Classification of EEG Signals Using a Neural Network Model and Comparison with MLP and LSTM Algorithms

Enver Kaan Alptürk[1*], Yakup Kutlu[1]
[1]Department of Computer Engineering, Iskenderun Technical University, Turkey
Emails: ealpturk.lee22@iste.edu.tr, yakup.kutlu@iste.edu.tr
*Corresponding author.

***Abstract*— This study introduces a neural network model designed for the classification of EEG signals. The model, developed using the Python programming language, is trained with the backpropagation algorithm and successfully predicts the correct outputs of signals. The research demonstrates that the model effectively predicts actions by training on EEG signal features recorded during hand clenching and unclenching actions. The accuracy of the developed neural network model was compared with Multilayer Perceptron (MLP), widely used in biological signal classification, and Long Short-Term Memory (LSTM), a low-error learning algorithm. The obtained results are presented in the Performance Analysis section.**

***Keywords*— EEG, Artificial Neural Networks, FFT, MLP, LSTM.**

I. INTRODUCTION

Eectroencephalogram (EEG) signals are electrophysiological recordings used to measure brain activity. First discovered in 1929 by Hans Berger, EEG has become a significant tool in brain research. EEG was one of the earliest methods to measure brain activity. In 1924, Hans Berger conducted experiments to measure electrical activity from the human brain. By 1929, Berger successfully recorded the brain's electrical activity by placing electrodes on the human scalp, producing the first EEG signals. Berger discovered that EEG signals exhibit fluctuations in different frequencies and amplitudes, categorizing these fluctuations into frequency bands such as "alpha" and "beta."

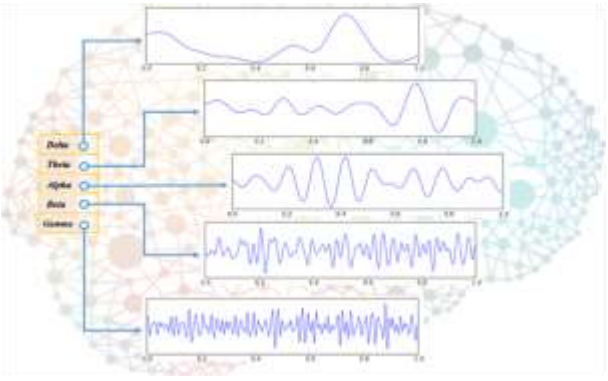

Figure 1: Sample signal graphs of EEG frequency ranges.

EEG signals are acquired by measuring brain activity via electrodes placed on the skin. Typically, electrodes are attached to a cap that ensures their proper placement on the scalp. The EEG device processes the electrical signals by amplifying and filtering them, which are then used for analysis [1].

EEG signals are typically observed as low-frequency and small-amplitude waves. Different wave types correspond to different states of brain activity. For instance, alpha waves are observed during relaxation, while beta waves increase during attention-demanding tasks. Additionally, delta and theta waves are prominent during sleep and some neurological disorders. Table 1 categorizes EEG signals based on their frequency ranges, and Figure 1 provides example graphs of these signals [1,9].

Table 1: Frequency Ranges of EEG Signals

Type	**Frequency Range (Hz)**
Delta	0.5 Hz – 3,5 Hz
Theta	4 Hz – 7 Hz
Alpha	8 Hz – 12 Hz
Beta	13 Hz – 30 Hz
Gamma	30+ Hz

This study involved recording EEG signals at a sampling frequency of 128 Hz for 1-second intervals during hand-clenching and unclenching actions using a portable EEG device designed in a previous study [2]. The signals underwent preprocessing and feature extraction before classification using a neural network model, the MLP algorithm, and the LSTM algorithm. The goal of this research is to convert brain signals generated during motor actions into meaningful data that can enable disadvantaged individuals to operate smart prostheses. This study represents the initial phase of developing advanced prosthetic devices..

II. MATERIALS AND METHODS

This study utilized EEG signals recorded during hand clenching and unclenching actions. The signals were sampled at 128 Hz. for 1-second intervals using a portable EEG device developed in a prior study. Preprocessing and feature extraction techniques were applied to the signals, followed by classification using an Artificial Neural Network (ANN) model, a Multilayer Perceptron (MLP), and a Long Short-Term Memory (LSTM) algorithm.

To prepare the data for classification, the Fast Fourier Transform (FFT) was applied to the recorded signals. The FFT outputs were divided into 4 Hz. intervals, and statistical features such as mean, maximum, minimum, standard deviation, skewness, and kurtosis were computed for each interval. These features formed the dataset used for model training and validation.

10 signals for each action (clenching and unclenching) were used as the training set, while 4 signals for each action were reserved for validation. The primary objective of this study was to interpret the brain signals produced during motor actions, enabling disadvantaged individuals to operate smart prostheses in the future.

A. Feature Extraction

Feature extraction involved processing the 1-second EEG signals captured during clenching and unclenching movements. These signals were transformed using the Fast Fourier Transform (FFT) and segmented into 4 Hz. frequency bands. For each segment, the following six statistical features were computed[1]:

1.Mean: The average value of the data within the 4 Hz. block.

2.Maximum Value: The highest data point in the block.

3.Minimum Value: The lowest data point in the block.

4.Standard Deviation: Calculated using the formula (Equation 1) [3]:

$$\sigma = \sqrt{\frac{1}{\mathrm{n}}\sum_{i=1}^{n}(x_i + \bar{x})^2} \quad \text{(Eq. 1)}$$

Here:

• N is the number of elements in the data set.

• x_i is the value at index i.

• $\bar{x}$ is the mean of the dataset.

5.Skewness: A measure of the asymmetry of the data distribution, calculated as:

(Equation 2) [3].

$$g_1 = \frac{m_3}{m_2^{\frac{3}{2}}} = \frac{\sqrt{n}\sum_{i=1}^{n}(x_i + \bar{x})^2}{(\sum_{i=1}^{n}(x_i + \bar{x})^2)^{\frac{3}{2}}} \quad \text{(Eq. 2)}$$

In this equation:

• $\bar{x}$ represents the sample mean.

• x_i is the value at index i.

• m_3 is the third central moment of the sample.

• m_2 is the sample standard deviation.

6.Kurtosis: A measure of the sharpness of the data distribution, determined using: (Equation 3) [3].

$$g_2 = \frac{m_4}{m_2^2} - 3 = \frac{n\sum_{i=1}^{n}(x_i + \bar{x})^4}{(\sum_{i=1}^{n}(x_i + \bar{x})^2)^2} - 3 \quad \text{(Eq. 3)}$$

In this formula:

• m_4 represents the fourth moment around the mean.

• m_2 represents the second moment around the mean.

These extracted features provided a comprehensive representation of the frequency-domain characteristics of the EEG signals. The feature set was then used as input for the machine learning models to classify the motor actions.

Fast Fourier Transform (FFT)

The Fourier transform is a mathematical operation that reveals the frequency components of a signal. Signals often consist of amplitude and frequency variations over time, and the Fourier transform decomposes these signals into their frequency components. However, directly applying the Fourier transform can be computationally slow in converting signals from the time domain to the frequency domain. To address this, the Fast Fourier Transform (FFT) algorithm was developed, providing a faster alternative by utilizing computational shortcuts and symmetry properties.

FFT is widely used in audio processing, image processing, data compression, and spectral analysis. It analyzes the spectral content of signals, identifies frequency components, and performs filtering and transformations in the frequency domain.

The mathematical representation of the FFT is as follows:

X_k= Σ _(n=0)^(N-1)▒ 〖x(n)^(-i(2π×k×n/N)) 〗 (Eq. 4)

Here, N represents the length of the sequence, $x(n)$ denotes the nth element of the input signal, and X_k refers to the kth component in the frequency domain. In this equation, the values of n and k range from 0 to N – 1.

The FFT algorithm uses this equation to efficiently compute the Discrete Fourier Transform (DFT) of a sequence. It achieves this by recursively dividing the sequence into smaller subsequences, where the length of the sequence is a power of two. These smaller subsequences are used to calculate the DFTs of shorter segments, which are then combined to obtain the DFT of the original sequenceT algorithm performs this division and combination process with a complexity of $O(N \log N)$. This ensures that as the data size increases, the computation time grows relatively slowly. Due to this efficiency, FFT is widely preferred for fast frequency analysis of large datasets.

B. Classification Approaches

Neural Network Model

The artificial neural network (ANN) model was developed using the Python programming language. The model consists of 96 input nodes and one output layer. The sigmoid activation function was employed, and weights were initialized randomly. Training was performed using the backpropagation algorithm.

The ANN model serves as a fundamental approach for classifying EEG signals. The specific ANN model used in this study was developed using Python and involves the following steps:

i. Dataset Definition

In this study, EEG signals recorded during 1-second intervals of hand-clenching and unclenching actions were processed to extract features. Each signal resulted in a total of 96 features, which were used as inputs to the ANN model.

ii. Model Construction

The model consists of one input layer, three hidden layers, and one output layer. Specifically, the layers include:

•Input Layer: 96 nodes representing the extracted features.

•Hidden Layers: 64, 64, and 32 nodes, respectively.

•Output Layer: 1 node.

The bias terms (e1, e2, e3, e4) are included to adjust the model. Initially, the weights connecting the nodes were initialized randomly. These weights represent connections between the input layer and hidden layers, as well as between hidden layers and the output layer. The dimensions of the weights were defined to match the model structure.

iii. Feedforward Process

The input dataset is fed into the ANN model. The inputs are multiplied by the weights and processed through the sigmoid activation function (Equation 5) to compute the outputs in the hidden layers. These outputs are then multiplied by subsequent weights and processed through the sigmoid function again to produce the final output at the output layer. This process results in the ANN's predictions.

iv. Backpropagation Algorithm

Backpropagation calculates the error between the predicted outputs and the actual outputs, adjusting the network weights accordingly.

1.The error at the output layer is computed first.

2.This error is propagated backward to the hidden layers to calculate their respective errors.

3.The computed errors are used to calculate delta values for updating weights and biases.

v. Weight and Bias Updates

Using the delta values obtained during backpropagation, weights and biases are updated to reduce the error. The updates follow the gradient descent principle, adjusting weights to minimize error. A defined learning rate is used for these updates.

vi. Training Loop

Steps 3-5 are repeated for all examples in the dataset. This ensures the network is trained using the entire dataset. The training process is repeated for a specified number of iterations (epochs), with each epoch representing one complete pass through the dataset.

This structured process allows the ANN model to learn from the EEG signal data and classify the motor actions effectively.

$$f(x) = \frac{1}{(1 + e^{-x})} \quad \text{(Eq. 5)}$$

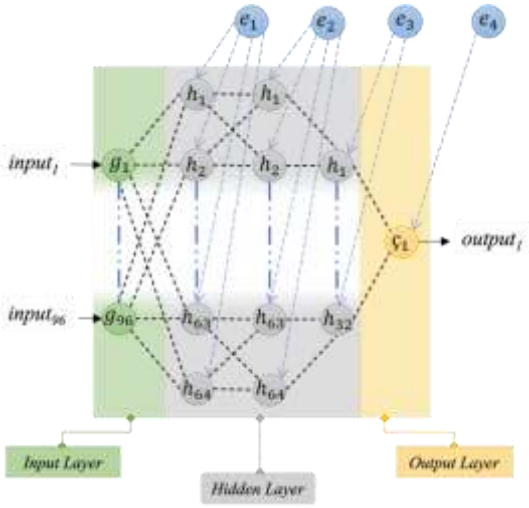

Figure 2: Layers and nodes of the created model.

$$net_h = a \times g_1 + a \times g_2 + \cdots + a \times g_{96} + e_1 \quad \text{(Eq. 6)}$$

$$h = f(net_h) \quad \text{(Eq. 7)}$$

$$E_ç = [Real\ Output] - ç \quad \text{(Eq. 8)}$$

$$\delta_ç = E_ç \times f'(ç) \quad \text{(Eq. 9)}$$

$$E_h = \delta_ç \times a \quad \text{(Eq. 10)}$$

$$\delta_h = E_h \times f'(h) \quad \text{(Eq. 11)}$$

$$a_ç = a_ç + net_h \times \delta_ç \quad \text{(Eq. 12)}$$

$$a_h = a_h + g \times \delta_h \quad \text{(Eq. 13)}$$

In the feedforward process, the numerical value for each node in the hidden layer is obtained by applying the sigmoid function to the net_h value, as shown in Equation 6. This process is similarly applied to compute the numerical value of the output node (Equation 7).

During backpropagation, the difference between the obtained output value and the actual output is calculated as the error (Equation 8). This error is then used to recompute the weights and hidden nodes.

To compute the error at the hidden nodes, a delta value for the output node (δ_ç) is first determined (Equation 9). This delta is multiplied by the weight associated with the corresponding node connection (Equation 10). Similarly, to recompute the value of a weight, a delta value for the hidden node (δ_h) is calculated (Equation 11).

The computed numerical values are used to update the weights between the output layer and the hidden layer (Equation 12) as well as the weights between the hidden layer and the input layer (Equation 13).

This iterative process is repeated for each weight and node across the specified number of iterations [5].

In the feedforward process, the numerical value for each node in the hidden layer is obtained by applying the sigmoid function to the net_h value, as shown in Equation 6. This process is similarly applied to compute the numerical value of the output node (Equation 7).

During backpropagation, the difference between the obtained output value and the actual output is calculated as the error (Equation 8). This error is then used to recompute the weights and hidden nodes.

To compute the error at the hidden nodes, a delta value for the output node (δ_ç) is first determined (Equation 9). This delta is multiplied by the weight associated with the corresponding node connection (Equation 10). Similarly, to recompute the value of a weight, a delta value for the hidden node (δ_h) is calculated (Equation 11).

The computed numerical values are used to update the weights between the output layer and the hidden layer (Equation 12) as well as the weights between the hidden layer and the input layer (Equation 13).

This iterative process is repeated for each weight and node across the specified number of iterations [5].

Multilayer Perceptron (MLP)

A Multilayer Perceptron (MLP) is a type of fully connected feedforward artificial neural network (ANN). An MLP consists of at least three layers: an input layer, a hidden layer, and an output layer. Each node, except for those in the input layer, is a neuron that applies a non-linear activation function. During the training process, a supervised learning technique called backpropagation is utilized. MLP differs from a simple perceptron by incorporating multiple layers and non-linear activation functions, which allows it to distinguish between data that is not linearly separables first passed to the input layer and then transferred to the first hidden layer. The hidden layers can be designed in various configurations depending on the complexity of the problem and the desired error rate. The output of each layer is passed as input to the next layer. Data emerging from the final hidden layer is transmitted to the output layer, where it is processed to determine the network's final output. Figure 3 illustrates a four-layer MLP model with two input nodes, one output node, and two hidden layers [6].

In MLPs, some neurons use non-linear activation functions to model the action potentials or firing frequencies of biological neurons. Two commonly used activation functions are sigmoid functions, defined in Equations 14 and 15. The first equation represents a hyperbolic tangent function that ranges from -1 to 1, while the second represents a logistic function ranging from 0 to 1. Here, y_i denotes the output of the ith neuron, and v_i represents the weighted sum of its input connections.

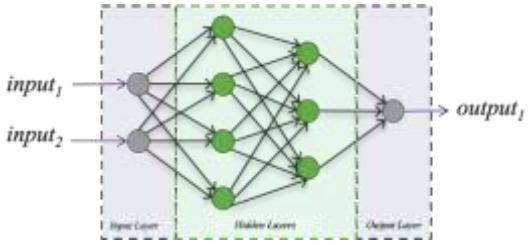

Figure 3: Four-layer MLP structure.

$$y(v_i) = \tanh(v_i) \quad \text{(Eq. 14)}$$

$$y(v_i) = (1 + e^{-v_i})^{-1} \quad \text{(Eq. 15)}$$

Long Short-Term Mermory (LSTM)

Developed in the late 1990s, Long Short-Term Memory (LSTM) is a specialized type of Recurrent Neural Network (RNN) designed for modeling sequential data. In an RNN, each data input is analyzed iteratively, taking into account the value of the previous output. However, despite claims that information from earlier time steps is considered, the architecture often struggles with the vanishing or exploding gradient problem, making it challenging to retain long-term dependencies. To address this issue, the LSTM architecture was developed to remember long-term information effectivelyTM architecture, illustrated in Figure 4, consists of repeating sequential blocks. Generally, the structure includes three key layers: the forget layer, the input layer, and the output layer [7].

In the LSTM architecture, the decision on which information to discard is made using the current input X_t and the previous hidden state h_t-1. This process occurs in the forget layer (f_t), as defined by Equation 16, where the sigmoid activation function is applied to determine the forget gate's output.

$$f_t = \sigma\left(W_{f,x} \times X_t + W_{f,h} \times h_f\right) \quad \text{(Eq. 16)}$$

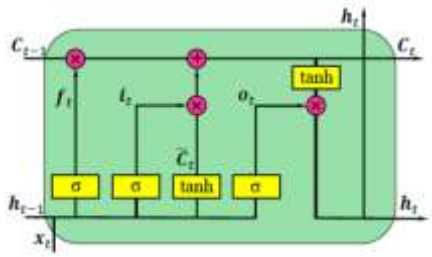

Figure 4: Long short-term memory architectural structure.

In the second step, the input layer determines new information. Initially, the information is updated using the sigmoid function, as described in Equation 17 (i_t). Then, new information is generated using the tanh function, as defined in Equation 18, which determines the candidate values for the state update.

$$i_t = \sigma\left(W_{i,x} \times X_t + W_{i,h} \times h_t - 1 + b_i\right) \quad \text{(Eq. 17)}$$

$$C_t = tanh\left(W_{c,x} \times X_t + W_{c,h} \times h_t - 1 + b_c\right) \quad \text{(Eq. 18)}$$

Finally, the output data is obtained in the output layer using Equations 19 and 20. These equations define the process of generating the LSTM's output based on the updated cell state and the hidden state.

$$o_t = \sigma\left(W_{o,x} \times X_t + W_{o,h} \times h_t - 1 + b_o\right) \quad \text{(Eq. 19)}$$

$$h_t = o_t \times tanh(C_t) \quad \text{(Eq. 20)}$$

The process described in the equations is repeated iteratively. The weight parameters (W) and bias parameters (b) are optimized by the model to minimize the difference between the actual training values and the LSTM's output values [8].

III. RESULTS

The neural network model developed in Python demonstrated effective classification of EEG signals. Figures 5 and 6 depict sample graphs of EEG signals recorded during fist-clenching and unclenching actions, respectively.

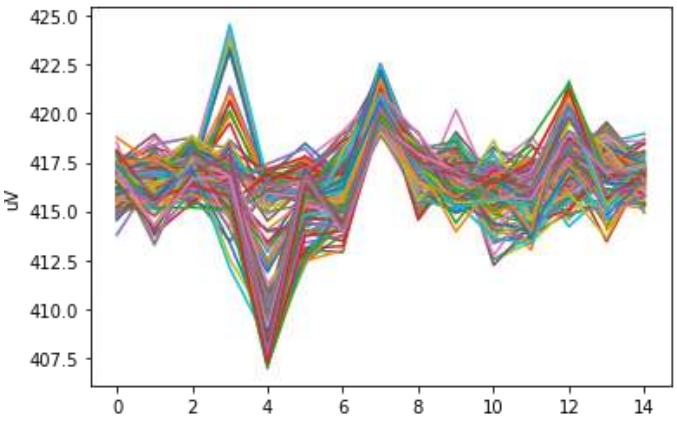

Figure 5: Signals recorded during the fist-clenching action.

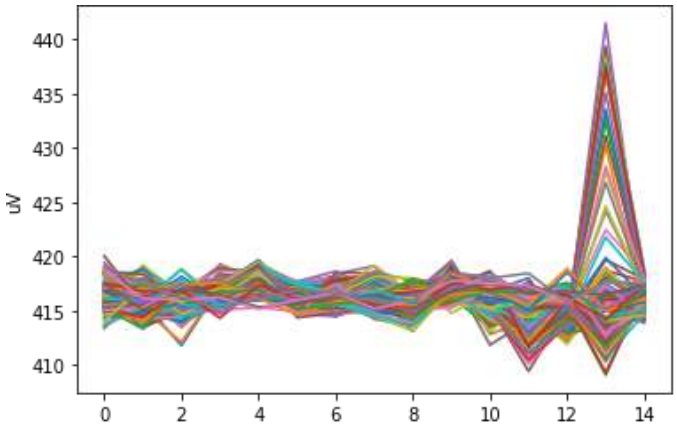

Figure 6: Signals recorded during opening of a clenched fist.

The features extracted from the signals constituted the input layer nodes of the created ANN model, LSTM model, and MLP model. Table 2 presents the parameters used for the three models and their corresponding performance results. Based on these results, the parameters yielding the most successful outcomes were identified as the final parameters of the proposed ANN model.

The learning rate of the ANN model was determined as 0.05 after a series of trials. Figure 7 shows the loss curves obtained from running the model three times with 1000 iterations. The results indicate that the error values of the model began to optimize after 400 iterations.

Figure 8 shows the accuracy graph obtained from running the LSTM algorithm for 700 iterations.The final accuracy was measured at 0.8800, which is considered an acceptable level for classification performance.

Figure 9 shows the accuracy and loss values of best MLP model. For the data set used, it was observed that the best activation function in the MLP algorithm was Tanh. In the LSTM and ANN algorithms, the best results were obtained using the Sigmoid activation function.

Table 3 presents comparing the final results of accuracy, Mean Squared Error (MSE), Mean Absolute Percentage Error (MAPE), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE) values obtained from all the algorithms..

Table 2. Parameters used in algorithms and results.

Run	Activation Function	Hidden Layers	Iterations	Learning Rate	MLP Loss-Accuracy	LSTM Loss-Accuracy	ANN Loss-Accuracy
1	Relu	(32,32)	500	0.5	7.11 – 0.5	0.81 – 0.5	0.5 – 0.6
2	Relu	(32,64)	500	0.5	0.7 – 0.63	0.46 – 0.66	0.48 – 0.6
3	Tanh	(64,32,32)	700	0.1	0.69 – 0.63	0.7 – 0.66	0.32 – 0.74
4	Tanh	(64,64,32)	1000	0.1	0.0 – 1.0	0.47 – 0.78	0.14 – 0.8
5	Sigmoid	(64,64,32)	1000	0.05	0.36 – 0.92	0.08 – 0.88	0.005 – 0.97

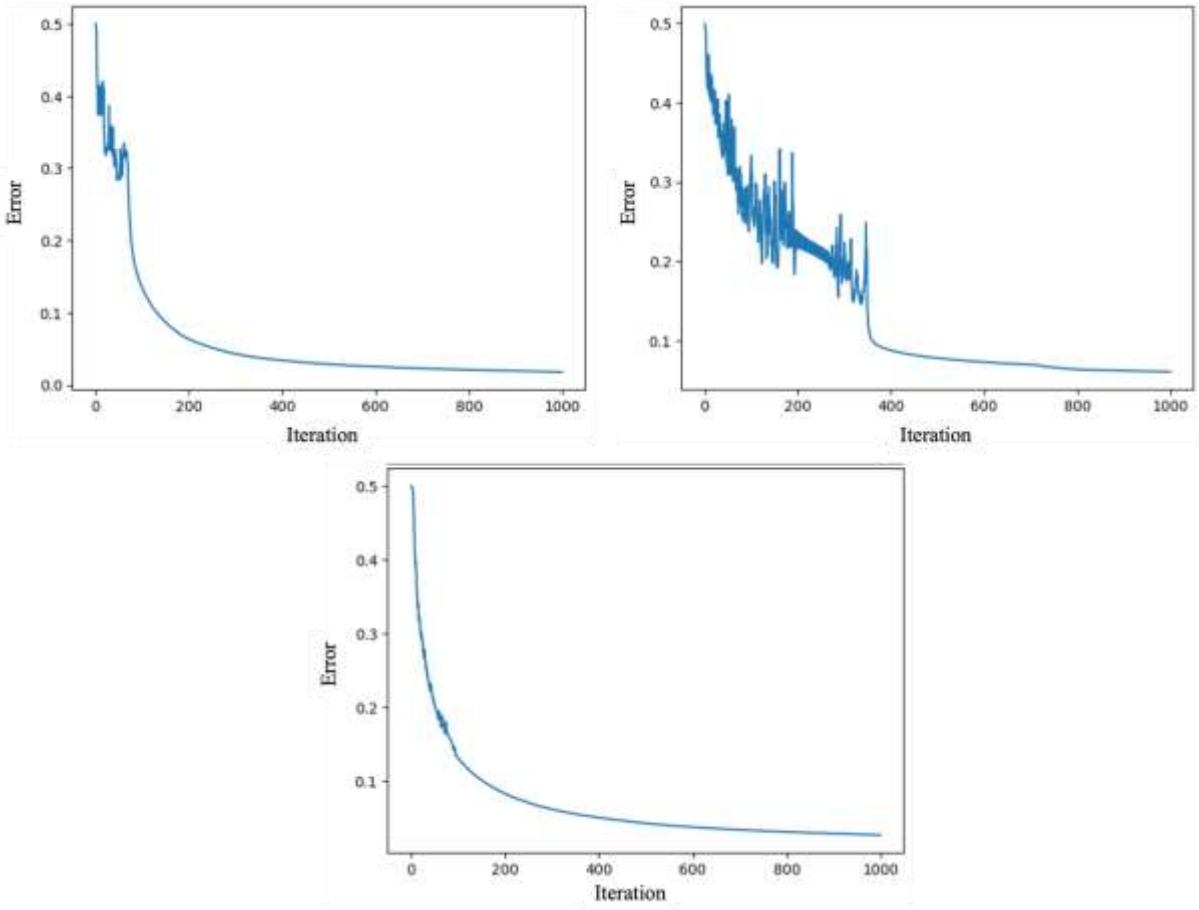

Figure 7: Loss values of the ANN model in 1000 iterations.

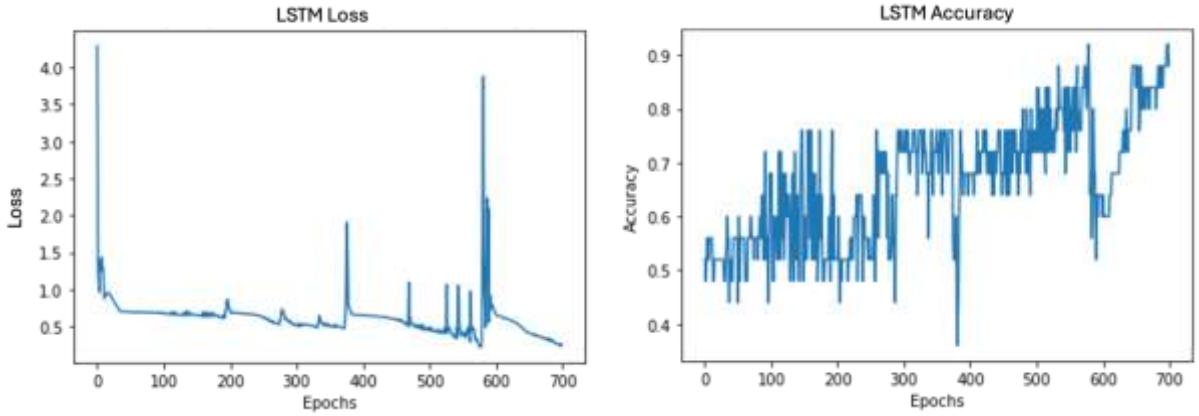

Figure 8: Accuracy rates and error/loss values obtained with the LSTM algorithm in 700 iterations.

Table 3. Final accuracy comparisons of algorithms.

	Accuracy	**MSE**	**MAPE**	**MAE**	**RMSE**
ANN	0,97	0,005	0,01	0,01	0,05
LSTM	0,84	0,082	0,15	0,15	0,28
MLP	1	0,0	0,0	0,0	0,0

IV. CONCLUSIONS

This study demonstrates the effective use of neural network models for EEG signal classification. Neural networks are powerful tools for solving complex problems and can successfully classify intricate data like EEG signals.

The results indicate high accuracy rates for all algorithms used. Feature extraction techniques applied to EEG signals produced reliable outcomes, affirming their potential for similar applications in the future. Neural networks, with their adaptability and precision, hold promise for solving increasingly complex problems.

REFERENCES

[1] Kutlu, Y., Yayik, A., Yildirim, E., & Yildirim, S. (2015). Orthogonal extreme learning machine based p300 visual event-related BCI. In Neural Information Processing: 22nd International Conference, ICONIP 2015, Istanbul, Turkey, November 9-12, 2015, Proceedings, Part II 22 (pp. 284-291). Springer International Publishing..

[2] Alptürk, E. K. (2022) Taşınabilir EEG cihazı tasarımı ve uygulamaları (Yüksek Lisans Tezi), Hatay: İskenderun Teknik Üniversitesi, 2022.

[3] M. R. Spiegel ve L. J. Stephens, Schaum's outline of theory and problems of statistics, Erlangga, 1999.

[4] M. T. Heideman, D. H. Johnson ve C. S. Burrus, «Gauss and the history of the fast Fourier transform,» IEEE ASSP Magazine, cilt 1, no. 4, pp. 14-21, 1984.

[5] M. Mazur, «A Step by Step Backpropagation Example,» 17 03 2015. [Çevrimiçi]. Available: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/.

[6] G. Cybenko , «Approximation by superpositions of a sigmoidal function,» Mathematics of Control, Signals and Systems, no. 2, p. 303–314, Aralık 1989.

[7] S. Hochreiter ve J. Schmidhuber, «Long short-term memory,» %1 içinde Neural computation, 1997.

[8] A. Graves, «Long Short-Term Memory,» %1 içinde Supervised Sequence Labelling with Recurrent Neural Networks, Springer, 2012, pp. 37-45.

[9] Alptürk, E. K. ve Y. Kutlu, Analysis of Relation between Motor Activity and Imaginary EEG Records, Journal of Artificial Intellicence with Application, no. 1, pp. 5-10, 2020.