
EEG Emotion Predictor Application

UNDERGRADUATE THESIS

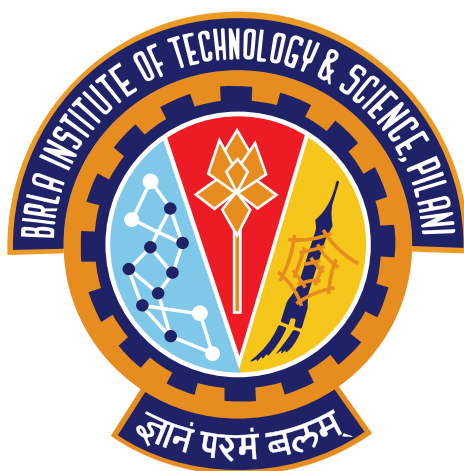
*Submitted in partial fulfillment of the requirements of
BITS F421T Thesis*

By

Satyam Saxena
ID No. 2020A3PS1781P

Under the supervision of:

Dr. Abhijit Asati
&
Dr. Dheerendra Singh, Dr.V.K. Chaubey



BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI, PILANI CAMPUS
March 2024

Declaration of Authorship

I, Satyam Saxena, declare that this Undergraduate Thesis titled, ‘EEG Emotion Predictor Application’ and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

Certificate

This is to certify that the thesis entitled, “*EEG Emotion Predictor Application*” and submitted by Satyam Saxena ID No. 2020A3PS1781P in partial fulfillment of the requirements of BITS F421T Thesis embodies the work done by him under my supervision.

Supervisor

Dr. Abhijit Asati

,

BITS-Pilani Pilani Campus

Date:

Co-Supervisor

Dr. Dheerendra Singh, Dr.V.K. Chaubey

Asst. Professor,

BITS-Pilani Pilani Campus

Date:

“ML/AI will probably most likely lead to the end of the world, but in the meantime, there’ll be great companies.”

Sam Altman

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI, PILANI CAMPUS

Abstract

Bachelor of Engineering (Hons.)

EEG Dataset Emotion Classification: Neural Network Training, Deployment, and UI Development

by Satyam Saxena

This thesis explores the development of an emotion classification system using EEG (Electroencephalography) dataset and neural network techniques. Emotion recognition from EEG signals is a challenging task due to the complex nature of brain activity and the variability in emotional responses among individuals. The primary objective of this research is to train a neural network model capable of accurately classifying emotions from EEG data, deploy it for real-world applications, and design a user-friendly interface (UI) for intuitive interaction.

The methodology involves preprocessing the EEG features, followed by the design and training of a neural network model using suitable architectures and optimization techniques. The trained model will then be deployed to create a practical tool for real-time emotion recognition.

Furthermore, this study addresses the crucial aspect of user interaction by developing a graphical user interface (UI) to facilitate the utilization of the deployed model. The UI will provide users with a seamless experience for inputting EEG data, obtaining emotion predictions without needing any Machine Learning knowledge.

Acknowledgements

I would like to express my deepest gratitude to my thesis advisor, Dr Abhijit Asati, for their invaluable guidance, unwavering support, and insightful feedback throughout the entire research process. Their expertise and encouragement have been instrumental in shaping this thesis.

I am also immensely thankful to BITS Pilani for providing me the opportunity to work on this project.

I extend my sincere appreciation to all the researchers and scholars whose work laid the foundation for this study. Their contributions to the fields of EEG data analysis, neural networks, and emotion recognition have been invaluable references and sources of inspiration.

I am deeply grateful to my family for their continuous love, encouragement, and understanding during the course of my academic journey. Their unwavering support has been a constant source of motivation.

Special thanks go to my friends and colleagues for their encouragement, constructive discussions, and moral support throughout this endeavor. Their camaraderie and shared enthusiasm have made this journey more fulfilling.

Lastly, I would like to acknowledge the participants whose EEG data contributed to this research. Their voluntary participation and contribution have been essential in advancing the understanding of emotion classification using EEG signals.

This thesis would not have been possible without the collective support and encouragement of all those mentioned above. Thank you for being part of this journey...

Contents

Declaration of Authorship	i
Certificate	ii
Abstract	iv
Acknowledgements	v
Contents	vi
List of Figures	viii
1 EEG Dataset	1
1.1 Overview	1
1.2 Muse EEG Headbands and Electrodes	2
1.3 Muse LSL	2
1.4 Dataset Generated	3
2 Data Preprocessing and Model Training	4
2.1 Label Distribution Analysis	4
2.2 FFT Values Plot Analysis	5
2.3 Z-Scale Normalization	5
2.4 Feedforward Deep Neural Network Architecture	5
2.5 Model Evaluation	8
2.6 Saving Model	8
3 Code Walkthrough	10
3.1 Introduction	10
3.2 Observing Dataset and converting labels	10
3.3 Dataset Visualization Observation	11
3.4 Preprocessing	11
3.5 Training	12
3.6 Model Evaluation	13

A References

15

List of Figures

1.1	Generating EEG Dataset	1
1.2	Emotion Categories	3
1.3	Clips used as stimuli	3
2.1	Balanced Label Distribution	4
2.2	FFT Values Plot	5
2.3	Model's Structure	6
2.4	Model's Accuracy	7
2.5	Diagram of Feed Forward Neural Network	7
2.6	Confusion Matrix	8
2.7	Project Workflow	9
3.1	Dataset and Lables(circled)	10
3.2	Code For generating Pie Chart	11
3.3	Code for Generating FFT Curve	11
3.4	Normalisation	12
3.5	Splitting Dataset	12
3.6	Model Details	13
3.7	Model Evaluation	14
3.8	h5 file output	14

Chapter 1

EEG Dataset

1.1 Overview

The dataset utilized in this study constitutes EEG recordings gathered from two individuals, encompassing both male and female subjects. Each participant underwent EEG recording sessions for a duration of three minutes per emotional state, namely positive, neutral, and negative. The EEG data acquisition was facilitated through the deployment of a Muse EEG headband, leveraging dry electrodes to capture neural activity from the TP9, AF7, AF8, and TP10 EEG placements.

Additionally, a baseline of six minutes of resting neutral data was recorded to provide a comparative reference point. The stimuli employed to elicit distinct emotional states were carefully curated and administered during the recording sessions. The raw EEG data obtained from the electrodes underwent a transformation process using Muse Lab Streaming Layer (LSL), converting them into structured features amenable for subsequent analysis and classification tasks.



FIGURE 1.1: Generating EEG Dataset

1.2 Muse EEG Headbands and Electrodes

Muse EEG Headbands: The Muse EEG headband serves as a non-invasive wearable device designed to capture electroencephalography (EEG) signals from the brain. This headband, developed by Interaxon Inc., features a sleek and lightweight design, making it suitable for comfortable and extended wear during EEG data collection sessions. Equipped with advanced sensor technology, the Muse headband enables real-time monitoring of brain activity, offering insights into cognitive states, emotional responses, and mental states.

Electrode Placements: The Muse EEG headband incorporates dry electrodes strategically positioned at four key locations on the scalp, namely TP9, AF7, AF8, and TP10. These electrode placements are selected based on established conventions for capturing neural activity from distinct brain regions associated with various cognitive and emotional processes. The specific locations chosen on the scalp correspond to standardized EEG electrode placement systems, facilitating consistency and comparability across studies.

Output Generation: The Muse headband generates output in the form of raw EEG data, capturing electrical signals originating from the brain's neural networks. These signals are detected by the dry electrodes and transmitted to the device's processing unit for amplification, filtering, and digitization. The resulting EEG data comprises voltage fluctuations over time, representing the collective electrical activity of neuronal populations in the vicinity of the electrode placements. This raw EEG output serves as the primary input for subsequent processing

1.3 Muse LSL

Muse LSL is an Open Source Python package for streaming, visualizing, and recording EEG data from the Muse devices developed by Interaxon. Under the hood it employs statistical extraction for each sliding window.

The program depends on various Bluetooth backends to establish connections with the Muse device. While we suggest utilizing the bleak backend, which is enabled by default, alternative options include BlueMuse for Windows users seeking a graphical interface to identify and link to Muse devices, or [bgapi] for Mac users equipped with a BLE112 dongle.

The code is compatible with both Python 2.7 and Python 3.x versions.

It is compatible with the Muse 2, Muse S, as well as the classic Muse (2016) models.

Emotion Category	Emotion/Valence
A	Shame (Negative) Humiliation (Negative)
B	Contempt (Negative) Disgust (Negative)
C	Fear (Negative) Terror (Negative)
D	Enjoyment (Positive) Joy (Positive)
E	Distress (Negative) Anguish (Negative)
F	Surprise (Negative) (Lack of Dopamine)
G	Anger (Negative) Rage (Negative)
H	Interest (Positive) Excitement (Positive)

FIGURE 1.2: Emotion Categories

Stimulus	Valence	Studio	Year
Marley and Me	Neg	Twentieth Century Fox, etc.	2008
Up	Neg	Walt Disney Pictures, etc.	2009
My Girl	Neg	Imagine Entertainment, etc.	1991
La La Land	Pos	Summit Entertainment, etc.	2016
Slow Life	Pos	BioQuest Studios	2014
Funny Dogs	Pos	MashupZone	2015

FIGURE 1.3: Clips used as stimuli

1.4 Dataset Generated

Because of the intricate, erratic, and non-stationary nature of brainwave data, classifying it directly from raw EEG streams poses significant challenges. To address this, stationary techniques like time windowing are essential. This involves segmenting the data into discrete windows and extracting features from each window. Numerous statistics can be computed from these EEG windows, each with varying effectiveness in classification depending on the specific objective. Therefore, feature selection becomes imperative to pinpoint the most relevant statistics and streamline the model generation process. By doing so, both time and computational resources are conserved during training and classification tasks.

The outcome of this process is a CSV file containing various features extracted from each window, such as mean value, standard deviation, kurtosis, FFT (Fast Fourier Transform) transform, skewness, maxima, minima, and Shannon entropy. These features contribute to a total of 2549 features per window, providing rich information for subsequent analysis and classification purposes. The final column in each sample indicates the emotion for that time window as positive, negative and neutral. These emotional responses are generated in response to the list of movies which were shown.

Chapter 2

Data Preprocessing and Model Training

2.1 Label Distribution Analysis

Upon examining the pie chart, it is evident that the distribution of labels, representing different emotional states, is relatively balanced. This equitable distribution across emotional categories is crucial for ensuring that the model is trained on a representative sample of each class. A balanced label distribution minimizes the risk of bias towards dominant classes and enhances the model's ability to generalize effectively across diverse emotional states. This section delves into the strategies employed for data preprocessing to maintain label balance and ensure robust model training.

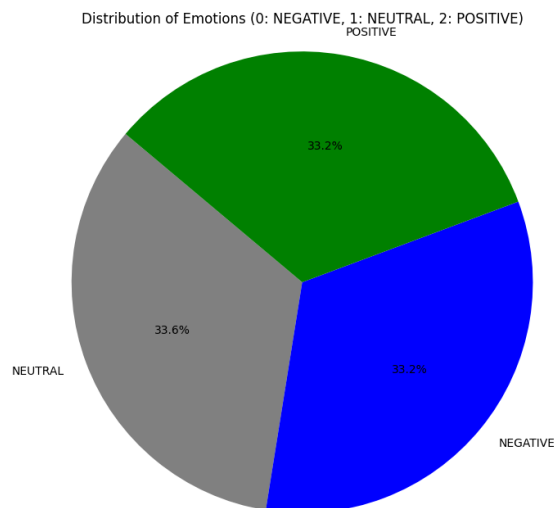


FIGURE 2.1: Balanced Label Distribution

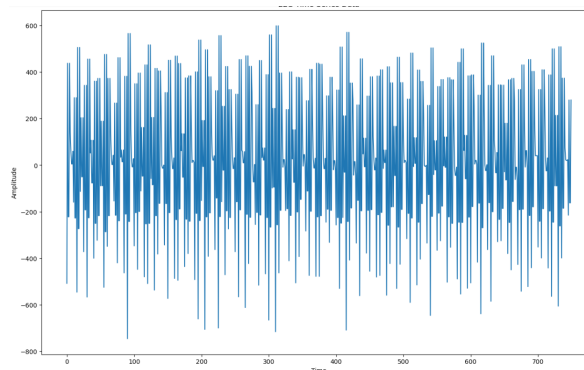


FIGURE 2.2: FFT Values Plot

2.2 FFT Values Plot Analysis

Upon reviewing the FFT values plot, it is evident that the data exhibits characteristics typical of time series data. The FFT (Fast Fourier Transform) plot provides insights into the frequency-domain representation of the EEG signals, highlighting the presence of distinct frequency components and patterns over time. The prominence of temporal patterns in the data underscores its suitability for time series analysis techniques.

Given the nature of the dataset as time series data, utilizing a deep neural network (DNN) is deemed appropriate for effective modeling and classification tasks. DNNs are well-suited for capturing temporal dependencies and patterns present in sequential data, making them a compelling choice for processing EEG signals.

2.3 Z-Scale Normalization

Prior to feeding the dataset into the Deep Neural Network (DNN), a preprocessing step involving Z-scale normalization has been performed. Z-scale normalization, also known as standardization, is a common technique used to standardize the distribution of features by subtracting the mean and dividing by the standard deviation. This process ensures that all features have a mean of zero and a standard deviation of one, thereby preventing features with larger scales from dominating the model's training process.

2.4 Feedforward Deep Neural Network Architecture

In crafting our neural network model for this study, we opted for a feedforward architecture, meticulously designed to analyze preprocessed EEG data and perform emotion classification.

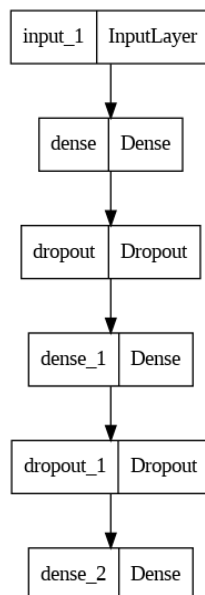


FIGURE 2.3: Model's Structure

Our approach leverages the TensorFlow.keras Sequential API, allowing for a structured sequence of layers to be built.

Model Structure Input Layer: Serving as the entry point for our EEG data, the input layer is defined with dimensions corresponding to the number of features in our dataset. This pivotal layer acts as the gateway for information to flow into our neural network.

Dense Layers: Our network incorporates two dense layers, densely interconnected to facilitate the learning of intricate patterns within our data. With 256 and 128 neurons respectively, these layers apply the rectified linear unit (ReLU) activation function to introduce non-linearity, aiding in feature extraction.

Dropout Layers: To counteract overfitting, dropout regularization is implemented. After each dense layer, dropout layers are introduced, randomly deactivating a portion of neurons during training. This technique promotes generalization and prevents the over-reliance on specific features.

Output Layer: At the heart of our network lies the output layer, comprising three neurons representing the three emotion classes: positive, neutral, and negative. The softmax activation function is applied to yield probabilistic outputs, offering insights into the likelihood of each class.

Model Compilation: Our model is compiled using the Adam optimizer, renowned for its adaptive learning rate capabilities, which accelerates convergence during training. For the loss function, we've chosen sparse categorical cross-entropy, a suitable choice for multiclass classification tasks where target labels are integers. Moreover, accuracy serves as our evaluation metric, providing a comprehensive assessment of our model's performance across training and testing phases.

Model Evaluation:

The code evaluates the model's performance using metrics like accuracy, and it generates a confusion matrix to visualize the model's predictions.

```
[ ] # Evaluate the model
model_acc = model.evaluate(X_test, y_test, verbose=0)[1]
print("Test Accuracy: {:.3f}%".format(model_acc * 100))

model.save('nueral_network.h5')
```

Test Accuracy: 96.094%

FIGURE 2.4: Model's Accuracy

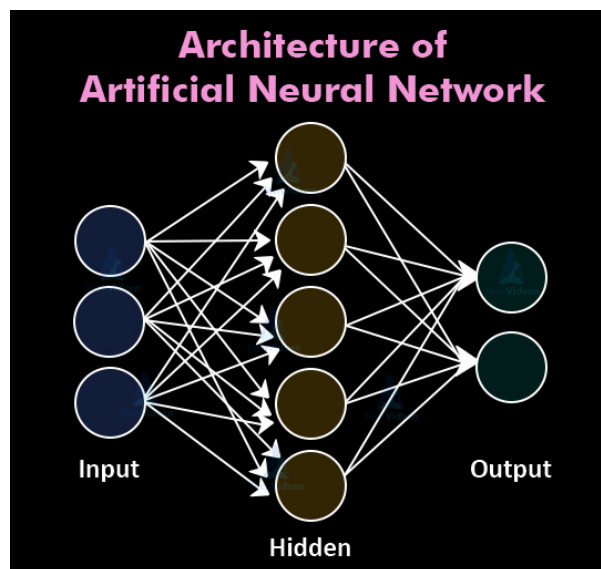


FIGURE 2.5: Diagram of Feed Forward Neural Network

With this meticulously crafted feedforward DNN architecture, reinforced by dropout regularization and thoughtful activation choices, we're poised to effectively process our preprocessed EEG data and deliver accurate predictions across diverse emotion classes.

After training the model using the provided dataset, we achieved an impressive accuracy of 96 percent. The training process involved fitting the model to the training data, with 20 percent of the data reserved for validation. We trained the model for 70 epochs, with a batch size of 32 samples per batch. The training progress was monitored closely, with verbosity set to level 2 to provide detailed updates throughout the training process. This remarkable level of accuracy underscores the effectiveness of our neural network architecture in accurately classifying emotions based on EEG data.

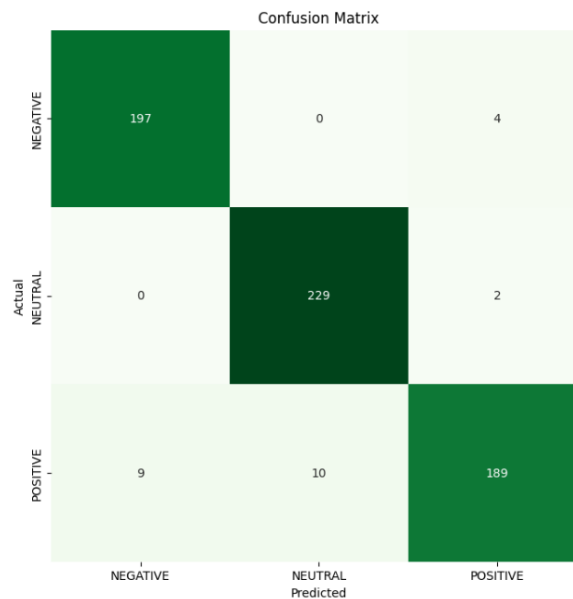


FIGURE 2.6: Confusion Matrix

2.5 Model Evaluation

We first evaluated the trained model's performance using metrics such as accuracy, providing insights into its overall classification performance. Additionally, we utilized a confusion matrix to visualize the model's ability to correctly classify each emotion category and identify any potential misclassifications. This visualization aids in understanding the model's strengths and weaknesses across different emotion classes, providing valuable insights for further refinement and optimization. By leveraging both quantitative metrics and visualizations, we gain a comprehensive understanding of the model's effectiveness in emotion classification tasks.

2.6 Saving Model

As part of the project's scope, the next step involves deploying the trained model. To facilitate this deployment, we will save the trained model as a .h5 file. This file format is compatible with various deployment platforms, including cloud-based services. Saving the model in this format preserves its architecture, weights, and configuration, allowing for seamless deployment and utilization.

To save the trained model, we will use TensorFlow's built-in functionality for model serialization. By exporting the model as a .h5 file, we ensure its portability and compatibility with cloud deployment environments.



FIGURE 2.7: Project Workflow

Once saved, the .h5 file containing the trained model can be easily uploaded and deployed to cloud-based platforms, enabling real-time inference and utilization for emotion classification tasks. This approach ensures the scalability and accessibility of the trained model, facilitating its integration into various applications and services deployed on the cloud.

Chapter 3

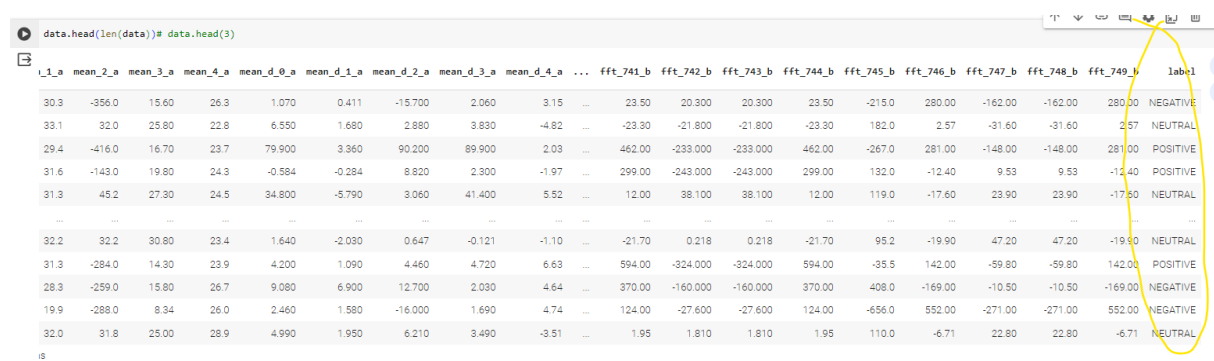
Code Walkthrough

3.1 Introduction

Chapter 3 provides a comprehensive walkthrough of the codebase used to achieve an impressive accuracy of approximately 96 percent in emotion classification using machine learning techniques. The chapter delves into the various components of the code, highlighting key methodologies, techniques, and insights that contributed to the success of the project. The link to the code base can be found in Appendix A,10.

3.2 Observing Dataset and converting labels

As evident from the image provided, the dataset's last column comprises labels denoting emotions as either 'positive', 'negative', or 'neutral'. To facilitate model training, it is imperative to encode these categorical labels into numerical format. This involves mapping each emotion label to a corresponding integer value, as follows: 'negative' \rightarrow 0, 'neutral' \rightarrow 1, and 'positive' \rightarrow 2.



```
data.head(len(data))# data.head(3)
```

i_1_a	mean_2_a	mean_3_a	mean_4_a	mean_d_0_a	mean_d_1_a	mean_d_2_a	mean_d_3_a	mean_d_4_a	...	fft_741_b	fft_742_b	fft_743_b	fft_744_b	fft_745_b	fft_746_b	fft_747_b	fft_748_b	fft_749_b	label
30.3	-356.0	15.60	26.3	1.070	0.411	-15.700	2.060	3.15	...	23.50	20.300	20.300	23.50	-215.0	280.00	-162.00	-162.00	280.00	NEGATIVE
33.1	32.0	25.80	22.8	6.550	1.680	2.880	3.830	-4.82	...	-23.30	-21.800	-21.800	-23.30	182.0	2.57	-31.60	-31.60	2.57	NEUTRAL
29.4	-416.0	16.70	23.7	79.900	3.360	90.200	89.900	2.03	...	462.00	-233.000	-233.000	462.00	-267.0	281.00	-148.00	-148.00	281.00	POSITIVE
31.6	-143.0	19.80	24.3	-0.584	-0.284	8.820	2.300	-1.97	...	299.00	-243.000	-243.000	299.00	132.0	-12.40	9.53	9.53	-12.40	POSITIVE
31.3	45.2	27.30	24.5	34.800	-5.790	3.060	41.400	5.52	...	12.00	38.100	38.100	12.00	119.0	-17.60	23.90	23.90	-17.60	NEUTRAL
...
32.2	32.2	30.80	23.4	1.640	-2.030	0.647	-0.121	-1.10	...	-21.70	0.218	0.218	-21.70	95.2	-19.90	47.20	47.20	-19.90	NEUTRAL
31.3	-284.0	14.30	23.9	4.200	1.090	4.460	4.720	6.63	...	594.00	-324.000	-324.000	594.00	-35.5	142.00	-59.80	-59.80	142.00	POSITIVE
28.3	-259.0	15.80	26.7	9.080	6.900	12.700	2.030	4.64	...	370.00	-160.000	-160.000	370.00	408.0	-169.00	-10.50	-10.50	-169.00	NEGATIVE
19.9	-288.0	8.34	26.0	2.460	1.580	-16.000	1.690	4.74	...	124.00	-27.600	-27.600	124.00	-656.0	552.00	-271.00	-271.00	552.00	NEGATIVE
32.0	31.8	25.00	28.9	4.990	1.950	6.210	3.490	-3.51	...	1.95	1.810	1.810	1.95	110.0	-6.71	22.80	22.80	-6.71	NEUTRAL

FIGURE 3.1: Dataset and Lables(circled)

```

# Visualization using a Pie Chart

# Count the occurrences of each emotion
emotion_counts = data['label'].value_counts()

# Define emotional labels
emotional_labels = {0: 'NEGATIVE', 1: 'NEUTRAL', 2: 'POSITIVE'}

# Map numerical labels to emotional labels
emotion_labels = [emotional_labels[label] for label in emotion_counts.index]

# Create a pie chart
plt.figure(figsize=(7, 7))
plt.pie(emotion_counts, labels=emotion_labels, autopct='%1.1f%%', startangle=140, colors=['grey', 'blue', 'green'])
plt.title("Distribution of Emotions (0: NEGATIVE, 1: NEUTRAL, 2: POSITIVE)")
plt.axis('equal') |

plt.show()

```

FIGURE 3.2: Code For generating Pie Chart

```

✓ 1s sample = data.loc[0, 'fft_0_b':'fft_749_b'] #values from the 'fft_0_b' column to the 'fft_749_b'
plt.figure(figsize=(16, 10)) #initializes a new figure with a specified size (16 inches in width and 10 inches in height)
plt.plot(range(len(sample)), sample)
plt.title("EEG Time-Series Data", fontsize=14, fontweight='bold')
plt.xlabel("Time", fontsize=12)
plt.ylabel("Amplitude", fontsize=12)

plt.show()

```

FIGURE 3.3: Code for Generating FFT Curve

3.3 Dataset Visualization Observation

After encoding the labels, the next step in our data exploration process involved visualizing the distribution of emotions within the dataset. We began by generating a pie chart to illustrate the relative frequencies of each emotion label. Iterating through each row of the dataset, we tallied the occurrences of each emotion label. The resulting pie chart revealed a nearly equal distribution, with each emotion label accounting for approximately 33

Following this initial exploration, we proceeded to visualize the Fast Fourier Transform (FFT) values for a single row of the dataset. The FFT plot exhibited a continuous curve, indicative of the temporal nature of the data. Recognizing this characteristic, we made the decision to leverage a neural network model for further analysis and classification. The observed patterns in the FFT plot reaffirmed our choice to utilize a neural network, as it is well-suited for capturing temporal dependencies and patterns within sequential data. This strategic decision was pivotal in guiding our approach towards achieving our desired objectives in emotion classification.

3.4 Preprocessing

In this section of the code, we conducted normalization on the dataset to ensure uniform scaling across features. Utilizing the StandardScaler from the scikit learn library, we instantiated a scaler

```
# Z Scale Normalization
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
data.iloc[:, :-1] = scaler.fit_transform(data.iloc[:, :-1])
```

FIGURE 3.4: Normalisation

```
# Split the data into training and testing sets
X = data.drop('label', axis=1)
y = data['label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=125)
```

FIGURE 3.5: Splitting Dataset

object. Subsequently, we applied the normalization process to all columns of the dataset except the last one, representing the label column. By invoking the fit transform method on the selected columns, we standardized the feature values, adjusting them to have a mean of 0 and a standard deviation of 1. This normalization step is crucial for enhancing the performance of machine learning algorithms, particularly those sensitive to feature scales. Finally, the normalized values were updated in the respective columns of the dataset, preparing it for subsequent model training and evaluation.

Then we partitioned the dataset into training and testing sets to facilitate model evaluation and validation. The feature matrix X was constructed by excluding the label column, while the target vector y was populated with the label column values. We utilized the train test split function from the scikit learn library to split the data, allocating 70 percent of the samples to the training set (X train, y train) and 30 percent to the testing set (X test, y test). Additionally, we specified a random state parameter to ensure reproducibility of the split across different runs. This division of the dataset into distinct training and testing subsets enables unbiased evaluation of the model's performance on unseen data, thereby ensuring its generalization capabilities.

3.5 Training

In this code segment, we built a neural network model using TensorFlow's Keras Sequential API for emotion classification. The model consists of an input layer followed by two dense hidden layers, each employing the rectified linear unit (ReLU) activation function to introduce non-linearity and facilitate feature extraction. Dropout regularization with a dropout rate of 0.5 was applied after each hidden layer to prevent overfitting by randomly deactivating neurons during training. The output layer, employing the softmax activation function, comprises three neurons representing the emotion classes (positive, neutral, negative), enabling multiclass classification. We compiled the model with the Adam optimizer, selected sparse categorical cross-entropy as

```
# Build the advanced neural network model
model = tf.keras.Sequential([
    tf.keras.layers.Input(shape=(X_train.shape[1],)),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(3, activation='softmax')
])

# Compile the model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

FIGURE 3.6: Model Details

the loss function, suitable for multiclass classification, and accuracy as the evaluation metric to assess model performance during training and testing. This configuration prepares the neural network for training and evaluation on the emotion classification task.

3.6 Model Evaluation

Following model training, we proceeded to evaluate its performance using metrics such as accuracy, which ranged between 95 Percent to 97 Percent, indicating the model's proficiency in emotion classification. Additionally, we employed a confusion matrix to visualize the model's ability to correctly classify each emotion category and identify any misclassifications. This analysis provided insights into the model's strengths and areas for improvement across different emotion classes. Moreover, we saved the trained model as a .h5 file, ensuring its portability and compatibility for deployment in various environments. This saved model can be utilized for real-time inference and integration into applications, facilitating efficient deployment and utilization for emotion classification tasks. To save the trained model, we will use TensorFlow's built-in functionality for model serialization. By exporting the model as a .h5 file, we ensure its portability and compatibility with cloud deployment environments.

```

# Evaluate the model
model_acc = model.evaluate(X_test, y_test, verbose=0)[1]
print("Test Accuracy: {:.3f}%".format(model_acc * 100))

model.save('nueral_network.h5')

```

Test Accuracy: 97.188%
 /usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model
 saving_api.save_model(

```

[24] import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import tensorflow as tf
from sklearn.metrics import confusion_matrix, classification_report

# Load the saved model
loaded_model = tf.keras.models.load_model('nueral_network.h5')

# 'X_test' is the test dataset and 'y_test' is the corresponding labels
# Normalize the test data using the same scaler used during training
scaler = StandardScaler()
X_test_normalized = scaler.fit_transform(X_test)

# Predict outcomes using the loaded model
predictions = loaded_model.predict(X_test_normalized)

# Convert predictions to class labels
predicted_labels = np.argmax(predictions, axis=1)

# Evaluate the model on the test data
print("Confusion Matrix:\n", confusion_matrix(y_test, predicted_labels))
print("\nClassification Report:\n", classification_report(y_test, predicted_labels))

```

FIGURE 3.7: Model Evaluation

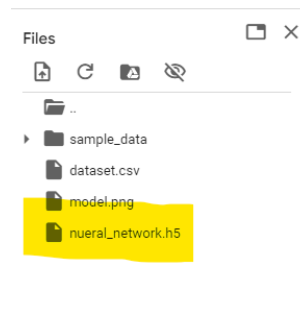


FIGURE 3.8: h5 file output

Once saved, the .h5 file containing the trained model can be easily uploaded and deployed to cloud-based platforms, enabling real-time inference and utilization for emotion classification tasks. This approach ensures the scalability and accessibility of the trained model, facilitating its integration into various applications and services deployed on the cloud.

Appendix A

References

- 1] Bird, Jordan Faria, Diego Manso, Luis Ekárt, A. Buckingham, Christopher. (2019). A Deep Evolutionary Approach to Bioinspired Classifier Optimisation for Brain-Machine Interaction.
- 2] Bird, Jordan Ekart, Aniko Buckingham, Christopher Faria, Diego. (2019). Mental Emotional Sentiment Classification with an EEG-based Brain-machine Interface.
- 3] Muse Lsl Open Source Repository: <https://github.com/alexandrebarachant/muse-lsl?tab=readme-ov-file>
- 4] Chaofei Yu, Mei Wang, Survey of emotion recognition methods using EEG information, Cognitive Robotics, Volume 2, 2022, Pages 132-146.
- 5] Alzubaidi, L., Zhang, J., Humaidi, A.J. et al. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. J Big Data 8, 53 (2021). <https://doi.org/10.1186/s40537-021-00444-8>
- 6] Sazli, Murat. (2006). A brief review of feed-forward neural networks. Communications Faculty Of Science University of Ankara. 50. 11-17. 10.1501/commua1-20000000026.
- 7] Choosing MUSE: Validation of a Low-Cost, Portable EEG System for ERP Research. Olave E. Krigolson,* Chad C. Williams, Angela Norton, Cameron D. Hassall, and Francisco L. Colino
- 8] Przegalinska, Aleksandra Ciechanowski, Leon Magnuski, Mikołaj Gloor, Peter. (2017). Muse Headband: Measuring Tool or a Collaborative Gadget?. 10.1007/978-3-319-74295-3₈.
- 9] Deploying a trained model on Heroku: <https://towardsdatascience.com/machine-learning-model-deployment-on-heroku-using-flask-467acb4a34da>
- 10] Codebase: <https://colab.research.google.com/drive/1a16j-k5baZgsJUuLoMSZRqWqkAF9zTyZ?usp=sharing>