electronics

MDPI

Article

Emotion Recognition from EEG Signals Using Recurrent Neural Networks

M. Kalpana Chowdary [1], J. Anitha [2] and D. Jude Hemanth [2,*]

[1] Department of CSE, Marri Laxman Reddy Institute of Technology, Dundigal, Hyderabad 500043, India; dr.kalpana@mlrinstitutions.ac.in
[2] Department of ECE, Karunya Institute of Technology and Sciences, Coimbatore 641114, India; anithaj@karunya.edu
* Correspondence: judehemanth@karunya.edu

Abstract: The application of electroencephalogram (EEG)-based emotion recognition (ER) to the brain–computer interface (BCI) has become increasingly popular over the past decade. Emotion recognition systems involve pre-processing and feature extraction, followed by classification. Deep learning has recently been used to classify emotions in BCI systems, and the results have been improved when compared to classic classification approaches. The main objective of this study is to classify the emotions from electroencephalogram signals using variant recurrent neural network architectures. Three architectures are used in this work for the recognition of emotions using EEG signals: RNN (recurrent neural network), LSTM (long short-term memory network), and GRU (gated recurrent unit). The efficiency of these networks, in terms of performance measures was confirmed by experimental data. The experiment was conducted by using the EEG Brain Wave Dataset: Feeling Emotions, and achieved an average accuracy of 95% for RNN, 97% for LSTM, and 96% for GRU for emotion detection problems.

Keywords: brain–computer interface; deep learning; RNN; LSTM; GRU

Citation: Chowdary, M.K.; Anitha, J.; Hemanth, D.J. Emotion Recognition from EEG Signals Using Recurrent Neural Networks. *Electronics* **2022**, *11*, 2387. https://doi.org/10.3390/electronics11152387

Academic Editor: Manohar Das

Received: 1 July 2022
Accepted: 26 July 2022
Published: 30 July 2022

1. Introduction

An important subfield of human-computer interaction is the brain–computer interface (BCI) [1]. In brain–computer interface systems, people with disabilities can operate devices using mental activities. In BCI techniques, electroencephalogram recordings from the brain are acquired, and the signals are analyzed to infer the subject's purpose. The application of EEG-based emotion recognition (ER) to the brain-computer interface (BCI) has become increasingly popular over the past decade. BCI works by acquiring electroencephalogram signals from a subject's brain and uses them to extract knowledge of the subject's intention. Every day, emotions affect the way we interact, make decisions, and think. The cognitive brain–computer interface (BCI) is developed and improved through emotion classification. Many areas, such as e-health care, EEG-based music playing, e-learning, EEG-based music therapy [2] and marketing, etc., can be enhanced by EEG-based emotion recognition in real-time. Various disorders such as stroke, brain tumors, and sleep disorders can be diagnosed or treated by monitoring EEG activity. From a psychological viewpoint, emotional states can be modeled using either discrete (joy, fear, anger, happiness, sadness, surprise) or dimensional (valence and arousal) models.

EEG signals can be analyzed using machine learning (ML) algorithms and deep learning (DL) algorithms. Traditional ML algorithms generally involve the steps of pre-processing and feature extraction, followed by classification. Even so, manual extraction does not cover all hidden features, and the formulas used to extract time and frequency domain features are often extremely complex [3]. Furthermore, EEG signals can be contaminated by electromyography artifacts, causing serious interference in the traditional machine learning techniques. Based on all these problems, our main purpose is to build a

low-complexity emotion recognition system with a low error rate and a high classification rate. Because of the aforementioned circumstances, certain deep learning methods are employed to overcome these issues. Machine learning allows a system to automatically learn and improve based on its previous experiences. Deep learning is a type of machine learning that involves the use of sophisticated algorithms and deep neural networks to train a model. The importance of deep learning is that it works with both structured and unstructured data, whereas machine learning only works with organized and semi-structured data. As the volume of data increases, the performance of machine learning algorithms declines; therefore, we need a deep learning method to retain the model's performance.

Deep learning has a wide range of applications in several industries. Although automating self-driving cars is a risky endeavor, it has lately come a step closer to being a reality. Deep learning-based models are trained and tested in simulated situations to evaluate progress on everything from recognizing a stop sign to identifying a pedestrian on the road. One very familiar deep learning application is virtual assistants. In our daily lives, we all use virtual assistants, such as Alexa, Microsoft's Cortana, Apple's Siri, and Google Assistant [4]. Deep learning is extremely valuable in pharmaceutical and medical firms for a variety of reasons, including quick diagnosis and image segmentation [5]. For example, MRI data, X-rays, and other images can be analyzed using a conventional neural network (CNN). Deep learning has risen to prominence in practically every industry. It is employed in a variety of industries, including e-commerce, advertising, chatbots, robotics, visual recognition, natural language processing, fraud detection, and manufacturing, etc. [6]. The performance of convolutional and recurrent neural networks for emotion recognition is implemented and analyzed in this study. The motivation to choose deep learning techniques, which is not possible with conventional ML techniques, is the advantage of automatic feature extraction. In this study, we investigate the possibility of improving the performance of convolution and recurrent neural networks with a focus on emotion recognition.

A list of nomenclature used throughout this paper is provided in Table 1 as follows.

Table 1. List of nomenclature used in this paper.

Nomenclature	Referred to
EEG	Electroencephalogram
BCI	Brain–Computer Interface
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-term Memory Network
GRU	Gated Recurrent Unit
DL	Deep Learning
ML	Machine Learning
FFNN	Feed Forward Neural Network
NLP	Natural Language Processing
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative

This paper is further subdivided into sections. Section 2 explains the methodology and the database; Section 3 explains the architecture of recurrent neural networks; Sections 4 and 5 explain the architecture of long short-term memory networks and gated recurrent neural networks; Section 6 describes the models used in this work for EEG-based emotion recognition: RNN, LSTM, and GRU; Section 7 discusses the experimental results; Section 8 is the comparative analysis; and Section 9 is the conclusion.

2. Methodology

The complete emotion recognition process involves data pre-processing, automatic feature extraction, and finally, classification using deep learning models. Figure 1 depicts the overall structure of this study.

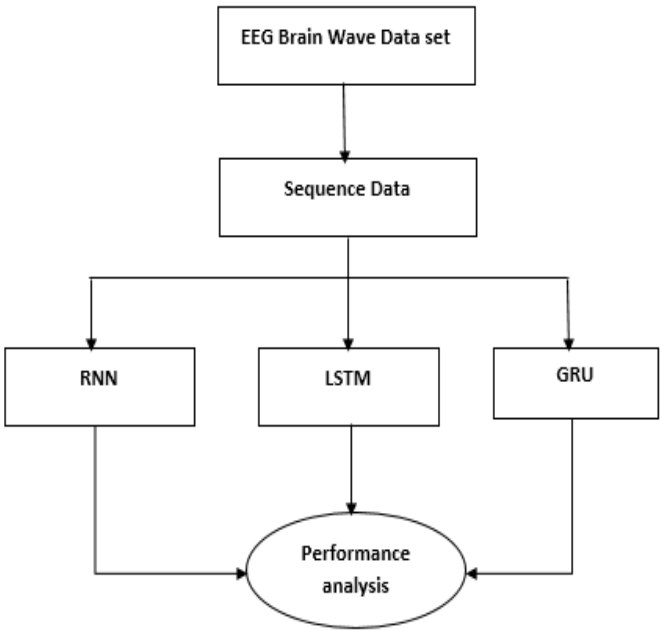

Figure 1. EEG-based emotion recognition.

Database (EEG Brain Wave Data Set: Feeling Emotions)

The data were collected for 3 min from two members, one male and one female, in each state—positive, neutral, and negative. Dry electrodes were used to record the TP9, AF7, AF8, and TP10 EEG placements using a Muse EEG headgear. Six minutes of resting data were also taken. This dataset was resampled through statistical extraction since waves must be defined mathematically in a temporal manner [7]. The web link of the database is below.

https://www.kaggle.com/datasets/birdy654/eeg-brainwave-dataset-feeling-emotions (accessed on 1 May 2022).

3. Recurrent Neural Networks (RNN)

The feed-forward neural network (FFNN) has a few flaws, which led to the development of RNN: sequential data cannot be handled, only the current input is considered, and prior inputs are not remembered. The RNN is the solution to these difficulties. An RNN can deal with sequential data by considering both present and the past inputs. Because of the internal memory, RNNs can memorize past inputs. The variation in the flow of information between an RNN and an FFNN is depicted in the Figure 2.

Some of the applications of RNN are image captioning [8], time series prediction [9], machine translation [10], and NLP [11,12], etc. RNNs are divided into four categories: one to one, many to one, one to many, and many to many [13]. Figure 3 shows the categories of RNN, and Figure 4 explains simple RNN.

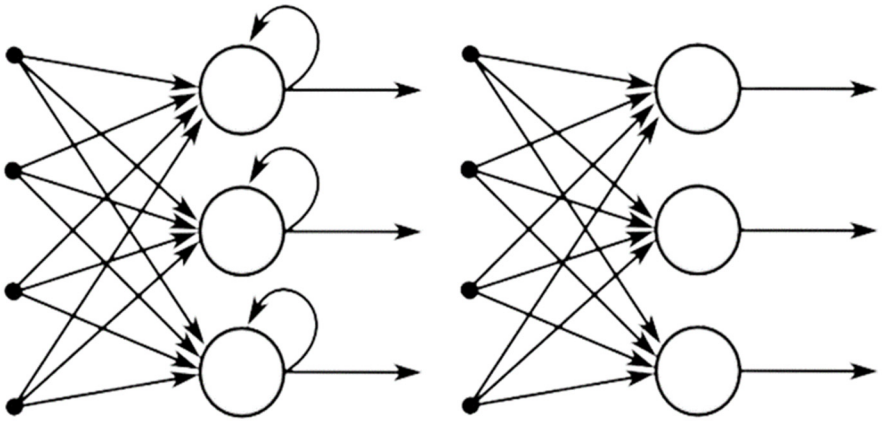

Figure 2. Recurrent neural network and feed-forward neural network.

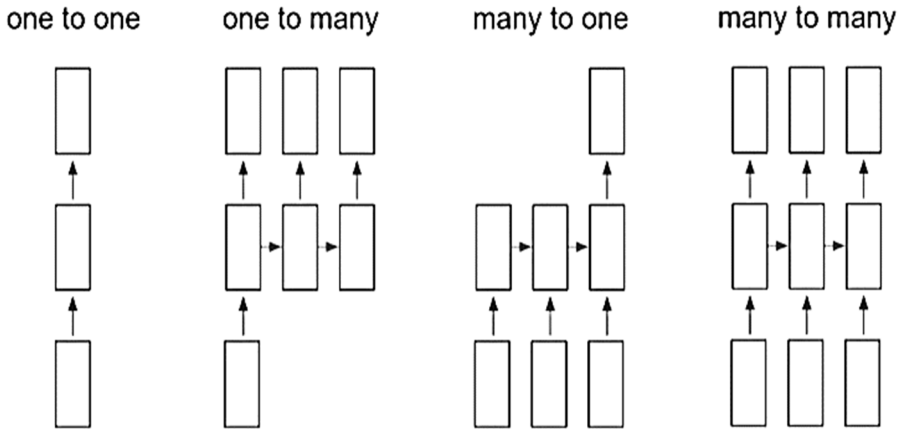

Figure 3. Different types of RNN.

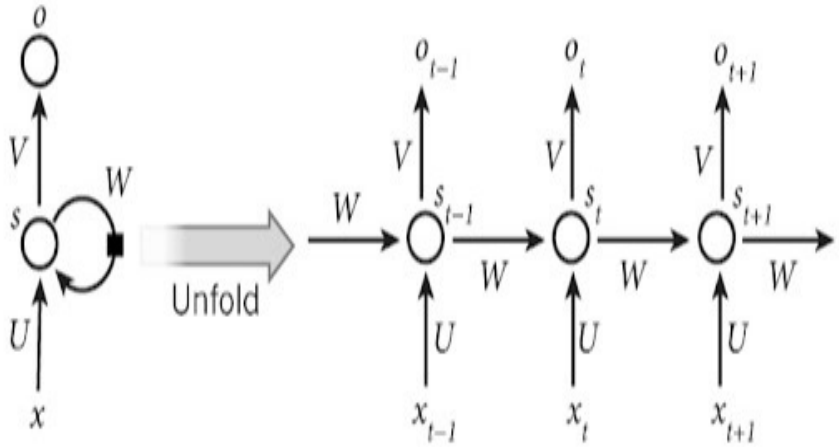

Figure 4. The simple RNN (recurrent neural network) model.

One to One: This type of neural network is used to solve general machine learning problems that have only one input and output.

Many to One: One single output is generated by this RNN based on the sequence of inputs. This type of network is used for sentiment analysis, in which a given sentence can be categorized as positive or negative based on its emotional content.

One to Many: This type of neural network has one input and several outputs. Image captions are a good example of this.

Many to Many: A series of inputs produce a series of outputs using this RNN. An example is the machine translation process.

In the above diagram, x is the input state, s is the hidden state, and O is the output [14,15]. The network's weights are U, V, and W. The internal architecture of the RNN cell is explained in Figure 5.

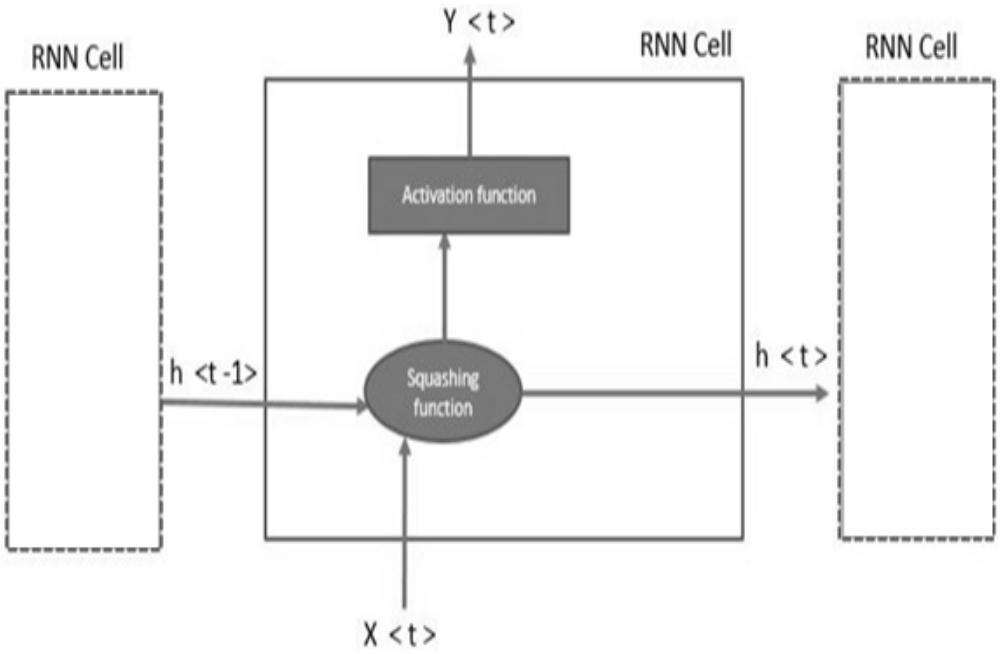

Figure 5. Working of the recurrent neural network.

The formula for calculating the current state is

$$h_t = f(h_{t-1}, X_t) \tag{1}$$

where x_t is the input state, h_t is the current state, and h_{t-1} denotes the previous state.

The formula for calculating the activation function is

$$h_t = \tan h(W_{hh}\, h_{t-1} + W_{Xh}\, X_t) \tag{2}$$

where W_{hh} stands for the recurrent neuron weights, and W_{Xh} stands for the input neuron weights.

The calculation formula for the output is

$$Y_t = W_{hy}\, h_t \tag{3}$$

Here, Y_t denotes the output and W_{hy} denotes the output layer weights.

Recurrent neural networks have problems with vanishing gradients and exploding gradients during back-propagation. An example of vanishing gradients is when a gradient is too small, so the model stops learning, or takes a long time to learn, because of it. Exploding gradients occur when the method assigns an absurdly high value to the weights for no apparent reason. A slightly modified version of RNNs with gated recurrent units and long short-term memory networks can resolve this issue. The architecture and working functionality of the LSTM network and GRU networks are explained in the next section.

4. Long Short-Term Memory Networks (LSTM)

A long short-term memory network, also known as LSTM, is an advanced RNN that can store information for a long period of time. It can handle the vanishing gradient problem encountered by RNNs. Depending on the data, the network may or may not

retain the memory. Through its gating mechanisms, the network maintains its long-term dependencies. In the network, the memory can be released or stored on demand based on the gating mechanism. Gates are the basic three components of an LSTM cell [16,17]. The forget gate is the first section, the input gate is the second, and the output gate is the third, as shown in Figure 6.

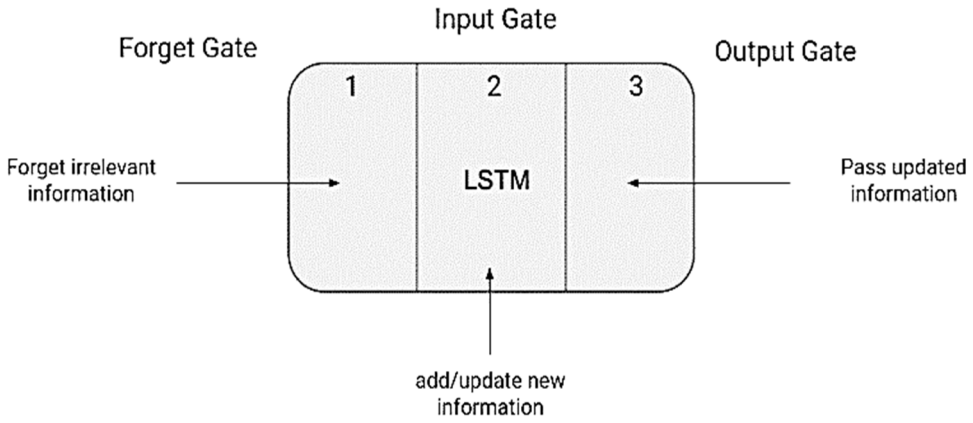

Figure 6. LSTM cell with gates.

An LSTM, like a simple RNN, contains a hidden state, with H_{t-1} representing the previous timestamp's hidden state, and H_t representing the current timestamp's hidden state. LSTMs also contain a cell state, which is represented by C_{t-1} and C_t, respectively, for past and current timestamps [18,19], as shown in Figure 7.

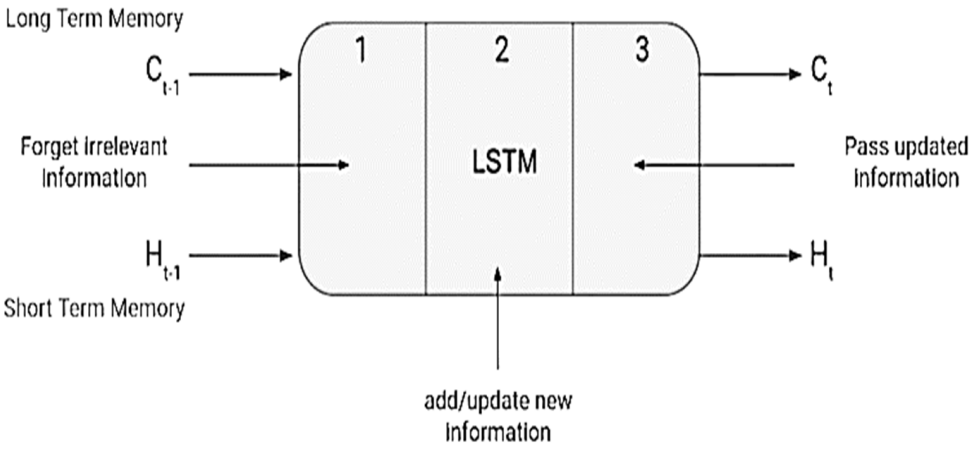

Figure 7. LSTM cell with hidden and cell states.

Here, hidden state refers to the short-term memory, whereas cell state refers to the long-term memory. An overview of how LSTM works is shown in Figure 8.

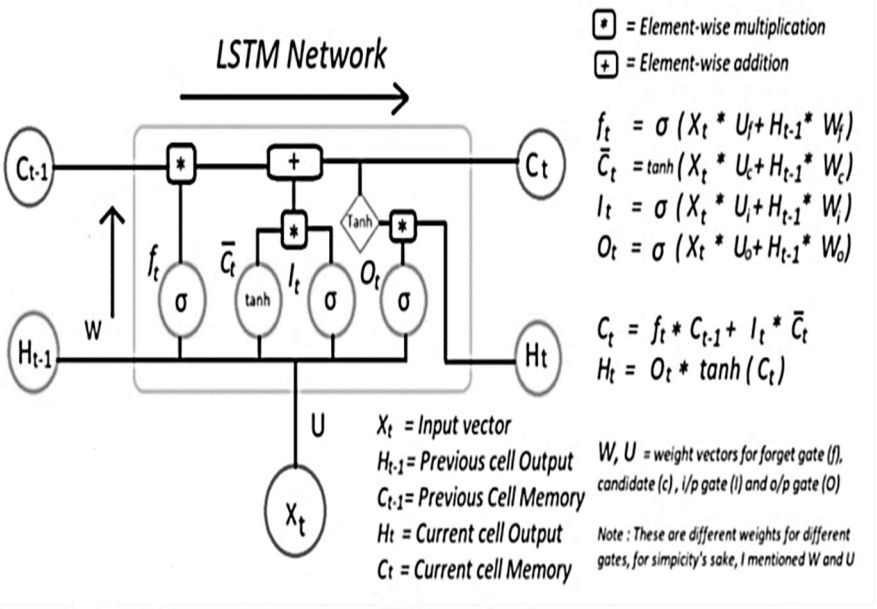

Figure 8. Internal Working of LSTM cell.

Forget gate: The forget gate is the first of the gates. If you close it, no previous memories will be saved. All old memories will pass through if you fully open this gate. It is actually an element-by-element multiplication. The forget gate equation is as follows:

$$f_t = \sigma\left(X_t * U_f + H_{t-1} * W_f\right) \tag{4}$$

If you multiply the old memory with a vector close to 0, you are attempting to erase the majority of the old memory. If you want to allow the old memory to pass, set the forget gate to 1.

$$C_{t-1} * f_t = 0 \; if \; f_t = 0 \tag{5}$$

$$C_{t-1} * f_t = C_{t-1} \; if \; f_t = 1 \tag{6}$$

where X_t and H_{t-1} are the current timestamp's input and the previous timestamp's hidden state. U_f and W_f are the weights associated with the input and hidden states.

Input gate: The input gate is the second gate. The second gate determines how much new input should be allowed in. By adjusting this gate, new and old memories should be differently affected. The input gate is used to measure the significance of new data carried by the input. The input gate's equation is as follows:

$$i_t = \sigma(X_t * U_i + H_{t-1} * W_i) \tag{7}$$

where, U_i and W_i are the weights associated with the current input and previous hidden states.

Cell state: The + operator comes next. Piecewise summation is the meaning of this operator. This action will blend the current input and the old memory. To form St., the element-wise summation of the old memory and present input are used.

$$\overline{C}_t = tanh\left(X_t * U_c + H_{t-1} * W_c\right) \text{ (New information)} \tag{8}$$

$$C_t = f_t * C_{t-1} + I_t * \overline{C}_t \text{ (Updating cell state)} \tag{9}$$

Output gate: The output for this LSTM unit must be generated. The new memory, previous output, and current input all control the output gate in this stage. This gate regulates the amount of new memory that should be sent to the following LSTM unit.

$$O_t = \sigma(X_t * U_O + H_{t-1} * W_O) \tag{10}$$

Because of the sigmoid function, its value will also be between 0 and 1. We will now utilize the modified cell states O_t and tanh to determine the current hidden state, as indicated in the equation below.

$$H_t = O_t * tanh\ (C_t) \tag{11}$$

The hidden state turns out to be a function of long-term memory (C_t) and current output. Apply the SoftMax activation to the hidden state Ht if you need to obtain the output of the current timestamp.

$$\text{Output} = \text{Softmax}\ (\text{Ht}) \tag{12}$$

5. Gated Recurrent Network (GRU)

GRUs are variants of RNN architecture that utilize gating mechanisms to control information flow between the cells of a neural network. LSTMs and GRUs operate on the same concept. In comparison to LSTM, they are quite new. This is why GRUs outperform LSTMs and have a more straightforward architecture. LSTMs are composed of two very different states, the cell state and the hidden state, which provide long- and short-term storage [20].

In GRUs, only one hidden state is transferred from one time step to another. As a result of the gating mechanisms and computations that the hidden state and input data undergo, it can simultaneously maintain both long- and short-term dependencies. Some of the applications of GRU are speech recognition, stock price prediction, machine translation, and sentiment analysis, etc. Figure 9 shows the structure of the GRU cell. At each timestamp t, it takes an input Xt and the hidden state H_{t-1} from the previous timestamp $t-1$. Later on, it outputs a new hidden state Ht, which is again passed along to the next timestamp.

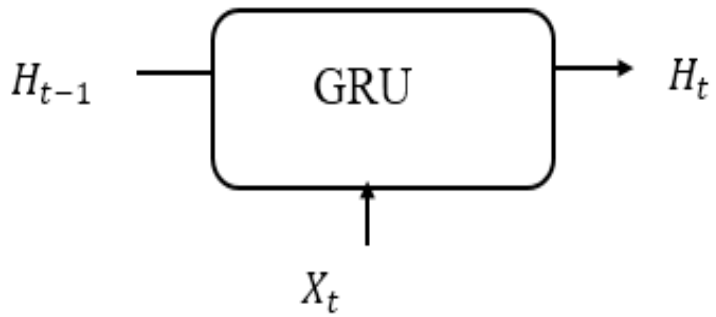

Figure 9. GRU cell.

GRU Architecture

The gated recurrent unit, in short GRU, has the same workflow as the RNN, but the gate operations are different. GRU addresses the problem of standard RNN by integrating two gates: the update gate and the reset gate, which are explained [21]. The internal architecture of the GRU cell is explained in Figure 10.

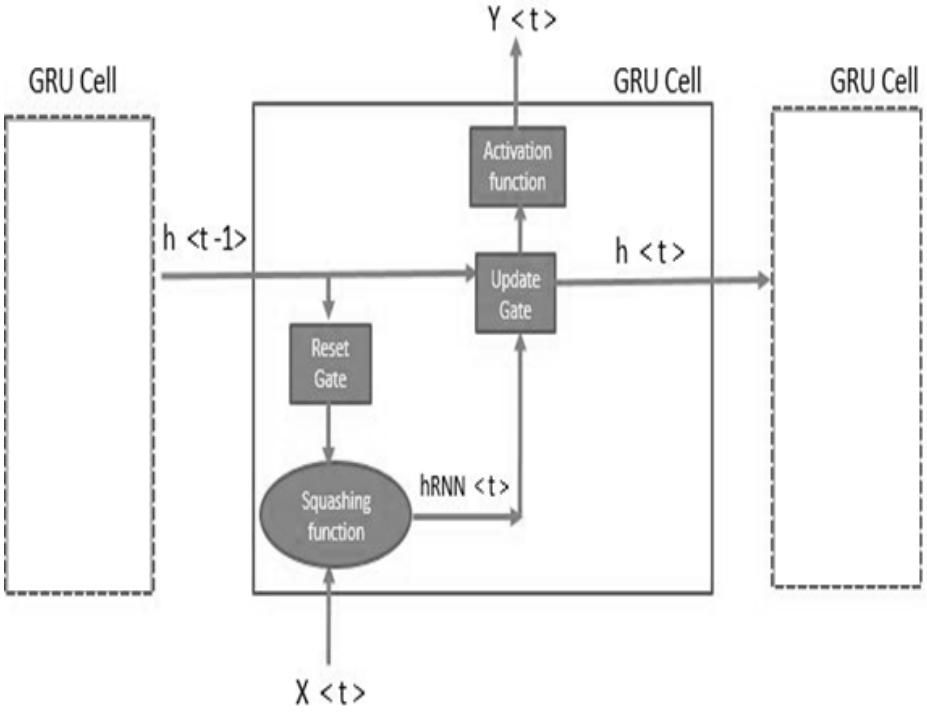

Figure 10. Function of gated recurrent network.

Update gate: The update gate functions similarly to an LSTM's input and forget gate. It determines the information that should be discarded or included. Below is the equation for the update gate that manages long-term memory.

$$U_t = \sigma(X_t * U_u + h_{t-1} * W_u) \tag{13}$$

The update gate's weight matrices are U_u and W_u.

Further, this value is compressed using the sigmoid function to maintain the range from 0 to 1. In this way, the update gate alleviates the problem of vanishing gradients.

Reset gate: Using the reset gate, we can decide how much past data must be ignored; in simple terms, it decides whether or not the previous cell state is relevant. Below is the equation for the reset gate that manages the short-term memory.

$$r_t = \sigma(X_t * U_r + h_{t-1} * W_r) \tag{14}$$

The reset gate's weight matrices are U_r and W_r.

The sigmoid function transforms values in the range of 0 to 1, with values closer to zero being ignored, and those closer to 1 are further processed.

Candidate hidden state: As a first step, the Hadmard product between the reset gate and hidden state from the previous timestamp was calculated. This was then fed into the tanh function to determine the candidate's hidden state. The formula for the candidate hidden state is shown below.

$$\overline{h} = \tan h(X_t\, U_h + r_t \odot h_{t-1}\, W_h) \tag{15}$$

Current hidden state: Compute the Hadmard product between the update gate and the hidden state vector from the previous timestamp. Subtract the update gate from one to create a new vector, and then calculate the Hadmard product of the new vector with the candidate hidden state. Finally, sum the two vectors to generate the currently hidden state vector.

$$h_t = U_t \odot h_{t-1} + (1 - U_t) \odot \overline{h} \tag{16}$$

where h_t indicates the current hidden state.

6. Architectures Used in This Study

In this study of EEG-based emotion recognition, three architectures are used: RNN (recurrent neural network), LSTM (long short-term memory network), and GRU (gated recurrent unit). The experiments were conducted using the EEG Brain Wave Dataset: Feeling Emotions.

6.1. RNN Architecture

The RNN algorithm used in this work consists of an RNN layer with 128 units, a flatten layer, and finally a dense layer with softmax activation.

Learning features are determined using the RNN layers, and a dense layer is used to classify them into emotions from raw EEG signals. Simple RNN is used in this study. It is a simplified version of the real RNN in Keras. The loss function is a sparse categorical cross-entropy, and the optimizer used is an Adam optimizer. Figure 11 shows the RNN model used in this work for EEG-based emotion recognition.

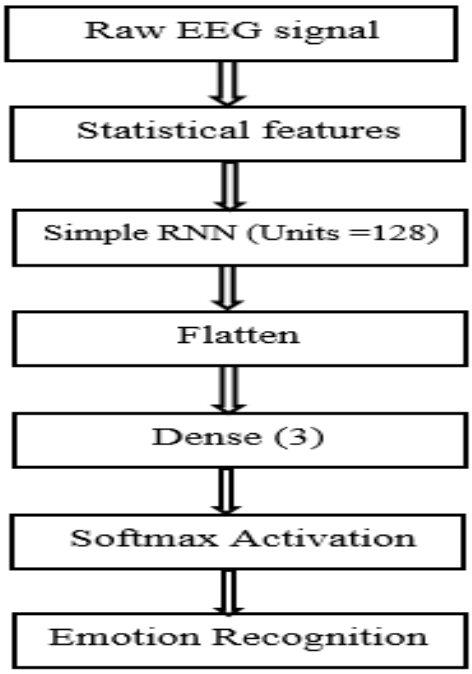

Figure 11. Detailed RNN model.

Table 2 shows the Keras implementation of the RNN architecture and a detailed explanation of the RNN model with input and output vector shapes, as shown in Figure 12.

Table 2. Keras implementation of the RNN model.

Model: "model"		
Layer (type)	**Output Shape**	**Param #**
input_1 (InputLayer)	(None, 2548)	0
tf.expand dims (TFOpLambda)	(None, 2548, 1)	0
simple rnn (SimpleRNN)	(None, 2548, 128)	16,640
flatten (Flatten)	(None, 326144)	0
dense (Dense)	(None, 3)	978,435
Total params: 995,075 Trainable param: 995,075 Non-trainable params: 0		

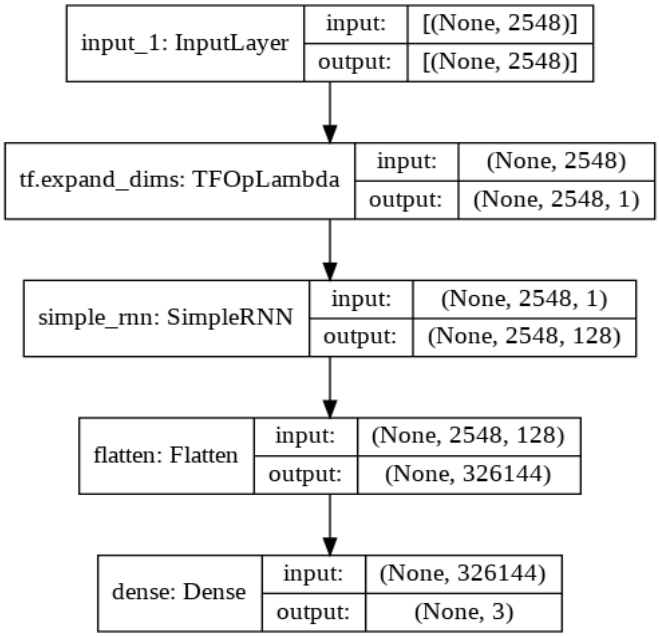

Figure 12. Detailed RNN architecture with input and output vector shapes.

6.2. LSTM Architecture

The LSTM algorithm used in this work consists of an LSTM layer with 128 units, a flatten layer, and finally, a dense layer with softmax activation.

Learning features are determined using the LSTM layers, and a dense layer is used to classify them into emotions from raw EEG signals. Figure 13 shows the LSTM model for EEG-based emotion recognition. From the Keras API layers, the LSTM layer and dense layers are imported. Here, we used the LSTM layer with 128 internal units, and the return sequence is kept true. For each input time step, return sequences generate the hidden state output. The loss function is sparse categorical cross-entropy and the optimizer used is an Adam optimizer.

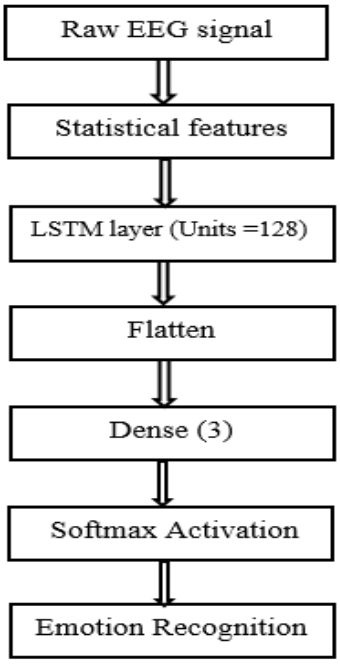

Figure 13. Detailed LSTM model.

Table 3 shows the Keras implementation of the LSTM architecture and the detailed explanation of the LSTM model with input and output vector shapes, as shown in Figure 14.

Table 3. Keras implementation of the LSTM model.

Model: "model_1"		
Layer (type)	**Output Shape**	**Param #**
input_2 (InputLayer)	(None, 2548)	0
tf.expand_dims_1 (TFOpLambda)	(None, 2548, 1)	0
lstm_1 (LSTM)	(None, 2548, 128)	66,560
flatten 1 (Flatten)	(None, 326144)	0
dense_1 (Dense)	(None, 3)	978,435
Total params: 1,044,995 Trainable param: 1,044,995 Non-trainable params: 0		

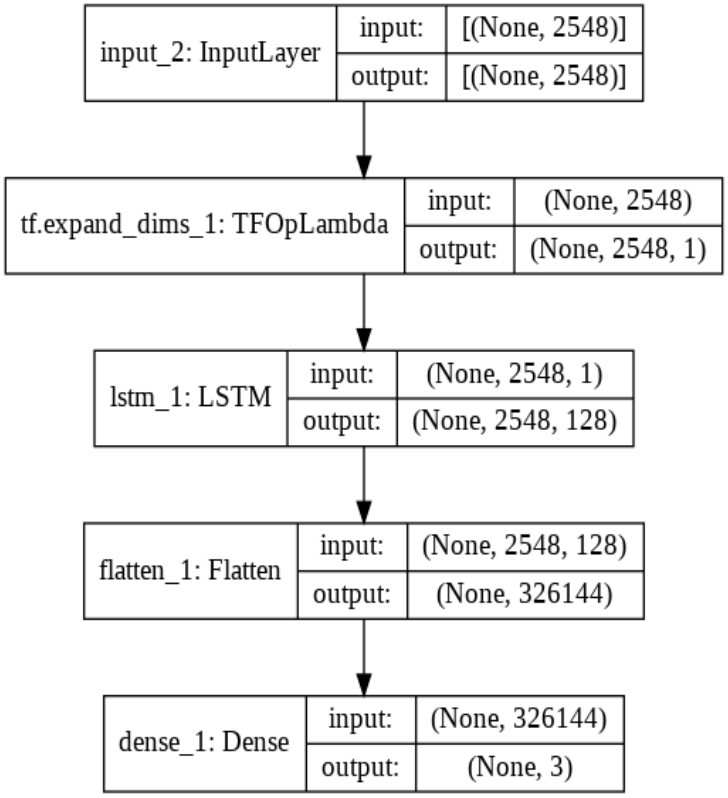

Figure 14. Detailed LSTM architecture with input and output vector shapes.

6.3. GRU Architecture

The GRU algorithm used in this study consists of a GRU layer with 128 units, a flatten layer, and finally, a dense layer with softmax activation. Learning features are determined using the GRU layers, and a dense layer is used to classify them into emotions from raw EEG signals. Figure 15 shows the GRU model used in this study for EEG-based emotion recognition.

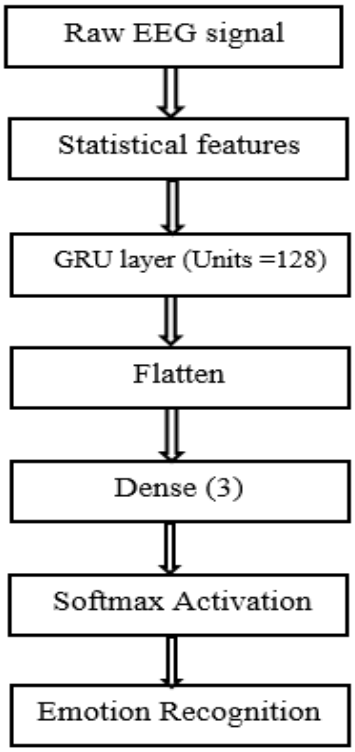

Figure 15. Detailed GRU model.

Table 4 shows the Keras implementation of the GRU architecture and the detailed explanation of the GRU model with input and output vector shapes, as shown in Figure 16.

Table 4. Keras implementation of the GRU model.

Model: "model_3"		
Layer (type)	**Output Shape**	**Param #**
input_4 (InputLayer)	(None, 2548)	0
tf.expand_dims_3 (TFOpLambda)	(None, 2548, 1)	0
gru_1 (GRU)	(None, 2548, 128)	50,304
flatten 3 (Flatten)	(None, 326144)	0
dense_3 (Dense)	(None, 3)	978,435
Total params: 1,028,739 Trainable param: 1,028,739 Non-trainable params: 0		

With less training data, GRUs perform better than LSTMs, particularly when applied to language modeling tasks. In cases where additional inputs are needed for the network, GRUs are easier to modify, and thus are simpler with less code. The GRUs also have fewer parameters than the LSTM, as evidenced by the model summaries. With 128 internal units, the total number of parameters for the LSTM model is 1,044,995, and for the GRU model, it is 1,028,739.

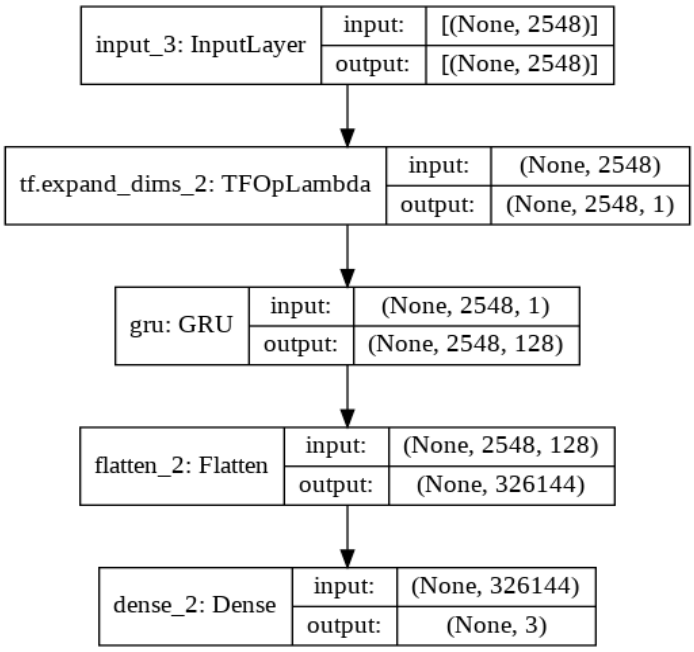

Figure 16. Detailed GRU architecture with input and output vector shapes.

7. Experimental Results and Discussions

This database has a total of 2132 samples for three emotions: positive (708 samples), negative (708 samples), and neutral (716 samples). The data were collected for 3 min durations per state. In this dataset, 1492 samples were used for training purposes and 640 were used for testing purposes. All the implementations use the Keras library with Tensor flow backend.

7.1. Results of RNN Architecture

Figure 17 shows the data fitting results using the RNN architecture.

```
Epoch 8/30
38/38 [==============================] - 101s 3s/step - loss: 0.3488 - accuracy: 0.9883 - val_loss: 6.0850 - val_accuracy: 0.9298
Epoch 9/30
38/38 [==============================] - 102s 3s/step - loss: 0.4215 - accuracy: 0.9899 - val_loss: 6.8325 - val_accuracy: 0.9164
Epoch 10/30
38/38 [==============================] - 101s 3s/step - loss: 0.0851 - accuracy: 0.9941 - val_loss: 5.9676 - val_accuracy: 0.9331
Epoch 11/30
38/38 [==============================] - 100s 3s/step - loss: 0.1022 - accuracy: 0.9975 - val_loss: 5.1344 - val_accuracy: 0.9298
Epoch 12/30
38/38 [==============================] - 103s 3s/step - loss: 1.0315e-04 - accuracy: 1.0000 - val_loss: 3.7099 - val_accuracy: 0.936
Epoch 13/30
38/38 [==============================] - 102s 3s/step - loss: 0.0197 - accuracy: 0.9983 - val_loss: 6.2676 - val_accuracy: 0.9298
Epoch 14/30
38/38 [==============================] - 101s 3s/step - loss: 0.0828 - accuracy: 0.9966 - val_loss: 3.9152 - val_accuracy: 0.9398
Epoch 15/30
38/38 [==============================] - 100s 3s/step - loss: 0.0342 - accuracy: 0.9983 - val_loss: 5.0354 - val_accuracy: 0.9465
```

Figure 17. Fitting results by using the RNN architecture.

Figure 18 shows the accuracy and loss of the RNN model. When the number of epochs changes, the loss and accuracy values also change. Table 5 displays the confusion matrix for the test data of 640 samples, and Table 6 shows the performance metrics of the model calculated by using the confusion matrix.

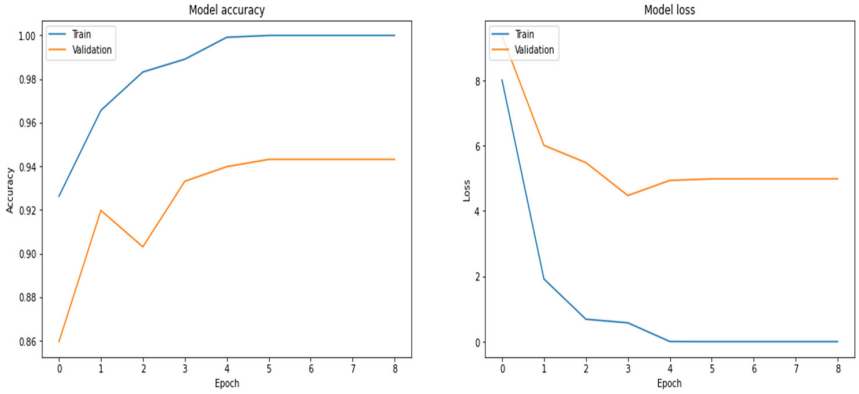

Figure 18. Accuracy and loss curves by using RNN.

Table 5. Confusion matrix by using RNN model.

	Negative	Neutral	Positive
Negative	181	1	19
Neutral	0	222	9
Positive	9	2	197

Table 6. Performance measures by using the RNN model.

	TP	TN	FP	FN	Sensitivity/Recall	Specificity	Precision	F1 Score	Accuracy
Negative	181	430	9	20	0.90	0.97	0.95	0.92	0.95
Neutral	222	406	3	9	0.96	0.97	0.98	0.96	0.98
Positive	197	404	28	11	0.94	0.97	0.87	0.89	0.93
Average results					0.93	0.97	0.93	0.92	0.95

The metrics of the model accuracy, specificity, and sensitivity are calculated by using true positive (TP), false positive (FP), false negative (FN), and true negative (TN) values.

From the above calculations, the F1 score is 0.92 and the accuracy of the model by using RNN is 95%.

7.2. Results of LSTM Architecture

Figure 19 shows the data fitting results of the LSTM architecture.

```
Epoch 1/30
38/38 [==============================] - 16s 382ms/step - loss: 0.6500 - accuracy: 0.9715 - val_loss: 0.8128 - val_accuracy: 0.9532
Epoch 2/30
38/38 [==============================] - 14s 358ms/step - loss: 0.1988 - accuracy: 0.9832 - val_loss: 1.0339 - val_accuracy: 0.9431
Epoch 3/30
38/38 [==============================] - 14s 359ms/step - loss: 0.3065 - accuracy: 0.9841 - val_loss: 1.9143 - val_accuracy: 0.9231
Epoch 4/30
38/38 [==============================] - 14s 360ms/step - loss: 0.0460 - accuracy: 0.9941 - val_loss: 0.7968 - val_accuracy: 0.9599
Epoch 5/30
38/38 [==============================] - 14s 358ms/step - loss: 0.0127 - accuracy: 0.9992 - val_loss: 0.5787 - val_accuracy: 0.9699
Epoch 6/30
38/38 [==============================] - 14s 359ms/step - loss: 1.2065e-06 - accuracy: 1.0000 - val_loss: 0.5818 - val_accuracy: 0.9732
Epoch 7/30
38/38 [==============================] - 14s 360ms/step - loss: 8.2225e-07 - accuracy: 1.0000 - val_loss: 0.5809 - val_accuracy: 0.9732
Epoch 8/30
38/38 [==============================] - 14s 362ms/step - loss: 6.3498e-07 - accuracy: 1.0000 - val_loss: 0.5803 - val_accuracy: 0.9732
Epoch 9/30
38/38 [==============================] - 14s 358ms/step - loss: 5.1591e-07 - accuracy: 1.0000 - val_loss: 0.5800 - val_accuracy: 0.9732
Epoch 10/30
38/38 [==============================] - 14s 359ms/step - loss: 4.4928e-07 - accuracy: 1.0000 - val_loss: 0.5796 - val_accuracy: 0.9732
```

Figure 19. Fitting results of the LSTM architecture.

Figure 20 shows the accuracy and loss of the graphs using the LSTM network.

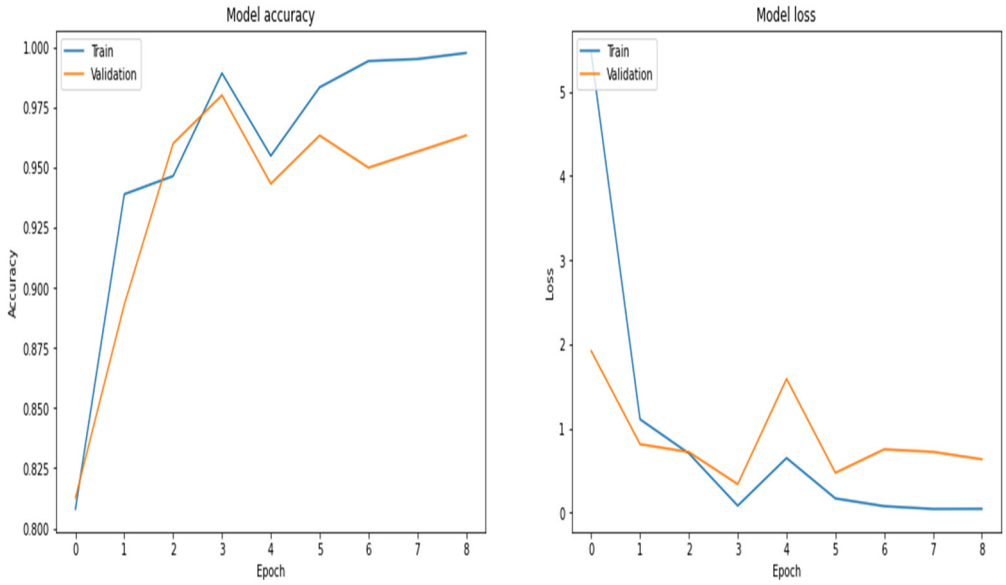

Figure 20. Accuracy and loss curves using the LSTM model.

Table 7 displays the confusion matrix for the test data of 640 samples for the three emotions of negative, neutral, and positive.

Table 7. Confusion matrix of LSTM model.

	Negative	Neutral	Positive
Negative	194	0	7
Neutral	0	227	4
Positive	8	2	198

Table 8 shows the performance measures of the LSTM model used in this study. The performance measures are calculated by using TP, TN, FP, and FN values that are taken from the confusion matrix.

Table 8. Performance measures of the LSTM model.

	TP	TN	FP	FN	Sensitivity/Recall	Specificity	Precision	F1 Score	Accuracy
Negative	194	431	8	7	0.96	0.98	0.96	0.95	0.97
Neutral	227	407	2	4	0.98	0.99	0.99	0.98	0.99
Positive	198	421	11	10	0.95	0.97	0.94	0.94	0.96
Average results					0.96	0.98	0.96	0.95	0.97

From the above calculations, the F1 score is 0.95, and the accuracy of the model by using LSTM is 97%.

7.3. Results of GRU Architecture

Figure 21 shows the data fitting results by using the GRU architecture.

```
Epoch 1/30
38/38 [==============================] - 15s 349ms/step - loss: 29.8543 - accuracy: 0.7309 - val_loss: 4.2715 - val_accuracy: 0.8294
Epoch 2/30
38/38 [==============================] - 13s 333ms/step - loss: 2.8968 - accuracy: 0.8952 - val_loss: 2.1216 - val_accuracy: 0.9331
Epoch 3/30
38/38 [==============================] - 13s 333ms/step - loss: 3.2328 - accuracy: 0.9120 - val_loss: 4.5265 - val_accuracy: 0.8997
Epoch 4/30
38/38 [==============================] - 13s 333ms/step - loss: 0.7447 - accuracy: 0.9665 - val_loss: 0.7030 - val_accuracy: 0.9632
Epoch 5/30
38/38 [==============================] - 13s 333ms/step - loss: 0.5200 - accuracy: 0.9723 - val_loss: 0.5436 - val_accuracy: 0.9666
Epoch 6/30
38/38 [==============================] - 13s 332ms/step - loss: 0.1764 - accuracy: 0.9841 - val_loss: 1.9942 - val_accuracy: 0.9298
Epoch 7/30
38/38 [==============================] - 13s 332ms/step - loss: 0.5237 - accuracy: 0.9715 - val_loss: 0.9960 - val_accuracy: 0.9599
Epoch 8/30
38/38 [==============================] - 13s 332ms/step - loss: 0.2461 - accuracy: 0.9874 - val_loss: 1.2653 - val_accuracy: 0.9498
Epoch 9/30
38/38 [==============================] - 13s 332ms/step - loss: 0.5987 - accuracy: 0.9749 - val_loss: 2.6677 - val_accuracy: 0.9532
Epoch 10/30
38/38 [==============================] - 13s 333ms/step - loss: 0.1817 - accuracy: 0.9908 - val_loss: 1.2528 - val_accuracy: 0.9632
```

Figure 21. Fitting results by using the GRU architecture.

Figure 22 shows the accuracy and loss of the graphs by using GRU network.

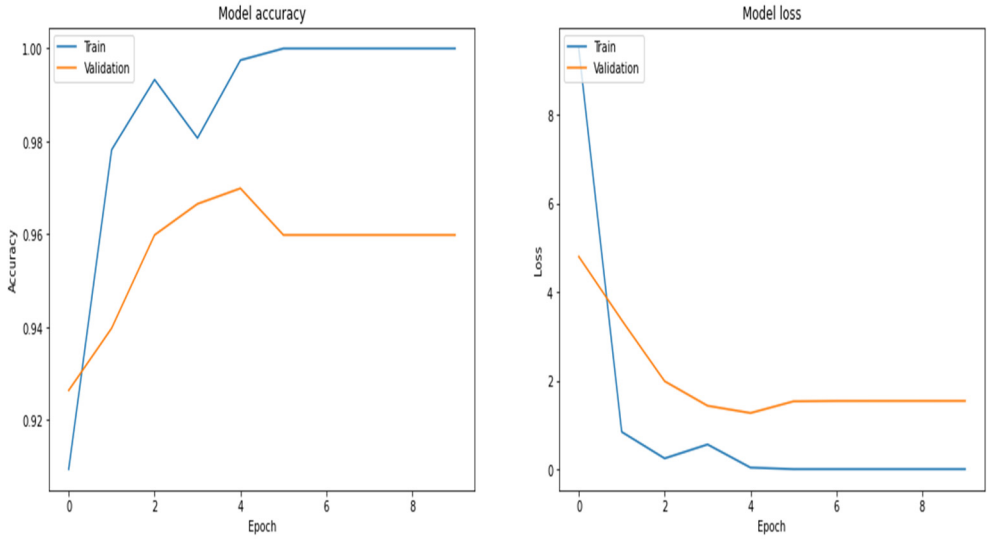

Figure 22. Accuracy and loss curves by using the GRU model.

Table 9 displays the confusion matrix for the test data of 640 samples for the three emotions of negative, neutral, and positive.

Table 9. Confusion matrix of the GRU model.

	Negative	Neutral	Positive
Negative	194	0	7
Neutral	0	228	3
Positive	12	4	192

The sensitivity, specificity, precision, F1 score, and accuracy are calculated by using TP, TN, FP and FN. Table 10 shows the performance measures of the GRU model.

Table 10. Performance measures of the GRU model.

	TP	TN	FP	FN	Sensitivity/Recall	Specificity	Precision	F1 Score	Accuracy
Negative	194	427	12	7	0.96	0.97	0.94	0.94	0.97
Neutral	228	405	4	3	0.98	0.99	0.98	0.98	0.98
Positive	192	422	10	16	0.92	0.97	0.95	0.93	0.95
Average results					0.95	0.97	0.95	0.95	0.96

From the above calculations, the F1 score is 0.95 and the accuracy of the model by using GRU is 96%.

8. Comparative Analysis of Recurrent Neural Networks Based Emotion Recognition with EEG Signals

For EEG emotion detection, the EEG Brain Wave Dataset is used in this work. The dataset contains a total of 2134 samples for three emotions: positive (708 samples), negative (708 samples), and neutral (716 samples). In this dataset, 1492 samples are used for training purposes and 640 are used for testing purposes. Table 11 shows the performance metrics of RNN, LSTM, and GRU models. The performance measures of sensitivity, specificity, precision, F1 score, and accuracy are calculated for every model. Performance measures are calculated using TP, TN, FP, and FN values that are obtained from the confusion matrix.

Table 11. Performance metrics of RNN, LSTM and GRU models.

S. No.	Network	Sensitivity	Specificity	Precision	F1 Score	Accuracy
1	RNN	0.93	0.97	0.93	0.92	0.95
2	LSTM	0.96	0.98	0.96	0.95	0.97
3	GRU	0.95	0.97	0.95	0.95	0.96

RNN has the drawbacks of vanishing and exploding gradients, so compared to the RNN network, both LSTM and GRU networks achieved a high accuracy. GRU uses fewer parameters than LSTM [22], so it uses less memory and is faster. However, LSTM is more accurate in a larger dataset. When dealing with large sequences, the LSTM algorithm is chosen, but when less memory is required and faster results are desired, the GRU algorithm may be used. In this study, LSTM achieved 1% more accuracy than GRU. Table 12 shows a comparative analysis of other existing works for EEG-based emotion recognition.

Table 12. Comparative analysis with other existing works for EEG emotion detection.

Method/Author	Average Accuracy Rate (%)
DNN+ Sparse Auto encoder/[23]	96
DCNN/[24]	85
Multi column CNN/[25]	90
3D CNN/[26]	88

9. Conclusions and Future Scope

Three models are developed for EEG signal-based emotion recognition: RNN, LSTM, and GRU models. The experiments were conducted using the EEG Brain Wave Database. The accuracy achieved is 95% using the RNN model, 97% using the LSTM model, and 96% using the GRU model. Compared to the RNN model, both LSTM and GRU models achieved a high accuracy. In this study, LSTM achieved 1% more accuracy than GRU. A limitation of this study was a lack of datasets, as most of the EEG datasets are not publicly available. If a greater number of samples is available, it is possible to recognize a higher number of emotions. In the future, these networks will be implemented for multimodal datasets and real time data to recognize emotions.

Author Contributions: Literature Search, Figures, Study Design and Data Analysis, M.K.C.; Data Interpretation and Data Validation, J.A.; Data Interpretation, Data Validation and Supervision, D.J.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: "EEG Brain wave Data Set: Feeling emotions" at https://www.kaggle.com/datasets/birdy654/eeg-brainwave-dataset-feeling-emotions (accessed on 1 May 2022).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ortiz-Echeverri, C.J.; Salazar-Colores, S.; Rodríguez-Reséndiz, J.; Gómez-Loenzo, R.A. A new approach for motor imagery classification based on sorted blind source separation, continuous wavelet transform, and convolutional neural network. *Sensors* **2019**, *19*, 4541. [CrossRef] [PubMed]
2. Liu, Y.; Sourina, O.; Nguyen, M.K. Real-time EEG-based emotion recognition and its applications. In *Transactions on Computational Science XII*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 256–277.
3. Sánchez-Reyes, L.M.; Rodríguez-Reséndiz, J.; Avecilla-Ramírez, G.N.; García-Gomar, M.L.; Robles-Ocampo, J.B. Impact of eeg parameters detecting dementia diseases: A systematic review. *IEEE Access* **2021**, *9*, 78060–78074. [CrossRef]
4. Hoy, M.B. Alexa, Siri, Cortana, and more: An introduction to voice assistants. *Med. Ref. Serv. Q.* **2018**, *37*, 81–88. [CrossRef] [PubMed]
5. Lundervold, A.S.; Lundervold, A. An overview of deep learning in medical imaging focusing on MRI. *Z. Med. Phys.* **2019**, *29*, 102–127. [CrossRef] [PubMed]
6. Sarker, I.H. Ai-based modeling: Techniques, applications and research issues towards automation, intelligent and smart systems. *SN Comput. Sci.* **2022**, *3*, 158. [CrossRef] [PubMed]
7. Bird, J.J.; Ekart, A.; Buckingham, C.D.; Faria, D.R. Mental emotional sentiment classification with an eeg-based brain-machine interface. In Proceedings of the International Conference on Digital Image and Signal Processing (DISP'19), Oxford, UK, 23–30 April 2019.
8. Liu, X.; Xu, Q.; Wang, N. A survey on deep neural network-based image captioning. *Vis. Comput.* **2019**, *35*, 445–470. [CrossRef]
9. Giles, C.L.; Lawrence, S.; Tsoi, A.C. Noisy time series prediction using recurrent neural networks and grammatical inference. *Mach. Learn.* **2001**, *44*, 161–183. [CrossRef]
10. Singh, S.P.; Kumar, A.; Darbari, H.; Singh, L.; Rastogi, A.; Jain, S. Machine translation using deep learning: An overview. In Proceedings of the 2017 International Conference on Computer, Communications and Electronics (Comptelix), Jaipur, India, 1–2 July 2017; pp. 162–167.
11. Pattanayak, S. Natural language processing using recurrent neural networks. In *Pro Deep Learning with TensorFlow*; Apress: Berkeley, CA, USA, 2017; pp. 223–278.
12. Yin, W.; Kann, K.; Yu, M.; Schütze, H. Comparative study of CNN and RNN for natural language processing. *arXiv* **2017**, arXiv:1702.01923.
13. Medsker, L.; Jain, L.C. (Eds.) *Recurrent Neural Networks: Design and Applications*; CRC Press: Boca Raton, FL, USA, 1999.
14. Graves, A. Generating sequences with recurrent neural networks. *arXiv* **2013**, arXiv:1308.0850.
15. Ebrahimi Kahou, S.; Michalski, V.; Konda, K.; Memisevic, R.; Pal, C. Recurrent neural networks for emotion recognition in video. In Proceedings of the 2015 ACM on International Conference on Multimodal Interaction, Seattle, WA, USA, 9–13 November 2015; pp. 467–474.
16. Graves, A. Long short-term memory. In *Supervised Sequence Labelling with Recurrent Neural Networks*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 37–45.
17. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
18. Sainath, T.N.; Vinyals, O.; Senior, A.; Sak, H. Convolutional, long short-term memory, fully connected deep neural networks. In Proceedings of the 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), South Brisbane, QLD, Australia, 19–24 April 2015; pp. 4580–4584.
19. Malhotra, P.; Vig, L.; Shroff, G.; Agarwal, P. Long short term memory networks for anomaly detection in time series. In Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, Bruges, Belgium, 22–24 April 2015; Volume 89, pp. 89–94.
20. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv* **2014**, arXiv:1412.3555.
21. Rana, R. Gated recurrent unit (GRU) for emotion classification from noisy speech. *arXiv* **2016**, arXiv:1612.07778.
22. Yang, S.; Yu, X.; Zhou, Y. Lstm and gru neural network performance comparison study: Taking yelp review dataset as an example. In Proceedings of the 2020 International Workshop on Electronic Communication and Artificial Intelligence (IWECAI), Shanghai, China, 12–14 June 2020; pp. 98–101.
23. Liu, J.; Wu, G.; Luo, Y.; Qiu, S.; Yang, S.; Li, W.; Bi, Y. EEG-based emotion classification using a deep neural network and sparse autoencoder. *Front. Syst. Neurosci.* **2020**, *14*, 43. [CrossRef] [PubMed]

24. Shao, H.-M.; Wang, J.-G.; Wang, Y.; Yao, Y.; Liu, J. EEG-Based Emotion Recognition with Deep Convolution Neural Network. In Proceedings of the 2019 IEEE 8th Data Driven Control and Learning Systems Conference (DDCLS), Dali, China, 24–27 May 2019; pp. 1225–1229.
25. Yang, H.; Han, J.; Min, K. A multi-column CNN model for emotion recognition from EEG signals. *Sensors* **2019**, *19*, 4736. [CrossRef] [PubMed]
26. Salama, E.S.; El-Khoribi, R.A.; Shoman, M.E.; Shalaby, M.A.W. EEG-based emotion recognition using 3D convolutional neural networks. *Int. J. Adv. Comput. Sci. Appl.* **2018**, *9*, 329–337. [CrossRef]