# Processing EEG Data with CNN, CNN-LSTM and VAE

Aurosweta Mahapatra 206070948 aurosweta99@ucla.edu
Shrey Mathur 205928673 shreyzmail@gmail.com
Syam Sundar Kirubakaran 805905367 syamk@ucla.edu
Vaibhav Tiwari 105912719 vaibhavtiwari@ucla.edu

## Abstract

*This project suggests three methods for creating a neural network model for the EEG Net dataset - using CNN, CNN+LSTM and Variational Autoencoders (VAEs). The study evaluates and compares the performance of both methods in classifying motor imagery. The results show that the CNN+LSTM method outperforms the VAE method in terms of accuracy. However, the VAE method has the advantage of preserving the crucial features of the EEG signals while reducing their dimensionality. Both methods have their respective advantages and limitations and can be used depending on the specific requirements of the application. In addition to the above two methods we have also implemented the Random Forest for this dataset to give a comparative analysis on accuracy achievements for ML and DL models.*
***Index terms:*** *Machine Learning(ML), Deep Learning(DL), VAE(Variational Autoencoder), Long Short Term Memory Networks(LSTM), Electroencephalogram(EEG)*

## 1.Overview

In this project we are exploring methods to classify the EEG Net dataset[1]. Three neural network models are proposed: CNN, CNN+LSTM, and VAE. CNN+LSTM method outperformed the other methods in terms of accuracy. DL models were superior to ML models for BCI applications[2].

### 1.1. Convolutional Neural Network

CNN works well for the EEG Net dataset because it can effectively capture spatial dependencies in the data. CNN can take advantage of spatial arrangement of electrodes to extract spatial features from the data, which can be useful in tasks such as EEG classification[3]. In order to improve accuracy of CNN we reduced the no. of convolution layers from 4 to 3 and added a Fully Connected Layer. We felt that 3 convolution layers (with relatively more parameters) would be sufficient for feature extraction and an additional FC layer will further improve classification. To improve performance, we increased the no. of parameters. We increased the no. of filters of the first 3 layers. We also increased the kernel size to increase the receptive field (allowing the neurons in the layer to get information from relatively more neurons from the previous layers, resulting in better learning). We have also increased the max pool size after the 2nd convolution layer to (5,1) in order to further reduce the no. of parameters, resulting in reduction in computational cost. Our model was giving accuracy of around 68%. Since the train and validation accuracy were both increasing at the end of training, we decided to increase the no. of epochs. On increasing the no. of epochs to 100, we got an accuracy of 71.16%.
Our final model has the following architecture: The first convolution layer of this model(CL1) has 50 filters of size(10,1), CL2 has 100 filters of size(15,1) and CL3 has 150 filters of size(15,1). All the convolution layers are followed by maxpool layers. The fully connected layer(FC1) has 200 neurons. The fully connected output layer with 4 neurons. Dropout is 0.5 for all layers, batch size and 4 no. of epochs is 64 and 100 respectively. The test accuracy is 71.16%

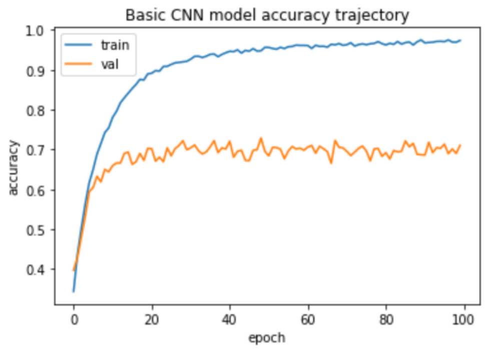

***Figure 1: Accuracy vs Epoch plot for CNN model***

### 1.2. Convolutional Neural Network with LSTM

Convolutional Neural Network (CNN) layers for feature extraction on input data are paired with LSTMs to facilitate sequence prediction in the CNN LSTM architecture. A CNN LSTM can be created by first adding CNN layers, then LSTM layers, and finally a Dense layer at the output. The CNN Model is responsible for feature extraction and the LSTM Model for feature interpretation[4]. Initially, in order to improve our model, we tried increasing only the no. of epochs. Increasing the epochs lead to some improvement (66-67%). We then changed the model architecture in order to increase the complexity of the model, to better understand the distribution. We added 1 more convolution layer, fully connected layer and increased the no. of neurons in the LSTM layer. In addition, we increased the no. of filters in each layer. We also increased the kernel size of neurons in the LSTM layer. In addition, we increased the no. of filters in each layer. We also increased the kernel size to increase the receptive field (allowing the neurons in the layer to get information from relatively more neurons from the previous layers, resulting in better learning). We had a model with 1.3 million parameters. This model was overfitting the dataset ( as train accuracy was increasing but validation accuracy remained low). In order to prevent overfitting, we reduced the no. of parameters (in the FC and CNN layers), increased the batch size(to reduce no. of updates) and increased the dropout (for regularization). We tried various combinations until the data was no longer overfitting. Our model was giving accuracy of around 69%. Since the train and validation accuracy were both increasing at the end of training, we decided to increase the no. of epochs. On increasing the no. of epochs to 400, we got an accuracy of 71.6%.

Our final model has the following architecture: The first convolution layer of this model (CL1) has 50 filters of size(25,1), CL2 has 50 filters of size (20,1), CL3 has 100 filters of size(15,1), CL4 has 100 filters of size (10,1) and CL5 has 100 filters of size (10,1). All the convolution layers are followed by a maxpool layer of size (3,1). The first fully connected layer (FC1) has 200 neurons and FC2 has 100 neurons. The LSTM has 20 neurons followed by a fully connected output layer with 4 neurons. The dropout for all the layers is 0.6. The batch size and no. of epochs is 128 and 400 respectively.

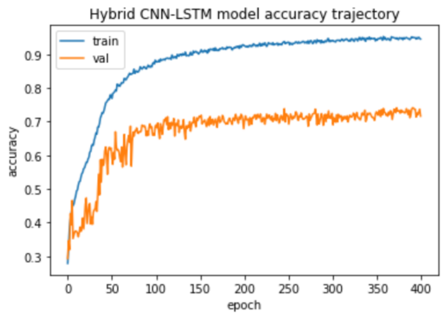

***Figure 2: Accuracy vs Epoch plot for CNN-LSTM mode***

## 1.3. Variational Autoencoder(VAE)

VAE (Variational Autoencoder) is a generative model that can learn the underlying distribution of the input data and generate new samples from that distribution. It has 3 important components: an Encoder, a latent space and a Decoder. The encoder maps the high-dimensional EEG signals into a lower-dimensional latent space, where the features that capture the essential characteristics of the EEG signals are represented. The compressed representation can then be used as input to a decoder which then generates new signals. Unlike loss functions used in classification models like CNN, VAEs have 2 loss functions. The reconstruction loss (makes sure the output signal doesn't deviate too much from the input signal) in this case it is the mean squared error and the Latent loss (makes sure the vector representation takes only a fixed range of values) in this case it is the KL divergence.

$$\frac{1}{2}\left[-\sum_i \left(\log \sigma_i^2 + 1\right) + \sum_i \sigma_i^2 + \sum_i \mu_i^2\right]$$

***Formula 1: Latent loss - KL Divergence***

We built the VAE with no preprocessing of the received EEG data and generated a number of artificial signals but we ran into the mode collapse problem as all the channels collapsed onto a same single signal, we suspected that preprocessing by adding white noise can avoid falling into this trap but we still ran into collapsing channels as we reached overfitting quite early. On further investigation this could be fixed using an unrolled GAN or using GAN with Wasserstein loss.[6]

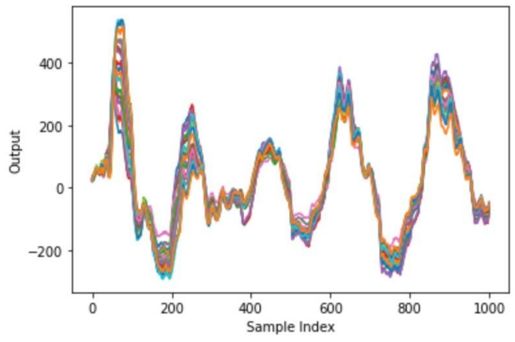

***Figure 3: Artificial Signal without preprocessing***

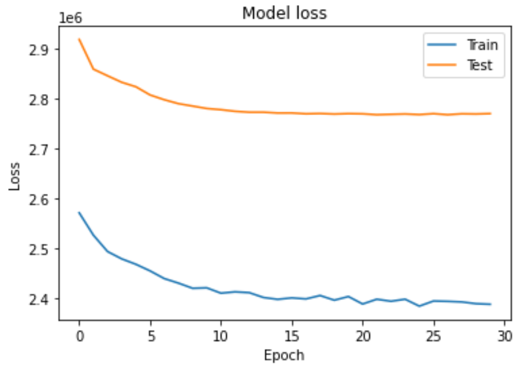

***Figure 4:*** **VAE** ***Loss without preprocessing***

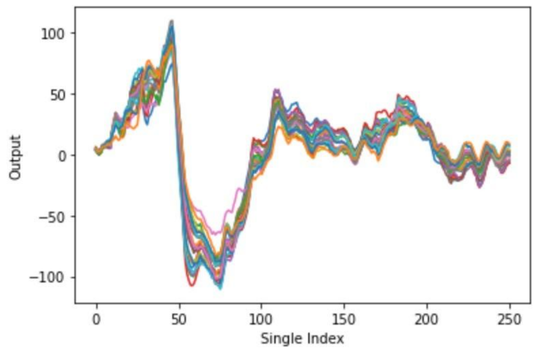

***Figure 5: Artificial Signal with preprocessing***

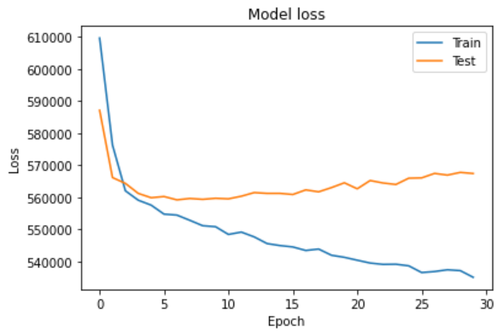

***Figure 6:*** **VAE** ***Loss with preprocessing***

Our final model has the following architecture for encoder: The first convolution layer of this model(CL1) has 40 filters of size(1, 50) and CL2 has 80 filters of size(22, 1) followed by a dense layer then a fully connected layer(FC1) has 16 neutrons which yields the latent. Also, each convolutional layer is followed by a leaky ReLU activation function. The decoder takes the latent as the input and has dense layer and a reshape layer followed by transpose 2D convolutional layer with 80 filters of size (22, 1) followed by transpose 2D convolutional layer with 40 filters of size (1, 50) followed by transpose 2D convolutional layer with output dimensions similar to the input.

### 1.4. Random Forest

The Random Forest Algorithm is a supervised machine learning technique used for classification and regression problems. It comprises numerous decision trees on various subsets of the dataset to improve predictive accuracy. The steps involved in Random Forest algorithm in this case are 1)we selected a subset of data points and a subset of features for constructing each decision tree 2) Then, Individual decision trees were constructed for each sample 3) The final output considered was based on the majority voting of the output produced by each decision tree. The test accuracy we got for Random Forest algorithm is 41.37%

## 2. Results and Observations

Processing EEG data using deep learning models has shown promising results in accurately classifying EEG signals. Among the three models tested, CNN-LSTM performed the best with an accuracy of 71.6%, surpassing the accuracy achieved by the standalone CNN model. Additionally, the use of the random forest algorithm resulted in lower accuracy, indicating that deep learning models are more effective for analyzing EEG data.

### 2.1. CNN vs CNN-LSTM

As discussed before, EEG data is a type of time-series data that requires capturing both spatial and temporal dependencies. EEG data unlike regular images are not pixels but time-series samples. CNN-LSTM is well-suited for EEG data because it can handle both spatial and temporal dependencies in the input data. The CNN component can extract spatial features from each time-series sample, while the LSTM component can capture temporal dependencies across multiple time-series samples.

### 2.2. ML vs DL

EEG data requires capturing both spatial and temporal dependencies. DL models, such as CNN-LSTM, are better suited for this task, as they can capture both spatial and temporal features, while ML models lack the ability to explicitly capture long-term temporal dependencies. This is why DL models generally outperform ML models for EEG datasets.

## 3. Conclusion

In conclusion, EEG data was processed using three different neural network architectures, CNN, CNN-LSTM, and VAE. The accuracy achieved for CNN was 71.16%, while the accuracy achieved for CNN-LSTM was 71.6%, indicating that the addition of LSTM improved performance. Additionally, the study included the use of the random forest algorithm, which achieved an accuracy of only 41.37%, highlighting the superiority of deep learning models over machine learning models for processing EEG data with the EEG Net [5] architecture.

| **Model** | **Accuracy** |
|---|---|
| CNN | 71.16% |
| CNN-LSTM | 71.3% |
| Random Forest | 41.37% |

***Table 1: The table above represents the performance comparison of models used in the project. CNN-LSTM performs the best out of all the models.***

Overall, the study demonstrated the effectiveness of deep learning models, especially CNN-LSTM, for processing EEG data.

## 4. References

[1] Data Description – BCI Competition 2008 – Graz data set A, bbci.de/competition/iv/desc_2a.pdf

[2] Data Set – BCI Competition IV, bbci.de/competition/iv/

[3] Deep learning with convolutional neural networks for brain mapping and decoding of movement-related information from the human EEG, arxiv.org/pdf/1703.05051.pdf

[4] A Novel CNN-LSTM Hybrid Model for Prediction of Electro-Mechanical Impedance Signal Based Bond Strength Monitoring, mdpi.com/1424-8220/22/24/9920

[5] EEGNet: a compact convolutional neural network for EEG-based brain-computer interfaces, pubmed.ncbi.nlm.nih.gov/29932424/

[6] Unrolled GANs, arxiv.org/pdf/1611.02163.pdf

## 5. Architecture Diagrams:

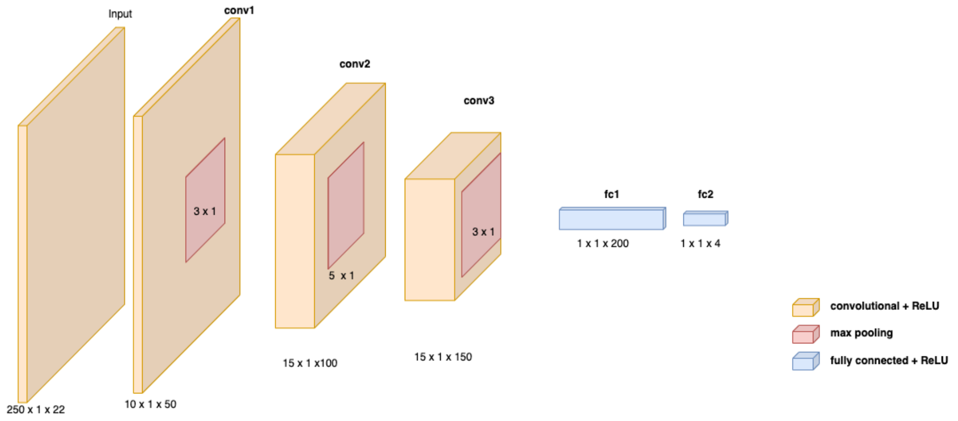

*Figure 7:* Architecture diagram of CNN

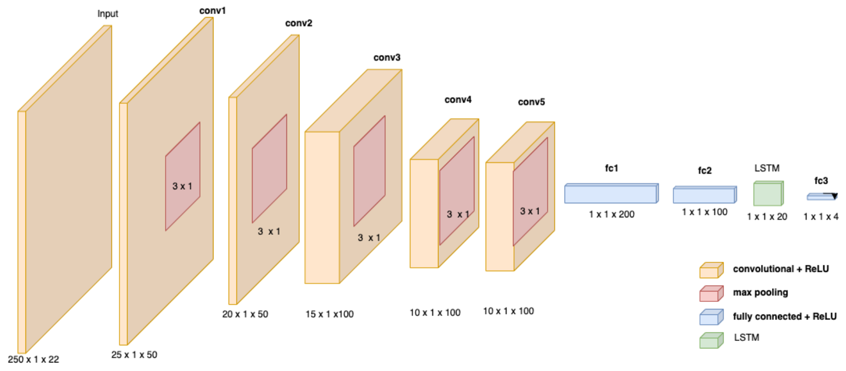

*Figure 8:* Architecture diagram of CNN-LSTM

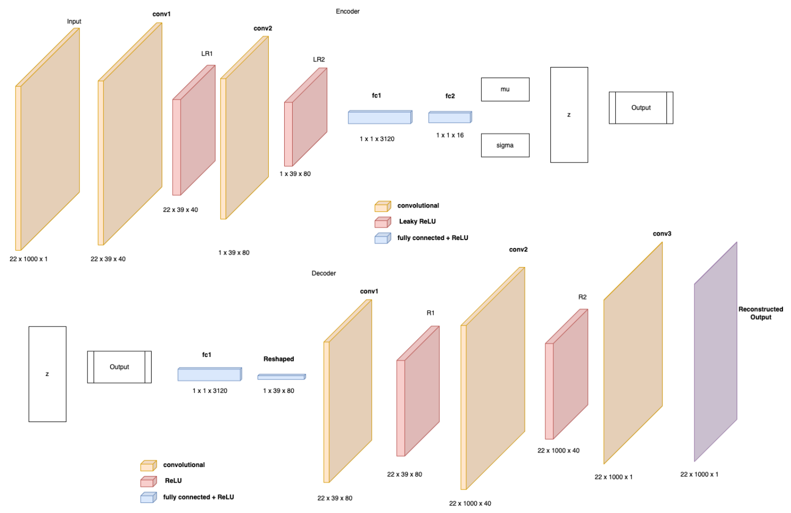

*Figure 9:* Architecture diagram of VAE