Evolution in Electrical and Electronic Engineering Vol. 3 No. 2 (2022) 844-855
© Universiti Tun Hussein Onn Malaysia Publisher's Office

EEEE

Homepage: http://publisher.uthm.edu.my/periodicals/index.php/eeee
e-ISSN : 2756-8458

Identification of Epilepsy from EEG Signal using Recurrent Neural Network

Nurul Izyan Ainaa Jumari[1], Ashok Vajravelu[1]*

[1]Department of Electronic Engineering, Faculty of Electrical & Electronic Engineering
Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA

*Corresponding Author Designation

DOI: https://doi.org/10.30880/eeee.2022.03.02.099
Received 29 June 20212; Accepted 20 September 2022; Available online 30 October 2022

Abstract: The implementation of a Deep Neural Network (DNN) is widely used for a long time ago. Nowadays, the DNN still undergoes improvements in a lot of research for different subjects including epilepsy. This disorder remains one of the diagnoses that can be done by using DNN. For this study, one of the DNN models which is the Recurrent Neural Network (RNN) is utilized to make identification for epilepsy based on the Electroencephalography (EEG) dataset. The dataset is attained from UCI Machine Learning Repository for Epileptic Seizure Recognition Data which is an open-source dataset for epilepsy. The dataset with epileptic can be distinguished from the non-epileptic one by using the LSTM model, the upgrade version of RNN. The performance evaluation in terms of accuracy also obtained a value of more than 98%.

Keywords: EEG, epilepsy, RNN, LSTM, ANN

1. Introduction

When the brain fails to function normally, there will be some disorders or abnormalities. One of the disorders is epilepsy. Epilepsy is a chronic disease of the brain and a central nervous system disorder that can cause unprovoked seizures condition in patients [1]. During epilepsy, the cognitive and social behavior of the human will be an issue. The brain is a complex organ that is related to almost all functions in our bodies. It has control over sensory organs, systems and more. Thus, if the brain becomes scrambled and cannot function properly, creating abnormalities, the human body will be in a terrible state. It is planned to attempt to identify in which lobe of the brain, the impact will be more by using Recurrent Neural Network (RNN), one of the DNN models.

In [7], Convolutional Neural Network is used to detect epileptic seizures based on the University of Bonn EEG dataset. CNN uses the image as an input, assigning importance to various aspects of the image and also being capable to differentiate one from another for classification. CNN has four layers, which the first three layers being used to detect or learn about the input's features while the last layer is used for classification. Study [2] was also conducted with CNN but with a different pre-processing

Corresponding author: ashok@uthm.edu.my method. [3] used Short-Term Fourier Transform (STFT) spectrogram and Continuous Wavelet Transformation (CWT) scalogram but [2] used Gramian Angular Summation Field (GASF) and Gramian Angular Difference Field (GADF) methods to transform EEG signal into a 2D image for CNN input.

Long Short-Term Memory (LSTM) is popular in studies related to epilepsy identification [4]- [6]. A hybrid neural network like [5] can also be used, which utilized LSTM with CNN, to create a hybrid neural network named Deep C-LSTM. These past studies show that there are a lot of ways to utilize neural networks for epilepsy identification. DNN works with a lot of structures like CNN, RNN and more. The networks also can work together in one system like CNN and LSTM which produces a C-LSTM structure. Aside from that, the process before classifying by using the network, like pre-processing and feature extraction process also can be done in different ways. It can be concluded that DNN can function in a variety of ways, but with the correct choice of technique and process, the best results can be acquired.

As per the literature study, LSTM, which is the upgrade of RNN, is identified as one of the best DNNs to proceed with this study. This study is using EEG as a base to construct LSTM. The EEG signal is identified either as epileptic or normal through this network application. With the identification result obtained with the chosen network model, a performance evaluation is done with some comparison with the other DNN models constructed specifically for this purpose plus with previous related research.

2. Materials and Methods

Long Short-Term Memory (LSTM) is a part of RNN which is widely used for sequential data. Sequential data is the type of data, in which points in the dataset rely on other datasets' points. Time-series data like EEG is a type of sequential data. This means the usage of EEG is very suitable for this network [6]. A typical neural network deduces that the dataset used in the network is independent of each other. However, RNN is different as the output from this model depends on previous input, within a sequence. The network is said to have "memory" as it still remembers the previous input to influence the current input and output as in Figure 1. This network shares the same parameters across the layers of the network. The structure of RNN is shown in Figure 2.

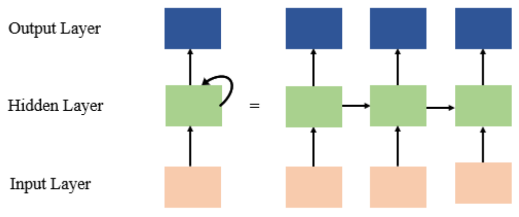


Figure 1: RNN and its unfold structure (right).

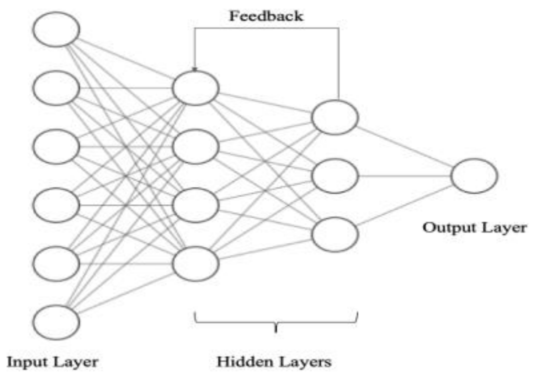


Figure 2: RNN full structure [5].

LSTM is the improvised version of RNN with the existence of memory extension. LSTM mainly depends on three gates which are input, forget and output. With the forget gate, the network can decide either to take or leave the data in the network depending on the importance of the data. Over time, the network learns which information to delete. The input gate decides whether to take the new input or not. These decisions on the two gates affect the output gates.

Next, LSTM can perform backpropagation. The forward propagation is performed from the input, through the network, then to the output. Backpropagation is the reverse of forwarding propagation. The weights of the network can be fine-tuned or adjusted based on the error estimation obtained while performing backpropagation. The weights of the network are the parameters specified to transform the input in the hidden layer. So, with the weights adjusted to a better specification, an error can be reduced and the model can be more accurate.

The difference between normal RNN and LSTM can be seen in Figures 3 and 4. The use of different layers in LSTM like the amount of tanh definitely can produce a satisfying result for its particular purpose.

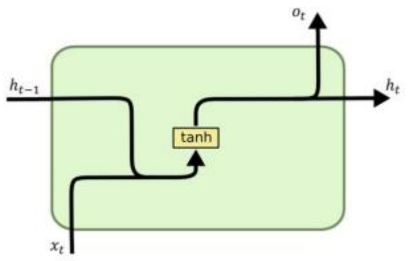


Figure 3: RNN neuron [7]

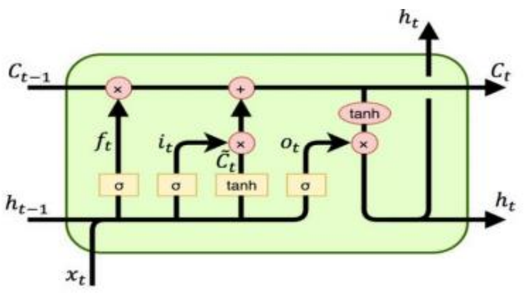


Figure 4: LSTM neuron [7]

2.1 Dataset

The EEG dataset is obtained from the UCI Machine Repository-Epileptic Seizure Recognition Dataset. This dataset contains five different sets or classes for different conditions of EEG recordings. Class 1 is for EEG recordings with epileptic seizures recorded from the patients. Classes 2 and 3 are for epileptic EEG signals but each recording is taken with sensors at different spatial locations. Class 2 is taken from the epileptogenic zone, but Class 3 is from the other side of the zone. Class 4 is for surface EEG recordings of healthy subjects with eyes closed while Class 5 is when the eyes are opened. This data is set into a .csv file for easy access [8].

2.3 Tools

As this study is fully software-based, the software that has been chosen for this study is Jupyter Notebook which implements Python language. Python has a lot of libraries that focus on the application of Machine Learning, Deep Learning and Data Science like keras and tensorflow. Thus, this software is selected to build the DNN models needed for this study. A suitable environment of Python is made with the libraries needed for this study, which are:

a) tensorflow
b) keras
c) numpy
d) matplotlib
e) pandas
f) sklearn

2.4 Workflow

The flow of this study works as Figure 5.

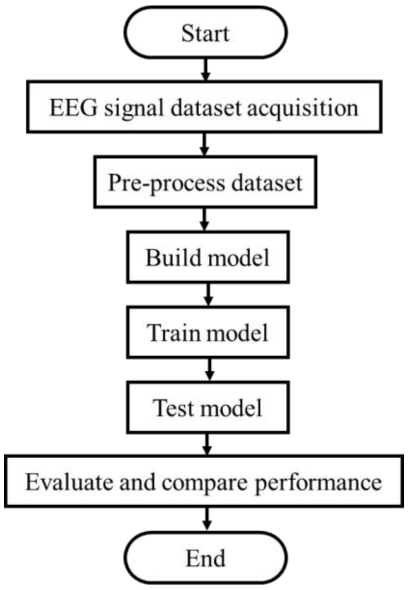

Start
EEG signal dataset acquisition
Pre-process dataset
Build model
Train model
Test model
Evaluate and compare performance
End

Figure 5: Workflow.

Before starting to build the LSTM model, data pre-processing is needed. The dataset obtained from UCI Machine Learning Repository has been segmented and transformed into a .csv file for easy access. The dataset is then split randomly into train and test data with a ratio of 80:20.

Feature scaling or standardization is then applied to both train and test data. This step is done to normalize or rescale data into a certain range, like 0 to 1. As the dataset used is big and varies in values, this normalization step helps to equalize the data. There are two ways of standardization that have been applied in this study. First, by using the direct Eq. 1 and by using a built-in function which is StandardScaler. This function works like Eq. 1.

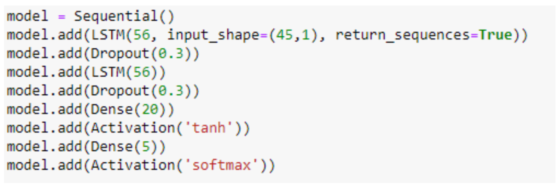
$$X = \frac{X - \mu}{\sigma} \qquad Eq. 1$$

$$X = value\ for\ X\_train/X\_test/Y\_test\ (data)$$
$$\mu = mean\ of\ data$$
$$\sigma = standard\ deviation\ of\ data$$

The pre-processed data then proceeds to be used in the neural network model for the classification of epileptic and non-epileptic signals.

2.5 LSTM Model

As LSTM is a sequential network, function Sequential from keras library is used. This is because Sequential is a model with a linear stack of layers that has one input and one output of tensor, suitable for LSTM. The specifications for the layers are shown in Figure 6. The LSTM model is built layer by layer with lstm, dropout, dense and activation layers.



```
model = Sequential()
model.add(LSTM(56, input_shape=(45,1), return_sequences=True))
model.add(Dropout(0.3))
model.add(LSTM(56))
model.add(Dropout(0.3))
model.add(Dense(20))
model.add(Activation('tanh'))
model.add(Dense(5))
model.add(Activation('softmax'))
```

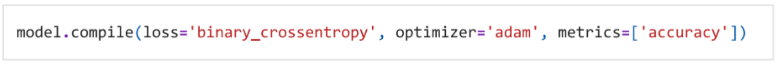
Figure 6: LSTM model specification.

The LSTM input shape was a 3D tensor shape with timesteps and features of 45 and 1, respectively. A tensor refers to a matrix or vector that represents the data. The LSTM output is expected to produce a unit of 56. After that, dropout is added. Dropout functions by ignoring random units and setting the units to 0 during training, plus it helps prohibit the model from overfitting [9]. Usually, a rate ranging from 20% to 50% is used for the layer.

The dense layer is operating as the hidden and output layers for this model. Dense layers with units of 20 and 5 are used. The unit represents the output size of the layer. This layer is used to change the dimension or size of the output. It is also connected and received all inputs from the previous layer. Along with each dense layer, an activation function is prepared after the layer.

The activation function is set up to two types, tanh and softmax. The hyperbolic tangent activation function, tanh, which is commonly used in LSTM [10], is utilized as it is compatible with LSTM [11]. For output, the activation function chosen is softmax. It specializes in the classification of two or more classes [12]. So, the softmax function is used for training and classification as it is suitable. Overall, an activation layer is added to control how the model is trained and with a suitable layer, the output layer can predict the prominent outcome.

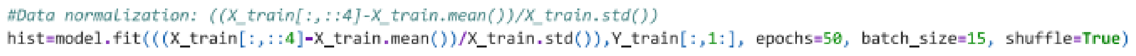
Next, for model training, the model is compiled with loss, optimizer and metrics configurations as shown in Figure 7. The loss function measures the difference between the predicted and actual values. The chosen loss function for this model is Binary Crossentropy (BCE), which is specified for binary classification (0-1). The optimizer is needed to modify features like weights or learning rates to decrease any possible loss in model training. Adam (Adaptive Moment Estimation) optimizer is chosen to complete the task. The learning rate for the Adam optimizer is 0.001. Then, the metrics for model performance are specified to accuracy.



```
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Figure 7: Compilation of LSTM model.

A fit function is utilized for model training with epochs set to 50. An epoch refers to a round or cycle of full model training with the data. As training with an entire dataset for an epoch might be too big, the training has to be done in batches. So, the batch size is set to 15 as depicted in Figure 8. The shuffle is set to true, so the data used is shuffled before starting each epoch. With this, model training can be done. The loss and accuracy findings for each epoch from the model training can be evaluated in Section 3, Results and Discussion. The trained model is then proceeded to be used for model testing.



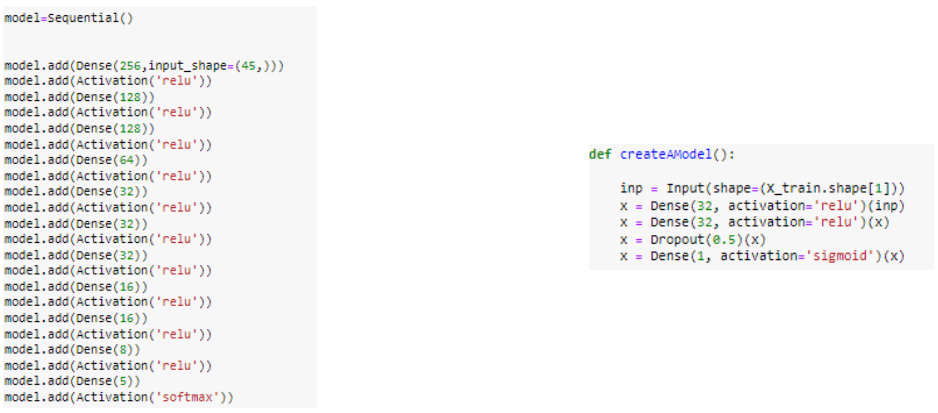
```
#Data normalization: ((X_train[:,::4]-X_train.mean())/X_train.std())
hist=model.fit(((X_train[:,::4]-X_train.mean())/X_train.std()),Y_train[:,1:], epochs=50, batch_size=15, shuffle=True)
```

Figure 8: Train LSTM model.

For model testing, function prediction is used to identify the epileptic and non-epileptic signals. To identify these signals, the test data is used in the trained model to make predictions of the epileptic and non-epileptic signals. Then, the prediction result is compared with the actual test data to obtain the accuracy of the prediction result from the trained model. The overall process is repeated for the other two Artificial Neural Network (ANN) models with different structures of layers and specifications (Section 2.6). These two models are used for performance comparison in model training and testing with the LSTM model.

2.6 ANN Model for Performance Comparison

There are two ANN models used for comparison with the main model, each is shown in Figure 9 by using the same dataset with an 80:20 ratio for the train and test data. Dataset pre-processing is also done in the same manner. However, both models have different specifications of layers and input/output shapes. ANN has a different build or construction compared to LSTM. The ANN is built with input, hidden and output layers [13]. The hidden layers can be more than one as this is to add more depth to the network. However, the same hyperparameters for model training are applied, the same as the main model, LSTM.



```
model=Sequential()


model.add(Dense(256,input_shape=(45,)))
model.add(Activation('relu'))
model.add(Dense(128))
model.add(Activation('relu'))
model.add(Dense(128))
model.add(Activation('relu'))
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dense(32))
model.add(Activation('relu'))
model.add(Dense(32))
model.add(Activation('relu'))
model.add(Dense(32))
model.add(Activation('relu'))
model.add(Dense(16))
model.add(Activation('relu'))
model.add(Dense(16))
model.add(Activation('relu'))
model.add(Dense(8))
model.add(Activation('relu'))
model.add(Dense(5))
model.add(Activation('softmax'))
```

```
def createAModel():

    inp = Input(shape=(X_train.shape[1]))
    x = Dense(32, activation='relu')(inp)
    x = Dense(32, activation='relu')(x)
    x = Dropout(0.5)(x)
    x = Dense(1, activation='sigmoid')(x)
```

(a) **(b)**

Figure 9: Specifications of (a) Model 1 and (b) Model 2.

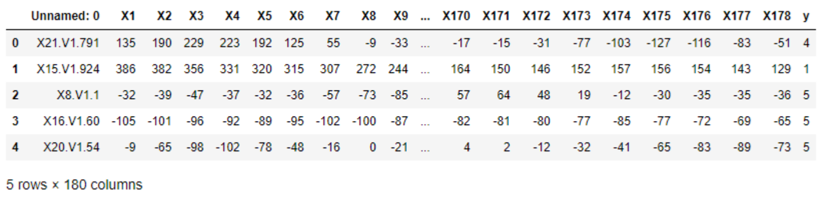
Model 1 has a lot of hidden layers (dense layers). The input/output shape is set to be reduced as the layers are finished towards the output layer. The activation used for hidden layers is Rectified Linear Activation (ReLU) and softmax. ReLU is one of the most commonly used functions aside from sigmoid and tanh as it is famous for its simplicity [14]. Model 2 does not use a lot hidden layers as Model 1. The activation layers used are ReLU and sigmoid. The sigmoid function is placed at the last dense layer as it is useful for binary classification [15].

3. Results and Discussion

This section shows the result of pre-processing the dataset, which is used for model training and testing. The accuracy-loss values from model training and accuracy value from model testing are also shown. With these values, performance comparison and evaluation can be done.

3.1 Pre-processing Dataset

The dataset taken from the UCI Machine Repository for Epileptic Seizure Recognition Dataset is already undergone pre-processing for segmentation as shown in Figure 10. For data splitting, the train data takes 80% (9200 samples) and the other 20% (2300 samples) for test data.



	Unnamed: 0	X1	X2	X3	X4	X5	X6	X7	X8	X9	...	X170	X171	X172	X173	X174	X175	X176	X177	X178	y
0	X21.V1.791	135	190	229	223	192	125	55	-9	-33	...	-17	-15	-31	-77	-103	-127	-116	-83	-51	4
1	X15.V1.924	386	382	356	331	320	315	307	272	244	...	164	150	146	152	157	156	154	143	129	1
2	X8.V1.1	-32	-39	-47	-37	-32	-36	-57	-73	-85	...	57	64	48	19	-12	-30	-35	-35	-36	5
3	X16.V1.60	-105	-101	-96	-92	-89	-95	-102	-100	-87	...	-82	-81	-80	-77	-85	-77	-72	-69	-65	5
4	X20.V1.54	-9	-65	-98	-102	-78	-48	-16	0	-21	...	4	2	-12	-32	-41	-65	-83	-89	-73	5

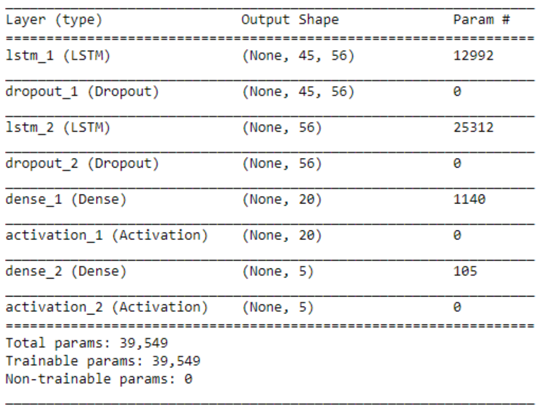
5 rows × 180 columns

Figure 10: Data representation in .csv file

3.2 Model Training

3.2.1 LSTM Model

The built model can be summarized into an output table like in Figure 11 which consists of a layer, its output shape and parameter. The chosen layers type is shown with their respective output shape which is produced based on the specifications shown in Section 2.

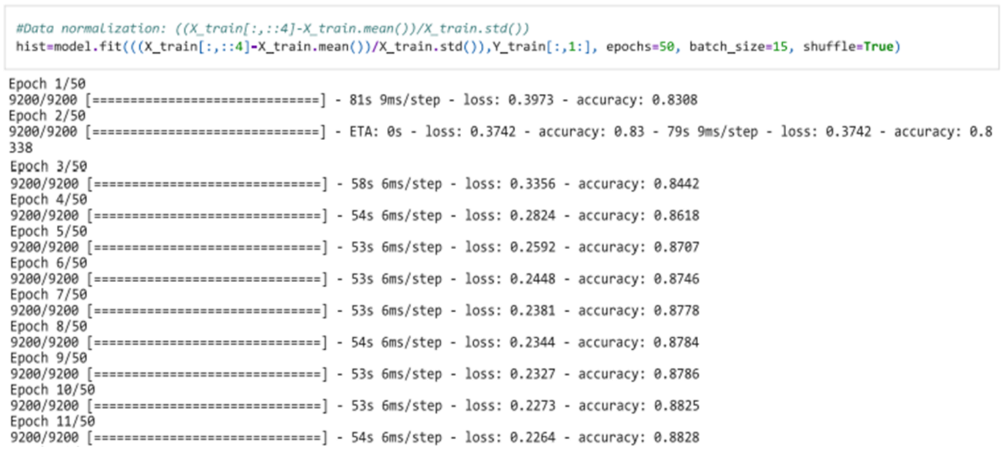


```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 45, 56)            12992
_________________________________________________________________
dropout_1 (Dropout)          (None, 45, 56)            0
_________________________________________________________________
lstm_2 (LSTM)                (None, 56)                25312
_________________________________________________________________
dropout_2 (Dropout)          (None, 56)                0
_________________________________________________________________
dense_1 (Dense)              (None, 20)                1140
_________________________________________________________________
activation_1 (Activation)    (None, 20)                0
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 105
_________________________________________________________________
activation_2 (Activation)    (None, 5)                 0
=================================================================
Total params: 39,549
Trainable params: 39,549
Non-trainable params: 0
_________________________________________________________________
```

Figure 11: LSTM model layer by layer

Next, the training model by using the normalized dataset is proceeded, with 50 epochs (Figure 12). For each epoch, 9200 samples of train data are used for the network model training. Upon completion of one epoch, its execution time is shown alongside the values of accuracy and loss for model training. The LSTM model training took more than 50s to complete one epoch thus making the overall execution time for training longer. It can also be seen that the loss decreased, and accuracy increased towards the end of epochs. These two values are then presented in graph form for further discussion in Section 3.4.1.

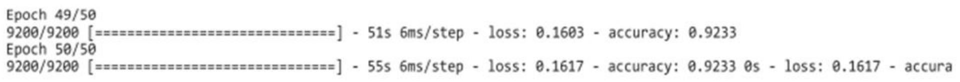
The next step is model testing which produced the accuracy of epilepsy identification made by the trained model in Section 3.4.2.



```
#Data normalization: ((X_train[:,::4]-X_train.mean())/X_train.std())
hist=model.fit(((X_train[:,::4]-X_train.mean())/X_train.std()),Y_train[:,1:], epochs=50, batch_size=15, shuffle=True)
```

```
Epoch 1/50
9200/9200 [==============================] - 81s 9ms/step - loss: 0.3973 - accuracy: 0.8308
Epoch 2/50
9200/9200 [==============================] - ETA: 0s - loss: 0.3742 - accuracy: 0.83 - 79s 9ms/step - loss: 0.3742 - accuracy: 0.8
338
Epoch 3/50
9200/9200 [==============================] - 58s 6ms/step - loss: 0.3356 - accuracy: 0.8442
Epoch 4/50
9200/9200 [==============================] - 54s 6ms/step - loss: 0.2824 - accuracy: 0.8618
Epoch 5/50
9200/9200 [==============================] - 53s 6ms/step - loss: 0.2592 - accuracy: 0.8707
Epoch 6/50
9200/9200 [==============================] - 53s 6ms/step - loss: 0.2448 - accuracy: 0.8746
Epoch 7/50
9200/9200 [==============================] - 53s 6ms/step - loss: 0.2381 - accuracy: 0.8778
Epoch 8/50
9200/9200 [==============================] - 54s 6ms/step - loss: 0.2344 - accuracy: 0.8784
Epoch 9/50
9200/9200 [==============================] - 53s 6ms/step - loss: 0.2327 - accuracy: 0.8786
Epoch 10/50
9200/9200 [==============================] - 53s 6ms/step - loss: 0.2273 - accuracy: 0.8825
Epoch 11/50
9200/9200 [==============================] - 54s 6ms/step - loss: 0.2264 - accuracy: 0.8828
```

(a)



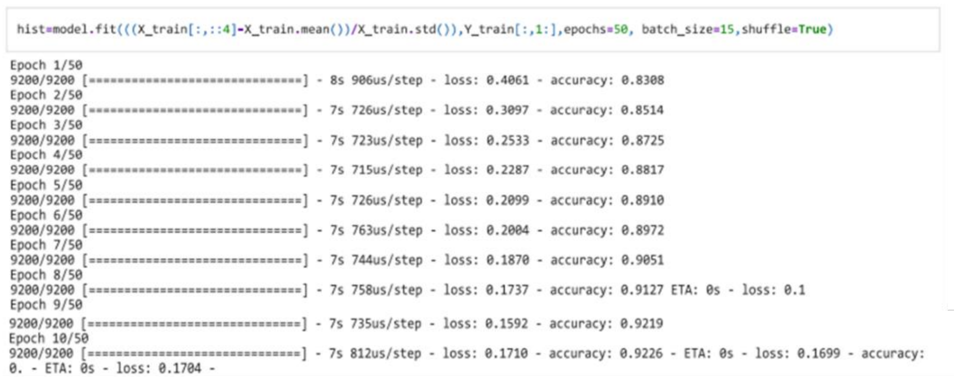
```
Epoch 49/50
9200/9200 [==============================] - 51s 6ms/step - loss: 0.1603 - accuracy: 0.9233
Epoch 50/50
9200/9200 [==============================] - 55s 6ms/step - loss: 0.1617 - accuracy: 0.9233 0s - loss: 0.1617 - accura
```

(b)

Figure 12: (a) Start and (b) finish of LSTM model training at 50 epochs

3.3.2 ANN Model

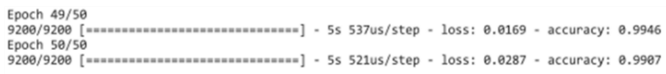
Based on Figures 13 and 14, Model 1 took a longer execution time per epoch compared to Model 2. This might happen because of the total of hidden layers used. Nevertheless, the execution time for ANN model training is still shorter than LSTM. The loss decreased and accuracy increased towards the end of 50 epochs.



```
hist=model.fit(((X_train[:,::4]-X_train.mean())/X_train.std()),Y_train[:,1:],epochs=50, batch_size=15,shuffle=True)
```

```
Epoch 1/50
9200/9200 [==============================] - 8s 906us/step - loss: 0.4061 - accuracy: 0.8308
Epoch 2/50
9200/9200 [==============================] - 7s 726us/step - loss: 0.3097 - accuracy: 0.8514
Epoch 3/50
9200/9200 [==============================] - 7s 723us/step - loss: 0.2533 - accuracy: 0.8725
Epoch 4/50
9200/9200 [==============================] - 7s 715us/step - loss: 0.2287 - accuracy: 0.8817
Epoch 5/50
9200/9200 [==============================] - 7s 726us/step - loss: 0.2099 - accuracy: 0.8910
Epoch 6/50
9200/9200 [==============================] - 7s 763us/step - loss: 0.2004 - accuracy: 0.8972
Epoch 7/50
9200/9200 [==============================] - 7s 744us/step - loss: 0.1870 - accuracy: 0.9051
Epoch 8/50
9200/9200 [==============================] - 7s 758us/step - loss: 0.1737 - accuracy: 0.9127 ETA: 0s - loss: 0.1
Epoch 9/50
9200/9200 [==============================] - 7s 735us/step - loss: 0.1592 - accuracy: 0.9219
Epoch 10/50
9200/9200 [==============================] - 7s 812us/step - loss: 0.1710 - accuracy: 0.9226 - ETA: 0s - loss: 0.1699 - accuracy:
0. - ETA: 0s - loss: 0.1704 -
```

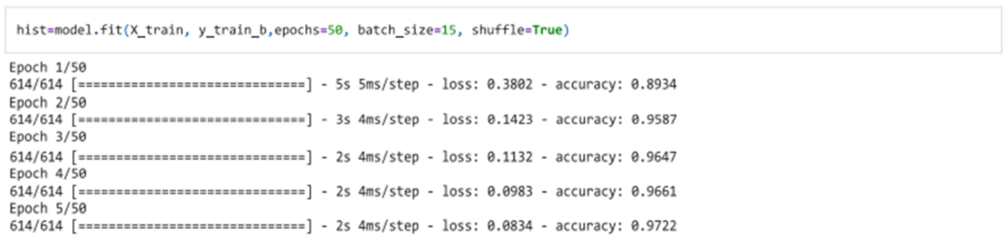
(a)



```
Epoch 49/50
9200/9200 [==============================] - 5s 537us/step - loss: 0.0169 - accuracy: 0.9946
Epoch 50/50
9200/9200 [==============================] - 5s 521us/step - loss: 0.0287 - accuracy: 0.9907
```

(b)

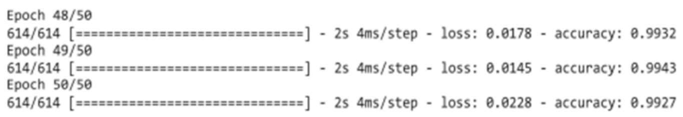
Figure 13: (a) Start and (b) finish of Model 1 training at 50 epochs



```
hist=model.fit(X_train, y_train_b,epochs=50, batch_size=15, shuffle=True)
```

```
Epoch 1/50
614/614 [==============================] - 5s 5ms/step - loss: 0.3802 - accuracy: 0.8934
Epoch 2/50
614/614 [==============================] - 3s 4ms/step - loss: 0.1423 - accuracy: 0.9587
Epoch 3/50
614/614 [==============================] - 2s 4ms/step - loss: 0.1132 - accuracy: 0.9647
Epoch 4/50
614/614 [==============================] - 2s 4ms/step - loss: 0.0983 - accuracy: 0.9661
Epoch 5/50
614/614 [==============================] - 2s 4ms/step - loss: 0.0834 - accuracy: 0.9722
```

(a)



```
Epoch 48/50
614/614 [==============================] - 2s 4ms/step - loss: 0.0178 - accuracy: 0.9932
Epoch 49/50
614/614 [==============================] - 2s 4ms/step - loss: 0.0145 - accuracy: 0.9943
Epoch 50/50
614/614 [==============================] - 2s 4ms/step - loss: 0.0228 - accuracy: 0.9927
```

(b)

Figure 14: (a) Start and (b) finish of Model 2 training at 50 epochs.

3.4 Performance evaluation

3.4.1 Accuracy-Loss Graph for Model Training

The loss function and accuracy for the train data are then displayed in graph form as shown in Figure 15. This graph is called learning curves and can be used to evaluate performance for model training. The progression of the model in terms of its loss and accuracy usually is done because it is important to check whether the two values are acceptable or not. This is based on the expectation that the loss decreases and accuracy increases as the training process is done towards the end of epochs. For all models, the loss and accuracy progression meet those expectations, proven by Figure 3.6. However, the curves show some differences in their form. Based on the LSTM graph, the curves seem to show some linear characteristics after 10 epochs, unlike both ANN models. Their accuracy curves seem to reach stability after 20 and 10 epochs for Models 1 and 2 respectively. The loss curves also show the same behavior. This behavior shows the optimal fit for learning curves which means the model training is in a great performance.

This study only uses train and test data as it focuses on the specification for a model which is based on past research that suggests dividing both of the data in train and test data ratio only [16]-[17]. However, for better performance analysis, the learning curves for validation data can be added for future works. This is because the gap between train and validation data curves can show whether the model is overfitting, underfitting, or in optimal condition. Overfitting happens when the network model can work well with train data but not with test data because it learns too many details and noise from the train data. While underfitting happens when the network model cannot learn the patterns in train data properly thus affecting the performance with test data.

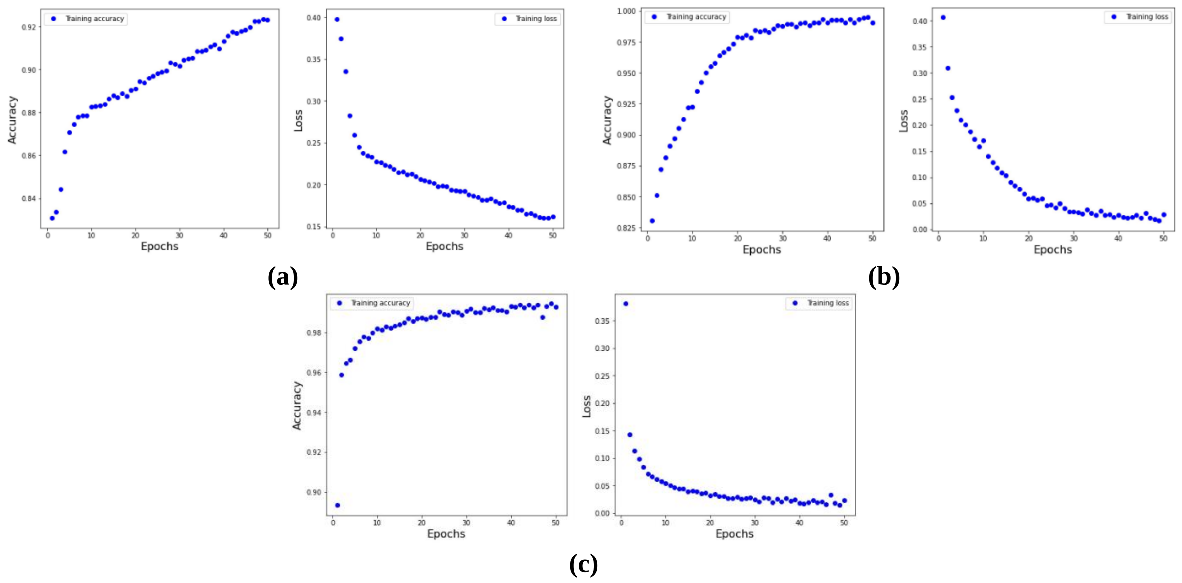


Figure 15: Training accuracy and loss graph against epochs for (a) LSTM, (b) ANN Model 1 and (c) Model 2.

3.4.2 Accuracy score

Upon the completion of the training process with satisfying accuracy and loss values, the models then undergo testing to prove the predictions on each trained model with an accuracy value. Table 1 shows the accuracy obtained for each network model. LSTM has the highest accuracy score. This proves the fact that the use of the LSTM model is better than ANN models in identifying epilepsy for this study.

Table 1: Accuracy for each model

Model	Accuracy (%)
LSTM	98.87
ANN Model 1	97.78
ANN Model 2	97.57

Because of the different aspects in the model build, like the number of layers, activation function and hyperparameters for the model training, the model prediction's performance can be different. These aspects become variable in a neural network construction as its performance can differ depending on the determination of the aspects. A lot of research has been conducted by focusing on these kinds of aspects like the variant of LSTMs [11],[18], dropout layer [19], activation function [11] and other various parameters like hidden neurons and layers, connection and weight [20].

As an example, extra observation on the accuracy of this study network model is done to test the effect of aspects change. Some aspect changes are made on the fit function hyperparameters (epochs, batch size and shuffle) which are used for model training. The original hyperparameters are 50 epochs, 15 batch sizes and enable shuffle. The accuracy obtained after the changes is shown in Table 2.

For 50/15 of epoch/batch size pairs with shuffle set as true, network models ANN show higher accuracy than shuffle set as false. The same goes with 40/64 pairs, LSTM and ANN-Model 1 show higher accuracy with shuffle set as true.

The table shows that the change of aspects affects model performance. If the hyperparameters are changed, the accuracy value also can be changed. Even so, the hyperparameters solely cannot decide the performance of every model network. Different models with the same hyperparameter setup usually have different performances. Hyperparameter set for a simple network has different effects on a complex network so usually, it is optimized by considering previous experience and research also with try-and-error [20]. Most importantly, this comparison is made only for the purpose to prove the fact that accuracy obtained for the prediction can also rely on other aspects instead of model types like LSTM and ANN.

Table 2: Accuracy comparison between original and changed hyperparameters

Network model	Epoch/Batch size			
	50/15		40/64	
LSTM	98.87 %	98.87 %	98.89 %	98.91 %
ANN-Model 1	97.78 %	97.74 %	98.22 %	97.87 %
ANN-Model 2	97.57 %	97.48 %	97.35 %	97.39 %

Shuffle	
True	🟩
False	🟧

Aside from this study, there is a lot of research regarding the identification of epilepsy by using different neural networks. The comparison of epilepsy identification or classification accuracy with past research is shown in Figure 16. The highest accuracy obtained from Table 1 is chosen to be compared in the graph. The accuracy obtained for this study might not be the highest but the performance that has been shown deserves to be considered a suitable method for epilepsy identification.

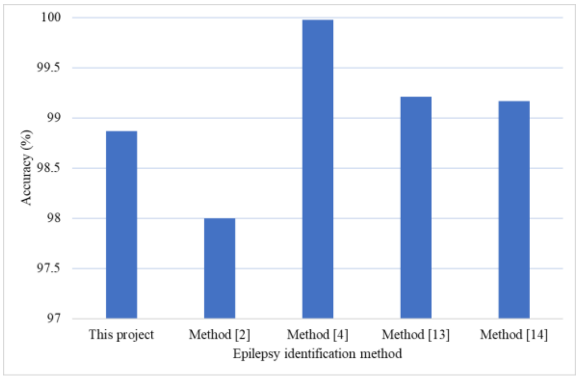


Figure 16: Comparison of accuracy with the previous study

4. Conclusion

In this study, the LSTM model built for the identification of epilepsy shows a great accuracy value (98.87%). With the correct and suitable layers, the LSTM model can be functioned and be trained efficiently. The performance evaluation for the model is helpful as the model can be justified for prediction done by it with the help of learning curves of the trained model and accuracy value. Plus, by using the pre-processed data, the process becomes more smooth. Comparison done with ANN models is also helpful as the difference in model build and aspects which might affect the performance can be a reference for future work.

In addition, some measures can be done to improve the whole system. To build a more significant RNN model, the dataset should be split into train, validation and test data. Providing a suitable ratio of data for validation is important and test data is better not used in model training as validation data because the model can become overfit. The evaluation performance for this study by doing the comparison with other DNN models is a great action. So, for future work other kinds of model algorithms like CNN, DBN and others can be used. In addition, the processing timing is expected to be reduced for the respective number of system configurations with the use of a real-time dataset. Then from the algorithm, an evaluation can be done. The future study also recommended being upgraded so the system can distinguish the severity of epileptic seizures based on the EEG signal.

Acknowledgement

The author would like to express deep gratitude to UTHM and also thank the Faculty of Electrical and Electronic Engineering, Universiti Tun Hussein Onn Malaysia for the facilities.

References

[1] E. Beghi, "The Epidemiology of Epilepsy", *Neuroepidemiology*, vol. 54, no. 2, pp. 185-191, 2019. Available: 10.1159/000503831.

[2] A. Shankar, H. K. Khaing, S. Dandapat and S. Barma, "Epileptic Seizure Classification Based on Gramian Angular Field Transformation and Deep Learning," 2020 IEEE Applied Signal Processing Conference (ASPCON), pp. 147-151, 2020.

[3] M. Rashed-Al-Mahfuz, M. Moni, S. Uddin, S. Alyami, M. Summers and V. Eapen, "A Deep Convolutional Neural Network Method to Detect Seizures and Characteristic Frequencies Using Epileptic Electroencephalogram (EEG) Data", *IEEE Journal of Translational Engineering in Health and Medicine*, vol. 9, pp. 1-12, 2021. Available: 10.1109/jtehm.2021.3050925.

[4] M. U. Abbasi, A. Rashad, A. Basalamah and M. Tariq, "Detection of Epilepsy Seizures in Neo-Natal EEG Using LSTM Architecture," in IEEE Access, vol. 7, pp. 179074-179085, 2019.

[5] Y. Liu et al., "Deep C-LSTM Neural Network for Epileptic Seizure and Tumor Detection Using High-Dimension EEG Signals", *IEEE Access*, vol. 8, pp. 37495-37504, 2020. Available: 10.1109/access.2020.2976156

[6] S. Ryu and I. Joe, "A Hybrid DenseNet-LSTM Model for Epileptic Seizure Prediction", *Applied Sciences*, vol. 11, no. 16, p. 7661, 2021. Available: 10.3390/app11167661.

[7] S. Vinayak E, S. A and N. A, "Epilepsy Prediction using a Combined LSTM – XGBoost System on EEG Signals", *International Journal of Innovative Technology and Exploring Engineering*, vol. 10, no. 1, pp. 18-24, 2020. Available: 10.35940/ijitee.a8086.1110120.

[8] R. G. Andrzejak, K. Lehnertz, F. Mormann, C. Rieke, P. David and C. E. Elger, "Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state", *Physical Review E*, vol. 64, no. 6, 2001. Available: 10.1103/physreve.64.061907.

[9] Y. Zhang, C. Pan, J. Sun and C. Tang, "Multiple sclerosis identification by convolutional neural network with dropout and parametric ReLU", *Journal of Computational Science*, vol. 28, pp. 1-10, 2018. Available: 10.1016/j.jocs.2018.07.003

[10] S. Poornima and M. Pushpalatha, "Prediction of Rainfall Using Intensified LSTM Based Recurrent Neural Network with Weighted Linear Units", *Atmosphere*, vol. 10, no. 11, p. 668, 2019. Available: 10.3390/atmos10110668.

[11] D. Kent, and F. M. Salem, "Performance of Three Slim Variants of The Long Short-Term Memory (LSTM) Layer", *Neural and Evolutionary Computing*. 2019. Available: https://doi.org/10.48550/arXiv.1901.00525

[12] K. Adem, S. Kiliçarslan and O. Cömert, "Classification and diagnosis of cervical cancer with stacked autoencoder and softmax classification", *Expert Systems with Applications*, vol. 115, pp. 557-564, 2019. Available: 10.1016/j.eswa.2018.08.050.

[13] J. Feng and S. Lu, "Performance Analysis of Various Activation Functions in Artificial Neural Networks", *Journal of Physics: Conference Series*, vol. 1237, no. 2, p. 022030, 2019. Available: 10.1088/1742-6596/1237/2/022030.

[14] A. D. Rasamoelina, F. Adjailia and P. Sincak, "A Review of Activation Function for Artificial Neural Network", *2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, 2020. Available: 10.1109/sami48414.2020.9108717.

[15] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation Functions: Comparison of trends in Practice and Research for Deep Learning", *Computer Vision and Pattern Recognition: Machine Learning*, 2018. Available: https://doi.org/10.48550/arXiv.1811.03378.

[16] D. Sikdar, R. Roy and M. Mahadevappa, "Epilepsy and seizure characterisation by multifractal analysis of EEG subbands", *Biomedical Signal Processing and Control*, vol. 41, pp. 264-270, 2018. Available: 10.1016/j.bspc.2017.12.006.

[17] W. Kurdthongmee, "Optimisation of deep neural networks for identification of epileptic abnormalities from electroencephalogram signals", Heliyon, vol. 6, no. 12, p. e05694, 2020. Available: 10.1016/j.heliyon.2020.e05694.

[18] S. Opałka, D. Szajerman and A. Wojciechowski, "LSTM multichannel neural networks in mental task classification", *COMPEL - The international journal for computation and mathematics in electrical and electronic engineering*, vol. 38, no. 4, pp. 1204-1213, 2019. Available: 10.1108/compel-10-2018-0429.

[19] G. Cheng, V. Peddinti, D. Povey, V. Manohar, S. Khudanpur and Y. Yan, "An Exploration of Dropout with LSTMs", *Interspeech 2017*, 2017. Available: 10.21437/interspeech.2017-129.

[20] T. K. Gupta and K. Raza, "Optimization of ANN Architecture: A Review on Nature-Inspired Techniques", *Machine Learning in Bio-Signal Analysis and Diagnostic Imaging*, pp. 159-182, 2019. Available: 10.1016/b978-0-12-816086-2.00007-2.