BCI Signal reconstruction and ehancemennt using GAN

Francesco Citeroni[1], Giuseppe Mancini[1]

[1]University of Rome 'La Sapienza', Rome, Italy

Abstract

Electroencephalography (EEG) is a vital tool for recording the brain's electrical activity and is the primary source of signals for brain-computer interfaces (BCIs). BCIs facilitate communication between the brain and external devices by converting thoughts into actions through specialized processing techniques. However, these recorded signals are often heavily affected by noise, rendering them unusable and restricting the widespread use of BCIs. This study aims to explore the application of Generative Adversarial Networks (GANs) to reconstruct and enhance BCI signals. GANs, known for their ability to generate high-quality data, will be trained using high-quality brain recordings. The goal of the project is to develop and implement a system that can process lower-quality or partially corrupted BCI signals, with the objective of improving and restoring them. This approach could provide substantial benefits, especially in scenarios where signal quality is compromised, such as with wireless or portable BCI devices. Dataset(s) used: Multi-Modality Emotion Recognition, Abnormal Data.

Keywords

LaTeX class, paper template, paper formatting, CEUR-WS

1. Introduction

Linear methods for signal enhancement, like least mean square (LMS) algorithms and their variants, are commonly used for noise reduction because of their simplicity. However, their effectiveness is limited when dealing with nonlinear signals, and they often fail to reach a global optimum for eliminating noise and interference. On the other hand, nonlinear techniques, such as wavelet transform with adaptive thresholding, are popular due to their strong local time-frequency analysis capabilities. These methods work by decomposing signals into multiple frequency bands and filtering out noise based on a selected threshold.

It has been proven that deep neural networks (DNNs) not only exhibit frequency selectivity but also have the ability to learn and analyze the noise and interference patterns within wireless channels. This makes deep learning particularly advantageous in complex wireless environments with significant noise and interference. Similarly, it was found that autoencoders can effectively reduce noise in wireless channels, further confirming the potential of deep learning methods for signal enhancement.

Overall, current techniques focus on learning the noise or interference distributions in either the time or frequency domains and filtering them out. However, the unpredictable nature of dynamic transmission channels can degrade performance or even render fixed signal enhancement systems ineffective. A recent breakthrough in deep learning, particularly in the field of generative modeling, is the emergence of Generative Adversarial Networks (GANs), which hold promise for more adaptive and robust signal enhancement solutions.

The proposed wireless signal enhancement based on Generative Adversarial Networks (GANs) offers several key advantages:

1. Adaptability to Nonlinear Systems: Unlike traditional linear methods, which struggle with nonlinear signals, GANs can adapt to the nonlinearities present in wireless communication systems. This allows for more effective noise and interference suppression, even in complex signal environments.

2. Dynamic Learning: GANs are capable of continuously learning and adapting to the characteristics of signals and noise in time-varying systems. This makes them more flexible in handling the unpredictable and dynamic nature of wireless transmission channels, where conventional methods often fail due to their fixed design.

3. Enhanced Noise and Interference Reduction: The GAN framework allows for a more nuanced approach to noise reduction by generating a model of the "clean" signal and using it to distinguish and remove noise and interference more effectively. This can lead to improved signal quality, especially in environments with severe interference.

4. Global Optimization: GANs can overcome the limitations of traditional algorithms, such as least mean square (LMS), which may not always achieve a global optimum. Through adversarial training, GANs are better positioned to reach more optimal solutions for noise and interference elimination.

5. Improved Performance in Complex Wireless Channels: Deep learning techniques, including GANs, have shown superior performance in complex wireless channels where traditional methods are less effective. The ability of GANs to learn and model intricate patterns and

Woodstock'21: Symposium on the irreproducible science, June 07–11, 2021, Woodstock, NY

citeroni.1758276@studenti.uniroma1.it (F. Citeroni); mancini.@studenti.uniroma1.it (G. Mancini)

characteristics of the wireless environment allows for more robust signal enhancement.

6. Future-Proofing for Wireless Networks: As wireless networks become more complex, with increasing interference and noise, the GAN-based approach offers a scalable and forward-looking solution. It can be trained and refined as new challenges arise, making it a promising tool for next-generation wireless communication systems.

In summary, the main advantages of using GANs for wireless signal enhancement lie in their adaptability, dynamic learning capabilities, ability to handle nonlinearities, and improved performance in complex, noisy environments. In this report we will give a brief introduction of the model, focusing in particular on the GAN model. By taking advantage of recent advancements in GANs, we have developed a pipeline that can denoise artefacts from EEG time series data, effectively mapping noisy EEG signals to clean ones based on the specific nature of the artefact. We showcase the effectiveness of our network using a specific dataset.

2. GAN MODEL

Generative Adversarial Networks (GANs) are a type of generative model used to create synthetic data without requiring detailed domain-specific knowledge. They were introduced by Goodfellow et al. (2014), in which a multilayer perceptron was employed for both the generator and the discriminator networks. These two networks engage in a competitive process, often referred to as a minimax game, as described by the objective function in Equation (1). The generator aims to increase the error rate of the discriminator by producing data that resembles real samples, while the discriminator tries to accurately differentiate between real and generated samples. This interaction is illustrated in Figure 2. GANs are widely applied to generate new, previously unseen data, either for augmenting existing datasets or to ensure the privacy of the training data.

$$\min_G \max_D V(G,D) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))] \quad (1)$$

2.1. EEG Signal reconstruction model - LSTM layer

The first `Generator` implementation was a neural network-based model for EEG signal reconstruction, which utilizes a Long Short-Term Memory (LSTM) layer to model the temporal dependencies in the signal. The model architecture is designed to process sequential EEG data. It consists of an LSTM layer followed by a fully connected (linear) layer, with an optional Tanh activation function for output regularization.

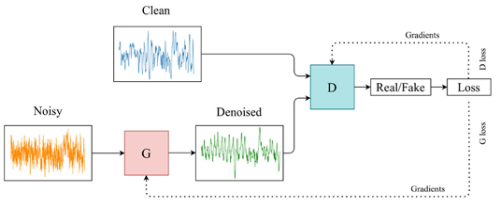

Figure 1: Generative Adversarial Network (GAN) Architecture for Converting Noisy EEG Signals to Clean EEG Data.

The LSTM layer is the core component for learning temporal dependencies in EEG signals. Given an input sequence $X = \{x_1, x_2, \ldots, x_T\}$ where x_t represents the input at timestep t, the LSTM produces hidden states h_t and cell states c_t at each timestep. The LSTM's internal operations are governed by the following equations:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \quad \text{(forget gate)} \quad (2)$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \quad \text{(input gate)} \quad (3)$$

$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \quad \text{(candidate cell state)} \quad (4)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \quad \text{(cell state update)} \quad (5)$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \quad \text{(output gate)} \quad (6)$$

$$h_t = o_t \odot \tanh(c_t) \quad \text{(hidden state)} \quad (7)$$

Here, σ is the Sigmoid function, $\odot$ denotes element-wise multiplication, and W, U, and b are the learnable weight matrices and biases for each gate. The LSTM is initialized with zero-valued hidden states and cell states, h_0 and c_0, for each sequence.

Once the LSTM processes the entire sequence, the output is passed through a fully connected (FC) layer to map the hidden state at each timestep h_t to the desired output size y_t, represented as:

$$y_t = W_{fc} h_t + b_{fc} \quad (8)$$

where W_{fc} and b_{fc} are the weights and biases of the FC layer. This operation converts the LSTM's output to a format suitable for signal reconstruction.

The Sigmoid activation function is applied to the output layer to constrain the output to a specific range (e.g., for normalized EEG signals)::

$$y'_t = \sigma(y_t) \quad (9)$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function. This ensures that the model's output is bounded between 0 and 1, which is useful for many signal processing applications.

The combination of the LSTM's ability to capture long-term dependencies and the flexibility of the fully connected layer makes this model highly suitable for EEG signal reconstruction tasks, where capturing both short- and long-term temporal patterns is crucial for accurate performance.

2.2. EEG Signal reconstruction model - Residual Neural Network

The second `Generator` implementation was designed to transform the input sequence into a refined output using a series of convolutional and deconvolutional layers with intermediate residual blocks to enhance feature representation. The network begins with an initial convolutional layer, which applies a 1D convolution with a kernel size of 3 and 64 filters, followed by a batch normalization layer to standardize the output. The activation function used is the rectified linear unit (ReLU), which is defined as:

$$\text{ReLU}(x) = \max(0, x) \tag{10}$$

This activation function ensures that non-negative inputs pass through while negative values are set to zero, allowing for non-linearity in the model. The output of the first convolutional layer can be represented as:

$$y_1 = \text{ReLU}(\text{BN}(\text{Conv1D}(x))) \tag{11}$$

where BN denotes batch normalization and Conv1D represents the 1D convolution operation.

Following the initial convolutional layer, the generator passes the data through a series of 16 residual blocks. Each residual block consists of two convolutional layers with kernel size 3, batch normalization, and ReLU activations. The primary advantage of residual blocks is the inclusion of skip connections, allowing the input to bypass the convolutional layers and be directly added to the output. This can be mathematically represented as:

$$y_{k+1} = \text{ReLU}(y_k + \text{Block}(y_k)) \tag{12}$$

where $\text{Block}(y_k)$ represents the output of the convolutional block in the k-th residual block.

Once the signal passes through the residual blocks, it is processed by a deconvolutional layer, which expands the dimensionality of the sequence. This deconvolution layer uses a transposed convolution operation with a stride of 2, effectively upsampling the data. The formula for the output after the deconvolutional layer is:

$$y_2 = \text{ReLU}(\text{BN}(\text{ConvTranspose1D}(y_1))) \tag{13}$$

Finally, the generator outputs the signal using a 1D convolutional layer with a sigmoid activation function, which ensures that the final output is scaled between 0 and 1. Thus, the final output of the generator is:

$$y_{\text{final}} = \sigma(\text{Conv1D}(y_2)) \tag{14}$$

Overall, the architecture effectively combines convolutional layers, residual blocks, and deconvolution operations to refine the input signal and produce the desired output. The use of residual connections helps mitigate the vanishing gradient problem, allowing the network to learn more effectively.

2.3. Discriminator

The Discriminator model is designed as a deep convolutional neural network (CNN) with the objective of distinguishing real data sequences from those generated by the adversarial model. The input to the Discriminator is a one-dimensional sequence of size *seq_len* with *input_size* features. The model architecture follows a sequence of convolutional layers, each increasing in complexity to progressively extract higher-level features from the input sequence.

The network consists of eight convolutional layers. The first layer applies a convolution with 64 filters, each of size 3, using a stride of 1, followed by batch normalization to stabilize learning and LeakyReLU activation to introduce non-linearity. As the layers progress, the number of filters doubles while the stride alternates between 1 and 2, which reduces the sequence length. The convolutional operation in each layer can be expressed as:

$$y(t) = \sum_{k=1}^{K} w_k \cdot x(t - k + 1) + b \tag{15}$$

where $y(t)$ is the output at time step t, w_k is the convolution filter, $x(t-k+1)$ represents the input values, and b is the bias term. Padding ensures that the output dimensions are maintained in layers with a stride of 1, while layers with stride 2 downsample the input, effectively reducing the temporal resolution of the sequence.

The final layers in the convolutional block output feature maps of size $1024 \times \frac{\textit{seq\_len}}{16}$. These are flattened and passed through a fully connected layer to produce a scalar value and at the end was applied a sigmoid function.

The architecture is optimized for binary classification, where the goal is to correctly classify real and generated sequences. By progressively reducing the sequence length while expanding the feature space through convolutions, the Discriminator captures essential temporal dependencies and feature hierarchies crucial for distinguishing between real and generated data.

3. MATERIALS AND METHODS

3.1. Computing Platform

The experiments for this project were run on an Nvidia rtx 3080 with PyTorch and Google Colaboratory to make the project readily deployable.

3.2. Datasets

This dataset contains two subcategories. The first one is Healthy dataset contains 64 channels and consists of over 1500 one and two minutes EEG recordings, obtained from 109 volunteers. Subjects performed different motor/imagery tasks while 64-channel EEG were recorded using the BCI2000 system [1]. Each subject performed 14 experimental runs: two one-minute baseline runs (one with eyes open, one with eyes closed), and three two-minute runs of each of the four following tasks: [Task 1] A target appears on either the left or the right side of the screen. The subject opens and closes the corresponding fist until the target disappears. Then the subject relaxes. [Task 2] A target appears on either the left or the right side of the screen. The subject imagines opening and closing the corresponding fist until the target disappears. Then the subject relaxes. [Task 3] A target appears on either the top or the bottom of the screen. The subject opens and closes either both fists (if the target is on top) or both feet (if the target is on the bottom) until the target disappears. Then the subject relaxes. [Task 4] A target appears on either the top or the bottom of the screen. The subject imagines opening and closing either both fists (if the target is on top) or both feet (if the target is on the bottom) until the target disappears. Then the subject relaxes. Each event code includes an event type indicator (T0, T1, or T2) that is concatenated to the Task * it belongs with (i.e TASK1T2). The event type indicators change definition depending on the Task * it is associated with. For example, TASK1T2 would correspond to the onset of real motion in the right fist, while TASK3T2 would correspond to onset of real motion in both feet: [T0] corresponds to rest. [T1] corresponds to onset of motion (real or imagined) of: the left fist (in runs 3, 4, 7, 8, 11, and 12; for Task 1 (real) and Task 2 (imagined)); both fists (in runs 5, 6, 9, 10, 13, and 14; for Task 3 (real) and Task 4 (imagined)); [T2] corresponds to onset of motion (real or imagined) of: the right fist (in runs 3, 4, 7, 8, 11, and 12; Task 1 (real) and Task 2 (imagined)); both feet (in runs 5, 6, 9, 10, 13, and 14; for Task 3 (real) and Task 4 (imagined));

This dataset consists of raw 18-channel EEG from 7 human participants with orthopedic Impairment during motor imagery(MI). The participants performed a series of MI-related trials across three sessions. These sessions comprised 40 trials, of which four different MI tasks were presented in random order (e.g., Reach → Twist → Lift → Reach → Grasp → Grasp → Twist → Reach → Lift → Reach). Each trial began with 3 s of fixation cross. The monitor then displayed a 4 s visual cue, followed by 3 s of letters indicating the ready state with a gray screen to eliminate the afterimage. The participants were then instructed to perform the imaginary movement for 5 s in the given order.

[1] http://www.bci2000.org

3.3. Preprocessing

The preprocessing of the dataset was a very important thing for this project and involves several critical steps to ensure the quality and relevance of the data used for subsequent analysis. Initially, we select a random tab-separated values (TSV) file from the provided directory, which serves as a reference for replacing and aligning data in the comma-separated values (CSV) files. The preprocessing function, *My Unhealthy Dataset* reads each CSV file while preserving the last column, which contains labels for different conditions. A random TSV file is selected to replace the first row of each CSV file with a truncated version of the first column from the TSV file. This step is crucial for aligning data from different sources.

Subsequently, the data is filtered to include only specific rows based on predefined criteria: for unhealthy datasets, rows with labels 'S 1', 'S 4', 'S 8', or 'S 10' are retained, while for healthy datasets, rows are filtered to include 'TASK1T1' or 'TASK1T2' labels, we use those labels since they represent the same task. The last column of the CSV files, which is not required for further analysis, is removed to streamline the dataset.

A further step involves removing channels that are not common across healthy and unhealthy datasets. The *remove-channels* function identifies these common channels by comparing the first column of the TSV files corresponding to healthy and unhealthy conditions. DataFrames are then filtered to include only the columns corresponding to these common channels, ensuring that the datasets are aligned for accurate comparison.

Finally, the preprocessed data is printed along with its associated labels to verify the correctness of the preprocessing steps. This comprehensive preprocessing pipeline prepares the dataset for effective analysis and modeling, ensuring that the data is clean, consistent, and aligned with the objectives of the study.

To ensure consistency and improve the quality of our analysis, we employed a series of preprocessing steps. First, we normalized the dataframes using Min-Max scaling to rescale the feature values to the range $[0, 1]$. This step is crucial as it standardizes the data and prevents features with larger ranges from dominating the analysis. The normalization process is applied individually to each

dataframe in both the unhealthy and healthy datasets, as demonstrated in the provided code.

After normalization, we focused on reducing the dimensionality of our data by retaining only the channels that have 'C' on the column name from each dataframe. This step simplifies the analysis and concentrates on channels that cover the motor cortex and could give better results. The updated dataframes are then printed for verification, confirming that the 'C3' column is consistently included across all dataframes.

In addition to preprocessing the dataframes, we also addressed the preprocessing of tensors. We implemented a filtering function to ensure that each tensor meets the minimum required shape. Tensors smaller than the defined threshold are resized or cropped accordingly. This ensures uniformity across the dataset and prepares the data for subsequent analysis steps. The filtered tensors are printed along with their labels and shapes to facilitate verification and ensure that preprocessing has been applied correctly.

These preprocessing steps are essential for preparing our datasets for further analysis and modeling. By normalizing the data, focusing on specific features, and ensuring consistent tensor dimensions, we enhance the quality and reliability of our results.

3.4. Band Pass Filter

To enhance the quality of the EEG signals and isolate relevant frequency components, a bandpass filter is applied to the dataset. The filtering process is crucial for removing unwanted noise and focusing on the specific frequency range of interest. In this study, a bandpass filter with a passband of 8 to 30 Hz is employed, which is commonly used to target the alpha and beta frequency bands in EEG analysis.

The filtering procedure begins by extracting channel names from the first row of each DataFrame. The DataFrame is then converted into an MNE Raw object, which is a data structure specifically designed for handling EEG data. Using the MNE-Python library, the bandpass filter is applied to the Raw object. This step involves specifying the low and high cutoff frequencies (8 Hz and 30 Hz, respectively) and the sampling frequency (128 Hz).

After filtering, the channel names are reinserted into the DataFrame, and the filtered data is extracted from the Raw object. The data is then converted back into a DataFrame format with the original channel names. This ensures that the filtered data maintains the same structure as the input DataFrames.

The bandpass filter is applied to both unhealthy and healthy datasets, refining the signals and preparing them for further analysis. This preprocessing step is essential for enhancing signal quality and improving the accuracy of subsequent modeling and analysis phases.

4. Training

4.1. EEG-GAN Training

The training process for the EEG-GAN involves alternating updates between the generator and the discriminator, following the principles of adversarial training. The generator aims to produce denoised EEG signals from noisy inputs, while the discriminator distinguishes between real clean EEG signals and the denoised signals generated by the model. The training is performed using two loss functions: binary cross-entropy loss for the adversarial task and mean squared error (MSE) for the content loss between the generated and real signals. Additionally, an optional Kullback-Leibler (KL) divergence term is included to enhance the generator's performance in approximating the true distribution of clean EEG signals.

For each epoch, the discriminator is first trained by minimizing the binary cross-entropy loss function:

$$\begin{aligned}\mathcal{L}_D =& \frac{1}{2}(\mathcal{L}_{BCE}(D(x_{\text{real}}), 1) \\ &+ \mathcal{L}_{BCE}(D(G(x_{\text{noisy}})), 0))\end{aligned} \tag{16}$$

where x_{real} represents clean EEG data, x_{noisy} represents noisy EEG data, G is the generator, and D is the discriminator. The generator is then updated using a combination of the adversarial loss, content loss, and KL-divergence:

$$\begin{aligned}\mathcal{L}_G =& \mathcal{L}_{BCE}(D(G(x_{\text{noisy}})), 1) \\ &+ 0.5 \cdot \mathcal{L}_{MSE}(G(x_{\text{noisy}}), x_{\text{real}}) \\ &+ 0.1 \cdot \mathcal{L}_{KL}(G(x_{\text{noisy}}), x_{\text{real}})\end{aligned} \tag{17}$$

The adversarial loss encourages the generator to produce EEG signals that the discriminator cannot distinguish from real signals, while the content loss ensures that the generated signal closely matches the clean EEG in terms of amplitude. The KL-divergence further regularizes the generator's output to align with the distribution of clean EEG signals.

The model is trained for *num_epochs* iterations, and the losses for both the generator and discriminator are recorded. Every *plot_interval* epochs, a comparison between the noisy and denoised signals is visualized to track the model's performance. The denoised signals are normalized and overlaid with the noisy signals to visually assess the model's ability to remove noise.

4.2. WGAN-EEG Training

The training process for the WGAN model to denoise EEG signals consists of two primary components: the generator and the critic (or discriminator). The generator takes noisy EEG input and aims to produce a clean version, while the critic attempts to differentiate between real clean EEG data and the generator's output. The loss

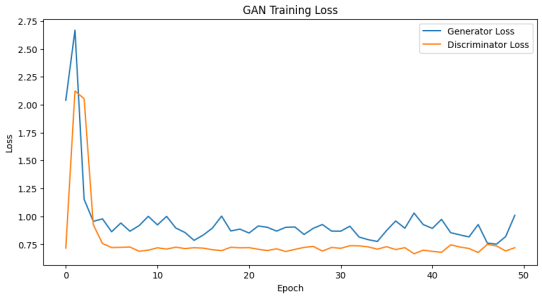

Figure 2: Generative Adversarial Network (GAN) Architecture for Converting Noisy EEG Signals to Clean EEG Data.

functions used during training are the Wasserstein GAN (WGAN) loss for both the generator and the critic, as well as a content loss for the generator, which ensures the denoised output closely resembles the ground truth.

The critic is updated multiple times per iteration to maintain the Lipschitz constraint by clipping its weights within a range $[-0.1, 0.1]$. The critic loss is calculated as:

$$L_C = -\mathbb{E}_{x \sim P_{\text{real}}}[C(x)] + \mathbb{E}_{\hat{x} \sim P_{\text{fake}}}[C(\hat{x})], \tag{18}$$

where $C(x)$ represents the critic's output for real EEG data x, and $C(\hat{x})$ represents its output for the generated data $\hat{x}$.

For the generator, the adversarial loss encourages it to produce realistic EEG signals, while a content loss ensures the denoised signal closely resembles the clean EEG. The combined generator loss is:

$$L_G = -\mathbb{E}_{\hat{x} \sim P_{\text{fake}}}[C(\hat{x})] + \lambda \cdot \text{MSE}(\hat{x}, x), \tag{19}$$

where $\lambda = 0.5$ is the weight of the content loss (MSE).

The model is trained for a set number of epochs, and at regular intervals, the noisy and denoised signals are plotted for visual inspection. Additionally, signals are normalized between [0,1] to allow better comparison between noisy and denoised EEG over time.

5. RESULTS

5.1. Network convergence

The convergence of the EEG-GAN model during training is analyzed by observing the loss curves of the generator and the critic (discriminator) over 50 epochs, as shown in Figure 2. Initially, both the generator and discriminator exhibit high losses, indicating the difficulty in distinguishing real from generated data and the poor quality of the generated EEG signals. However, within the first few epochs, a rapid decrease in both losses occurs, signifying that the model quickly learns to produce more realistic outputs and to differentiate between real and generated EEG signals.

The discriminator loss stabilizes around a value of 0.75, while the generator loss shows fluctuations but remains relatively steady after approximately 10 epochs. The consistency of the discriminator loss indicates that the critic reaches a stable point where it effectively maintains the Wasserstein distance between real and generated data. The fluctuations in the generator loss suggest that the model continues to explore better mappings for denoising EEG signals, although the improvements become more gradual over time.

The overall trend of the losses suggests successful training, with the model not exhibiting signs of mode collapse or divergence. The stability of the critic's loss around a low value, combined with the generator's loss remaining within a manageable range, indicates convergence toward an optimal solution for the denoising task.

5.2. Reconstruction results

The reconstructions presented in this study were performed using the same dataset that was used for training the models. However, to ensure that the models did not overfit and were capable of generalizing to unseen data, we conducted testing on signals from different channels of both unhealthy and healthy brain activity. This allowed us to assess whether the models could accurately reconstruct signals of a different nature compared to those used during training. Specifically, the models were tested on channels that were not included in the training set, thereby providing a more rigorous evaluation of their robustness and generalization ability. By doing so, we could verify if the models, particularly EEG-GAN and WGAN-EEG, could successfully reconstruct and distinguish between healthy and unhealthy signals, demonstrating their potential for practical applications in EEG signal processing. The results indicated that both models were able to generalize effectively, showing minimal overfitting and successfully reconstructing different types of signals.

The Table 1 shows the comparison of signal reconstructions generated by both EEG-GAN and WGAN-EEG models for noisy and denoised signals. Each subplot represents the overlaid noisy and denoised EEG signals, showcasing the models' capacity for reconstructing cleaner signals.

Subplot (a) demonstrates the EEG-GAN model's reconstruction performance. Here, we observe that the denoised signal (orange curve) captures the general trends of the noisy input (blue curve) but tends to smooth out some of the sharper peaks. This indicates that the EEG-GAN model is effective at reducing noise but may sac-

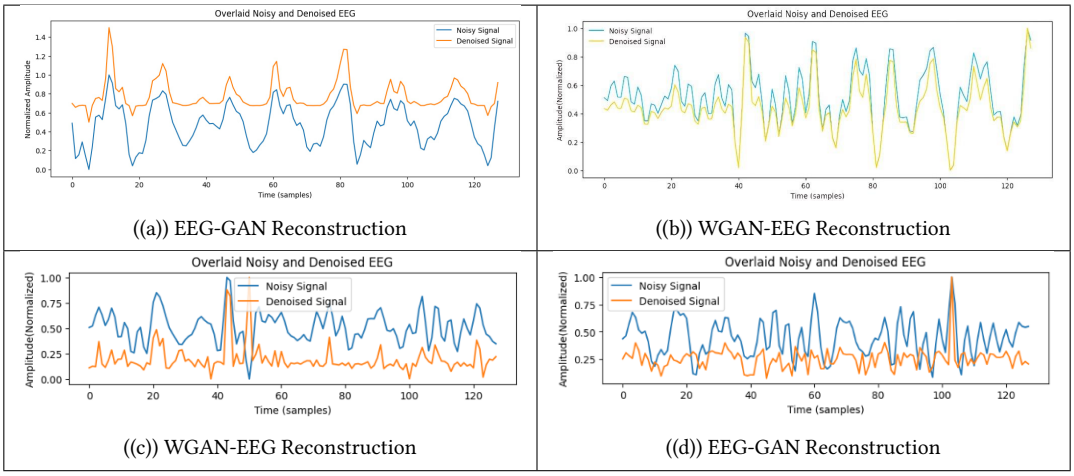

Table 1
Comparison of noisy and denoised EEG signal reconstructions using EEG-GAN and WGAN-EEG models.

rifice some high-frequency components of the signal in the process. The model seems to excel at maintaining the primary oscillations of the signal, particularly in the middle region of the time series.

Subplot (b) presents the WGAN-EEG reconstruction, which differs slightly from the EEG-GAN output. The denoised signal is closely aligned with the noisy signal, even in sections where the EEG-GAN model exhibits smoothing. WGAN-EEG retains more of the signal's intricate features, especially in regions with rapid oscillatory activity, such as the segments between time samples 40 and 80. The ability to preserve high-frequency details in this model suggests that the WGAN framework may offer superior performance in scenarios where subtle signal dynamics are essential for analysis.

Subplot (c), another WGAN-EEG reconstruction, reveals the model's consistency in reconstructing signals across different datasets or channels. Here, the denoised signal (orange curve) follows the noisy signal (blue curve) with higher fidelity compared to EEG-GAN. Importantly, the WGAN-EEG model avoids overfitting, managing to generalize well even on new data channels, thus demonstrating robustness. The reduction of noise is evident without substantial distortion of the underlying signal patterns, particularly in the lower amplitude regions.

Subplot (d) returns to the EEG-GAN model on another dataset, showcasing its capacity for handling noisy signals. While the denoised signal is again effective in removing noise, similar to (a), there is a noticeable drop in sensitivity to minor signal fluctuations. For instance, the sharp spike near sample 100 in the noisy signal is significantly flattened in the denoised output. This behavior reinforces the observation that EEG-GAN tends to emphasize larger, slower components at the cost of some fine-grained details.

5.3. Classification results

Table 1 summarizes the evaluation metrics for the signals reconstructed by the EEG-GAN and WGAN-EEG models, providing a comparison based on Spectral RRMSE, Temporal RRMSE, and Signal-to-Noise Ratio (SNR).

Spectral RRMSE (Relative Root Mean Square Error): This metric evaluates how well the models maintain the spectral properties of the original signals. Lower values indicate better preservation of frequency-domain information. We observe that WGAN-EEG (Figures a and b) consistently achieves lower Spectral RRMSE values compared to EEG-GAN (Figures c and d), particularly in Figure b where the value is 0.000977, the lowest among the four signals. This suggests that WGAN-EEG is more effective at retaining the spectral structure of the original signals.

Temporal RRMSE: This metric assesses the reconstruction accuracy in the time domain. Similar to the spectral RRMSE, lower values reflect better performance. The WGAN-EEG reconstruction in Figure b again outperforms the other models, with a Temporal RRMSE of 0.245178, indicating that this model excels not only in frequency preservation but also in temporal accuracy. On the other hand, EEG-GAN reconstructions (Figures c and d) have higher Temporal RRMSE values, suggesting slightly inferior performance in capturing the time-domain features of the signal.

Signal-to-Noise Ratio (SNR): SNR measures the strength of the signal relative to the background noise. Higher SNR indicates a cleaner, more accurate reconstruction. WGAN-EEG in Figure b delivers the best per-

Table 2
EEG-GAN/WGAN Signal Evaluation

Signal	Spectral RRMSE	Temporal RRMSE	SNR
Fig a	0.000920	0.289406	13.604142
Fig b	0.000977	0.245178	14.957250
Fig c	0.001257	0.329908	12.297100
Fig d	0.001515	0.351424	10.297100

formance with an SNR of 14.957250, reinforcing the observation that this model achieves superior denoising while maintaining signal integrity. In contrast, the EEG-GAN model, particularly in Figure d, produces lower SNR (10.297100), showing that it introduces more residual noise compared to WGAN-EEG.

6. Conclusion

In this paper, we have proposed a contemporary deep neural network that uses a GAN/WGAN framework with a MSE/BCE/KL-Divergence based loss function for EEG signal reconstruction. Both models show promising results in denoising EEG signals, but they differ in how they treat signal complexity. EEG-GAN appears to prioritize smoothness, likely making it a good candidate for applications where the broader oscillatory structure is more important than fine-grained details. On the other hand, WGAN-EEG maintains higher fidelity to the original noisy signal and is able to retain smaller-scale variations, suggesting it might be better suited for applications requiring the preservation of rapid signal changes or high-frequency information. This analysis reveals that model selection should be based on the specific characteristics of the signals and the requirements of the downstream tasks.

Acknowledgments

I would like to express my sincere gratitude to Professor Christian Napoli and his assistant Imad Eddine Tibermacine for their invaluable support and guidance throughout the development of this project. Their insightful feedback, encouragement, and expertise were instrumental in the successful completion of this work. Their dedication and commitment to excellence have greatly inspired and motivated me. Thank you for providing the opportunity to work on this project and for all the assistance provided along the way.

References

A. Online Resources

The sources for this project are available via:

- GitHub,
- Dataset