A Novel Hybrid Deep Neural Network Classifier for EEG Emotional Brain Signals

Mahmoud A. A. Mousa[1], Abdelrahman T. Elgohr[2], Hatem A. Khater[3]
Faculty of Engineering, Zagazig University, Zagazig, Egypt[1, 2]
Faculty of Engineering, Horus University, Damietta Egypt[2, 3]
School of Mathematical and Computer Sciences, Heriot Watt University, Dubai, UAE[1]

Abstract—**The field of brain computer interface (BCI) is one of the most exciting areas in the field of scientific research, as it can overlap with all fields that need intelligent control, especially the field of the medical industry. In order to deal with the brain and its different signals, there are many ways to collect a dataset of brain signals, the most important of which is the collection of signals using the non-invasive EEG method. This group of data that has been collected must be classified, and the features affecting changes in it must be selected to become useful for use in different control capabilities. Due to the need for some fields used in BCI to have high accuracy and speed in order to comply with the environment's motion sequences, this paper explores the classification of brain signals for their usage as control signals in Brain Computer Interface research, with the aim of integrating them into different control systems. The objective of the study is to investigate the EEG brain signal classification using different techniques such as Long Short-Term Memory (LSTM), Convolutional Neural Networks (CNN), as well as the machine learning approach represented by the Support Vector Machine (SVM). We also present a novel hybrid classification technique called CNN-LSTM which combines CNNs with LSTM networks. This proposed model processes the input data through one or more of the CNN's convolutional layers to identify spatial patterns and the output is fed into the LSTM layers to capture temporal dependencies and sequential patterns. This proposed combination uses CNNs' spatial feature extraction and LSTMs' temporal modelling to achieve high efficacy across domains. A test was done to determine the most effective approach for classifying emotional brain signals that indicate the user's emotional state. The dataset used in this research was generated from a widely available MUSE EEG headgear with four dry extra-cranial electrodes. The comparison came in favor of the proposed hybrid model (CNN-LSTM) in first place with an accuracy of 98.5% and a step speed of 244 milliseconds/step; the CNN model came in the second place with an accuracy of 98.03% and a step speed of 58 milliseconds/step; and in the third place, the LSTM model recorded an accuracy of 97.35% and a step speed of 2 sec/step; finally, in last place, SVM came with 87.5% accuracy and 39 milliseconds/step running speed.**

Keywords—BCI; EEG; Brain Signals Classification; SVM; LSTM, CNN; CNN-LSTM

I. Introduction

Brain Computer Interface (BCI) is a technology that enables direct communication between the brain and an external device using signals generated from the brain. It has been proposed as a potential therapeutic treatment for various neurological disorders and a tool for efficient human-computer interaction. BCI technology can be used to control assistive devices such as wheelchairs, prostheses and communication systems, as well as to monitor brain activity and diagnose neurological diseases. Moreover, BCI technology can be used to provide a more natural form of human-computer interaction, allowing users to control computers with thoughts [1]. BCI technology can be divided into two main categories as shown in Fig.1: invasive and noninvasive. Invasive BCI requires the insertion of electrodes into the brain in order to capture brain signals, which is a risky and complicated process. On the other hand, noninvasive BCI relies on measuring signals from the scalp or other parts of the body to detect brain activities. Noninvasive BCI is more commonly used and includes electroencephalography (EEG), magnetoencephalography (MEG), and functional near-infrared spectroscopy (fNIRS). EEG is the most widely used BCI technique and is based on electrical signals generated by the brain [2].

EEG signals are a type of electrical activity that can be measured from the brain. They are used in a variety of engineering fields, including medical, robotics, and computer engineering. In medical engineering, EEG signals are used to diagnose and monitor neurological conditions. EEGs can be used to detect seizures, diagnose sleep disorders, and monitor brain activity during surgery. EEGs can also be used to measure brain activity during cognitive tasks, such as memory tests. This can help doctors better understand how the brain works and how to treat neurological conditions [3]. In robotics engineering, EEG signals are used to control robotic devices. By measuring the electrical activity of the brain, robots can be programmed to respond to certain commands. This can be used to create robots that can interact with humans in a more natural way. For example, robots can be programmed to respond to facial expressions or voice commands [4]. In computer engineering, EEG signals are used to create brain-computer interfaces. These allow users to control computers with their thoughts. This technology is still in its early stages, but it has the potential to revolutionize the way we interact with computers [5].

The most common EEG signal classification methods as shown in Fig. 2 are supervised learning algorithms, such as Support Vector Machines (SVMs), Artificial Neural Networks (ANNs), and decision trees. These algorithms are used to identify patterns in EEG signals that can be used to diagnose and monitor neurological conditions [6]. For example, SVMs can be used to classify EEG signals into different categories, such as normal or abnormal, or to detect changes in EEG signals over time. ANNs, which include Convolutional Neural Network (CNN), can be used to identify patterns in EEG signals that can be used to diagnose and monitor neurological conditions. Decision trees can be used to identify patterns in EEG signals that can be used to diagnose and monitor neurological conditions [7].

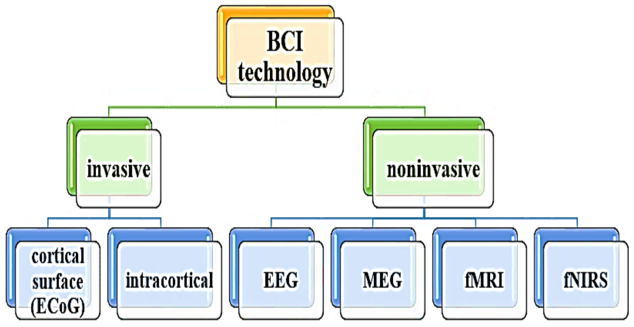

Fig. 1. Brain computer interface technology [1].

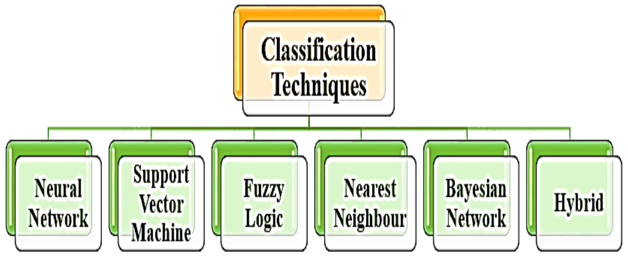

Fig. 2. Dataset classification techniques [8].

A. *Related Work*

Z. -T. Liu et al. (2019) tested a proposed approach on DEAP dataset, classifying Valance and Arousal emotional states using K-nearest neighbor and support vector machine. The experiments compare temporal windows of different lengths and three EEG signal rhythms. The results show that the EEG signal with one temporal window has the highest recognition accuracy of 86.46%. A multimodal emotional communication-based humans-robots interaction system would use the suggested approach for real-time emotion identification [9].

T. Song et al. (2020) to recognize emotions in multichannel EEG data used a dynamical graph convolutional neural network (DGCNN). Our EEG emotion recognition method uses a graph to describe multichannel EEG data and classify emotions using this model. EEG emotion recognition is improved by learning new features from the adjacency matrix. Emotion EEG datasets SEED and DREAMER were extensively studied. The proposed recognition method is more accurate than current methods. On SEED, it averaged 90.4% in subject-dependent experiments and 79.95% in subject-independent cross-validation [10].

In 2020, S. K. Khare and colleagues introduced an adaptive tunable Q wavelet transform for selecting tuning parameters automatically. Grey wolf optimization identifies the best tuning parameters. GWO tuning parameters divide EEG signals into sub bands. Time-domain properties of SB are inputted into a multiclass least-squares support vector machine. Evaluating the classification accuracy of four main emotions - happiness, fear, sadness, and relaxation - compared to current methods. A radial basis function kernel that outperforms prior methods on the same dataset achieves an accuracy of 95.70%. This article presents a nonparametric method for decomposing EEG signals to improve efficiency. This approach can enhance the progress of BCI system development by utilizing machine learning techniques [11].

Chowdary MK, et al., (2022) aim to classify emotions from electroencephalogram signals by utilizing different recurrent neural network structures. Three architectures employed in this study for emotion recognition using EEG signals are RNN (recurrent neural network), LSTM (long short-term memory network), and GRU (gated recurrent unit). Experimental data confirmed the efficiency of these networks in terms of performance measures. The study utilized the EEG Brain Wave Dataset: Feeling Emotions and obtained an average accuracy of 95% for RNN, 97% for LSTM, and 96% for GRU in detecting emotions [12].

EEG capture and emotion categorization in a simulated driving environment is suggested by Chen J. et al. (2024) to study panic emotion and accident-avoidance skills. The program models obstacle avoidance at different risk levels using vehicle speed. The system models the brain's physiological structure for data processing using graph neural networks (GNN) with functional connection and attention mechanisms. Various research compared entropy and power properties. The top single-label F1 score was 76.7%, and the three-class classification was 75.26 % accurate. Binary classification had 91.5% accuracy and the highest F1 score for a single label was 91.86%. Deep learning algorithms can accurately mimic hazardous events, record the driver's EEG data, and quickly track emotional states, according to experiments [13].

This research investigates classifying brain signals for use as control signals in Brain-Computer Interface (BCI) systems designed for various robotic applications. The aim is to compare four methods for multi-class classification: Long Short-Term Memory (LSTM) and Convolutional Neural Networks (CNN) from deep learning, a proposed hybrid CNN-LSTM approach, and Support Vector Machine (SVM) from machine learning. Ultimately, this research seeks to determine the most effective method for classifying emotional brain signals that reflect the user's emotional state.

The rest of this paper is organized as follows: Section II demonstrates the main concepts for signals classification overview; Section III presents the classification models; Section IV describes the dataset; Sections V and VI elaborate the classification results and a discussion of the results generated from the tests; Section VII mentions the applications that can benefit from this research topic; Section VIII concludes the paper and presents the future work.

II. Classification Overview

Dataset classification is a process of organizing data into categories based on certain characteristics. It is a way of organizing data into meaningful groups so that it can be more easily analyzed and understood. Dataset classification is used in a variety of fields, including data mining, machine learning, and artificial intelligence.

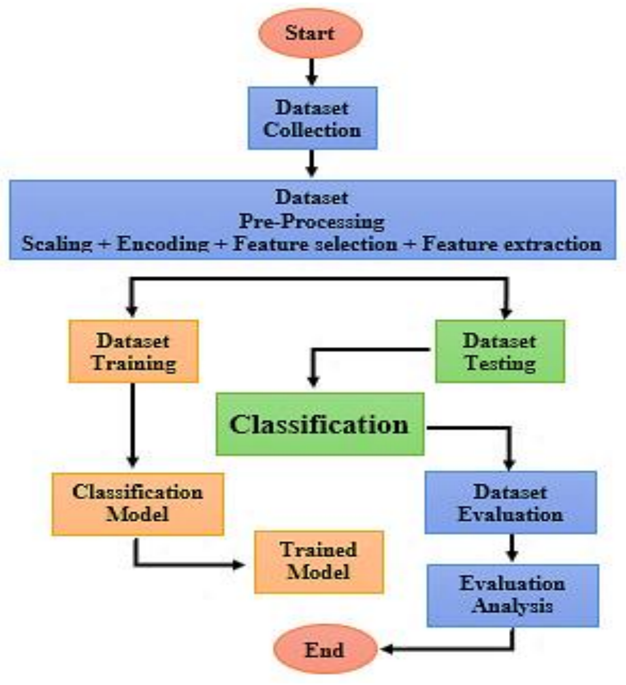

Fig. 3. Dataset classification process overview [14].

The process of dataset classification, as shown in Fig. 3 begins with the identification of the data that needs to be classified. This data can come from a variety of sources, such as databases, text documents, images, and audio files. Once the data has been identified, it is then divided into categories based on certain characteristics. These characteristics can include size, type, content, and other attributes.

Once the data has been divided into categories, it is then analyzed to determine the relationships between the different categories. This analysis can be done using a variety of techniques, such as clustering, decision trees, and neural networks. The goal of this analysis is to identify patterns and trends in the data that can be used to make predictions or decisions. Once the data has been classified and analyzed, it can then be used for a variety of purposes. For example, it can be used to create predictive models, to identify customer segments, or to detect anomalies in the data. It can also be used to create visualizations of the data, such as charts and graphs, which can be used to better understand the data [15].

III. Classification Models

A. *Support Vector Machine (SVM)*

Support Vector Machines (SVMs) are a powerful and versatile machine learning algorithm used for classification and regression tasks. SVMs are a supervised learning algorithm that can be used to classify data into two or more classes. They are based on the concept of finding a hyperplane that best divides a dataset into two classes. The main advantage of SVMs is that they are very effective in high dimensional spaces. This is because they use a kernel trick to map the data into a higher dimensional space, where it can be separated by a hyperplane. This allows them to capture complex relationships between the data points [16]. SVMs are also very robust to overfitting. This is because they use a regularization parameter which helps to reduce the complexity of the model and prevent overfitting. SVMs are also very efficient in terms of both time and memory. This is because they only need to store a subset of the training data, which makes them very efficient in terms of memory usage. In addition, SVMs are very versatile and can be used for a variety of tasks such as classification, regression, and outlier detection [17].

Building a Support Vector Machine (SVM) algorithm with Python as shown in Algorithm 1 [18], is a relatively straightforward process. The first step is to import the necessary libraries. The most common libraries used for SVM in Python are Scikit-learn, Numpy, and Matplotlib. Once the libraries are imported, the next step is to prepare the data. This involves loading the data into a Pandas Data Frame, cleaning the data, and splitting it into training and testing sets. It is important to ensure that the data is properly scaled and normalized before training the model. The next step is to create the SVM model. This is done by instantiating an SVM classifier object from the Scikit-learn library. The classifier object can then be fitted to the training data using the fit() method. Once the model is trained, it can be used to make predictions on the test data. This is done by calling the predict() method on the classifier object. The predictions can then be evaluated using a variety of metrics such as accuracy, precision, recall, and F1 score [19].

Algorithm 1: SVM model

```
Input: X (array of input data (features)), Y (array of output data
(classes - labels))
Output: performance of model (accuracy – precision – confusion
matrix)
1.  Function:
    Training _ SVM
            clf = svm.SVC(kernel='kernal type')
            clf.fit(X_train, y_train)

2.  Initialize:
    Learning rate – Number of runs (epoch)
        for i in X array
            if (Y(i) x X(i) x q) > 1
            then
update: q = q + learning rate x ((X(i)*Y(i))*(-2*(1/epoch)*q)
            else
update: q = q + learning rate x (-2*(1/epoch)*q)
            end if
            end
```

In the context of multi-class classification, SVMs can be used to construct a maximum-margin hyperplane that divides the feature space into regions, each corresponding to a particular class. The algorithm then searches for the optimal hyperplane that maximizes the margin between the classes. This hyperplane is then used to classify new data points. The advantage of SVMs is that they can be used to classify data with a large number of features and classes, as well as data with non-linear boundaries. Furthermore, SVMs are robust to outliers and can be used to classify data with a high degree of accuracy [19], [20].

B. Long Short Term Memory (LSTM)

The Long Short-Term Memory (LSTM) classifier is a powerful deep learning algorithm that can be used to classify data. It is a type of recurrent neural network (RNN) that is capable of learning long-term dependencies in data. The LSTM classifier is a powerful tool for predicting and classifying data, and it has been used in a variety of applications, such as natural language processing, speech recognition, time series forecasting, and classifying sequences of data, such as text, audio, and video. As shown in Fig. 4, the LSTM classifier is composed of a series of memory cells, each of which contains a set of weights and biases. The weights and biases are adjusted during the training process to learn the patterns in the data [21]. The memory cells are connected in a chain, and each cell is connected to the next cell in the chain. This allows the network to remember information from previous cells and use it to make predictions [22]. The LSTM classifier is trained using a supervised learning algorithm. During the training process, the network is presented with a set of input data and the desired output. The network then adjusts the weights and biases of the memory cells to learn the patterns in the data. Once the training is complete, the network can be used to make predictions on new data [23].

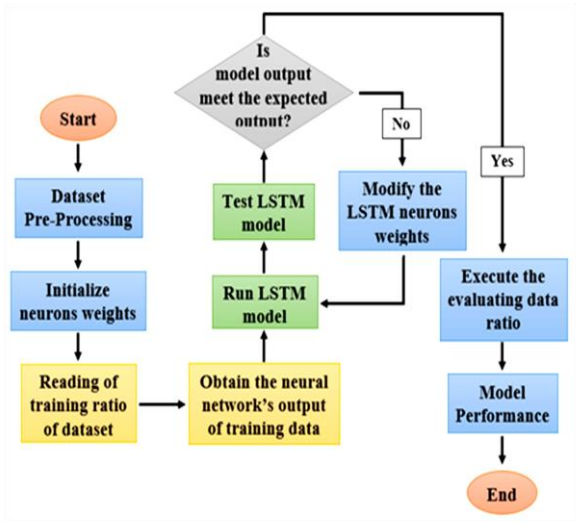
Start
Dataset Pre-Processing
Initialize neurons weights
Reading of training ratio of dataset
Obtain the neural network's output of training data
Run LSTM model
Test LSTM model
Is model output meet the expected output?
No
Modify the LSTM neurons weights
Yes
Execute the evaluating data ratio
Model Performance
End

Fig. 4. LSTM classifier model flowchart [21].

Building an LSTM model with Python is a great way to get started with deep learning. The first step in building an LSTM model with Python is to import the necessary libraries. The most popular library for deep learning in Python is TensorFlow, which provides a high-level API for building and training neural networks. Other popular libraries include Keras, PyTorch, and Theano. Once the libraries are imported, the next step is to prepare the data. This involves loading the data, preprocessing it, and splitting it into training and test sets. It is important to ensure that the data is properly normalized and scaled before training the model. The next step is to define the model architecture. This involves specifying the number of layers, the number of neurons in each layer, the type of activation functions, and the type of optimizer. It is also important to specify the input and output shapes of the model. Once the model architecture is defined, the next step is to compile the model. This involves specifying the loss function, the optimizer, and the metrics to be used for evaluating the model. Finally, the model can be trained. This involves specifying the number of epochs, the batch size, and the validation split. It is important to monitor the training process to ensure that the model is not overfitting or underfitting the data [22].

C. Convolutional Neural Network (CNN)

A convolutional neural network (CNN) is a type of artificial neural network used in deep learning that is specifically designed to process data that has a grid-like structure, such as tabular and images datasets. CNNs are composed of multiple layers of neurons that each perform a specific task as shown in Fig. 5. The first layer of neurons is responsible for detecting edges and other basic features in the input image. The second layer of neurons is responsible for detecting more complex features, such as shapes and patterns. The third layer of neurons is responsible for recognizing objects in the image. The fourth layer of neurons is responsible for recognizing more complex objects, such as faces or animals [24].

CNNs are particularly useful in robotics because they are able to process large amounts of data quickly and accurately. For example, a CNN can be used to identify objects in an image or video feed. It can also be used to analyze cognitive data represented in a database that enables robots to understand the surrounding environment and also understand the commands stored within it and classify them according to the event the robot is exposed to. This is useful for robots that need to identify objects in their environment in order to navigate or interact with them. CNNs can also be used to classify objects in a scene, which is useful for robots that need to recognize and interact with objects in their environment [25].

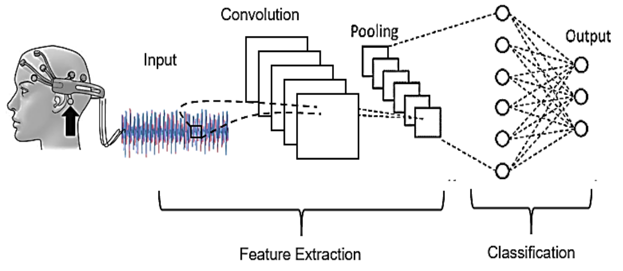

Fig. 5. Convolutional Neural Network (CNN) architecture.

Building a Convolutional Neural Network (CNN) model is a complex process that requires a lot of knowledge and experience. However, with the right guidance, it can be done relatively easily. The following steps outline the process of building a CNN model as described in Algorithm 2 [26]:

- Data Preparation: The first step in building a CNN model is to prepare the data. This includes gathering the data, cleaning it, and formatting it into a suitable format for the model. This step is important as it ensures that the model is trained on the most accurate and up-to-date data.

- Model Architecture: The next step is to decide on the model architecture. This includes deciding on the number of layers, the type of layers, and the number of neurons in each layer. This step is important as it determines the complexity of the model and how well it will perform.

- Training: Once the model architecture is decided, the next step is to train the model. This involves feeding the data into the model and adjusting the weights and biases of the neurons in order to minimize the error. This step is important as it ensures that the model is able to accurately predict the output given the input.
- Evaluation: After the model is trained, the next step is to evaluate the model. This involves testing the model on unseen data and measuring its performance. This step is important as it allows us to determine how well the model is performing and if it needs to be improved.
- Deployment: The final step is to deploy the model. This involves making the model available to users so that they can use it to make predictions. This step is important as it allows the model to be used in real-world applications.

These are the basic steps for building a CNN model. However, there are many other steps that can be taken to improve the model, such as hyperparameter tuning, regularization, and data augmentation. With the right guidance and experience, building a CNN model can be a relatively straightforward process.

Algorithm 2: CNN model

```
Input: tabular EEG emotional brain signal dataset
Output: confusion matrix and model testing accuracy
1.    Import necessary libraries
(Numpy as np, pandas as pd, tensorflow as tf, Sequential, Dense,
Conv1D, MaxPooling1D, and Flatten)
2.    Load the emotional dataset
dataset = pd read _ datatype ('dataset . datatype')
3.    Analysis the dataset
Input signals = dataset drop (columns = ['target columns'])
Labels = dataset ['last column']
4.    Split the dataset into training and testing sets
data train, data test, labels train, labels test = train test split (data,
labels, test size = test ratio to complete dataset, random state = no. of
states)
5.    Build the CNN model
model = Sequential ([
   Conv1D parameter definition (filters, kernel size, activation
functions)
      Input shape = X train shape.
      MaxPooling1D size.
      Dense (output layer count, output activation function
6.    Train the model
Training history = model fit (data train, labels train, epochs number,
batch size, validation data (data test, labels test))
7.    Evaluate the model
loss accuracy = model evaluate (data test, labels test)
print Test Loss
print Test Accuracy
print confusion matrix
```

D. CNN-LSTM Hybrid Model

The CNN-LSTM model, which combines Convolutional Neural Networks (CNNs) with Long Short-Term Memory (LSTM) networks, excels at modeling the interdependence of spatial and temporal data. This powerful combination leverages CNNs' ability to extract spatial features and LSTMs' strength in temporal modeling, leading to high effectiveness across various domains.

This hybrid model finds applications in tasks involving complex sequential data. It utilizes CNNs for spatial analysis and LSTMs for understanding temporal sequences. The CNN-LSTM model processes input data through one or more convolutional layers to identify spatial patterns. The output from these layers then feeds into LSTM layers to capture temporal dependencies and sequential patterns. Finally, dense layers are often used for classification or regression tasks. Algorithm 3 lists the whole process of proposed model.

The model's strength lies in the specialized functions of its layers. CNNs excel at extracting features from spatial data, while LSTMs represent complex temporal connections. This combination allows the model to learn both spatial and temporal characteristics simultaneously, enabling a comprehensive interpretation of the data. However, achieving optimal performance requires careful hyperparameter tuning for both CNN and LSTM components, and ensuring compatibility between the input data shape and both layer types. A small code example using the Keras library shows how to sequentially add CNN and LSTM layers for spatiotemporal modelling [27].

Algorithm 3: CNN-LSTM model

```
Input: tabular EEG emotional brain signal dataset
Output: confusion matrix and model testing accuracy
1.    Import necessary libraries
(Numpy as np, pandas as pd, tensorflow as tf, Sequential, Dense,
Conv1D, MaxPooling1D, and Flatten)
2.    Load the emotional dataset
dataset = pd read _ datatype ('dataset . datatype')
3.    Analysis the dataset
Input signals = dataset drop (columns = ['target columns'])
Labels = dataset ['last column']
4.    Split the dataset into training and testing sets
data train, data test, labels train, labels test = train test split (data,
labels, test size = test ratio to complete dataset, random state = no. of
states)
5.    Build the CNN-LSTM model
model = Sequential ([
•     Conv1D parameters definition (filters, kernel size, activation
functions)
         Input shape = X train shape.
         MaxPooling1D size.
•     LSTM parameters definition (units' size, return sequences)
•     Dense (output layer count, output activation function)
6.    Train the model
Training history = model fit (data train, labels train, epochs number,
batch size, validation data (data test, labels test))
7.    Evaluate the model
loss accuracy = model evaluate (data test, labels test)
print Test Loss
print Test Accuracy
print confusion matrix
```

IV. Dataset Description

Datasets can be used to analyze trends, identify patterns, and make predictions. They can also be used to compare different groups of people or different types of data, store information about people, places, events, and other topics, and create visualizations like charts, graphs, and maps. They can even be used to generate reports and presentations.

Datasets can be used to analyze trends, identify patterns, and make predictions. They can also be used to compare different groups of people or to compare different types of data. They can also be used to store information about people, places, events, and other topics. Datasets can be used to create visualizations, such as charts, graphs, and maps. They can also be used to create reports and presentations. They can also be used to store information about people, places, events, and other topics [28].

The dataset used in this research is a mental emotional sentiment dataset that was collected by other researchers using a commercial MUSE EEG headband which was used with a resolution of four (TP9, AF7, AF8, TP10) electrodes. To collect the data, researchers used a widely available MUSE EEG headgear with four dry extra-cranial electrodes. As can be seen in Fig. 6, micro voltage readings are taken from electrodes TP9, AF7, AF8, and TP10. Two individuals (1 male, 1 female, aged 20-22) each provided 60 seconds of data for each of the 6 film segments, for a total of 12 minutes (720 seconds) of brain activity data (6 minutes for each emotional state). A total of 36 minutes of EEG data was obtained from each individual, including six minutes of "neutral brainwave" data. The brain's waves were captured at a variable frequency and then resampled to 150Hz, yielding a collection of 324,000 data points. The positive and negative valence descriptors were evaluated instead of the emotions themselves to determine which activities were most likely to elicit. For a third category, representing the subject's baseline emotional state, neutral data were also obtained before any data on emotions were gathered (to prevent contamination from the latter). We only gathered data from each participant for three minutes every day to minimize the influence of a baseline emotional state [29].

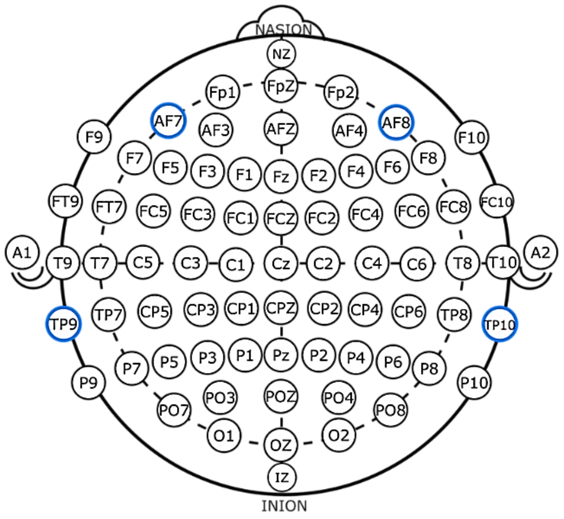

Fig. 6. Position of used EEG electrodes on human skull [29].

V. Classification Results

Classification results are the outcomes of a classification process, which is a type of data mining technique used to identify patterns and relationships in data. Classification results are used to make predictions about future data points, and can be used to make decisions about how to best utilize resources. Classification results are typically presented in the form of a confusion matrix, which is a table that shows the number of true positives, false positives, true negatives, and false negatives [30]. The confusion matrix is used to evaluate the accuracy of the classification model, and can be used to identify areas where the model is performing well or poorly. Classification results can also be used to identify important features in the data that are driving the model's predictions. This can be done by looking at the feature importance scores, which are calculated by the model and indicate how important each feature is in making the prediction. This can be used to identify which features are most important for making accurate predictions, and can be used to inform decisions about which features to focus on when building a model [31], [32].

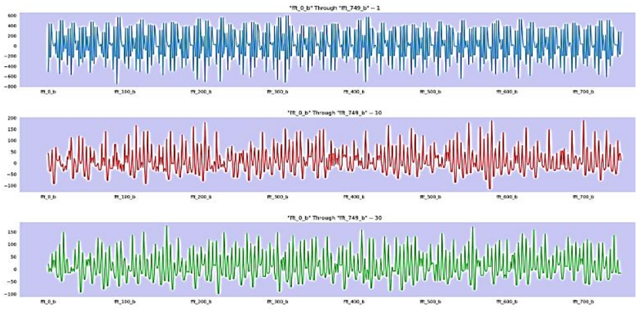

Fig. 7. Classes appearance analysis.

Finally, classification results can be used to compare different models and determine which one is the best for a given task. This can be done by looking at the accuracy scores of each model, as well as other metrics such as precision, recall, and F1 score. Comparing the results of different models can help identify which model is best suited for a given task, and can help inform decisions about which model to use [32].

To classify the dataset, it must first understand its details, the resultant classes from each row of input, and the number of instances of each class over the whole dataset. As the information from the dataset were analyzed, it was discovered that there are three separate classes as a consequence of all the input rows, which are positive, negative, and neutral, as they represent an indicator of the subject's emotional state. After each class was counted, it was discovered that the positive case occurred 708 times, the negative case appeared 708 times, and the neutral case appeared 716 times as shown in Fig. 7. These statistics reveal the dataset's balance, from which the difference in findings may be precisely calculated. As the last stage in studying the dataset, a sample may be obtained for each class using a variety of inputs, as illustrated in Fig. 8.

A. SVM Results

When implementing the SVM algorithm, the tensorflow library was used for deep learning in Python, on the Kaggle coding website. And by classifying the studied dataset, and specifying each of the training data percentage as 50%, the testing data percentage as 25%, the validation data percentage as 25%, and 100 epochs to test the algorithm and conclude the best classification result.

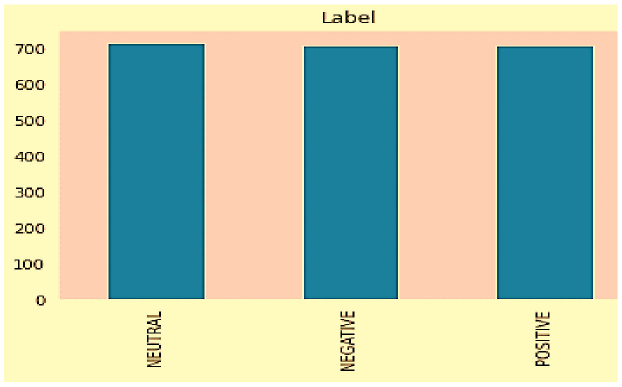

Fig. 8. Classes sample.

The classification results of the algorithm came with an accuracy of 87.5%, after only 15 epochs (early stop), radial basis function (RBF) kernel [33], and 39 ms/step running speed. When viewing the confusion matrix as shown in Fig. 9, which describe matching between actual label and predicted label. It can be seen that the results are not mixed up, or in another sense, the algorithm is not confused between dataset classes when determining the result significantly.

B. LSTM Results

The Long Short-Term Memory (LSTM) model was developed on the Kaggle coding platform using the Python tensorflow deep learning framework. By categorising the examined dataset, designating the percentage of each training data as 50%, the percentage of test data as 25%, and the percentage of validation data as 25%, 100 epochs for model testing, and selecting the best classification result. This model was built to contain the input layer, and the last layer is responsible for the output, and because the dataset contains more than two expected results (3 classes), the Softmax Activation Function is used [34].

The algorithm's classification results showed an accuracy of 97.35% after just 38 epochs (early stopping) and a running speed of 2 s/step, which is an excellent result. This result was reached by setting the learning rate to 0.001 and using Adam as the model's optimization library. In addition, through Fig. 10, it can be noted that the classification model was very sharp in showing the results, as it was not confused with the actual result of implementing the classification except in very simple cases that do not exceed 0.06 for each class confused with other classes.

C. CNN Results

On the Kaggle coding platform, the tensorflow deep learning library in Python was utilized to create the convolutional neural network model. By classifying the investigated data set, defining the percentage of each training data as 50%, the proportion of test data as 25%, and the proportion of validation data as 25%, 100 epochs for model testing, and identifying the best classification result.

This model was built to contain the input layer, five hidden layers all of which contain an activation function of the type Rectified Linear Unit (ReLU) [35], and this function is considered one of the best choices in the selection of the activation. The last layer is responsible for the output, and because the dataset contains more than two expected results (3 classes), the Softmax Activation Function is used.

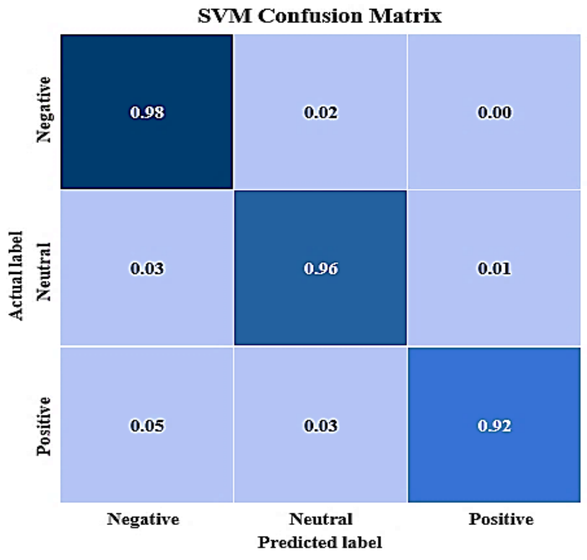

Fig. 9. Confusion Matrix of SVM model result.

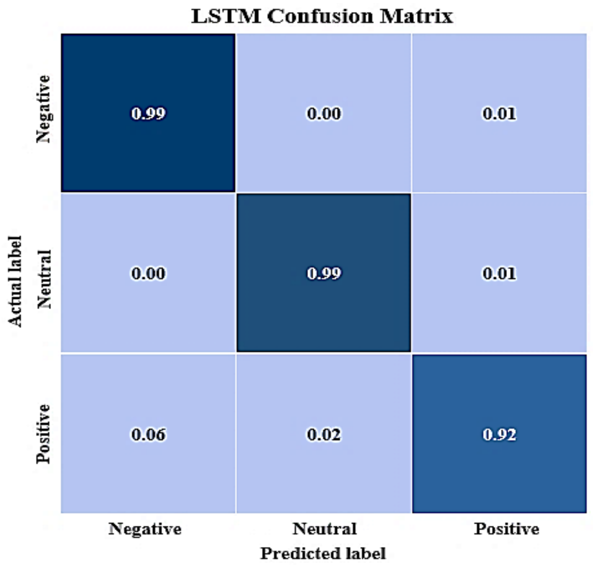

Fig. 10. Confusion matrix for LSTM model results.

The algorithm's classification results came with an accuracy of 98.03 % after only 38 epochs (early stopping) and 58 ms/step running speed, which is a great result. This result was obtained as a result of setting the learning rate to 0.001 and using Adamax as the optimization library on the model. Moreover, it can be shown in Fig. 11 that the classification model was extremely crisp in displaying the results, as it was not confused with the real result of implementing the classification except in very basic examples where 0.03 for each class confused with other classes was not exceeded.

D. CNN-LSTM Results

The CNN-LSTM model was implemented on the Kaggle coding platform using the Python tensorflow deep learning framework. By classifying the dataset, allocating 50% of the data for training, 25% for testing, and 25% for validation, conducting 50 epochs for model evaluation, and choosing the optimal classification outcome. This model was constructed with an input layer, three CNN layers utilizing the ReLU activation function and progressively increasing filter sizes starting from 64 bits. It also includes LSTM layers and a final output layer. Since the dataset consists of three distinct classes, the Softmax Activation Function is employed.

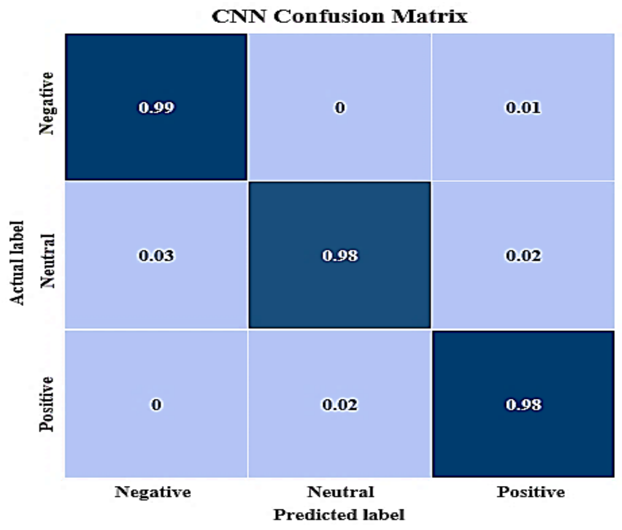

Fig. 11. Confusion matrix for CNN model results.

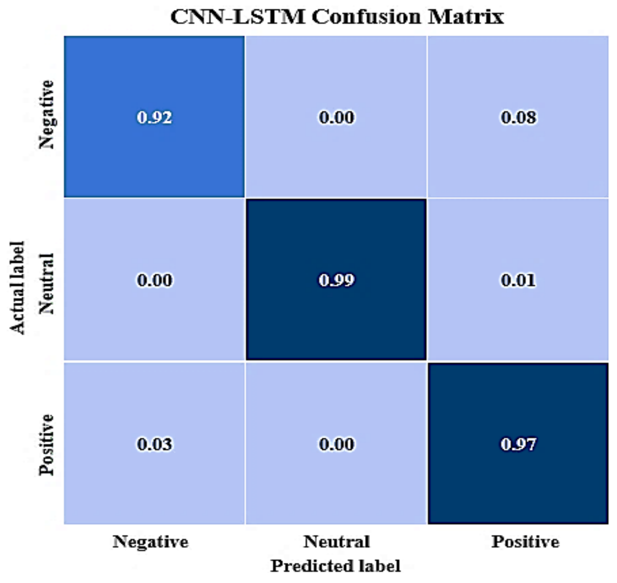

Fig. 12. CNN-LSTM confusion matrix.

The algorithm achieved a classification accuracy of 98.50% after completing 100 epochs without early stopping. Additionally, it demonstrated a running speed of 244 ms/step, which is considered an outstanding outcome. The attainment of this outcome was accomplished by configuring the learning rate to 0.001 and employing Adam as the optimization library for the model. Furthermore, Fig.12 demonstrates that the classification model exhibited a high level of accuracy in presenting the findings, as it only encountered confusion with the actual outcome of the classification in rare instances that did not surpass 0.06 for each misclassified class.

VI. Result and Discussion

When comparing the results of different models to classify the studied dataset (SVM, LSTM, CNN, and CNN-LSTM), more than one aspect can be relied on for comparison, the first and most important of which is the accuracy of the classification when implementing the model, the second is the confusion matrix, which is related in one way or another to the first factor in the comparison, and the third factor that was taken into account when comparing is speed of implementation of the model, as measured by the speed of the test steps and also the speed of early stopping when testing the model. Table I compiles these features for all models utilized in the paper.

TABLE I. Comparision Factor Conclusion

Model	**Comparison factors**		
	Accuracy	***Test speed***	***Early stop***
SVM	87.5 %	39 ms/step	15 epochs
LSTM	97.35 %	2 sec/step	15 epochs
CNN	98.03 %	58 ms/step	38 epochs
CNN-LSTM	98.50 %	244 ms/step	No

With regard to the first factor in the comparison, which is the accuracy of the model in implementation, it came in the foreground, and it is considered one of the best classification results applied to the studied data set. It is the result of classification using CNN-LSTM with an accuracy of 98.50%. Then it comes in second place, and not by a large difference, is the result of classification using CNN, with an accuracy of 98 %, while in third place was the LSTM model with 97.35 % accuracy, finally SVM where the classification accuracy was not good enough compared to the previous three models with an accuracy of 87.5 %.

As mentioned previously, the confusion matrix is linked to the accuracy of the classification, or in other words, this matrix is a breakdown of the characteristics of the classification result that lead to its accuracy. The order of the models when comparing the results based on the quality of the matrix came in the same order as the models in terms of accuracy.

As for the third factor in the comparison, it is actually divided into two different factors, which are the speed of implementation by step and the speed of implementation in early stopping when testing the model. In view of the speed of execution by step, the SVM model came in first place with a speed of 39 ms/step, and in second place came the CNN model with 58 ms/step, and the CNN-LSTM model came in third place with a large difference from its predecessors with 244 ms/step. and in last place LSTM with extreme test step speed time with 2 sec/step However, when looking at the speed of early stopping, the order can differ relatively, as the SVM model comes in first place equally with the LSTM model by stopping after only 15 epochs out of 100 epochs that were specified for implementation, and the CNN-LSTM model remains in last place with no early stop out of 100 epochs also for implementation.

As a result of this comparison, it can be concluded that although the classification of the CNN-LSTM is the best as a model for classification, it must be taken into account that the previous results are dependent on the input factors of each model, which were fixed in all cases, so that comparison can be made based on the equality of classification characteristics.

The CNN-LSTM was the best of them, as it has the highest accuracy, which is the most important factor in the comparison. In addition, the execution speed (step speed) was not bad (between the other models), but despite that, it was the most in the number of epochs that the model needed to infer the best classification result (training the model) but this did not significantly affect the outcome of the total execution time of the model.

And if we exclude the factor of execution speed, then CNN-LSTM, CNN and LSTM are very close in the result of classification accuracy, then these models can be equally reliable on the classification of the data set. In contrast, if the accuracy factor is excluded, the SVM model is the fastest in step speed and the least in the number of epochs required to train the model equally with the LSTM model, so SVM can be said that it is the best in the speed factor.

Considering that the CNN-LSTM result is the best model studied in this paper in terms of accuracy, which is the most reliable factor in the comparison. In the end, this model can also be compared with the similar models previously published on a similar database and with the different models. Through the previous literature study during previous years, published research showed limited accuracy of the techniques used, as K-nearest neighbor recorded an accuracy of 86.46% in Z-T. Liu. et al.'s 2019 research. T. Song et al. also recorded in their research published in 2020, an accuracy of 90.4% using DGCNN. Also, it was followed by the accuracy of the research of S.K. Khare et al. 2020, which reached 95.7% using the LSSVM, and then the RNN, LSTM, and GRU models obtained an accuracy of 95%, 97%, and 96% respectively in the research of Chowdary MK et al. 2022. Finally, Chen J. et al. 2024 are achieved 75.26% multi-classification accuracy by using GNN model in their published study. On the other hand, expectations were raised for an impressive result using the studied model (CNN-LSTM), where a classification accuracy was obtained that achieved a gorgeous mark in prediction. A summary of the results of the reviewed researches appears in Table II, where each row indicates the research person, the model used in it, and the model's accuracy result during testing.

TABLE II. LITERATURE RESULTS SUMMARY COMPARED WITH PROPOSED MODEL

Author	**Comparison**	
	Model	***Accuracy***
Z-T. Liu et al. (2019) [9]	K-NN	86.46 %
T. Song et al. (2020) [10]	DGCNN	90.40 %
S.K. Khare et al. (2020) [11]	LSSVM	95.70 %
Chowdary MK et al. (2022) [12]	RNN	95.00 %
	LSTM	97.00 %
	GRU	96.00 %
Chen J. et al. (2024) [13]	GNN	75.26 %
Proposed model	CNN-LSTM	98.50 %

VII. APPLICATIONS

Brain Computer Interfacing (BCI) is a technology, which enables communication between humans and machines through the direct interpretation of brain activities. This technology has a wide range of potential applications as shown in Fig. 13 and has been explored by researchers in fields such as medical diagnostics, prosthetics, human-machine interaction, and communication aids [36].

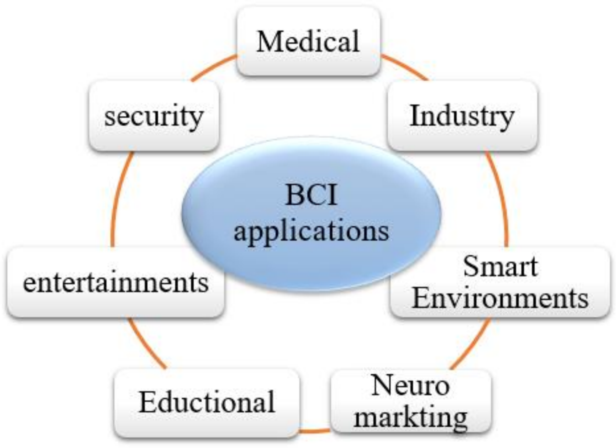

Fig. 13. BCI applications.

After classifying the data set, the classification model may be used to connect the model's inputs, which are brain signals, and any application that can be controlled by the dataset's distinct classifications. When looking at the field of medical industries, it is possible to coordinate between commands to control prosthetic limbs through brain signals directly, through three different commands linked to the three groups of the data set, for patients with paraplegia or total paralysis who are unable to move their natural organs, or move the muscles to control the Industrial limb [37]–[39]. Among the applications is Brain control in industrial robots in smart industries. It is also possible to link the results of classification (one of the classes for the data set) and a set of successive commands that include a path for a complete industrial process, so that the controller has the ability to control the brain in three separate industrial processes [40]. In addition, brain control technology can be used to control laboratory robots, and this can be used in research and scientific projects that allow the formation of innovative systems of intelligent control [41].

One of the most exciting applications of BCI in gaming and entertainment is the ability to control game characters and objects with your thoughts. This could allow players to control their characters in a more natural and intuitive way, as well as allowing for more complex interactions with the game world. For example, a player could use their thoughts to control a character's movements, or to manipulate objects in the game world. This could open up a completely new level of immersion and interaction with games [42].

Brain-Computer Interface (BCI) technology might change how individuals see themselves and their surroundings. BCI technology raises ethical and legal issues. BCI technology may enhance quality of life and give therapeutic advantages, but also presents privacy, autonomy, and informed consent problems [43].

VIII. Conclusion and Future Work

The domain of Brain-Computer Interface (BCI) stands as an exceptionally captivating realm of scientific inquiry due to its potential intersections with diverse industries, particularly those requiring intelligent control, such as industry and medicine. Various methodologies are employed to assemble datasets of cerebral signals for the comprehensive understanding of the intricate signals emanating from the brain. Among these methodologies, the non-invasive Electroencephalogram (EEG) method holds particular significance. The acquired dataset necessitates meticulous categorization, wherein the identification of influential characteristics responsible for inducing changes becomes imperative for its applicability across diverse control modalities.

Furthermore, the demand for precision and expeditious processing in BCI applications, especially in alignment with dynamic environmental motion sequences, prompted a comparative evaluation of four alternative classification models, namely Support Vector Machine (SVM), Long Short-Term Memory (LSTM), Convolutional Neural Network (CNN), and the hybrid CNN-LSTM model. The findings of this comparative analysis underscore the notable efficacy of the CNN-LSTM model, manifesting an accuracy of 98.5% alongside an operational speed of 244 milliseconds per step. Following suit, the CNN model secured the second position, achieving an accuracy of 98% with a step speed of 58 milliseconds per step. Occupying the third position, the LSTM model demonstrated an accuracy of 97.35%, albeit at a comparatively slower step speed of 2 seconds per step. Conclusively, the SVM model finalized the comparison, registering an accuracy of 87.5% and a step speed of 39 milliseconds per step. These findings accentuate the CNN-LSTM model's prowess in BCI applications, positioning it as the preeminent choice for striking a commendable equilibrium between accuracy and processing speed within dynamic environmental contexts.

Future work can involve two avenues: exploring the accuracy of other classification models and developing entirely new ones to advance the field. Additionally, we can investigate the creation of an integrated system utilizing brain signals for control and evaluate its overall performance in accurately executing commands based on the underlying classification accuracy.

Acknowledgment

We would like to thank Heriot-Watt University for its support to publish this paper. We also thank Horus University (Egypt) for its explicit support throughout this research and for providing access to high featured computer labs for models' implementation.

Kaggle platform was used as a workspace for modeling the studied approaches and implementing the proposed models.

References

[1] S. N. Abdulkader, A. Atia, and M. S. M. Mostafa, “Brain computer interfacing: Applications and challenges,” Egyptian Informatics Journal, vol. 16, no. 2. Elsevier B.V., pp. 213–230, Jul. 01, 2015. doi: 10.1016/j.eij.2015.06.002.

[2] Anupama H S, N. K. Cauvery, and Lingaraju G M, “Brain Computer Interface and Its Types-A Study,” Int J Adv Eng Technol, vol. 3, no. 2, pp. 739–745, 2012, Accessed: Feb. 22, 2024. [Online]. Available: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=15cd5fc6fd521f60cc35a2f4079b14360601117a.

[3] M.-P. Hosseini, A. Hosseini, and K. Ahi, “A Review on Machine Learning for EEG Signal Processing in Bioengineering,” IEEE Rev Biomed Eng, vol. 14, pp. 204–218, 2021, doi: 10.1109/RBME.2020.2969915.

[4] L. Bi, X.-A. Fan, and Y. Liu, “EEG-Based Brain-Controlled Mobile Robots: A Survey,” IEEE Trans Hum Mach Syst, vol. 43, no. 2, pp. 161–176, 2013, doi: 10.1109/TSMCC.2012.2219046.

[5] R. Bhavsar, Y. Sun, N. Helian, N. Davey, D. Mayor, and T. Steffert, “The correlation between EEG signals as measured in different positions on scalp varying with distance,” in Procedia Computer Science, Elsevier B.V., 2018, pp. 92–97. doi: 10.1016/j.procs.2018.01.015.

[6] J. das C. Rodrigues, P. P. R. Filho, E. Peixoto, A. K. N, and V. H. C. de Albuquerque, “Classification of EEG signals to detect alcoholism using machine learning techniques,” Pattern Recognit Lett, vol. 125, pp. 140–149, 2019, doi: https://doi.org/10.1016/j.patrec.2019.04.019.

[7] H. Dose, J. S. Møller, H. K. Iversen, and S. Puthusserypady, “An end-to-end deep learning approach to MI-EEG signal classification for BCIs,” Expert Syst Appl, vol. 114, pp. 532–542, 2018, doi: https://doi.org/10.1016/j.eswa.2018.08.031.

[8] J. Abdillah, I. Asror, and Y. F. A. Wibowo, “Emotion Classification of Song Lyrics using Bidirectional LSTM Method with GloVe Word Representation Weighting,” Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi), vol. 4, no. 4, pp. 723–729, 2020, Accessed: Feb. 22, 2024. [Online]. Available: http://jurnal.iaii.or.id/index.php/RESTI/article/download/2156/284.

[9] Z.-T. Liu, Q. Xie, M. Wu, W.-H. Cao, D.-Y. Li, and S.-H. Li, “Electroencephalogram Emotion Recognition Based on Empirical Mode Decomposition and Optimal Feature Selection,” IEEE Trans Cogn Dev Syst, vol. 11, no. 4, pp. 517–526, Dec. 2019, doi: 10.1109/TCDS.2018.2868121.

[10] T. Song, W. Zheng, P. Song, and Z. Cui, “EEG Emotion Recognition Using Dynamical Graph Convolutional Neural Networks,” IEEE Trans Affect Comput, vol. 11, no. 3, pp. 532–541, Jul. 2020, doi: 10.1109/TAFFC.2018.2817622.

[11] S. K. Khare, V. Bajaj, and G. R. Sinha, “Adaptive Tunable Q Wavelet Transform-Based Emotion Identification,” IEEE Trans Instrum Meas, vol. 69, no. 12, pp. 9609–9617, Dec. 2020, doi: 10.1109/TIM.2020.3006611.

[12] M. K. Chowdary, J. Anitha, and D. J. Hemanth, “Emotion Recognition from EEG Signals Using Recurrent Neural Networks,” Electronics (Basel), vol. 11, no. 15, Jul. 2022, doi: 10.3390/electronics11152387.

[13] J. Chen, X. Lin, W. Ma, Y. Wang, and W. Tang, “EEG-based emotion recognition for road accidents in a simulated driving environment,” Biomed Signal Process Control, vol. 87, no. 8, Jan. 2024, doi: 10.1016/j.bspc.2023.105411.

[14] B. Chakravarthi, S. C. Ng, M. R. Ezilarasan, and M. F. Leung, “EEG-based emotion recognition using hybrid CNN and LSTM classification,” Front Comput Neurosci, vol. 16, Oct. 2022, doi: 10.3389/fncom.2022.1019776.

[15] M. Besserve, K. Jerbi, F. Laurent, S. Baillet, J. Martinerie, and L. Garnero, “Classification methods for ongoing EEG and MEG signals,” Biol Res, vol. 40, no. 4, pp. 415–437, 2007, doi: http://dx.doi.org/10.4067/S0716-97602007000500005.

[16] M. Yoshikawa, M. Mikawa, and K. Tanaka, “A myoelectric interface for robotic hand control using support vector machine,” in 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007, pp. 2723–2728. doi: 10.1109/IROS.2007.4399301.

[17] T. N. T. Thu and V. D. Xuan, “Using support vector machine in FoRex predicting,” in 2018 IEEE International Conference on Innovative Research and Development (ICIRD), 2018, pp. 1–5. doi: 10.1109/ICIRD.2018.8376303.

[18] K. Harimoorthy and M. T, “Multi-disease prediction model using improved SVM-radial bias technique in healthcare monitoring system,” J Ambient Intell Humaniz Comput, vol. 12, Feb. 2021, doi: 10.1007/s12652-019-01652-0.

[19] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, “Support vector machines,” IEEE Intelligent Systems and their Applications, vol. 13, no. 4, pp. 18–28, 1998, doi: 10.1109/5254.708428.

[20] D. Tomar and S. Agarwal, “A comparison on multi-class classification methods based on least squares twin support vector machine,” Knowl Based Syst, vol. 81, pp. 131–147, 2015, doi: https://doi.org/10.1016/j.knosys.2015.02.009.

[21] J. Hernandez, D. López, and N. Vera, “Primary user characterization for cognitive radio wireless networks using long short-term memory,” Int J Distrib Sens Netw, vol. 14, p. 155014771881182, Feb. 2018, doi: 10.1177/1550147718811828.

[22] S. Tortora, S. Ghidoni, C. Chisari, S. Micera, and F. Artoni, “Deep learning-based BCI for gait decoding from EEG with LSTM recurrent neural network,” J Neural Eng, vol. 17, no. 4, Aug. 2020, doi: 10.1088/1741-2552/ab9842.

[23] H. Almutairi, G. M. Hassan, and A. Datta, “Classification of Obstructive Sleep Apnoea from single-lead ECG signals using convolutional neural and Long Short Term Memory networks,” Biomed Signal Process Control, vol. 69, p. 102906, 2021, doi: https://doi.org/10.1016/j.bspc.2021.102906.

[24] E. Pranav, S. Kamal, C. Satheesh Chandran, and M. H. Supriya, “Facial Emotion Recognition Using Deep Convolutional Neural Network,” in 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), 2020, pp. 317–320. doi: 10.1109/ICACCS48705.2020.9074302.

[25] K. Noda, H. Arie, Y. Suga, and T. Ogata, “Multimodal integration learning of robot behavior using deep neural networks,” Rob Auton Syst, vol. 62, no. 6, pp. 721–736, 2014, doi: 10.1016/j.robot.2014.03.003.

[26] B. Cao, H. Niu, J. Hao, and G. Wang, “Building EEG-based CAD object selection intention discrimination model using convolutional neural network (CNN),” Advanced Engineering Informatics, vol. 52, p. 101548, 2022, doi: https://doi.org/10.1016/j.aei.2022.101548.

[27] G. A. Marcoulides, “Discovering Knowledge in Data: an Introduction to Data Mining,” J Am Stat Assoc, vol. 100, no. 472, pp. 1465–1465, Dec. 2005, doi: 10.1198/jasa.2005.s61.

[28] Daniel T. Larose, Discovering Knowledge in Data : An Introduction to Data Mining, Second edition. Wiley-Interscience, Hoboken, N.J., ©2005, 2017.

[29] J. Bird, A. Ekart, C. Buckingham, and D. Faria, “Mental Emotional Sentiment Classification with an EEG-based Brain-machine Interface,” Feb. 2019.

[30] R. Ahmed Hegazii, E. Abdelhalim, and H. El-Din Mostafa, “A Proposed Technique for Breast Cancer Prediction and Classification Based on Machine Learning Section A-Research paper Eur,” Chem. Bull, vol. 2023, no. 8, pp. 7648–7656, doi: 10.48047/ecb/2023.12.8.619.

[31] T. Kaur and T. K. Gandhi, “Deep convolutional neural networks with transfer learning for automated brain image classification,” Mach Vis Appl, vol. 31, no. 3, p. 20, 2020, doi: 10.1007/s00138-020-01069-2.

[32] F. Hemmatian and M. K. Sohrabi, “A survey on classification techniques for opinion mining and sentiment analysis,” Artif Intell Rev, vol. 52, no. 3, pp. 1495–1545, 2019, doi: 10.1007/s10462-017-9599-6.

[33] Y.-W. Chang, C.-J. Hsieh, K.-W. Chang, M. Ringgaard, and C.-J. Lin, “Training and Testing Low-degree Polynomial Data Mappings via Linear SVM,” 2010.

[34] B. Gao and L. Pavel, “On the Properties of the Softmax Function with Application in Game Theory and Reinforcement Learning,” Apr. 2017, [Online]. Available: http://arxiv.org/abs/1704.00805

[35] V. Nair and G. E. Hinton, “Rectified Linear Units Improve Restricted Boltzmann Machines,” in Proceedings of the 27th International Conference on International Conference on Machine Learning, in ICML’10. Madison, WI, USA: Omnipress, 2010, pp. 807–814.

[36] S. N. Abdulkader, A. Atia, and M.-S. M. Mostafa, “Brain computer interfacing: Applications and challenges,” Egyptian Informatics Journal, vol. 16, no. 2, pp. 213–230, 2015, doi: https://doi.org/10.1016/j.eij.2015.06.002.

[37] J. H. Jeong, K. H. Shim, D. J. Kim, and S. W. Lee, “Brain-Controlled Robotic Arm System Based on Multi-Directional CNN-BiLSTM Network Using EEG Signals,” IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol. 28, no. 5, pp. 1226–1238, May 2020, doi: 10.1109/TNSRE.2020.2981659.

[38] M. A. A. Mousa, A. Elgohr, and H. Khater, “Path Planning for a 6 DoF Robotic Arm Based on Whale Optimization Algorithm and Genetic Algorithm,” Journal of Engineering Research, vol. 7, no. 5, pp. 160–168, Nov. 2023, doi: 10.21608/erjeng.2023.237586.1256.

[39] M. A. A. Mousa, A. T. Elgohr, and H. A. Khater, “Trajectory Optimization for a 6 DOF Robotic Arm Based on Reachability Time,” Annals of Emerging Technologies in Computing, vol. 8, no. 1, pp. 22–35, Jan. 2024, doi: 10.33166/AETiC.2024.01.003.

[40] M. A. Elazab, hamouda Abueldahab, A. Elgohr, and M. S. Elhadidy, “A Comprehensive Review on Hybridization in Sustainable Desalination Systems,” *Journal of Engineering Research*, vol. 7, no. 5, pp. 89–99, Nov. 2023, doi: 10.21608/erjeng.2023.235480.1238.

[41] B. Zhang, J. Wang, and T. Fuhlbrigge, “A review of the commercial brain-computer interface technology from perspective of industrial robotics,” in 2010 IEEE International Conference on Automation and Logistics, 2010, pp. 379–384. doi: 10.1109/ICAL.2010.5585311.

[42] S. Dutta, T. Banerjee, N. D. Roy, B. Chowdhury, and A. Biswas, “Development of a BCI-based gaming application to enhance cognitive control in psychiatric disorders,” Innov Syst Softw Eng, vol. 17, no. 2, pp. 99–107, 2021, doi: 10.1007/s11334-020-00370-7.

[43] S. Burwell, M. Sample, and E. Racine, “Ethical aspects of brain computer interfaces: a scoping review,” BMC Med Ethics, vol. 18, no. 1, p. 60, 2017, doi: 10.1186/s12910-017-0220-y.