Recurrent Neural Network Optimization for EEG-Based Classification of Brain Activity

Mehek Niwas

Deep Learning, MTH 4320, Fall 2023
Project 3

As an introduction of a more objective analysis for EEG interpretation, recurrent neural networks can help provide a second opinion for clinicians and specialists looking for abnormalities in brain activity. In this project, a recurrent neural network is progressively optimized to classify the Epileptic Seizure Recognition Dataset.

1 Introduction

Brain conditions such as epilepsy or tumor-related abnormalities are often misdiagnosed by clinicians. Along with a population study [11] reporting a 26% misdiagnosis rate for epilepsy alone, other studies [4] have found a low inter-rater reliability, or rate of agreement, amongst even specialists like epileptologists when interpreting electroencephalograms (EEGs). EEGs have become one of the main diagnostic tests to a variety of brain-related disorders because of their ability to directly measure and record electrical activity. However, EEG use is often abused by clinicians through over-interpretation of results. Introducing a more objective analysis may be an approach to mitigate this observation error, such as the classifications of EEG recordings made by a trained recurrent neural network.

Unlike traditional neural networks, RNNs have the ability to store relevant "memory", or information from previous time steps in the data to make predictions about a given time sequence such as an EEG recordings. Some of the most popular methods to determine the relevant memory, often called the "hidden state", are Long Short Term Memory (LSTM) cells and Gated Recurrent Unit (GRU) cells. These two variants incorporate computational gates with functions like sigmoid or hyperbolic tangent to selectively retain or forget information over time. While computationally intense, this process allows the network to effectively learn and utilize long-range dependencies in sequential data. The networks behave differently based on use of different techniques, architectures, and parameters.

With so many components to manipulate, optimization for any specific dataset is necessary for a recurrent neural network to be efficiently accurate. Often requiring a substantial amount of computation to train, runtime must also be taken into account for feasible and practical performance.

In this project, a recurrent neural network was optimized to classify the type of brain activity as 1 of 5 classes from a publicly available dataset consisting of 11,500 instances of 1 second EEG recordings. A secondary goal of the project was to ensure practicality in terms of computational expense. Different types of cells (GRU or LSTM), dropout rates, batch sizes, number of layers, hidden sizes, and types of weight initialization were investigated. Some components, like the decision to implement GRU and a batch size of 15, were determined through shortest runtime with highest yield in terms of accuracy. However, for selection in other components such as number of layers and hidden size, the distribution of accuracies between the 5 classes had to be analyzed. The hidden size was chosen to be 32, but sizes both above and below were found to have a lower performance in terms of class accuracy distribution. This phenomenon was also observed when selecting the number of layers, reinforcing the idea that some components must be optimized to a certain extent for a specific dataset.

By the end of the project, a final accuracy of 71.65% was achieved on a test set of unseen data by the RNN.

2 Related Works

Neural networks have been used in the past 15 years to create highly accurate object detection models, revolutionize recommender algorithms, and make leaps in computer vision. The computational process begins by assigning a weight parameter for each input received, which is optimized with a loss function in the training stage to calculate a weight sum equal to the output. This traditional outline of a basic neural network is modified by a recurrent neural network (RNN) to better predict outputs with sequential data, such as the temporal data that EEG recordings provide. Recurrent neural networks have many components that must be optimized for datasets to achieve best performance.

For classification of EEGs, literature particularly varies on the choice of GRU or LSTM cells. Johari et. al [7] implemented GRU to classify 1 of 3 emotional states from EEG data, stating "GRU provides a promising direction for advancing emotion analysis and understanding affective responses encoded in brainwave signals". On the other hand, Ahmed &

Sinha [1] chose an LSTM model to classify a similar dataset, explaining that "LSTM models have shown their ability to learn from sequential data". Performance of both models has been suggested to be similar, with a faster computation time for GRU due to the use of a hidden state rather than separate cell state calculations. One comparison study [2] in particular achieved an accuracy of 97% with LSTM and 96% with GRU for EEG emotion detection. A similar trend in LSTM and GRU models was found for an EEG regression problem by Jusseaume & Valova [8]. The similarity in accuracy indicates that selection of GRU or LSTM may depend on the nature of the dataset and the nuances in the researcher's goals for predictions or classifications.

A major component of RNNs that seems to be more specific to the dataset are the hidden sizes of the GRU, also called memory cell size when referring to LSTM cells. These sizes are the number of time steps to be stored as "memory" in the network, so it must be set within the total number of time steps from the input data itself. Ahmed & Sinha [1] compared the effect of the number of memory cells in an LSTM, finding that accuracy was 85%, 89%, and 90% for 32, 64, and 128 cells respectively. The slight increase or decrease in accuracy exemplifies the particular importance of tuning the hidden size with respect to a particular model or dataset.

The number of layers and complexity in an RNN also have an interesting response to accuracy. Greaves [5] explored this concept with EEG data to predict the 3-D perception of a patient. It was found that after comparing both simple and complex models, the simpler models with 1 or 2 layers were able to achieve a significantly higher overall accuracy than more complex versions. Greaves [5] concludes that "it is not straightforward to apply RNNs to EEG data".

Despite the differences in selections of RNNs, some common techniques across studies [3] are the use of ReLU and Softmax activation functions, which are known to be effective for neural networks in simplifying calculations to binary-like values, much like the action potential computations in biological neurons. Another commonality found in a review study [10] was the use of the Adaptive Moment Estimation (Adam) optimizer to minimize cross-entropy loss. Used in a variety of EEG classification models, the Adam optimizer is popular because it utilizes momentum to push through flat regions or shallow valleys in the gradients of the loss function. This characteristic of Adam allow models to have faster and smoother training.

Experiments conducted in this project will explore nuances between architecture selections and techniques such as the RNN type, hidden size, and number of layers, in addition to batch size and dropout rate that have not been specifically discussed in related works for EEG classification. Each experiment will continue to utilize certain common characteristics such as Adam optimizers and activation functions as well.

3 Methods

3.1 Dataset & Preparation

The Epileptic Seizure Dataset was originally sourced from the UCI Machine Learning Repository, by authors Wu and Fokou. It has now been removed, but still available on Kaggle. It was donated in 2017, and has been used in papers focusing on variations of explainable AI in RNNs.

The dataset was previously pre-processed and consists of 23 second EEG recordings from 500 patients. Each second of each recording was then vectorized into 178 data points of EEG values to maximize the full dataset for a total of 11,500 feature-vectors (23 x 500 = 11,500). This means that each feature-vector contains 178 EEG values to represent 1 second of a full 23 second recording. Thus, the RNN will require 178 time steps. Each feature-vector was pre-labeled into 1 of 5 classes of brain activity/abnormality: seizure, tumor, healthy, eyes closed, and eyes open.

Since each feature-vector only contains EEG values from 1 second and not all 23 seconds collected from a patient, there can be multiple feature-vectors that stem from a singular patient. This is why proper randomized shuffling of the data was especially necessary for the neural network, which was done with the use of a PyTorch dataloader object. The train_test_split function from sklearn was also used to randomly split the data into 60% (6,900 feature-vectors), 20% (2,300 feature-vectors), 20% (2,300 feature-vectors) splits for training, cross-validation, and testing respectively. A random seed was used to ensure the exact same randomized splits for every portion of experimentation.

PyTorch and CUDA were used in order to be able to run the network as fast as possible. Data preparation included removing extraneous index or ID labeling numbers, conversion into integers, and conversion into data loader objects for batch initialization in the neural network.

3.2 Approach to Experimentation

GPU, CUDA, and PyTorch were utilized to allow for faster creation of batches and faster results during experimentation.

Experimentation was set-up by focusing on a singular hyperparameter or variable at a time and using the previous best

Table 1 Epileptic Seizure Dataset Inputs and Outputs

	Rows	Cols
X = Input [EEG Recording] (feature-vectors can stem from same patient)	11,500	178
Y = Output [Brain Activity Recording] (classes: seizure, tumor, healthy, eyes closed/open)	11,500	1

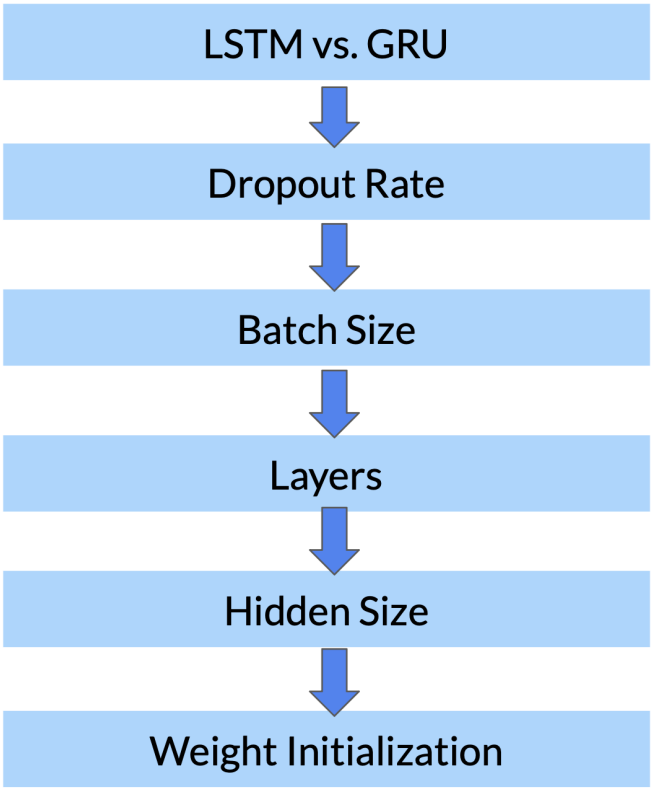

Fig. 1 Planned Model Architecture Progression through Experiments

model as a constant in the subsequent experiments, for a total of 6 experiments.

First, a benchmark model was established using one simple RNN layer (not LSTM or GRU cell type), one linear layer to interpret the results from the RNN layer, one batch for all of the data, a hidden size of 32 time steps, and 0 dropout rate.

Experiment 1 compares an LSTM cell versus a GRU cell with variables of the benchmark model as constants. Criteria and methods to determine the "best" model are described in section 3.3. Experiment 2 compares different dropouts of 0.2 and 0.5 while maintaining all other constants from the best model selected in Experiment 1. This methodology of using the best model from the previous experiment for the next was continued throughout all 6 experiments, as shown in Fig. 1.

In this fashion, Experiment 1 compared RNN cell types (LSTM vs. GRU), Experiment 2 focused on different dropout rates (0.2 vs. 0.5), Experiment 3 evaluated differences across batch sizes (10, 15, 25, or 50), Experiment 4 determined the number of layers used (4 vs. 5), Experiment 5 determined the hidden size (28, 32, 40, or 48), and Experiment 6 determined the weight initialization (random vs. Xavier). Highest accuracy was achieved by the best model selected from Experiment 6.

3.3 Evaluation & Selection of Models

As the goal was to achieve a high accuracy in a computationally inexpensive method, both runtime and overall accuracy of the model were recorded.

A decrease in overfitting and increase in accuracy was prioritized. Overfitting progression over epochs was calculated by subtracting the training loss/accuracy from the validation loss/accuracy, since the model usually performs worse on the validation data split. The validation data was data that the model did not have access to during training, so it is able to provide an estimate of how the model would perform, or generalize for predictions on unseen data. In some experiments, the accuracy and overfitting between different models was also very similar, so training runtime became prioritized and the best model was chosen based on lowest runtime.

Balance between the 5 individual class accuracies was prioritized in experiments as an additional way to differentiate performance among models with the previously discussed metrics of similar accuracy, overfitting, and runtime. This was found by graphing the class accuracy distribution and calculating standard deviation among the individual classes. This metric also allowed for a better analysis on the behavior of the network and effect of the parameter being tested in an experiment. The class accuracies provided insight into the specific classes prone to lower performance and overfitting by different models and common trends as well.

Once the last experiment was completed, predictions on the testing data split were made by the final best model as a final metric to assess the generalization of unseen data by the model.

4 Results

Results of the project indicated that different variations of parameters in models affect the overall accuracy and runtime, as well as the accuracies of the individual classes and estimated overfitting by the model. Each experiment suggested different trends and peformances based on the component or parameter of the network being optimized.

In experiment 1, the best model was selected based on the accuracy and runtime, since the accuracy was very similar and runtime for the GRU model was much shorter. In experiment 2, the model with a dropout rate of 0.5 was selected as the best performing model because it had a much higher accuracy of 39% than the 0.2 dropout rate model that only achieved an accuracy of 32%, along with a higher rate of overfitting. In experiments 3-6, the accuracy, runtime, and overfitting had to be graphed in order to select the best model. In experiments 4-6, the individual class accuracy distribution was also graphed to select the best model based on the best balance between classes since other metrics showed very similar results.

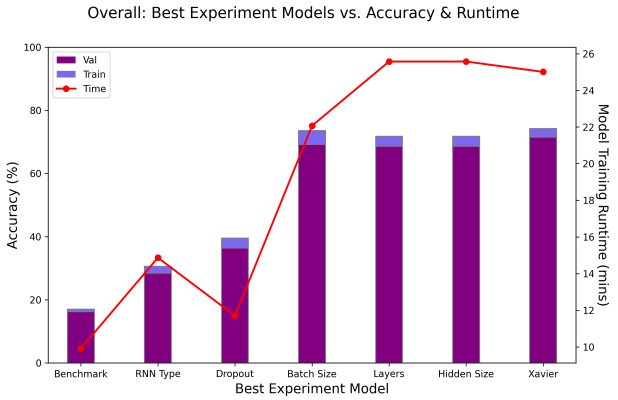

Fig. 2 Best Experiment Models vs. Accuracy & Runtime. Includes Benchmark Model results

The overall plot of best models from each experiment versus accuracy and runtime, Fig. 2 shows how runtime increases as accuracy increases, and varies from model to model. The color difference in the plot also indicates the estimated overfitting of each model. It highlights difference between the accuracy achieved on the validation and training data splits. The amount of overfitting by the models seems to have slight changes throughout the experiments.

Table 2 Exp. 3 - Batch Size vs. Accuracy & Runtime

Model	Train Acc.	Val Acc.	Runtime
batch size = 10	74%	70%	39 s
batch size = 15	73%	69%	22 s
batch size = 25	70%	67%	15 s
batch size = 50	70%	67%	9 s

The results of experiment 3 from Table 2 indicate runtime decreases with increase in batch size, while overfitting seems to be similar. The highest overall accuracies were achieved by models with batch size 10 and batch size 15, however the accuracies of both models were nearly identical. Therefore the best model was selected between these two models based on the runtime, and the model with batch size 15 had a much shorter runtime than the model with batch size 10 by nearly 20 minutes.

The table of results in experiment 4 (Table 3) indicate very similar accuracies and overfitting. Although the plot seems to show a large difference in model training runtime due to scaling, the actual difference between all models is only 8 minutes.

Table 3 Exp. 4 - Number of Layers vs. Accuracy & Runtime

Model	Train Acc.	Val Acc.	Runtime
layers = 2	73%	69%	22 s
layers = 4	71%	68%	25 s
layers = 5	71%	68%	29 s

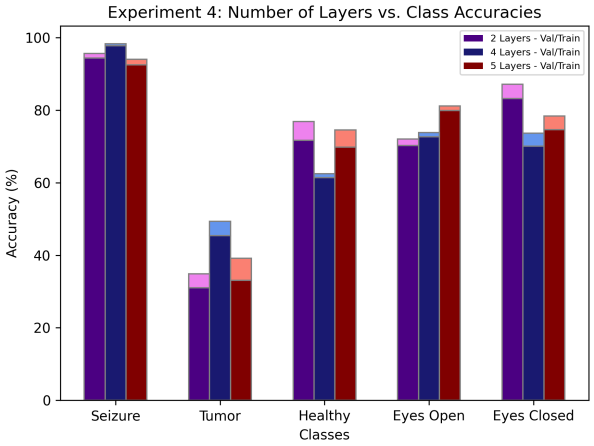

Fig. 3 Experiment 4 — Number of Layers vs. Class Accuracies

To select the best model of the experiment, accuracies and overfitting of each individual class was graphed (Fig. 3). In the graph, the Tumor class seems to have the worst performance across all models, but the model with 4 layers shows the highest performance for the Tumor class. The model with 4 layers also has the lowest standard deviation between classes, indicating a better balance among them. As a result, the model with 4 layers was selected as the best performing model and used in the subsequent experiment.

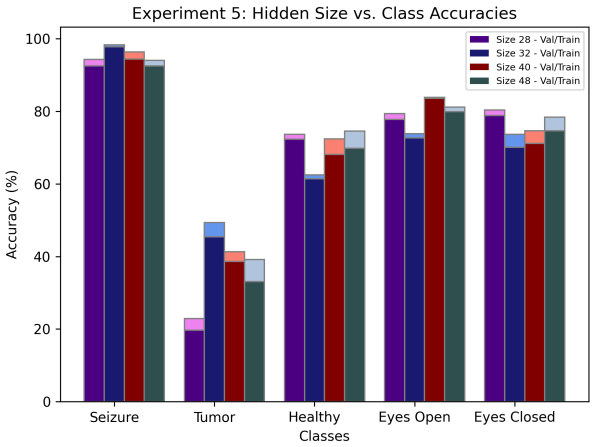

Fig. 4 Experiment 5 — Hidden Size vs. Class Accuracies

Table 4 Exp. 5 - Hidden Size vs. Accuracy & Runtime

Model	Train Acc.	Val Acc.	Runtime
hid. size = 28, epcs. = 20	70%	67%	27 s
hid. size = 32, epcs. = 20	71%	68%	25 s
hid. size = 40, epcs. = 15	73%	70%	37 s
hid. size = 48, epcs. = 15	73%	68%	43 s

Data from experiment 5 (Table 4) indicates very similar accuracies, overfitting, and runtime among hidden sizes 28 and 32. Hence, the class accuracies of all models were graphed to determine the best performing model (Fig. 4). As in the previous experiment, the Tumor class had the worst performance of all classes for all models. The model with a hidden 32 was able to achieve the highest accuracy in the Tumor class among all other models and had a lower standard deviation between class accuracies achieved, supporting the indication that it had a better balance in performance than the other models. The model with a hidden size of 32 was selected as the best performing model in Experiment 5.

Table 5 Exp. 6 - Weight Initialization vs. Accuracy & Runtime

Model	Train Acc.	Val Acc.	Runtime
init. type = random	71%	68%	25 s
init. type = Xavier	74%	71%	25 s

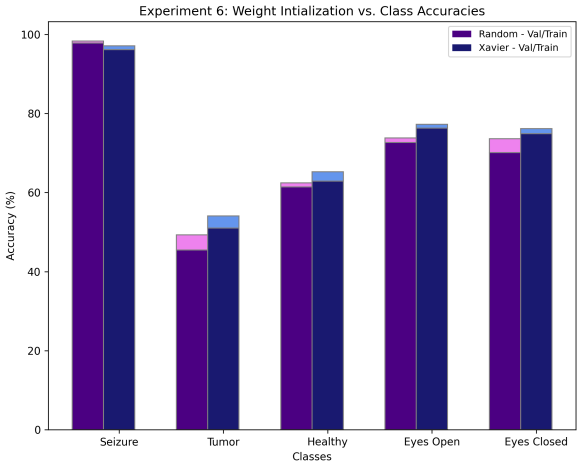

Fig. 5 Experiment 6 — Hidden Size vs. Class Accuracies

Two types of weight initialization (random and Xavier) were compared in Experiment 6. This experiment was purposefully completed as the last experiment to suggest whether a change in weight initialization has a marginal effect on the results of the model even after all of the other optimizations and adjustments already made. Analyzing the accuracy and runtime, the data (Table 5) indicates a slight increase in accuracy after using Xavier initialization. However, after graphing the individual class accuracies and overfitting for both models (Fig. 5), a substantial difference is shown in the accuracies for the Tumor class. As seen in previous experiments, models tend to perform the worst in the Tumor class, and the model with Xaiver weight initialization showed an increase in accuracy compared to random weight initialization. This suggests that a change in weight initialization is able to improve model performance, despite many parts of the model having already been optimized from previous experiments. After Experiment 6, the final architecture of the best performing model overall was concluded and the model was evaluated on the test data set to provide a final estimate on the generalization of the model.

5 Discussion & Conclusion

The finalized architecture was achieved by evaluating a different component or parameter of the model in each experiment (Fig. 6). The best performing model was determined in each experiment and the tested parameter was kept constant in the subsequent experiments. A simple RNN benchmark model was run in the beginning of the project, achieving a training accuracy of 17.03% with a training runtime of around 9 minutes. The best model from the end of the project achieved the highest training accuracy of 74.22% with a training runtime of 25 minutes. The finalized model architecture consisted of: GRU cell type, 0.5 dropout rate, batch size 15, 4 layers, hidden size 32, and Xavier weight initialization. This model was able to achieve a final accuracy of 71.25% on the testing data split, which indicates the estimated accuracy that should be expected if data of the same format is used to classify brain activity into 1 of the 5 classes. The individual class accuracies achieved on the testing data split was 94.90% for Seizure, 54.07% for Tumor, 60.09% for Healthy, 77.68% for Eyes Closed, and 71.58% for Eyes Open.

Overall, the project was able to demonstrate how model performance changes through accuracy, overfitting, runtime, and individual class accuracy from optimization of various components and parameters of a Recurrent Neural Network. It is important to note that this performance was achieved using a specific dataset and similar results should not necessarily be expected from real-world data due to the nuances that real-world data may contain.

6 Future Studies

An area of future experimentation may be to investigate the ratio of hidden size to the total number of time steps in the input

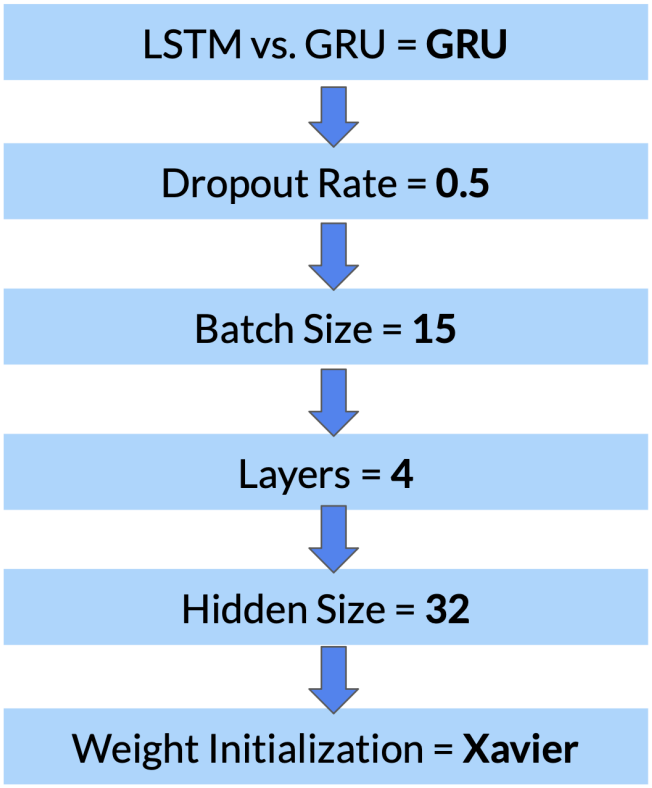

Fig. 6 Finalized Model Architecture Progression through Experiments

data. This would require by comparing results from different EEG datasets with a varying range of time steps in the input data to allow for generalization of this technique in reference to EEG classifications.

More experimentation can also be conducted with different variations of RNNs or combinations of different types of neural networks, such as a bidirectional RNN or convolutional recurrent neural network (ConvRNN). Bidirectional RNNs have been successful with many sources of sequential data and allow the neural network to make predictions on a sequence with the ability to consider future and past "memory". These models can be configured with different directionality options to achieve better results. Convolutional recurrent neural networks were utilized by Hu et. al [6] with emotion-focused EEG recordings to yield impressive results, so the architecture may be a point of research to classify presence of seizures and tumors as well.

Introduction of new techniques can be explored alongside these architectures, such as attention-mechanisms to stress certain data within the EEG time sequence. This technique has been used by Kim & Choi [9] with temporal data from EEG recordings, and was found to help increase accuracy for their particular dataset.

As more complex and different architectures and techniques are used in recurrent neural networks, there are more variables and components that can be manipulated to achieve the best performance possible for classification of EEG recordings. While use of such models in medical practice would not replace clinicians or specialists, the neural network classifications have potential to supplement decision-making to ultimately prevent the misdiagnosis of brain disorders and abnormalities.

Dataset Availability Statement

The dataset used in this project is publicly available on Kaggle. Link: kaggle.com/datasets/harunshimanto/epileptic-seizure-recognition.

References

[1] Ahmed, M., & Sinha, N. (2021). Eeg-based emotion classification using lstm under new paradigm. *Biomedical Physics Engineering Express*, *7*. https://doi.org/10.1088/2057-1976/ac27c4

[2] Chowdary, M., Anitha, J., & Hemanth, D. (2022). Emotion recognition from eeg signals using recurrent neural networks. *Electronics*, *11*(15). https://doi.org/10.3390/electronics11152387

[3] Craik, A., He, Y., & Contreras-Vidal, J. L. (3). Deep learning for electroencephalogram (eeg) classification tasks: A review. *Journal of neural engineering*, *16*. https://doi.org/10.1088/1741-2552/ab0ab5

[4] Grant, A. C., Abdel-Baki, S. G., Weedon, J., Arnedo, V., Chari, G., Koziorynska, E., Lushbough, C., Maus, D., McSween, T., Mortati, K. A., Reznikov, A., & Omurtag, A. (2014). Eeg interpretation reliability and interpreter confidence: A large single-center study. *Epilepsy behavior*, *32*, 102–107. https://doi.org/10.1016/j.yebeh.2014.01.011

[5] Greaves, A. S. (2015). Classification of eeg with recurrent neural networks. https://api.semanticscholar.org/CorpusID:1694197

[6] Hu, Z., Chen, L., Luo, Y., & Zhou, J. (2022). Eeg-based emotion recognition using convolutional recurrent neural network with multi-head self-attention. *Applied Sciences*, *12*. https://doi.org/10.3390/app122111255

[7] Johari, S., Namratha Meedinti, G., Delhibabu, M., & Joshi, D. (2023). Unveiling emotions from eeg: A gru-based approach. *Arxiv: Signal Procressing, Machine Learning*. https://doi.org/10.48550/arXiv.2308.02778

[8] Jusseaume, K., & Valova, I. (2022). Brain age prediction/classification through recurrent deep learning with electroencephalogram recordings of seizure subjects. *Sensors*, *22*. https://doi.org/10.3390/s22218112

[9] Kim, Y., & Choi, A. (2020). Eeg-based emotion classification using long short-term memory network with attention mechanism. *Sensors*, *20*. https://doi.org/10.3390/s20236727

[10] Maitin, A., Romero Muñoz, J., & García-Tejedor, Á. (2022). Survey of machine learning techniques in the analysis of eeg signals for parkinson's disease: A systematic review. *Applied Sciences*, *12*. https://doi.org/10.3390/app12146967

[11] Scheepers, B., Clough, P., & Pickles, C. (1998). The misdiagnosis of epilepsy: Findings of a population study. *Seizure*, *7*, 403–406. https://doi.org/10.1016/S1059-1311(05)80010-X