# Envelope reconstruction task using the EEG data.

Our task is to reconstruct the envelope of the speech using the EEG signals of the subject. The **SparrKULee EEG dataset** has been used for this task. The dataset contains 85 subjects EEG data while listening to the audio. The dataset is available at the[**kuleuven**](https://rdr.kuleuven.be/dataset.xhtml?persistentId=doi:10.48804/K3VSND) website. There are a total of 668 trails in total each of 15 min size.

## Data Preprocessing:

To increase the number of samples of the data, the data is augmented as a 5 sec sample each. The EEG and envelope data are down sampled to 64 hz of frequency. Thus, each sample is of size 320. While augmenting the data, for some models the data is overlapped with one second and for remaining models the data is not overlapped. The artefacts were removed in the overlapped data. The non-overlapped data is the raw data without removing the artefacts. The models are trained for both normalized and non-normalized data, as the normalization is not affecting the correlation matric, and normalized data is simpler for the models, we have finalized to use the normalized data.

The split ratio of the dataset is considered as follows:

Train data: 80%,

Validation data: 10%,

Test data: 10%.

## Performance metrices:

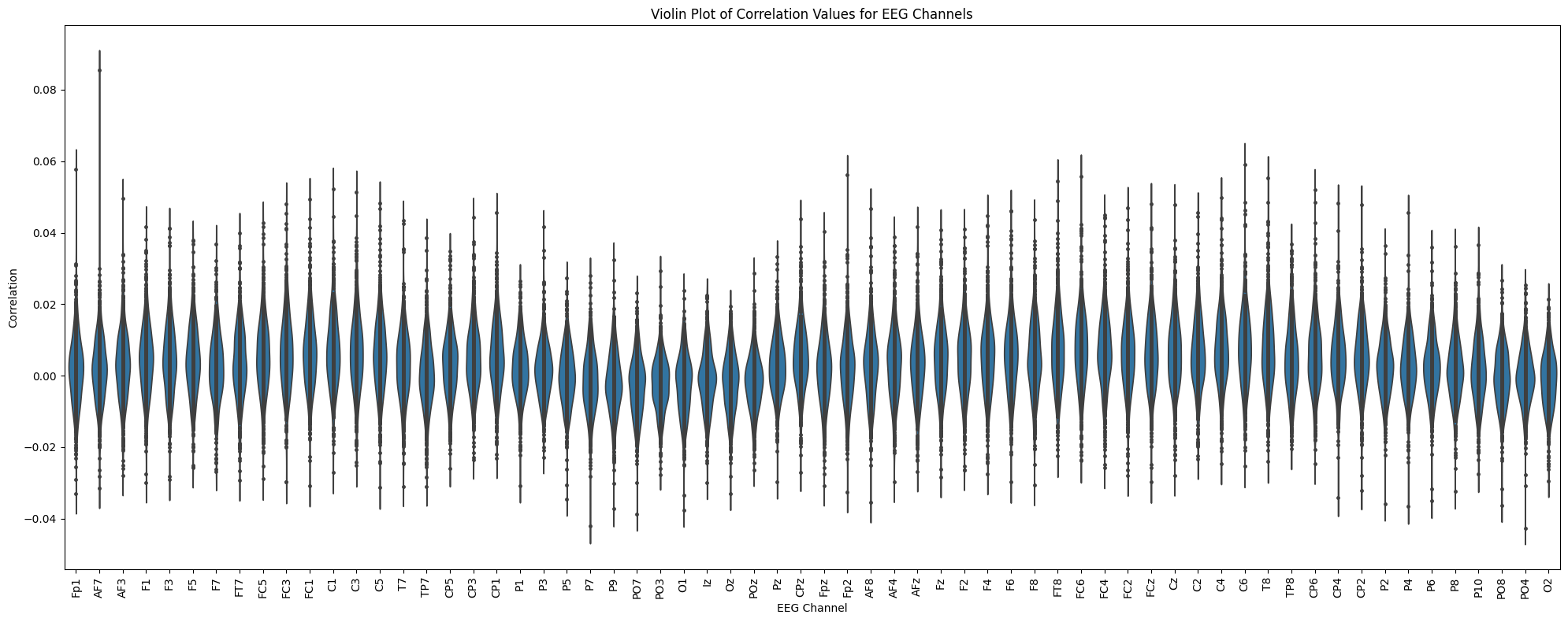
The performance metrices used in the different models were MSE, Pearson correlation and the Cosine similarity. Where our main target is to increase the Pearson correlation metric.

## Data analysis:

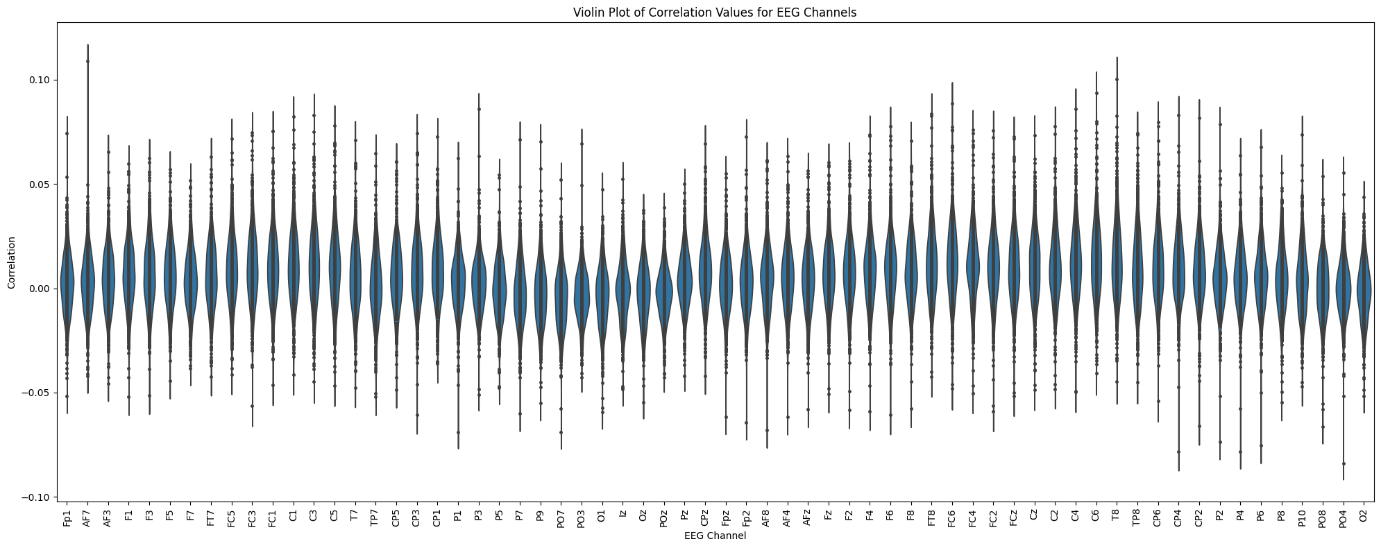
The EEG data is in the form of the 64 channels. These are the 64 electrodes places on the scalp. The names of the 64 channels are :

['Fp1', 'AF7', 'AF3', 'F1', 'F3', 'F5', 'F7', 'FT7', 'FC5', 'FC3', 'FC1','C1', 'C3', 'C5', 'T7', 'TP7', 'CP5', 'CP3', 'CP1', 'P1', 'P3', 'P5', 'P7', 'P9', 'PO7', 'PO3', 'O1', 'Iz', 'Oz', 'POz', 'Pz', 'CPz', 'Fpz', 'Fp2', 'AF8', 'AF4', 'AFz', 'Fz', 'F2', 'F4', 'F6', 'F8', 'FT8', 'FC6', 'FC4', 'FC2', 'FCz', 'Cz', 'C2', 'C4', 'C6', 'T8', 'TP8', 'CP6', 'CP4', 'CP2', 'P2', 'P4', 'P6', 'P8', 'P10', 'PO8', 'PO4', 'O2'].

To understand if there is any relationship between each individual channel of the EEG data and the target envelope, the Pearson correlation was computed between them and plotted. The analysis is made across 5 types of frequency bands, Delta[0.5 – 4], theta[4 – 8], Alpha[8 – 14], Beta[14 - 30], Broadband[0.5 – 32]. The resultant plots for the analysis are shown below.

Delta band[0.5 – 4]:  


Theta band[4 – 8]:

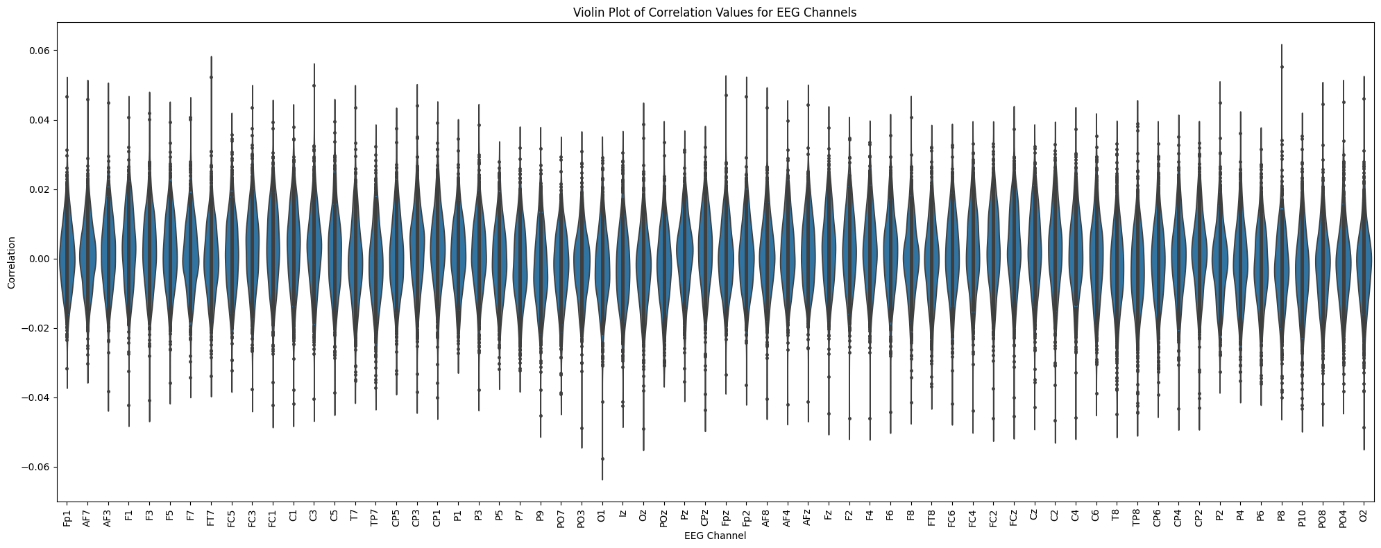


Alpha band[8 – 14]:

A blue and black sound wave

Description automatically generated

Beta band[14 - 30]:

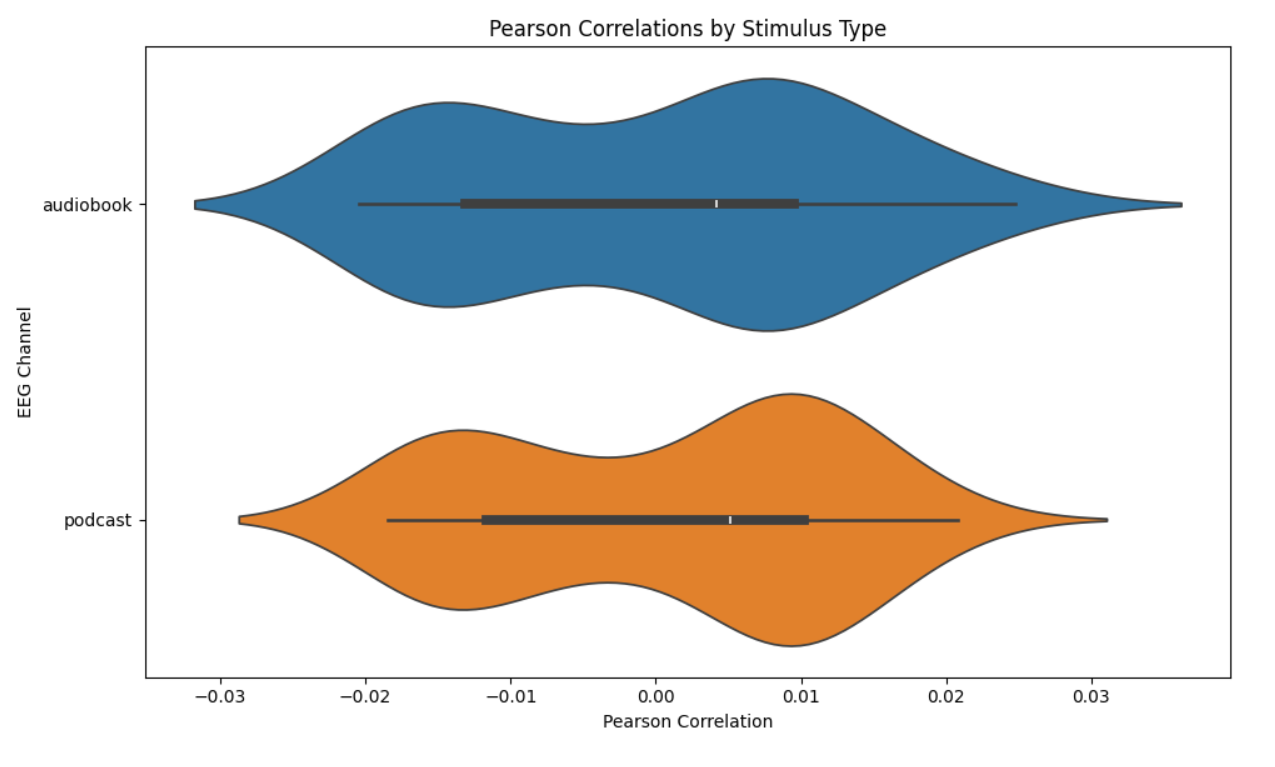


Broad band[0.5 – 32]:

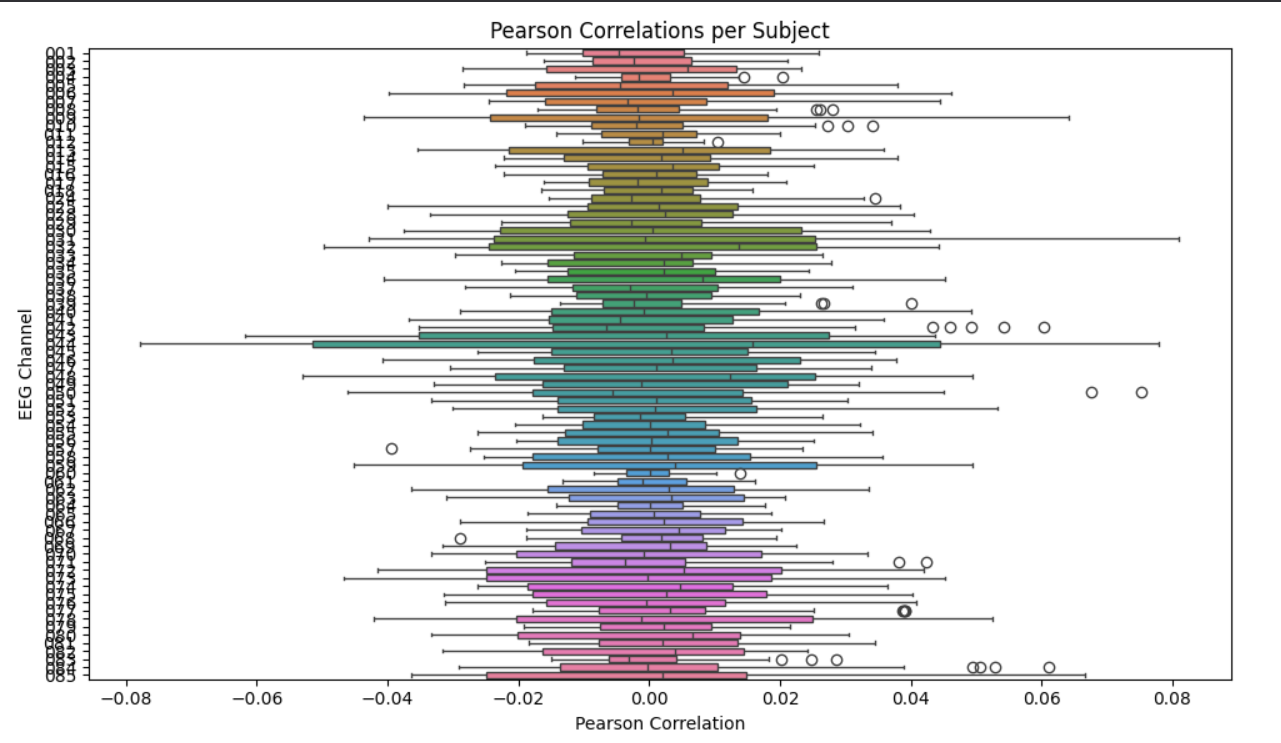
A blue and black sound wave

Description automatically generated

The SparrKULee data set has used mainly to types of audio files for all 85 subjects, they are audio books and podcasts. Below is the violin plot that shows the Pearson correlation according to the type of the audio with its corresponding envelope signal. The image specifies that the correlation between the speech envelope and the EEG signals does not depend much on the type of the audio. Both of their median is close to the 0.01 and the overall it is near 0. This specifies that there is no direct relationship between the envelope and the EEG signal.



The subject wise analysis has also been made to see if there is any direct relationship between the envelope and the EEG signal for any subjects. For every subject the median of the Pearson correlation is near 0. Thus, there is no direct relation.



### Analysis:

The above correlation analysis was made for both the raw data and the data after removing the artefacts. However, there is no significant difference was found between both types of the data and the envelope. The correlation between each channel and the envelope is between 0.10 to -0.10, which specifies that there is no relationship between the individual channel and the envelope. Since the model was not specifically developed based on any channels and all the 64 channels are considered in the models training.

## Pre-trained Models:

The codes for these pretrained models are available in this [github](https://github.com/exporl/auditory-eeg-dataset) repository.

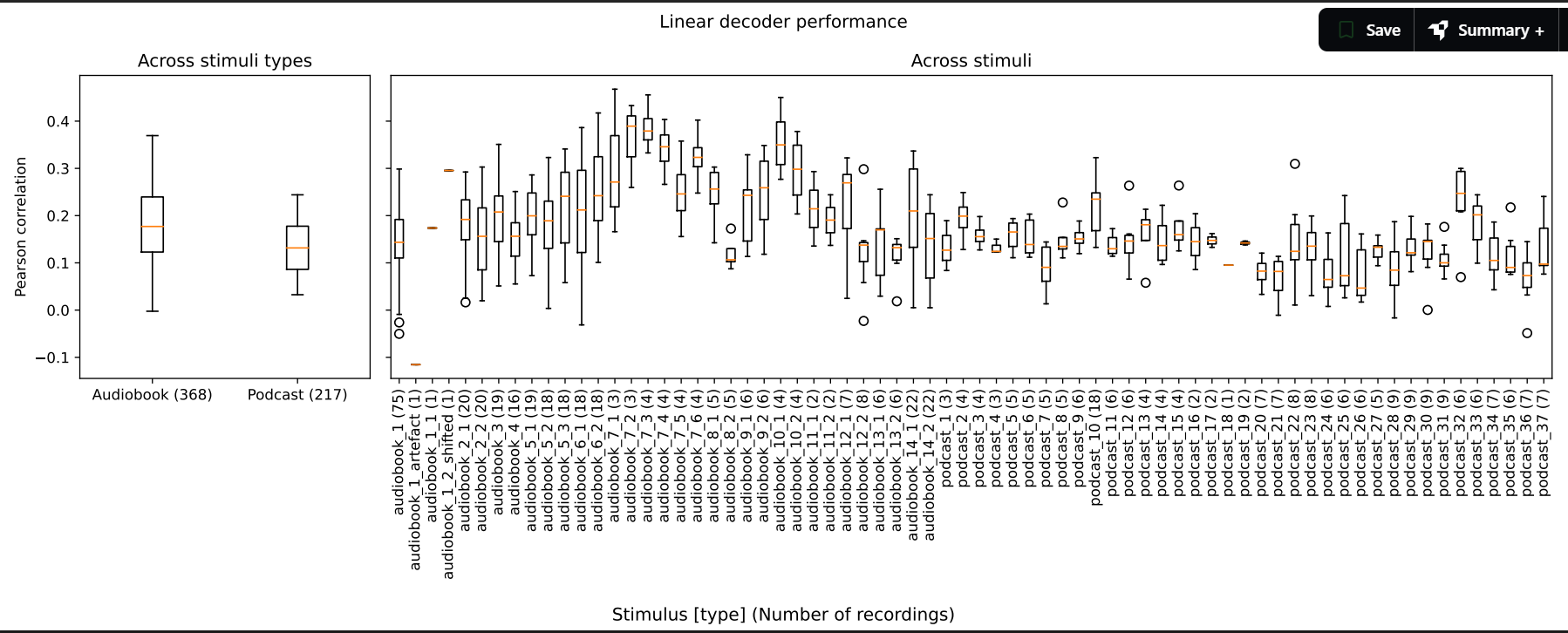
### VLAAI model:

A diagram of a graph

Description automatically generated

This shows a median Pearson correlation of 0.10. with overall correlation ranges in between 0 to 0.25.

### Linear backward model:



Using this model the model was developed to find the Pearson correlation across the stimuli and both types of stimuli which are Audiobook and podcast.

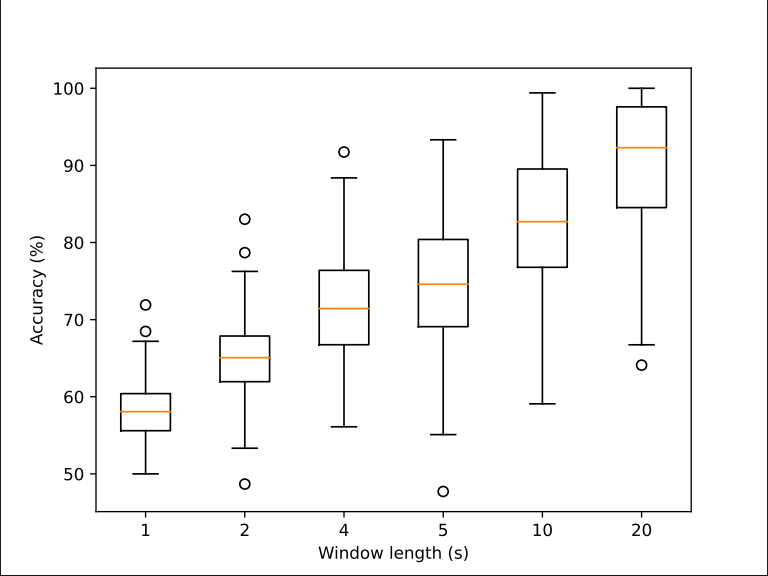
The model is trained with the Delta band to which produced a Pearson correlation of median close to 0.15 and ranges between 0.35 to -0.25.

A diagram of a linear decoder

Description automatically generated

### Dilated convolutional model:

This is a match mismatch task, which tried to classify the EEG signal between a matched stimulus and a mismatched stimulus. The model here mainly focuses on the Dilated convolutional layer. The model was trained across different window lengths.



The results shows that the window length of 20 seconds has the better accuracy of median over 90 percentage.

## Custom models:

These are the different models we have tried on the dataset for the envelope reconstruction task.

### Simple feed forward network:

This is the simplest model with only two fully connected layers or linear layers with 64, 128 neurons.

Data: The one second overlapped data without normalization .

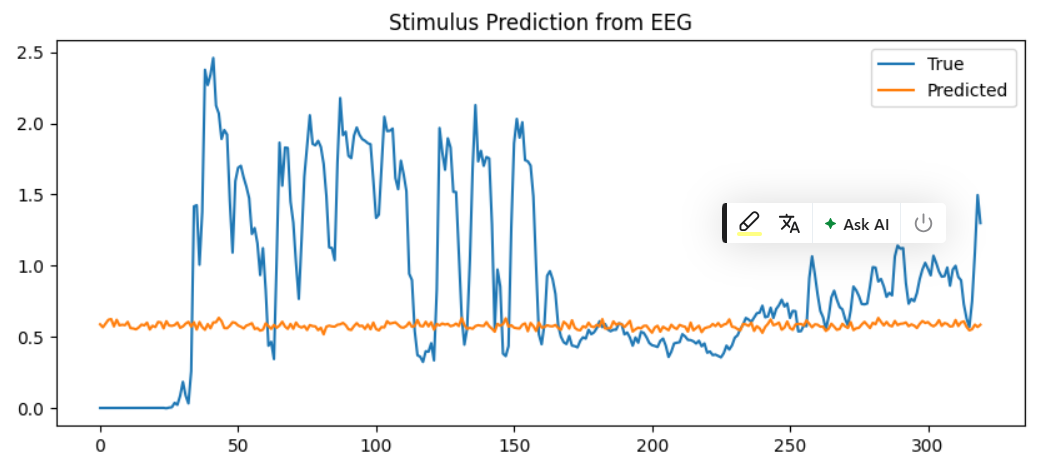
Hyperparameters:

1. Loss function: MSE loss
2. Epochs: 1
3. Activation function: relu
4. Batch size: 32

Results:

Val Loss: 0.2535 | Pearson: 0.0063 | Cosine: 0.7746

test Loss: 0.2301 | Pearson: 0.0551 | Cosine: 0.7735



Even the simplest model can get an average cosine similarity of 0.77. However, the Peason correlation is only 0.006. This specifies that there is no correlation between the EEG and envelope stimulus.

### EEG Transformers:

The EEG Transformers is a Transformer-based neural network. It projects multi-channel EEG inputs into a higher-dimensional space. This adds learned positional encodings, processes the sequence with a Transformer encoder, and outputs a scalar per time step.

Data: The one second overlapped data without normalization.

The **hyperparameters** are initialised as follows:

EEG channels: 64,

dimensionality of the feature space used by the transformers : 64,

number of attention heads: 8,

number of transformers encoded layers: 3,

dimension of the feed forward network layer with each transformer layer: 256,

dropout for regularization: 0.1,

weight decay: 1e-4,

epochs: 1,

batch size: 32.

Test on the val and test data:

val data: Epoch 01 | Val Loss: 0.2539 | Pearson: 0.0776 | Cosine: 0.7734

test data: Epoch 01 | test Loss: 0.2320 | Pearson: 0.0953 | Cosine: 0.7724

A graph showing a number of blue lines

Description automatically generated

Analysis:

The Pearson correlation is a little better than the simple feed forward network, however it is still near 0, which suggests that the model was unable to find any correlation between EEG and envelope stimulus.

### Hybrid model of 4 block to train and combine with 4 loss functions :

This model has 4 blocks, each block is trained separately with different loss functions. After training all 4 blocks, the final block combines the 4 blocks with a fully connected network. This is to try to learn all four metrices including MSE, MAE, Pearson correlation and cosine similarity.

Data: The one second overlapped data without normalization .

hyperparameters = {

'batch\_size': 16,

'num\_epochs': 1, # Low epoch for initial testing

'learning\_rate': 0.001,

'cnn\_filters': [64, 128, 256, 512], # CNN filters for each of the 4 CNN layers per block

'lstm\_hidden': 128, # LSTM hidden units

'mlp\_hidden': [256, 128, 64], # MLP hidden layers in block 4

'cnn\_kernel\_size': 3,

'lstm\_layers': 1,

'dropout\_rate': 0.3

}

Block 1: CNN + LSTM , loss function : Pearson correlation.

Block 2: CNN + LSTM , loss function: cosine similarity.

Block 3: CNN + LSTM , loss function: MSE.

Block 4: CNN + LSTM , loss function: MAE.

Final Block: MLP layer, inputs are the outputs of the 4 blocks above.

This is one idea that we have worked with. However, as we need to train 4 separate models individually, we would need 4 times the GPU memory. So worked on the CPU.

Epoch 1/1

Block 1 - Train MSE: 0.6813, Pearson: 0.0079, Cosine: -0.0108

Block 1 - Val MSE: 0.7644, Pearson: 0.0083, Cosine: -0.0146

Block 2 - Train MSE: 0.5402, Pearson: 0.0021, Cosine: 0.7921

Block 2 - Val MSE: 0.8436, Pearson: 0.0023, Cosine: 0.8032

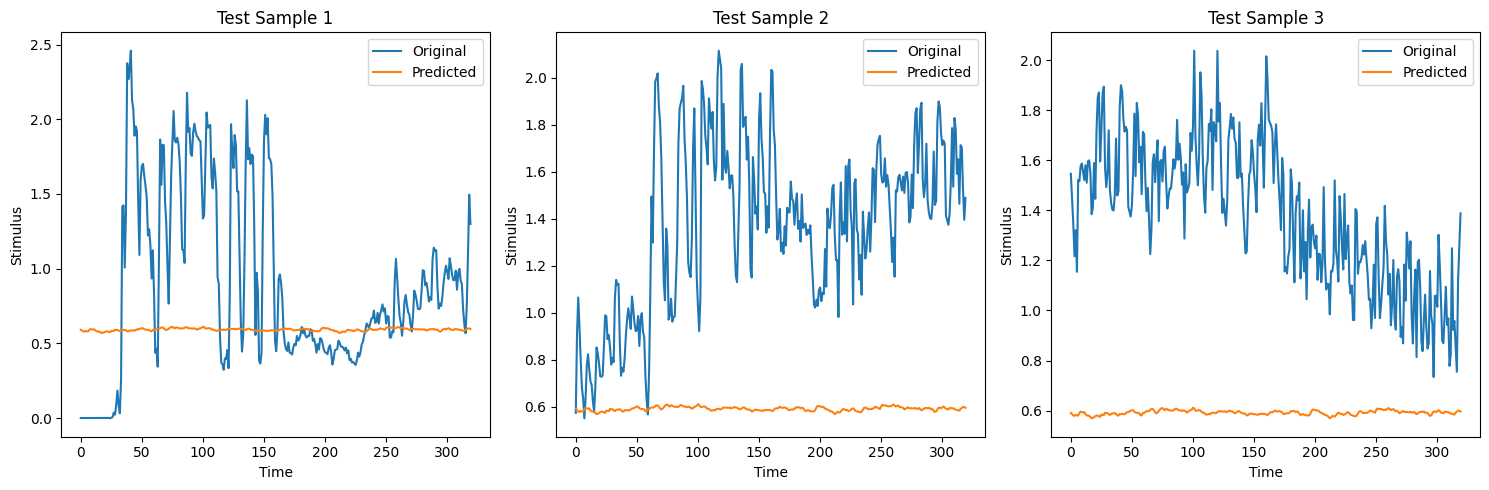
Block 3 - Train MSE: 0.2454, Pearson: 0.0013, Cosine: 0.7891

Block 3 - Val MSE: 0.2582, Pearson: 0.0020, Cosine: 0.8020

Block 4 - Train MSE: 0.2440, Pearson: 0.0011, Cosine: 0.7933

Block 4 - Val MSE: 0.2512, Pearson: -0.0021, Cosine: 0.8039

We got the highest Pearson value and low cosine similarity, when the loss function is the Pearson correlation. Other three times the cosine similarity is higher, and Pearson correlation is lower.



### Cultured and combined model:

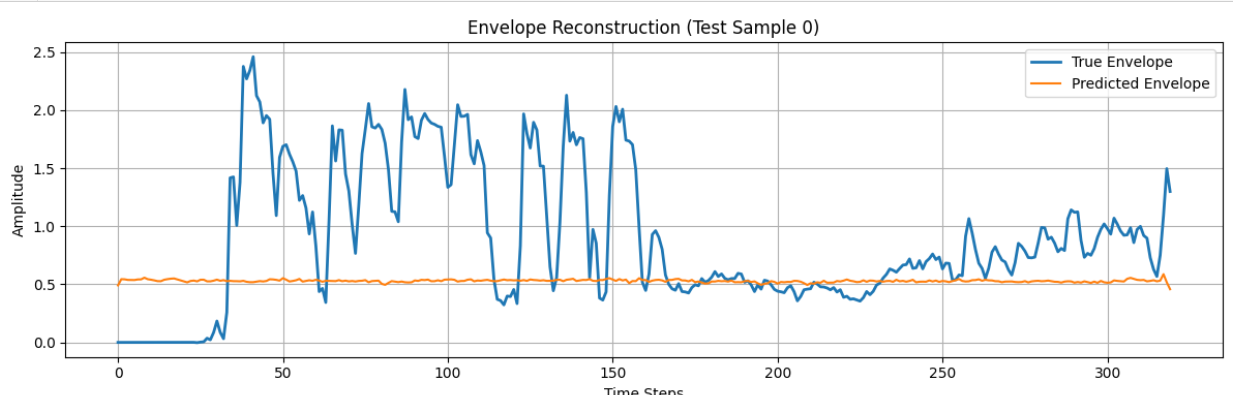
This is a hybrid approach, where the input data is clustered into k clusters using the auto encoders and k-means clustering, and the model is trained for each k cluster of the data and later combine with a final convolutional model. This approached taken to see if the input can be classified according to some properties like the amplitude of subject etc… Each clustered data is trained with a LSTM layer, and the final outputs of each cluster model is trained with a CNN layer.

Data: The one second overlapped data without normalization .

Epochs: 5

Loss function: MSE

Val MSE = 0.2345 | Pearson = 0.0067 | Cosine = 0.7980



This model is underfit, as the model could not find the correlation between the EEG and the envelope.

Observation:

After the clustering, 95% of the data is classified as a single cluster, hence the clustering is not useful for the model training. The remaining clusters data is negligible to train any parameters. Therefore, this model is just like the single cluster model.

### TGCN model:

A TGCN model is combination of the GCN model and a temporal module like GRU or LSTM. This model is considered to train on the data because the EEG data contains both the positional and temporal data. Where the temporal data is the time stamps, and the positional data is the position of the electrodes. The GCN requires an adjacency matrix of the data, which represents the adjacent electrodes to every electrode.

Data: used the non-normalised and one second overlap data.

parameters= {

# Model Architecture

"input\_channels": 64,

"hidden\_channels": 64,

"gcn\_out\_channels": 32,

"fc\_out\_channels": 1,

"kernel\_size": 5,

# Loss Weights

"alpha": 0.6, # Pearson correlation weight

"beta": 0.2, # Cosine similarity weight

"gamma": 0.1,

"delta":0.1,

# Training Parameters

"learning\_rate": 1e-3,

"batch\_size": 16,

"epochs": 100,

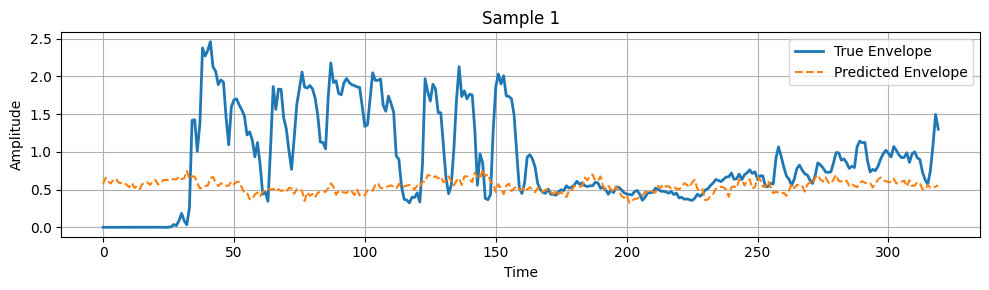
"use\_subset\_percentage": 1,

}

Loss function: alpha\* pearson corr + beta \* cosine sim + gamma\* MSE + delta \* MAE

Test Pearson Correlation: 0.0645

Test Cosine Similarity: 0.7930



Observation: this model is still underfit.

### GCN + LSTM + CNN:

CNN and LSTM are the main types of neural networks, and GCN is a positional neural network, these were combined to see if any information can be made.

Data: used the non-normalised and one second overlap data.

parameters = {

"input\_channels": 64,

"seq\_len": 320,

"gcn\_hidden": 64,

"gcn\_out": 32,

"cnn\_channels": 32,

"cnn\_kernel": 5,

"lstm\_hidden": 64,

"lstm\_layers": 1,

"fc\_out": 1,

"dropout": 0.3,

"learning\_rate": 1e-3,

"batch\_size": 32,

"epochs": 10,

"weight\_decay": 1e-5,

}

Loss functions: Pearson correlation.

Test Loss: 0.9014 | Test PCC: 0.0986 | Test CosSim: 0.7715

A graph of blue and orange lines

Description automatically generated

Observation:

From the above sample plot, we can see that the model is more overfitted. It scaled above the original values; this may be because of the non-normalised nature of the input data.

### GCN + LSTM + CNN 2:

This is the same model as the above but with different loss function.

# config.py or at the top of your notebook

config = {

"input\_channels": 64,

"seq\_len": 320,

"gcn\_hidden": 64,

"gcn\_out": 32,

"cnn\_channels": 32,

"cnn\_kernel": 5,

"lstm\_hidden": 64,

"lstm\_layers": 1,

"fc\_out": 1,

"dropout": 0.3,

"learning\_rate": 1e-3,

"batch\_size": 32,

"epochs": 10,

"weight\_decay": 1e-5,

# Loss weights

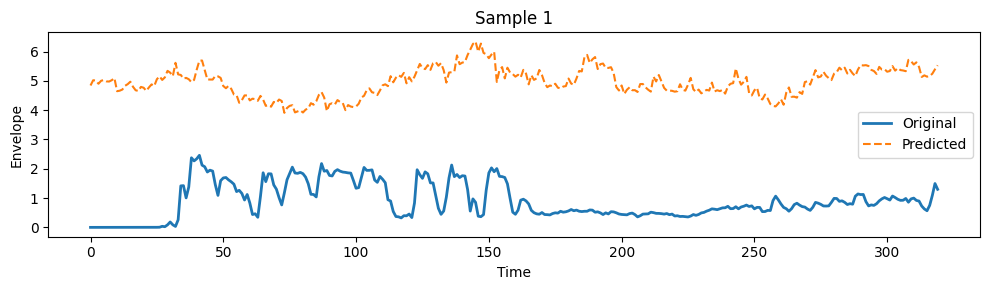
"alpha": 0.5, # Pearson

"beta": 0.5, # Cosine

}

Loss function: alpha\* pearson corr + beta \* cosine sim

Test Loss: 0.5502 | Test PCC: 0.0989 | Test CosSim: 0.8007



Observation:

Like the results of the previous model, the predicted output is scaled much above the amplitude of the original. this may be because of the non-normalised nature of the input data or could be the reason as the Pearson correlation and cosine similarity is the loss function and not the MSE.

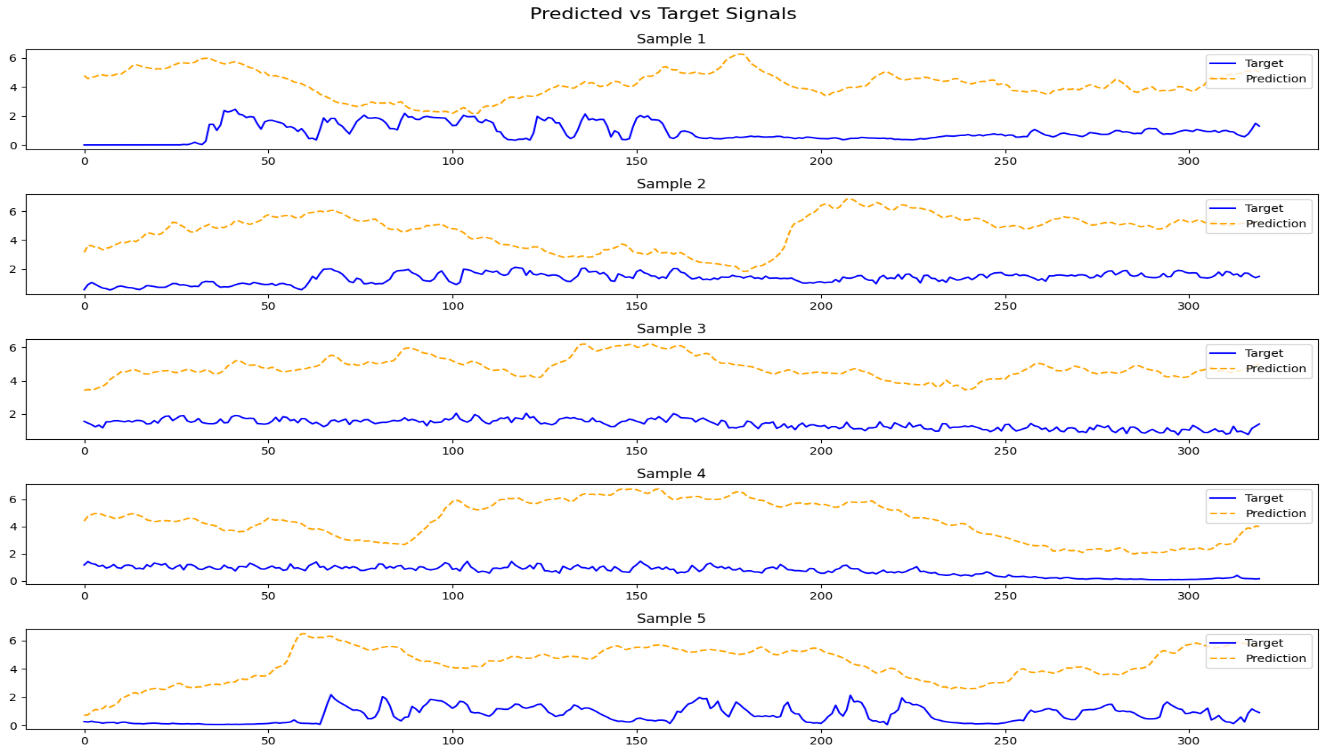
### CNN + LSTM:

This model contains 2 CNN layers and 2 LSTM layers. This is still not complex enough to make good result but consider seeing the working of the model if the loss function is Pearson correlation.

Data: non-normalised one second overlap data.

Epochs: 10

Loss function: Pearson correlation.



From the plots, the scaling is still above the original amplitude. while it does not affect the Pearson correlation, the scaling can still be observed. So, we can only say that without the MSE loss function the scaling will be neglected by the model as MSE loss function will try to keep the distance between the original and predicted to be smaller.

### 2 CNN + 2 LSTM:

This model contains 2 CNN layers and 2 LSTM layers.

Data: normalised and one second overlap.

hyperparameters = {

'input\_size': 64, # Number of EEG channels

'lstm\_hidden\_size': 128, # Number of units in LSTM hidden layers

'cnn\_channels': 64, # Number of filters in CNN layers

'kernel\_size': 3, # CNN kernel size

'num\_blocks': 4, # Number of LSTM+CNN blocks

'lstm\_layers': 2, # Number of LSTM layers per block

'cnn\_layers': 2, # Number of CNN layers per block

'dropout': 0.3, # Dropout rate

'learning\_rate': 0.001, # Learning rate

'batch\_size': 32, # Batch size

'num\_epochs': 5, # Number of epochs (low for initial testing)

'data\_fraction': 1 # Use 1% of the data

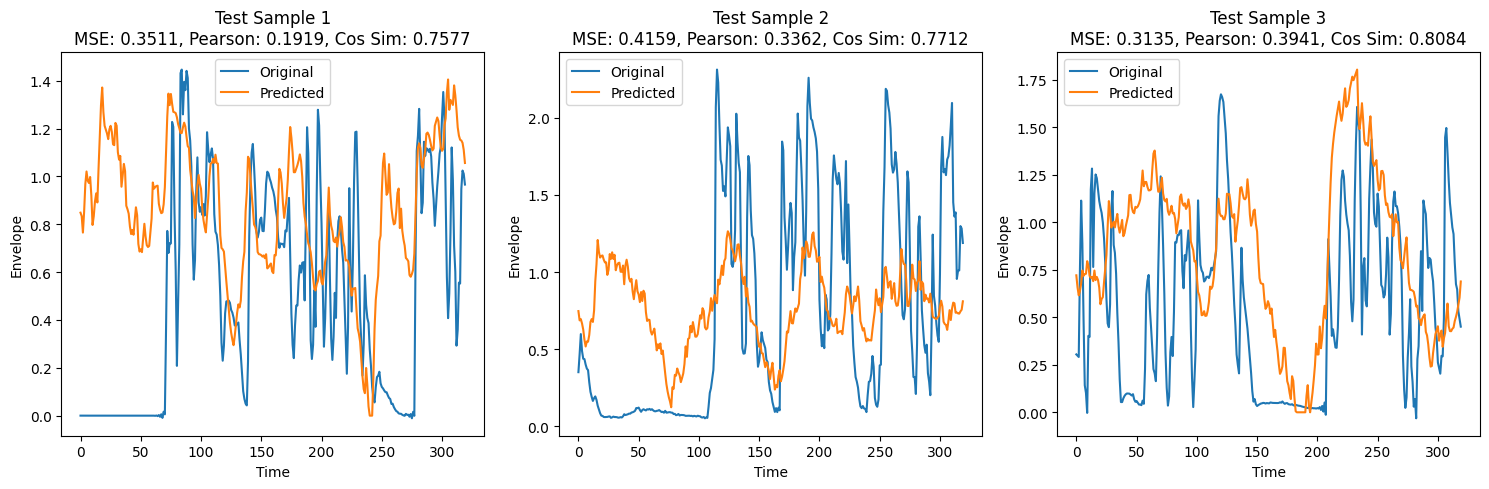
}

Val Loss: 0.8871

Val MSE: 0.3746

Val Pearson: 0.1129

Val Cosine Similarity: 0.7848



A diagram of a rhombus

Description automatically generated

Observations:

The highest Pearson correlation is more that 0.75, however the median is still only near 0.10, and there are some data, that got the negative correlation. This model cannot be told to be a good fit.

### CNN + LSTM + Transformers:

This model is a combination of the CNN, LSTM, transformers networks.

Data: normalized and one second overlapped data.

# Hyperparameters

hyperparameters = {

# Data

'fraction': 1, # Fraction of training data to use (1%)

'batch\_size': 32, # Batch size for training, validation, and test

# Model Architecture

'cnn\_filters': [32, 64], # Number of filters in CNN layers

'cnn\_kernel\_size': (3, 3), # Kernel size for CNN layers

'cnn\_padding': 1, # Padding for CNN layers

'lstm\_hidden\_size': 128, # Hidden units per direction in LSTM

'lstm\_num\_layers': 2, # Number of LSTM layers

'lstm\_bidirectional': True, # Whether LSTM is bidirectional

'transformer\_d\_model': 256, # Dimension of Transformer model

'transformer\_nhead': 4, # Number of attention heads in Transformer

'transformer\_dim\_feedforward': 512, # Feedforward dimension in Transformer

'transformer\_num\_layers': 2, # Number of Transformer encoder layers

'transformer\_dropout': 0.1, # Dropout rate in Transformer

# Training

'num\_epochs': 50, # Number of training epochs

'learning\_rate': 1e-3, # Learning rate for Adam optimizer

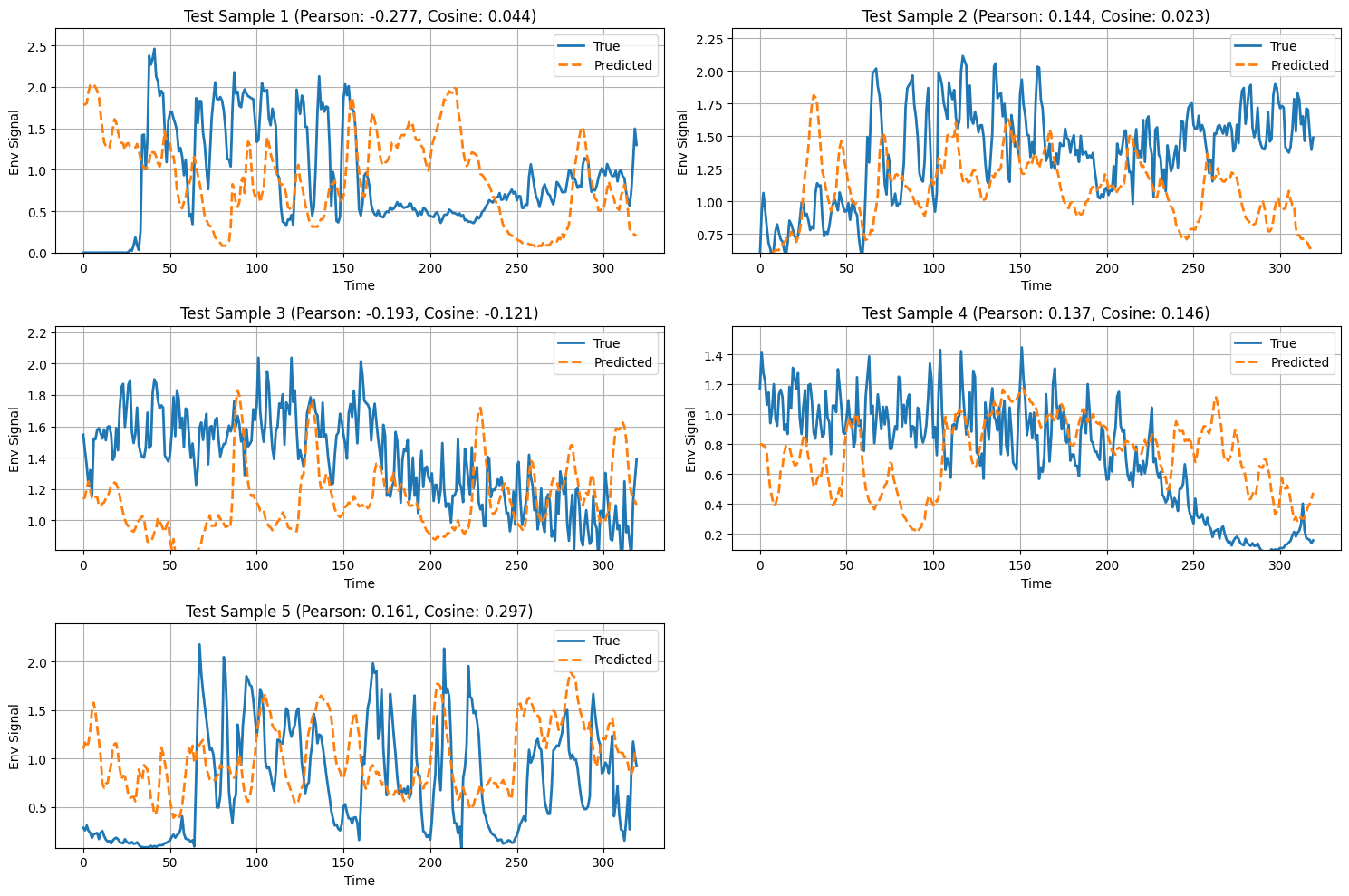
'loss\_function': 'MSE', # Loss function (Mean Squared Error)

}

Val Loss: 0.3708, Val MSE: 0.3709, Val MAE: 0.4911, Val Pearson: 0.0788, Val Cosine: 0.4173

A comparison of a plot with a blue rhombus

Description automatically generated with medium confidence



Here the cosine similarity is worse compared to the previous model, average Pearson correlation is also decreased.

## Currently working on:

As we have tried and analysed different neural networks models, I would like to keep a model fixed and work for improving the complexity of that model. As for the novelty of the model, I would like to use the Graphical convolution network(GCN), which integrate the space information using a graphical structure of the EEG electrodes. This model has not been used yet for the envelope reconstruction using EEG signal; however, this is used in tasks like, Auditory attention detection in cocktail party scenario, match mismatch classifications, emotion recognition etc... Following are some reference papers that uses the GCN along with EEG signals:

1. S. Cai, R. Zhang, M. Zhang, J. Wu and H. Li, "EEG-Based Auditory Attention Detection With Spiking Graph Convolutional Network," in *IEEE Transactions on Cognitive and Developmental Systems*, vol. 16, no. 5, pp. 1698-1706, Oct. 2024, doi: 10.1109/TCDS.2024.3376433
2. Aref Einizade, Mohsen Mozafari, Shayan Jalilpour, Sara Bagheri, Sepideh Hajipour Sardouie, Neural decoding of imagined speech from EEG signals using the fusion of graph signal processing and graph learning techniques, Neuroscience Informatics, Volume 2, Issue 3,2022,100091,ISSN 2772-5286, https://doi.org/10.1016/j.neuri.2022.100091.
3. S. Cai, R. Zhang and H. Li, "Robust Decoding of the Auditory Attention from EEG Recordings Through Graph Convolutional Networks," *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Seoul, Korea, Republic of, 2024, pp. 2320-2324, doi: 10.1109/ICASSP48485.2024.10447633.
4. Wang, Ruicong, Siqi Cai, and Haizhou Li. "EEG-based auditory attention detection with spatiotemporal graph and graph convolutional network." In *Proceedings of INTERSPEECH*, pp. 1144-1148. 2023.
5. Song, Tengfei, Wenming Zheng, Peng Song, and Zhen Cui. "EEG emotion recognition using dynamical graph convolutional neural networks." *IEEE Transactions on Affective Computing* 11, no. 3 (2018): 532-541.
6. Cai, Siqi, Tanja Schultz, and Haizhou Li. "Brain topology modeling with EEG-graphs for auditory spatial attention detection." *IEEE Transactions on Biomedical Engineering* 71, no. 1 (2023): 171-182.
7. Raeisi, Khadijeh, Mohammad Khazaei, Pierpaolo Croce, Gabriella Tamburro, Silvia Comani, and Filippo Zappasodi. "A graph convolutional neural network for the automated detection of seizures in the neonatal EEG." *Computer methods and programs in biomedicine* 222 (2022): 106950.

By looking into these reference papers, I would like to use the Graph convolution networks for the EEG reconstruction tasks.