EMOTION DETECTION USING EEG SIGNALS : A NEW APPROACH

Daksh Singhal[1], Ashish Astang[1], Shivam Yadav[1]

Netaji Subhas University of Technology, Dwarka, Delhi, 110078, India

Abstract

This research investigates the effectiveness of different neural network architectures in conjunction with feature selection techniques for accurately classifying emotions from EEG signals. Leveraging the EEG Brainwave Dataset, we evaluate the performance of various combinations, focusing on Principal Component Analysis (PCA) with Long Short-Term Memory (LSTM), and Recurrent Neural Networks (RNN). PCA-LSTM and PCA-RNN achieve accuracies of 86% and 99%, respectively, emphasizing the robustness of LSTM in capturing temporal dependencies and the suitability of RNNs for this task. Through comprehensive analysis, including confusion matrices, we provide insights into each scenario's performance and the models' ability to distinguish between emotional states. These findings underscore the significance of feature selection strategies like PCA in conjunction with specific neural network architectures for precise emotion classification from EEG signals.

Keywords:
EEG Signals, Principal Component Analysis (PCA), Recurrent Neural Networks (RNN), Long Short Term Memory (LSTM), Activation Function , Loss Functions , Optimizers , Hyperparameter Tuning , TensorFlow , Keras

1. Introduction

Emotion recognition through EEG signals has emerged as a promising avenue for understanding human affective states. In this research paper, we delve into the intricate task of emotion prediction by utilizing the EEG Brainwave Dataset: Feeling Emotions. The dataset incorporates three distinct emotion classes - neutral, happy, and sad. Our objective is to harness the power of advanced neural network architectures, specifically recurrent neural network (RNN) and long short-term memory (LSTM) to unravel patterns within the EEG signals and predict emotional states accurately. Our approach involves a comprehensive preprocessing strategy to distill meaningful features from the raw EEG data. But due to huge noise in the input data it becomes extremely difficult to find meaningful patterns in the data. After applying FFT (Fast Fourier Transform) the noise reduction is significant and the performance of many commonly used Machine Learning models did improve but there is always a room for improvement. A commonly

Email address: daksh.singhal.ug21@nsut.ac.in, ashish.astang.ug21@nsut.ac.in, shivam_yadav.ug21@nsut.ac.in (Shivam Yadav)

used technique called PCA can be deployed here for this data preprocessing purpose. Use of PCA significantly reduces training time, the input dimensions of the data i.e. it ends up making compact and relevant data that compiles most of the features to reduce the dimensions of the problem thereby simplifying the problem dataset by great extent. Demonstrating its efficiency in handling the inherent complexity of EEG data, the detailed metrics, including confusion matrices, precision, recall, and F1 score, are elaborated. Through this research, we aim to contribute valuable insights into the domain of EEG-based emotion prediction, emphasizing the importance of thoughtful feature selection and dimensionality reduction techniques for improve model performance. Furthermore, to achieve improved results, we will be using RNN and LSTM as our models for training purposes. RNN and LSTM hold great potential for analyzing meaningful patterns within a data and are best suited for our problem in hand. Our aim in this paper is to explore the possibility of improving the performance of RNN and LSTM by performing some data preprocessing.

2. Literature Survey

The field of emotion recognition is increasingly using electroencephalography (EEG) due to its safety and real-time capabilities. EEG signals provide a window into the brain's emotional responses. Researchers are actively exploring ways to improve emotion prediction accuracy using EEG data. These efforts involve analyzing brain signal patterns with mathematical tools like averages, standard deviations, and Fast Fourier Transforms (FFTs). Additionally, computer models such as Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTMs), and standard Neural Networks (ANNs) are being tested for their effectiveness. Some studies also utilize Principal Component Analysis (PCA) to simplify the data and enhance machine learning models' learning process. Despite the progress, challenges remain, including individual brain signal variations and noise interference. Our research aims to contribute to this field by experimenting with different features, models, and data processing techniques to advance our understanding of emotions through brainwave signals. Previous studies by Klibi, Bird, Sarkar, and Yang have demonstrated the potential of machine learning (ML) in deciphering emotions from EEG data.Klibi achieved an impressive 97Building upon this existing research, our work aims to surpass these achievements. We will leverage the valuable insights from these studies to guide our own investigation. Additionally, several papers suggest that meticulous preprocessing of EEG signals can significantly improve results. This reinforces the notion that effective data preparation is crucial for success.

Research being carried out in the field of brain computer interface gearing more towards BCI applications [3]. The EEG research community is diversifying their applications into many different subdomains. EEG signals are normally used for detecting stress as seen in [16], [9], and they suggest a strong correlation between stress and EEG signals. Koelstra et al. [8] used Hilbert-Huang Transform (HHT) to remove artifacts and perform cleaning. HHT is a time-frequency analysis method, which extracts the intrinsic mode functions (IMFs) that produce well-behaved Hilbert transforms from the signals that have been extracted, using an empirical mode decomposition. With the Hilbert transformation, the IMF gives instantaneous frequencies as a function of time.

Their use of hierarchical Support Vector Machines (SVM) achieved much better results than a linear SVM, resulting in an accuracy of 89%. Taking into account multiple different parameters is necessary, even when considering classic machine learning algorithms. Their findings inspired us to look into the use of cross-validation within a grid search and apply it to our experiments. Liao et al. [9] made use of Fast Fourier Transform and deep learning with back propagation to figure out the best way to sooth stress, but only obtained a 75% precision. In the field of motor imagery, EEG signals can be utilized to operate robotic arms [7]. Fakhruzzaman et al.[6] explained how BCI can identify and distinguish brain waves of people when they are performing different tasks. Their implementation was based on an automative EPOC to test out if the classifier "SVM" can distinguish between two major training activities: moving the left hand and moving the right foot, and four more combinations of the same two actions coupled with the addition of noise, like nodding or moving right foot along with left hand movement.

It is also observed that faces can be seen even when there are none, suggesting EEG can be used while seeing optical illusions [5] and also to find out the more active areas of the brain, detecting early visual processing (before one can think subconsciously). P100 and N170 perceptual signatures are found to be of much more importance than others. In motor imagery, deep learning has been used as seen in [14], which proposed a high-level goal of finding robust representations from EEG data, that would be invariant to inter and intra-subject differences and to inherent noise associated with EEG data collection. Specifically, rather than representing low-level EEG features as a vector, they transformed the data into a sequence of topology-preserving multi-spectral images (EEG "movie"), as opposed to classic EEG analysis techniques that disregard such spatial info. EEG signals are also widely seen in sleep pattern analysis as in [4] , [1], which allowed for better detection and identification of situational causes where subjects had better or worse sleep cycles. This research lead to various sleep improvement mobile applications that help people develop different exercises to get the optimal sleep, an interesting note is that Convolutional Neural Networks were used to extract time-invariant features, and bidirectional-Long Short-Term Memory in order to automatically predict the sleep transition stages From EEG epoch data. They used two public sleep datasets and used a two-step training algorithm. The first step involved training the model to learn filters to extract time-invariant features from the raw uni-channel EEG epochs. The second step involved sequence residual learning, useful for encoding the stage transition rules of sleep from a sequence of EEG epochs in the extracted features. It is interesting to note that CNNs require images as raw data and to convert EEG signals to images of 2D projection map, or azimutul 2D projection as explained in [15].

Another recent classifier is ICLabel [11], which is open source and runs on MatLab. ICLabel improves upon existing classifiers by suggesting approaches that result in an increased accuracy of the computed label estimates and enhancing its computational efficiency. Their classifier outperforms the existing automated IC component classification method for all measured IC categories while increasing the speed of computation speed by nearly 10 times. More complicated methods involving Monte Carlo simulations were seen in [14], where a comparison on three different machine learning techniques were performed for detecting epileptic seizure risks and further goes on to determine if 1-h screening is more feasible, to identify patients with less seizure risk (less than 5% seizures risk in 48 h). The different methods they used were the elastic net regression (EN), Critical Care EEG Monitoring Research Consortium (CCEMRC), and multicenter with a dataset of 7716 continuous EEGs (cEEG), and neural networks and sparse linear integer model (RiskSLIM). These methods performed relatively similar in terms of evaluation metrics, but RiskSlim using 2HELPS2B achieves slightly better results. EEG signals are also seen to help prevent seizures and seizure type prediction [18, 19]. These articles discussed how data collected from intracranial EEG-based monitoring systems can be used to predict epilepsy episodes in patients up to an hour earlier. They have used a number of different classification techniques with a wide variety of features from the dataset, both univariate and bivariate for their prediction models.

Many other papers have explained various degrees of success, ranging from 50 to 85% through the use of different classification techniques. Further improvement in accuracy was seen in Belakhdar et al.[2] gave a maximum accuracy of 86.5% during the detection of drowsiness. Additionally SVM and ANN were used in comparison after performing fast Fourier transformations to get vector of 9 features, suggesting that ANN though more robust did not provide drastic accuracy improvements. In contrast the work by Li et al. [11] suggests that for the WAY-EEG-GAL dataset, an approach based on AlexNet works better giving an accuracy of 96%. However the approach used cannot exactly suggest that neural nets work better since the data preparation technique used was tailored more towards feature extraction before classification. An algorithm based on entropy called wavelet packet decomposition was seen in [10] which improved the accuracy of SVM to 87–93% for the sample subjects tested. Promising results were also seen in Jin et al. [13] which suggested that a combination of FFT, PCA, and SVM gave results of nearly 90%. Therefore in conclusion the accuracy of any model is highly reliant on the feature extraction stage and not necessarily on how complex of a classification technique used is as implied in [12]. Thus reliable accuracy and recall can even be achieved with classification techniques.

3. Algorithms, test problems and comparison criteria

Despite advancements in using brainwave signals (EEG) for emotion prediction, challenges persist in achieving accurate and efficient models. Existing studies explore various features and neural network architectures, yet there's a need to systematically understand their impact. Questions arise regarding the effectiveness of different feature sets, the choice of neural network models, and the role of preprocessing techniques like Principal Component Analysis (PCA). Additionally, challenges such as individual variability and signal noise pose hurdles. This study addresses these gaps by comprehensively investigating the interplay between features, models, and preprocessing methods to advance our understanding and improve the accuracy of EEG-based emotion prediction models. Algorithms: EEG Feature Extraction:

Time-domain features: Mean, standard deviation, skewness, kurtosis. Frequency-domain features: Power spectral density, spectral entropy. Time-frequency domain features: Wavelet transform coefficients. Neural Network Architectures: Convolutional Neural Networks (CNN). Long Short-Term Memory networks (LSTM). Hybrid CNN-LSTM architectures. Attention mechanisms in neural networks. Preprocessing Techniques: Principal Component Analysis (PCA). Independent Component Analysis (ICA). Wavelet denoising. Artifact removal techniques. Test Problems: Emotion Recognition Datasets: DEAP (Database for Emotion Analysis using Physiological signals). SEED (SEED dataset for EEG-based Emotion Recognition). MAHNOB-HCI (Multimodal Affective Human-Computer Interaction). Cross-validation Methods: k-fold cross-validation. Leave-one-subject-out cross-validation. Leave-one-trial-out cross-validation. Comparison Criteria: Accuracy and Performance Metrics: Classification accuracy. F1-score, precision, recall. Receiver Operating Characteristic (ROC) curves. Computational Efficiency: Training time. Inference time. Model complexity and resource requirements. Robustness to Noise and Variability: Performance on noisy EEG signals. Generalization across different subjects and sessions. Sensitivity to electrode placement and signal artifacts. Impact of Feature Sets: Comparative analysis of different feature extraction methods. Assessment of feature importance for emotion prediction. Effectiveness of Preprocessing Techniques: Comparison of models with and without preprocessing. Evaluation of the impact of dimensionality reduction techniques like PCA on model performance. Interpretability and Explainability: Visualizations of learned features or attention weights. Identification of discriminative EEG patterns associated with specific emotions. Scalability and Generalization: Assessment of model performance on unseen datasets. Investigation of transfer learning techniques for emotion prediction across different datasets.

4. Methodology

4.1. PCA

Principal Component Analysis (PCA) is a widely used technique in the field of statistics and machine learning for dimensionality reduction and data visualization. It is a linear transformation method that aims to identify the directions (or principal components) in which the data varies the most. PCA achieves this by finding the orthogonal axes along which the data has the maximum variance. The primary objective of PCA is to reduce the dimensionality of high-dimensional datasets while preserving as much of the original variance as possible. By projecting the data onto a lower-dimensional subspace spanned by the principal components, PCA enables a more compact representation of the data, facilitating easier visualization, analysis, and interpretation. For our problem statement , we used PCA as a method to reduce the dimensions of our initial Data and then applying suitable models for evaluating their accuracy.

4.2. MODELS USED:

1) **RNN:**
Recurrent Neural Networks (RNNs) represent a powerful class of artificial neural networks designed to handle sequential data processing tasks. Their architecture includes recurrent connections that enable information to persist over time, allowing them to model temporal dependencies effectively. The training of RNNs involves **backpropagation through time (BPTT)**, albeit facing challenges such as vanishing gradients. To address this issue, advanced variants like Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs) have been introduced, offering mechanisms to capture long-term dependencies and mitigate gradient-related problems.

In practical applications, RNNs have demonstrated remarkable success across diverse fields such as natural language processing, time series analysis, and speech recognition. Their ability to model sequential data has led to advancements in tasks like text generation, sentiment analysis, and predictive analytics. Moving forward, continued research in RNNs aims to enhance their performance, scalability, and interpretability, opening avenues for innovation in sequential data analysis and fostering breakthroughs in artificial intelligence. Our problem statement being a time series analysis problem, RNN is a promising candidate for evaluating the results of our experiment.

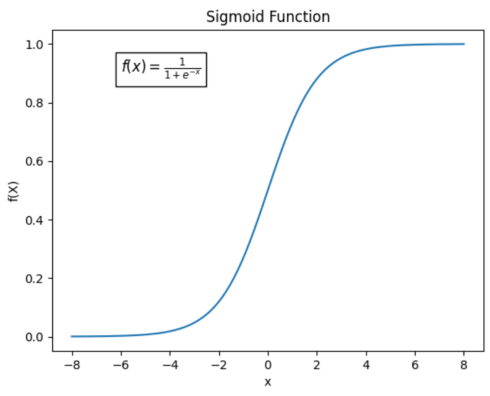
2) **LSTM:**
Long Short-Term Memory (LSTM) networks constitute a specialized variant of recurrent neural networks (RNNs) designed to address the limitations of standard RNNs in capturing long-term dependencies. LSTMs introduce a memory cell and gating mechanisms, including input, forget, and output gates, to regulate the flow of information within the network. This architecture enables LSTMs to retain information over extended time periods, making them particularly effective for tasks requiring the modeling of long-range dependencies. During training, LSTMs employ **backpropagation through time (BPTT)**, supplemented by mechanisms such as gradient clipping, to mitigate issues like vanishing gradients commonly encountered in RNN training.

In practical applications, LSTMs have exhibited outstanding performance across various domains, including natural language processing, time series prediction, and speech recognition. Their ability to capture complex temporal patterns has led to significant advancements in tasks such as language translation, stock market forecasting, and speech synthesis. In our case, LSTM seems to be a promising candidate for our problem statement.

4.3. ACTIVATION FUNCTIONS USED:

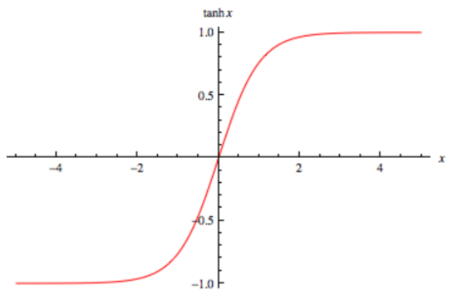
1)**SIGMOID:**

$$f(x) = \frac{1}{1 + e^{-x}} \tag{1}$$



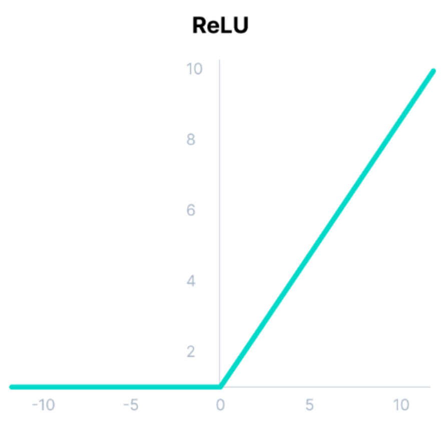
2)**TANH:**

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2}$$



3)**RELU:**

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$



where x is the input vector

*4.4. **OPTIMIZERS USED :***

1. **Adam:**

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \tag{4}$$

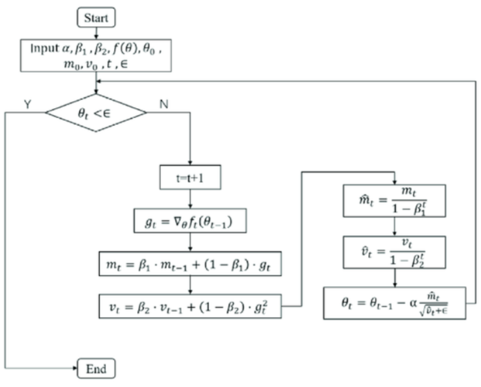
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \tag{5}$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{6}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{7}$$

$$\theta_{t+1} = \theta_t - \frac{\alpha \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{8}$$

where:
- m_t and v_t are the first and second moments of the gradients, respectively.
- g_t is the gradient of the parameters at time step t.
- β_1 and β_2 are exponential decay rates for the moment estimates.
- α is the learning rate.
- ϵ is a small constant to prevent division by zero.


Start
Input $\alpha, \beta_1, \beta_2, f(\theta), \theta_0$, m_0, v_0, t, ϵ
$\theta_t < \epsilon$
Y
N
t=t+1
$g_t = \nabla_\theta f_t(\theta_{t-1})$
$m_t = \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t$
$v_t = \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t^2$
$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$
$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$
$\theta_t = \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}}$
End

2. **Stochastic Gradient Descent (SGD):**

$$\theta_{t+1} = \theta_t - \alpha \cdot g_t \tag{9}$$

where:
- θ_t is the parameters at time step t.
- g_t is the gradient of the parameters at time step t.
- α is the learning rate.

3. **RMSProp:**

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma)g_t^2 \tag{10}$$

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{E[g^2]_t + \epsilon}} \cdot g_t \tag{11}$$

where:
- $E[g^2]_t$ is the exponentially decaying average of squared gradients.
- γ is the decay rate.
- α is the learning rate.
- ϵ is a small constant to prevent division by zero.

4. **Adadelta:**

$$E[g^2]_t = \rho E[g^2]_{t-1} + (1-\rho)g_t^2 \tag{12}$$

$$\Delta x_t = -\frac{\sqrt{E[\Delta x^2]_{t-1} + \epsilon}}{\sqrt{E[g^2]_t + \epsilon}} \cdot g_t \tag{13}$$

$$\theta_{t+1} = \theta_t + \Delta x_t \tag{14}$$

where:
- $E[g^2]_t$ is the exponentially decaying average of squared gradients.
- $E[\Delta x^2]_t$ is the exponentially decaying average of squared parameter updates.
- ρ is the decay rate.
- ϵ is a small constant to prevent division by zero.

4.5. LOSS FUNCTIONS USED :

1. **Binary Crossentropy:**

$$L(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)) \tag{15}$$

where: - N is the number of samples.
- y_i is the true label of the i-th sample (0 or 1).
- $\hat{y}_i$ is the predicted probability of the i-th sample being in class 1.
- log represents the natural logarithm.

2. **Sparse Categorical Crossentropy:**

$$L(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}\log(\hat{y}_i) \tag{16}$$

where: - N is the number of samples.
- $\hat{y}_i$ is the predicted probability distribution for the i-th sample.
- log represents the natural logarithm.

3. **Categorical Crossentropy:**

$$L(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log(\hat{y}_{ij}) \tag{17}$$

where: - N is the number of samples.
- C is the number of classes.
- y_{ij} is the true label for the i-th sample and j-th class (1 if the sample belongs to class j, 0 otherwise).
- $\hat{y}_{ij}$ is the predicted probability of the i-th sample being in class j.
- log represents the natural logarithm.

4. **Kullback-Leibler Divergence:**

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_i P(i)\log\left(\frac{P(i)}{Q(i)}\right) \tag{18}$$

where: - $P(i)$ and $Q(i)$ are the probabilities of the event i under the distributions P and Q, respectively.
- $\sum_i$ represents the sum over all events i.

5. **Hinge Loss:**

$$L(y, \hat{y}) = \frac{1}{N}\sum_{i=1}^{N}\max(0, 1 - y_i \cdot \hat{y}_i) \tag{19}$$

where: - N is the number of samples.
- y_i is the true label of the i-th sample (-1 or 1).

- $\hat{y}_i$ is the predicted score for the i-th sample.

6. **Weighted Categorical Crossentropy:**

$$L(y, \hat{y}) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C} w_j \cdot y_{ij} \log(\hat{y}_{ij}) \tag{20}$$

where: - N is the number of samples.
- C is the number of classes.
- w_j is the weight for class j.
- y_{ij} is the true label for the i-th sample and j-th class (1 if the sample belongs to class j, 0 otherwise).
- $\hat{y}_{ij}$ is the predicted probability of the i-th sample being in class j.
- log represents the natural logarithm.

7. **Focal Loss:**

$$L(y, \hat{y}) = -\frac{1}{N} \sum_{i=1}^{N} \alpha_i \cdot (1 - \hat{y}_i)^{\gamma} \log(\hat{y}_i) \tag{21}$$

where: - N is the number of samples.
- α_i is the balancing parameter for the i-th sample.
- γ is the focusing parameter.
- $(1 - \hat{y}_i)^{\gamma}$ is the modulating factor that reduces the loss for well-classified examples.

8. **Squared Hinge Loss:**

$$L(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} \max(0, 1 - y_i \cdot \hat{y}_i)^2 \tag{22}$$

where: - N is the number of samples.
- y_i is the true label of the i-th sample (-1 or 1).
- $\hat{y}_i$ is the predicted score for the i-th sample.

4.6. Parameters To Judge Performance of a Model

- **Precision**:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

- **Recall** (also known as Sensitivity or True Positive Rate):

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

- **F1-Score** (harmonic mean of precision and recall):

$$\text{F1-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- **Support**: Number of actual occurrences of the class in the specified dataset.
- **Accuracy**:

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}}$$

*4.7. **MISCELLANEOUS INFORMATION***

1. **Dataset and Data Acquisition:** The EEG Brainwave Dataset: Feeling Emotions, a publicly available dataset, was utilized for this research. EEG signals were captured using a commercially available MUSE EEG headband, employing four dry extra-cranial electrodes (TP9, AF7, AF8, and TP10) to record micro voltage measurements from the Frontal and Temporal lobes of the brain. Data was recorded for 60 seconds from two subjects (1 male and 1 female, aged 20-22) during the viewing of six film clips, resulting in 12 minutes (720 seconds) of brain activity data. Three film clips induced negative emotions, three contained positive scenes, and neutral emotion data were recorded during one minute rest intervals between emotional states. Subjects were instructed to refrain from conscious movements to prevent the influence of Electromyographic (EMG) signals on the EEG data.

2. **Preprocessing:** The dataset was prepossessed to extract relevant features, considering the full-time window, two half-windows, and quarter windows. The following features were extracted: Full-time window features: Sample mean, standard deviation, skewness, kurtosis, maximum, minimum, sample variances, and covariance of each signal. Eigen-values of the covariance matrix. Upper triangular elements of the matrix logarithm of the covariance matrix. Magnitude and frequency components obtained using Fast Fourier Transform (FFT). Frequency values of the ten most energetic components of the FFT. Two half-windows features: Changes in sample means, standard deviations, maximum and minimum values between the first and second half-windows for all signals. Quarter windows features: Sample mean of each quarter-window and paired differences of sample means between quarter windows for all signals. Maximum (minimum) values of each quarter window and paired differences of maximum (minimum) values between quarter windows for all signals.

3. **Neural Network Implementation**: Three different neural network architectures were implemented using TensorFlow: Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM)

4. **Scenarios:** Two scenarios were considered for model training and evaluation:
   (a) ***Case 1:*** Using just Fast Fourier Transform (FFT) features
   (b) ***Case 2:*** Using Principal Component Analysis (PCA) with 96% retained variance
5. **Finding out how many principal features are needed for 96% Variance retention:** we figure out this by repeatedly applying PCA on our data for finding out that at what minimum number of principal features, we can have a total explained variance of 96%. The optimal value was found to be n = 45.

6. **Performing Train Test Spilt:** The dataset under consideration was as 2132 row deep and 2549 column long. For this, 80% of data was used for training and 20 for testing i.e. 1705 rows for training and 427 rows for testing.

7. **Creating RNN and LSTM of relevant dimensions and depths**: We created a 2 layered RNN and LSTM using the inbuilt functionalities of TensorFlow and Keras. We Selected suitable Loss functions and optimizers along with the appropriate number of perceptron units in each layer.

Table 1: RNN WITH PCA MODEL SUMMARY

Layer (type)	**Output Shape**	**Param**
SimpleRNN	(None, 45, 65)	4355
SimpleRNN	(None, 50)	5800
Dense	(None, 3)	153
Total params:		10,308
Trainable params:		10,308
Non-trainable params:		0

Table 2: LSTM WITH PCA MODEL SUMMARY

Layer (type)	**Output Shape**	**Param**
LSTM	(None, 45, 65)	17,420
LSTM	(None, 65)	34,060
Dense	(None, 3)	198
Total params:		51,678
Trainable params:		51,678
Non-trainable params:		0

8. **Hyperparameter Tuning of RNN and LSTM:** We used the library called keras-tuner for performing hyperparameter tuning of the 3 parameters: Loss Function, Optimizer and most importantly : the number of perceptrons in each layer.

   SAMPLE CODE SNIPPET FOR HYPERPARAMETER TUNING:

```
activation  = ['relu','tanh']
loss_functions = [
    'binary_crossentropy',
    'sparse_categorical_crossentropy',
    'categorical_crossentropy',
    'kullback_leibler_divergence',
    'hinge',
    'weighted_categorical_crossentropy',
    'focal_loss',
    'squared_hinge'
]
```

```
Optimizer = [
            'adam',
            'sgd',
            'rmsprop',
            'adadelta']
def unit_set(hp):
    model = Sequential()

    model.add(SimpleRNN(units= 45,
                        activation=hp.Choice('activation_1',activation),
                        return_sequences=True,
                        input_shape=(45, 1)))

    model.add(SimpleRNN(units=30,
                        activation=hp.Choice('activation_2,',activation)))

    model.add(Dense(3,
                    activation='softmax'))

    model.compile(optimizer =hp.Choice('optimize',Optimizer),
                  loss = hp.Choice('loss',Loss),
                  metrics=['accuracy'])
    return model

tune = keras_tuner.GridSearch(hypermodel=unit_set,
                              objective=['val_accuracy'],
                              overwrite=True,
                              directory='RRN_HyperParameter_Data_With_PCA',
                              project_name = 'LossAndOptimizer')
tune.search(x_train,y_train,epochs=15,validation_data = (x_test,y_test))
hyp_param = tune.get_best_hyperparameters(num_trials=1)[0].values

def unit_set(hp):
    model = Sequential()
    unit1 = hp.Int('unit1',min_value = 45, max_value = 65, step = 10)
    unit2 = hp.Int('unit2',min_value = 10, max_value = 50, step = 10)
    model.add(SimpleRNN(units= unit1,
                        activation=hyp_param['activation_1'],
                        return_sequences=True,
                        input_shape=(45, 1)))

    model.add(SimpleRNN(units=unit2,
                        activation=hyp_param['activation_2,']))

    model.add(Dense(3,
                    activation='softmax'))
    model.compile(optimizer =hyp_param['optimize'],
                  loss = hyp_param['loss'],
                  metrics=['accuracy'])
    return model

tune = keras_tuner.GridSearch(hypermodel=unit_set,
                              objective=['val_accuracy'],
                              overwrite=True,
                              directory='RRN_HyperParameter_Data_With_PCA',
                              project_name = 'Units')

tune.search(x_train,y_train,epochs=15,validation_data = (x_test,y_test))

model = tune.get_best_models(num_models=1)[0]
history = model.fit(x_train,y_train,epochs=15,batch_size=40)
model.save("RNN_with_pca.h5")
```

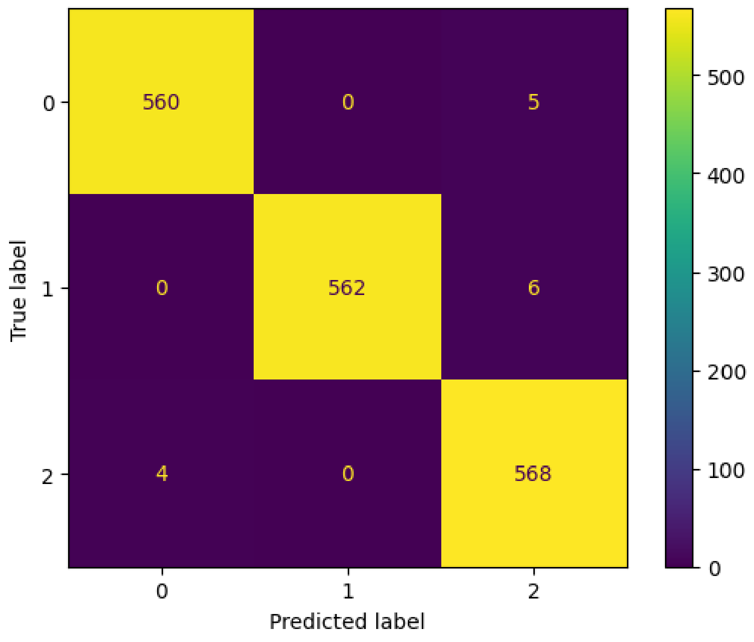
5. Conclusion and Future Work

5.1. RNN WITH PCA

Training Accuracy = 99.12%
The classification report for the training data is presented in Table 3.

Table 3: Classification Report for Training Data

Class	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.99	0.99	0.99	565
1	1.00	0.99	0.99	568
2	0.98	0.99	0.99	572
Accuracy			0.99	1705
Macro Avg	0.99	0.99	0.99	1705
Weighted Avg	0.99	0.99	0.99	1705

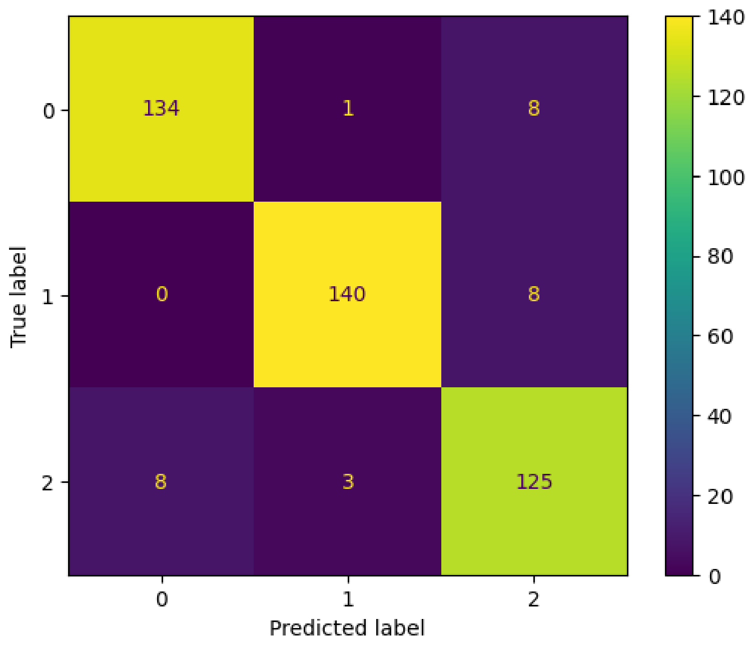


Testing Accuracy = 93.44%

The classification report for the Testing data is presented in Table 4.

Table 4: Classification Report for Testing Data

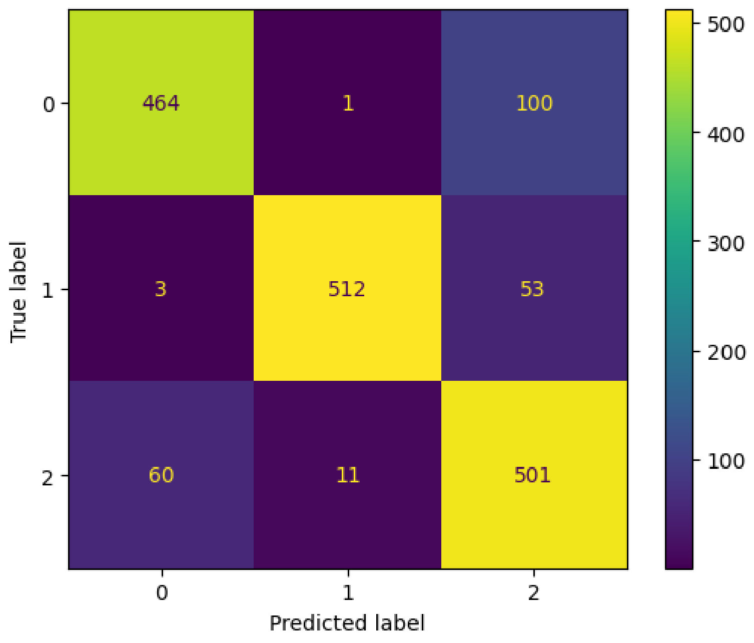
Class	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.94	0.94	0.94	143
1	0.97	0.95	0.96	148
2	0.89	0.92	0.90	136
Accuracy			0.93	427
Macro Avg	0.93	0.93	0.93	427
Weighted Avg	0.94	0.93	0.93	427



Training Accuracy = 86.62%
The classification report for the training data is presented in Table 5.

Table 5: Classification Report for Training Data

Class	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.88	0.82	0.85	565
1	0.98	0.90	0.94	568
2	0.77	0.88	0.82	572
Accuracy			0.87	1705
Macro Avg	0.87	0.87	0.87	1705
Weighted Avg	0.87	0.87	0.87	1705

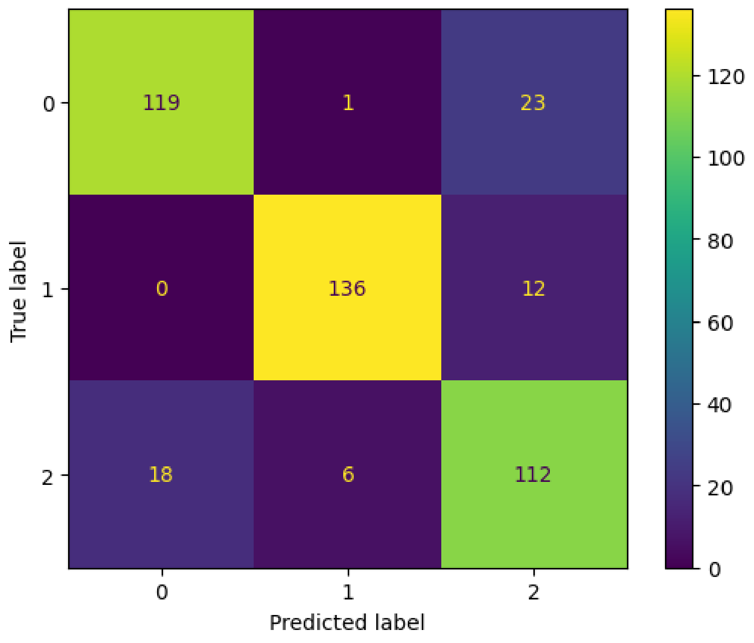


Testing Accuracy = 85.94%

The classification report for the testing data is presented in Table 6.

Table 6: Classification Report for Testing Data

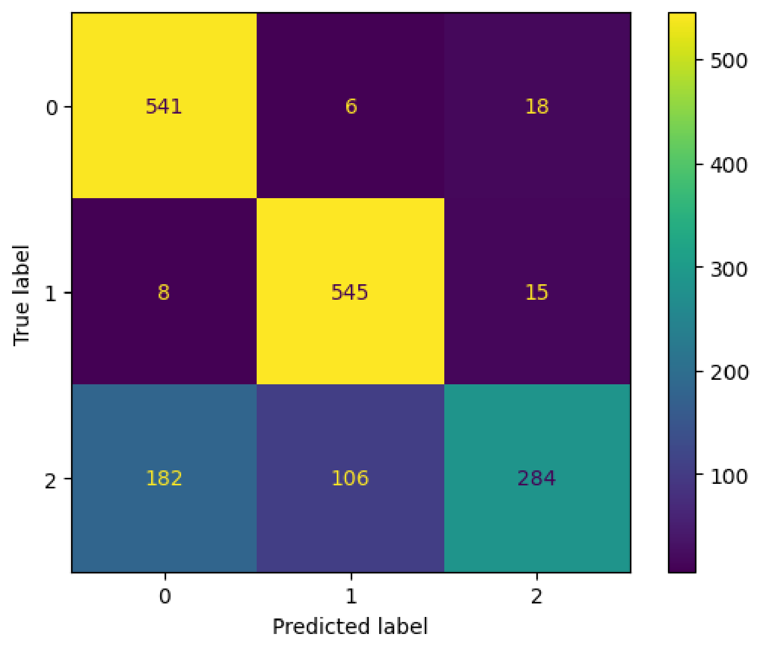
Class	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.87	0.83	0.85	143
1	0.95	0.92	0.93	148
2	0.76	0.82	0.79	136
Accuracy			0.86	427
Macro Avg	0.86	0.86	0.86	427
Weighted Avg	0.86	0.86	0.86	427



Training Accuracy = 80.35%
The classification report for the training data is presented in Table 7.

Table 7: Classification Report for Training Data

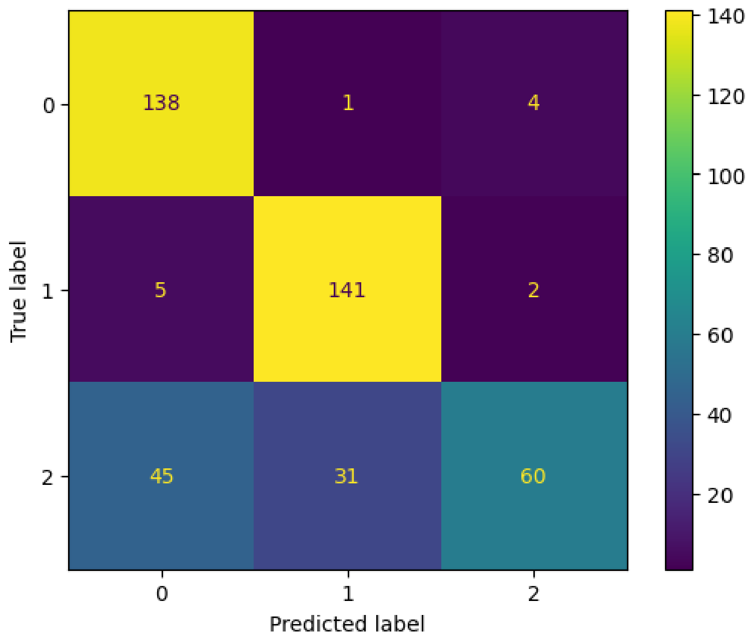
Class	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.74	0.96	0.83	565
1	0.83	0.96	0.89	568
2	0.90	0.50	0.64	572
Accuracy			0.80	1705
Macro Avg	0.82	0.80	0.79	1705
Weighted Avg	0.82	0.80	0.79	1705



Training Accuracy = 80.35%
The classification report for the testing data is presented in Table 8.

Table 8: Classification Report for Testing Data

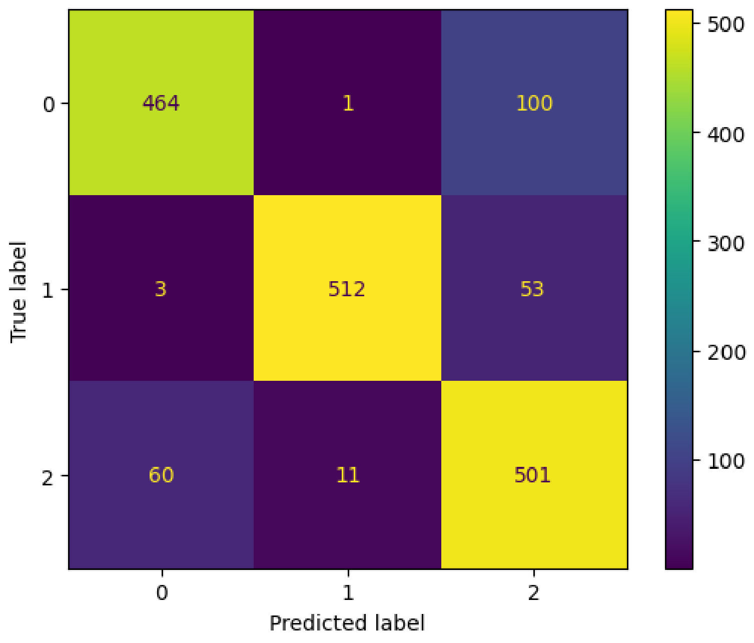
Class	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.73	0.97	0.83	143
1	0.82	0.95	0.88	148
2	0.91	0.44	0.59	136
Accuracy			0.79	427
Macro Avg	0.82	0.79	0.77	427
Weighted Avg	0.82	0.79	0.77	427



Training Accuracy = 81.62%
The classification report for the training data is presented in Table 9.

Table 9: Classification Report for Training Data

Class	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.88	0.82	0.85	565
1	0.98	0.90	0.94	568
2	0.77	0.88	0.82	572
Accuracy			0.82	1705
Macro Avg	0.80	0.83	0.82	1605
Weighted Avg	0.81	0.80	0.79	1665

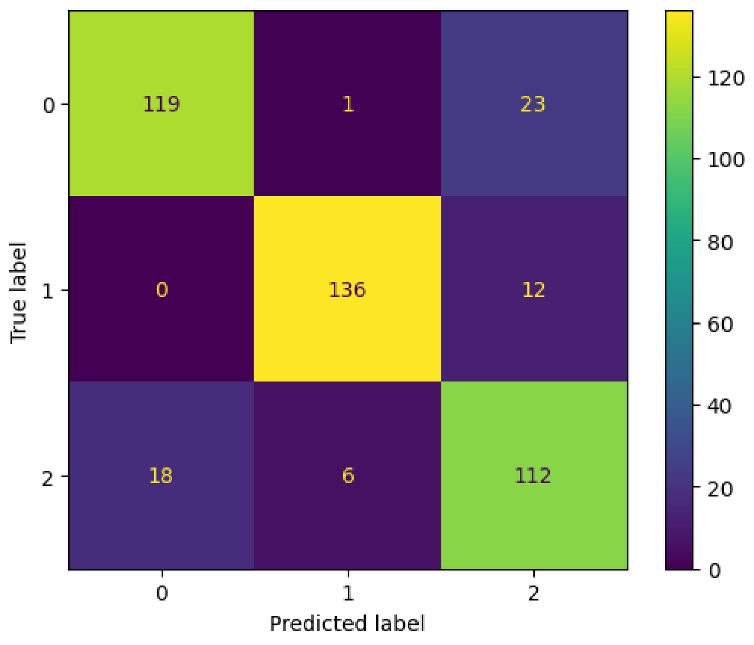


Testing Accuracy = 80.94%

The classification report for the testing data is presented in Table 10.

Table 10: Classification Report for Testing Data

Class	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.87	0.83	0.85	143
1	0.95	0.92	0.93	148
2	0.76	0.82	0.79	136
Accuracy			0.80	417
Macro Avg	0.81	0.84	0.80	427
Weighted Avg	0.82	0.80	0.76	404



5.5. Comparison of Accuracies with and without PCA

Table 11: Comparison of Accuracies with and without PCA

Model	**With PCA**	**Without PCA**
RNN	99.12%	81.62%
LSTM	86.62%	80.30%

Here we can clearly see that there is an remarkable improvement in the outcomes after applying PCA. on the EEG signals.

5.6. *Future Works and Research Gaps*

Despite of achieving high levels of accuracy , there's a huge scope of improvement . LSTM despite being a complex model fails to generalize for the cases where the Test case is happy and tends to misclassify the data . Further training such models is computationally very expensive therefore training complex , huge and deep model is far from reality without a state of the art infrastructure. A deep , complex and huge model may be appropriate in trapping meaningful patterns in the EEG data and help us exploring a pattern in it which may provide some meaningful insights to the emotional patterns of humans.

References

[1] Pouya Bashivan, Irina Rish, Mohammed Yeasin, and Noel Codella. Learning representations from eeg with deep recurrent-convolutional neural networks. *arXiv preprint arXiv:1511.06448*, 2015.

[2] Ikram Belakhdar, Walid Kaaniche, Raouia Djmel, and B Ouni. A comparison between ann and svm classifier for drowsiness detection based on single eeg channel. pages 443–6, 2016.

[3] Dimitris Bertsimas, Jack Dunn, and Athanasios Paschalidis. Regression and classification using optimal decision trees. In *2017 IEEE MIT undergraduate research technology conference (URTC)*, pages 1–4, 2017.

[4] Sylvain Chambon, Valentin Thorey, Pierre Jean Arnal, Emmanuel Mignot, and Alexandre Gramfort. A deep learning architecture to detect events in eeg signals during sleep. In *2018 IEEE 28th international workshop on machine learning for signal processing (MLSP)*, pages 1–6. IEEE, 2018.

[5] Alex Daros, Konstantine Zakzanis, and Anthony Ruocco. Facial emotion recognition in borderline personality disorder. *Psychol Med*, 43:1953–63, 2013.

[6] MN Fakhruzzaman, E Riksakomara, and H Suryotrisongko. Eeg wave identification in human brain with emotiv epoc for motor imagery. *Procedia Comput Sci*, 72:269–76, 2015.

[7] W Jia et al. Electroencephalography (eeg)-based instinctive brain-control of a quadruped locomotion robot. In *2012 annual international conference of the IEEE engineering in medicine and biology society*, pages 1777–81. IEEE, 2012.

[8] Simon Koelstra et al. Deap: a database for emotion analysis; using physiological signals. *IEEE Trans Affect Comput*, 3:18–31, 2011.

[9] C-Y Liao, R-C Chen, and S-K Tai. Emotion stress detection using eeg signal and deep learning technologies. In *2018 IEEE international conference on applied system invention (ICASI)*, pages 90–3. IEEE, 2018.

[10] Maja Pantic and Leon J Rothkrantz. Automatic analysis of facial expressions: the state of the art. *IEEE Trans Pattern Anal Mach Intell*, 22:1424–45, 2000.

[11] Luca Pion-Tonachini, Kenneth Kreutz-Delgado, and Scott Makeig. Iclabel: an automated electroencephalographic independent component classifier, dataset, and website. *NeuroImage*, 198:181–97, 2019.

[12] Kay Schaaff and Tanja Schultz. Towards emotion recognition from electroencephalographic signals. In *2009 3rd international conference on affective computing and intelligent interaction and workshops*, pages 1–6. IEEE, 2009.

[13] S Shariat, V Pavlovic, T Papathomas, A Braun, and P Sinha. Sparse dictionary methods for eeg signal classification in face perception. In *2010 IEEE international workshop on machine learning for signal processing*, pages 331–6. IEEE, 2010.

[14] Aaron F Struck et al. Comparison of machine learning models for seizure prediction in hospitalized patients. *Ann Clin Transl Neurol*, 67:1239–47, 2019.

[15] J Thomas et al. Eeg classification via convolutional neural network-based interictal epileptiform event detection. In *2018 40th annual international conference of the IEEE engineering in medicine and biology society (EMBC)*, pages 3148–51. IEEE, 2018.

[16] V Vanitha and P Krishnan. Real time stress detection system based on eeg signals. 2016.