for Emotion Detection, using PyTorch[6].
- Display the learning rates and validation results in graphical forms using plots.
- Evaluate the Transformer's results against those from Hoehn[3] et al.'s paper in spesific the Naive Bayes & Random Forest results.
- Compare the constructed transformer with Lim et al.'s transfer learning-based emotion detection model[5] to ensure comparative results in ED have been achieved with the fine-tuned BERT transformer model.

Visual aids, such as plots, will be utilized to display learning and validation outcomes, crucial for refining the Transformer's performance, in addition making necessary adjustments to hyperparameters in order to enhance performance.

A thorough comparison with Hoehn et al.'s and Lim et al.'s works[3][5] will validate our Transformer model's capability in capturing emotional nuances, showcasing its advantage over traditional ML models in understanding contextual dependencies.

2 Background

Emotion detection (ED) has garnered significant interest and investment in recent years, emerging as an important tool for understanding human behavior and enhancing communication effectiveness[7, 3]. Through the analysis of text, speech, or facial expressions, ED processes, and underlying machine learning (ML) models, aim to accurately identify and classify emotions expressed by individuals or groups, often in real-time.

To enhance ED we incorporated natural language processing (NLP) through the use of deep learning architectures, that can detect emotions with high precision given sufficient data[1]. The introduction of the Transformer[1] model by Vaswani et al. in 2017, through the paper "Attention is All You Need"[8], marked a significant advancement and established the transformer as the state-of-the-art model for various NLP tasks, including machine translation, text summarizing, and question & answering tasks.

3 Methods

In this section the mathematical, theoretical, and practical execution of this paper's transformer model is introduced and references to the technologies, datasets, and visualization techniques presented in the subsequent sub-sections.

[1] See Appendix **B** for more details on the Transformer

3.1 Data Preparation and Pre-Processing

The preliminary stage of the study involved data handling where the dataset was imported into a pandas DataFrame from a CSV file. Initial data exploration revealed superfluous columns which were subsequently removed to enhance dataset clarity and focus on English text (emotion_en, sentence_en) only. This data cleaning process was for streamlining downstream processing and analysis.

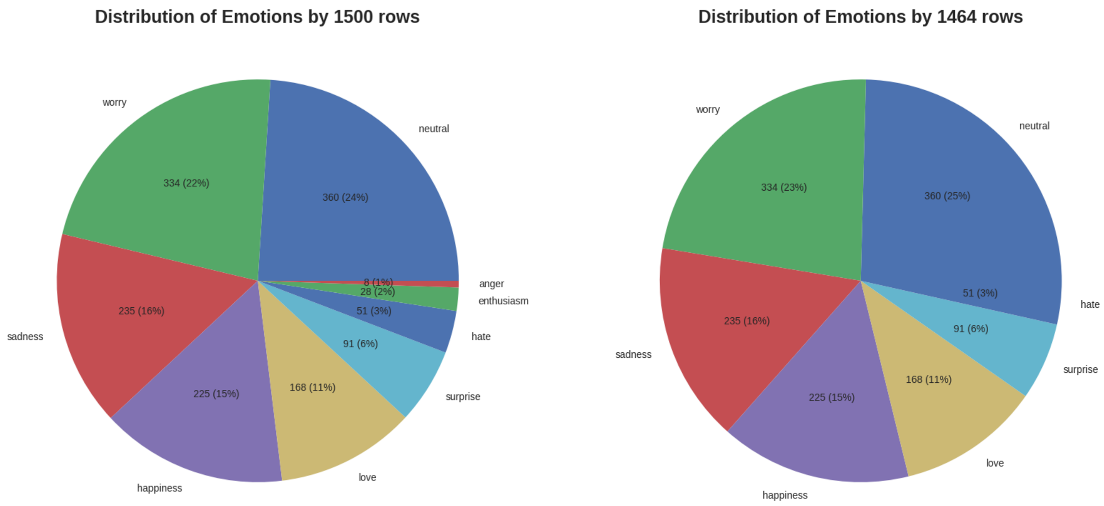


Figure 1: Emotion Distribution & Removal of Two (2) Emotions

Further into the data preparation phase, the distribution of emotions within the dataset was visualized using pie charts to understand the imbalance and distribution across different emotions. This was useful for assessing the need for potential resampling techniques to handle class imbalance, thus ensuring robust model training. The removal of "anger" and "enthusiasm" can be see in figure 2 above which left our classification mode with seven (7) emotions.

3.2 Feature Engineering and Label Encoding

Critical to the processing of textual data, label encoding was employed to convert categorical emotion labels into numerical format, thereby facilitating their use in machine learning algorithms. The `LabelEncoder` from SKLEARN.PREPROCESSING was utilized for this transformation, ensuring that each emotion label was uniquely represented by a numerical identifier. Label encoding is essential as it allows the model to interpret categorical data, which is inherently non-numeric, in a mathematical format that algorithms can process.

The simple use of the following code was performed by use of `LabelEncoder` in Python:

```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
data["labels"] = encoder.fit_transform(data["emotion_en"])
```

This feature engineering step is pivotal in optimizing the performance of the model, ensuring that the input data is structured in a way that maximizes the learning algorithm's effectiveness.

3.3 Sentence / Word Tokenization

In addition to label encoding, tokenization was performed using BERT's tokenizer[2]. This process converted text into tokens that could be fed into the neural network. Tokenization included the addition of special tokens necessary for BERT to understand sentence boundaries and involved truncating or padding sentences to a uniform length for batch processing.

[2] BertTokenizer from Hugging Face's transformers library

Upon reviewing the distribution (see figure 2) of word counts across different samples, it was determined that a token length of 64 was sufficient to capture the necessary information without any data loss, optimizing both performance and computational efficiency. This is true since Figure 2 illustrates that none of the emotions sentence word counts was greater than 40 words / tokens.

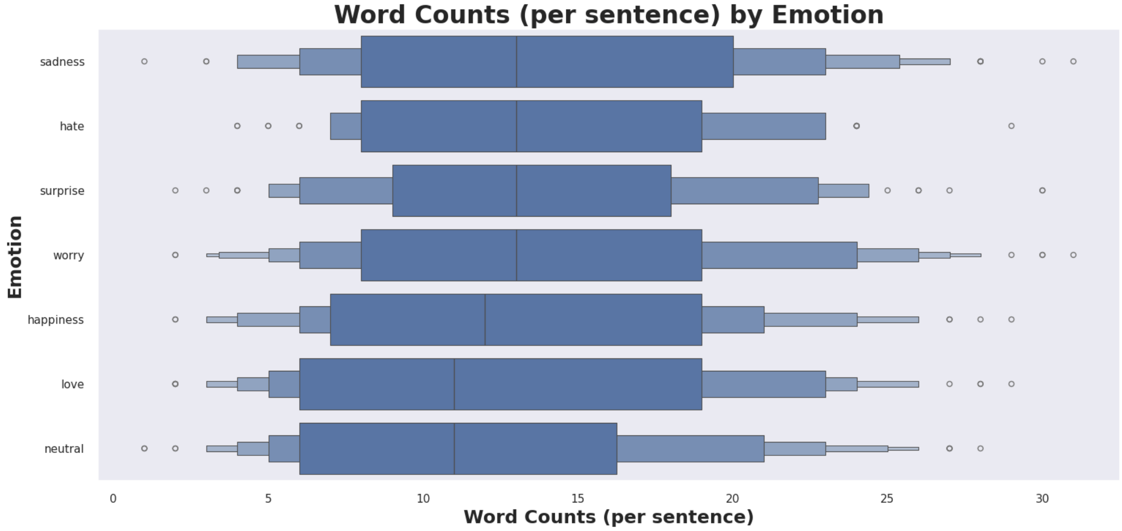


Figure 2: Sentence / Token word count distribution by emotion

Shortening the token length generally improves training efficiency by reducing the computational load and memory usage of memory, enabling faster iterations and optimization of the neural network model.

3.4 Model Training and Evaluation

The core of the methodology revolved around the deployment of a transformer-based model, specifically BERT (Bidirectional Encoder Representations from Transformers), for sequence classification. BERT builds upon the Transformer[3] by pre-training on a large corpus of text using a masked language model and next sentence prediction, enabling it to understand language context and nuances more effectively. The BERT model in our research was fine-tuned for the emotion detection task using a custom training loop encapsulated within a PyTorch Lightning framework to streamline training and evaluation processes.

Training involved dividing the data into training, validation, and test sets using a stratified[4] approach to maintain the distribution of labels across datasets. The model was trained on a CUDA-enabled device to leverage GPU acceleration, significantly speeding up the training process. Model performance was monitored through the computation of accuracy and F1-score metrics, and early stopping was employed to halt training upon saturation of validation loss improvements, thus preventing overfitting.

During the training, the softmax function was employed as the final layer's activation to convert the logits into probabilities for each class. The softmax function is defined as:

$$\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{1}$$

This transformation is essential for multi-class classification problems where the outputs can be interpreted as probabilities that sum to one. Subsequently, the cross-entropy loss function, which quantifies the difference between the predicted probabilities and the actual class labels, was used to compute the loss. The cross-entropy loss for a multi-class classification problem is given by:

[3] See Appendix **B** for more details on the Transformer
[4] See Appendix **A** for details on stratification

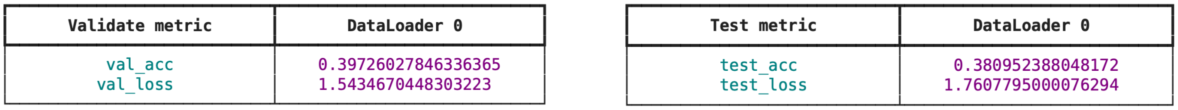
$$L = -\sum_{i=1}^{c} y_i \log(p_i) \tag{2}$$

where y_i is the true label in a one-hot encoded vector, and p_i is the predicted probability of each class for the sample.

Throughout the training phase, learning rate adjustments were managed by a scheduler to optimize the training dynamics, and the training process was logged for post-hoc analysis using the PyTorch Lightning logging utilities. This combination of softmax probabilities and cross-entropy loss is critical for effectively training deep learning models on classification tasks, as it guides the model towards better accuracy by minimizing the discrepancy between the predicted and actual labels.

4 Results

Overall the results where good. The training took place in little time with only four (4) epochs necessary to achieve convergence and not move into the over-fitting area. Based on the two outputs depicted in Figure 3 below it can be see that the validation accuracy is at about 39% and the test accuracy at 38%.



```
┏━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━┓
┃   Validate metric    ┃     DataLoader 0     ┃
┡━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━┩
│       val_acc        │ 0.39726027846336365  │
│       val_loss       │  1.5434670448303223  │
└──────────────────────┴──────────────────────┘
```

(a) Validation Dataset

```
┏━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━┓
┃     Test metric      ┃     DataLoader 0     ┃
┡━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━┩
│       test_acc       │  0.380952388048172   │
│      test_loss       │  1.7607795000076294  │
└──────────────────────┴──────────────────────┘
```

(b) Test Dataset

Figure 3: Validation & Test Datasets on Trained BERT Transformer

4.1 Convergence Trends of Loss and Accuracy

The examination of the model's learning curves (see Figure 4) presents an good visual of the fine-tuning of the model's performance across epochs. The training loss exhibits a decreasing trajectory, indicative of the model's increasing proficiency on the training dataset. Concurrently, the validation loss follows this descent initially, corroborating the model's capability to generalize beyond the training data. However, an observed plateau and marginal ascent in the validation loss at epoch 4 suggests the inception of overfitting. This phenomenon suggests the model's growing memorization of the training data, which could detrimentally affect its predictive performance on novel data.

The accuracy metrics further suggested this analysis that around epochs 4 the ascending curve of training accuracy signifies an improving recognition of training patterns. In contrast, the validation accuracy curve ascends initially fast but levels off at epochs 4, a possible suggestion of the model's limitations in generalizing these learned patterns to unseen data.

4.2 Learning Rate Dynamics

The learning rate decay graph delineates a step-wise reduction strategy, a deliberate choice to refine the model's weight adjustments as it progresses through the learning phases. The marked reduction after a specific epoch, delineated by a vertical dashed line, is indicative of an aggressive strategy to expedite the convergence process. Such a significant diminished, while potentially beneficial for escaping local optima, warrants careful optimization to ensure the model does not bypass the global minimum of the loss function.

4.3 Confusion Matrix Results

The confusion matrix indicates that 'neutral' emotion has the highest number of correct predictions (23), followed by 'worry' and 'happiness'. There are notable mis-classifications between 'happiness' and 'neutral',

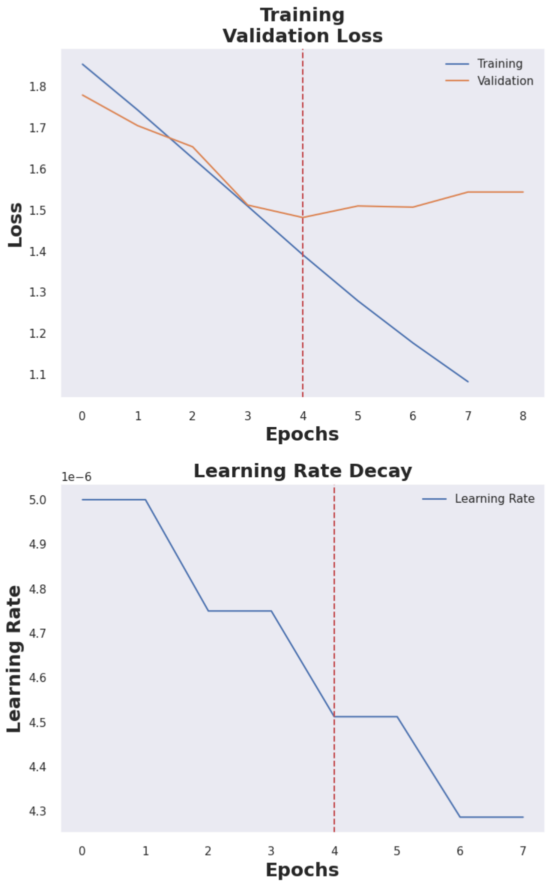


Figure 4: Validation of Loss & Accuracy During Training.
The vertical line at epoch 4 depicts the model's optimal weight & bias after finetuning.

'love' and 'neutral', and 'sadness' and 'worry'. These instances suggest potential challenges the model faces in distinguishing between certain emotions.

The emotion histogram showcases the frequency of each emotion within the test dataset. 'Neutral' and 'happiness' are the most prevalent emotions, while 'hate' has the least occurrence. This distribution is indicative of the dataset's composition and may have implications on the model's learning bias.

4.4 Model Training & Hyper-Parameter Tuning

In Table 1 the used for hyper-parameters in the BERT Transformer model setup are listed, these are pre-set from the Jacob Devlin et al. paper on "BERT: Pre-training of Deep Bidirectional Transformers"[2] which describes the parameter used.

BERT Transformer Parameters

Learning Rate	5×10^{-5}	Params	109M
Classes	7	Model Size	437.95MB
Number of Epochs	4	CUDA	YES
Optimizer	ADAM	Training (on CUDA)	~6min
Activation Func.	GeLU	Layers	4
Hidden Size	768	Batch Size	10

Table 1: Hyper-Parameter for Transformer Training

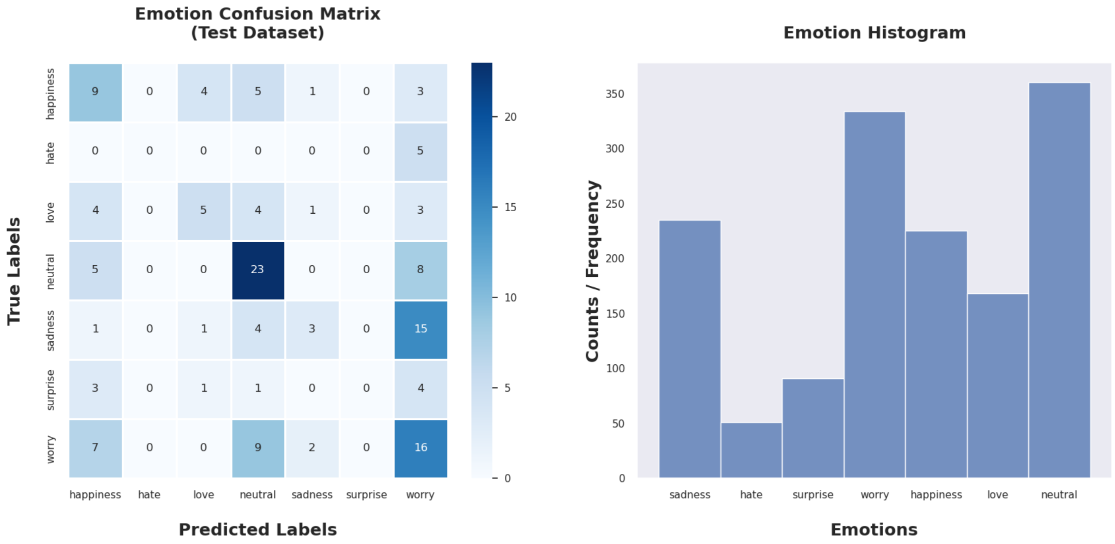


Figure 5: Confusion Matrix Results on Test Data

Furthermore, evidence of overfitting was observed after 4 epochs (see Figure 3), which is indicated by the loss curves for both training and validation.

5 Comparing to Naive Bayes & Random Forest

As stipulated in our introduction (see Section 1) the results of fine-tuning the BERT transformer performed better than Naive Bayes and Random Forest models that were trained in Hoehn et al. paper on Improving Emotion Detection"[3]. In table 2 one can see that BERT overall did better than the previous two models.

Comparisons - Test Accuracy & Model Training Time

Model	Accuracy	Training Time	Epochs
Naive Bayes	32.08%	∼30 min	150
Random Forest	29.72%	∼32 min	150
BERT	**38.09%**	∼6 min	**4**

Table 2: This table depicts the test accuracy, training, and required Epochs for Naive Bayes, Random Forest, and BERT

In addition to the improvements over the Naive Bayes and Random Forest models our BERT transformer performed lower on average than Lee[5] et al. "Transformer transfer learning emotion detection model" paper that used fine-tuning of the BERT model. Their results where substantially higher than our BERT model by over **25%** on average. We believe that due to the much larger training dataset[5] of Lee et al. research the accuracy's are in-line with ourś of only 1,500 sentences.

6 Discussion

The depicted learning dynamics suggest that the model is learning but is verging on overfitting, as evidenced by the loss and accuracy trends. Overall, this is to be expected with a smaller dataset such as ours that only entailed 1,500 sentences. The learning rate schedule merits a reevaluation to balance the trade-off between expedient convergence and the stability of the learning trajectory. Implementing regularization techniques, optimizing the learning rate decay, and employing early stopping are recommended to prevent overfitting and ensure a robust model.

However, it was impressive nonetheless to see that the BERT transformer outperformed the Naive Bayes and Random Forest models by over 6 - 10%, with far fewer epochs and hence less training time required.

7 Future Work

Further Dataset Expansion and Diversity
To improve the generalizability and robustness of the BERT model, it is crucial to train and test on larger and more diverse datasets. Expanding the dataset will help the model learn a wider variety of patterns, which is especially important for emotion detection tasks that can vary significantly across different contexts and cultures.

Regularization Techniques
Implementing more sophisticated regularization techniques could help mitigate the overfitting observed in the training process. Techniques such as dropout variations, L2 regularization (weight decay), or even newer methods like stochastic depth could be explored to enhance model performance without compromising on the ability to generalize.

Exploration of Learning Rate Schedules
The learning rate schedule has a significant impact on the training dynamics of deep learning models. Experimenting with different schedules, such as cyclical learning rates or super-convergence strategies, might lead to better optimization pathways and quicker convergence without falling into local minima.

Comparative Studies with Other Transformer Models
Further studies could compare the performance of BERT with other transformer-based models like RoBERTa, GPT-3, or T5 on the same tasks and limited dataset. Such comparisons could identify the strengths and weaknesses of each approach, which would lead to better choices in model selection for specific applications.

Acknowledgments

We extend our gratitude to Dr. Barbosa for making this CSCI 6350 class interesting and comprehensive in the area of NLP and it's uses in Artificial Intelligence (AI).

References

[1] Mengnan Chen et al. "Research on Emotion Recognition for Online Learning in a Novel Computing Model". In: *Applied Sciences* 12.9 (2022). ISSN: 2076-3417. DOI: 10.3390/app12094236. URL: https://www.mdpi.com/2076-3417/12/9/4236.

[2] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* 2019. arXiv: 1810.04805 [cs.CL].

[3] Richard Hoehn and Jaishree Ranganathan. *Improving Emotion Detection through Translation of Text to ML Models Trained in Different Languages.* Version 1.0.1. https://github.com/richardhoehn/mtsu.coms.qualifying-exam. Aug. 2023. URL: https://github.com/richardhoehn/mtsu.coms.qualifying-exam.

[4] Kaggle and data.world. *Emotion Detection from Text.* 2021. URL: https://www.kaggle.com/datasets/pashupatigupta/emotion-detection-from-text.

[5] Sanghyub John Lee et al. "Transformer transfer learning emotion detection model: synchronizing socially agreed and self-reported emotions in big data". In: *Neural Computing and Applications* 35.15 (2023), pp. 10945–10956. DOI: 10.1007/s00521-023-08276-8. URL: https://doi.org/10.1007/s00521-023-08276-8.

[6] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[7] Iqbal H. Sarker. "Machine Learning: Algorithms, Real-World Applications and Research Directions". In: *SN Computer Science* 2.3 (Mar. 2021), p. 160. ISSN: 2661-8907. DOI: 10.1007/s42979-021-00592-x. URL: https://doi.org/10.1007/s42979-021-00592-x.

[8] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

A Appendix – Stratification of Emotion Labels

In emotion recognition tasks, it is essential to ensure that the training and validation datasets are representative of the overall distribution of emotion labels. Stratification is a technique used to split the data in a way that maintains the proportion of each class label consistent across different subsets.

Stratification is particularly important in datasets where there is an imbalance in the distribution of labels. For example, some emotions like 'sadness' or 'anger' might be less frequent than 'happiness' or 'neutral'. Without stratification, there is a risk that the training or validation set might not contain any examples of the less frequent classes, which can lead to poor model performance and generalization.

Example Code

Below is an example Python code snippet demonstrating how to perform a stratified split:

```
from sklearn.model_selection import train_test_split
import pandas as pd

# Load your dataset
data = pd.read_csv('your_dataset.csv')

# Stratified split
X_train, X_val, y_train, y_val = train_test_split(
    X, y,
    test_size = 0.2,
    random_state = 42,
    stratify = y)
```

B Appendix – Transformer Architecture

The Transformer architecture is primarily composed of two main parts: the encoder and the decoder.

Encoder

The encoder consists of a stack of N identical layers. Each layer has two sub-layers:

- A multi-head self-attention mechanism
- A position-wise fully connected feed-forward network

Normalization and residual connections are used around each of the sub-layers.

Decoder

The decoder also consists of N identical layers. In addition to the two sub-layers in each encoder layer, each decoder layer includes a third sub-layer that performs multi-head attention over the encoder's output. Similarly, normalization and residual connections are also used.

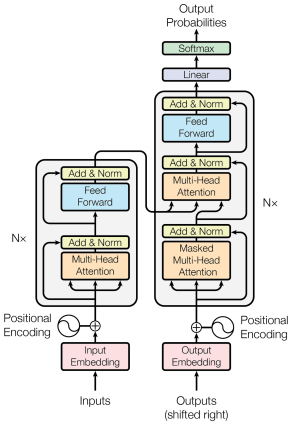


Figure 6:
The encoder-decoder structure of the Transformer architecture Taken from ”Attention Is All You Need”[8]

Multi-Head Attention

Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. This is achieved by projecting the queries, keys, and values h times with different, learned linear projections.

Positional Encoding

Since the model does not contain any recurrence or convolution, positional encoding are added to the input embedding at the bottoms of the encoder and decoder stacks. This is to make use of the order of the sequence.