Conditional Generative Adversarial Network for Motor Imagery EEG Data

Umair Arshad
19/04

BSc Data Science & Computing Project Report, Birkbeck College, University of London, 2024

Abstract

This project explores the potential of Conditional Generative Adversarial Networks (CGANs) to generate synthetic EEG data for motor imagery tasks, aimed at enhancing the data availability and improving the performance of EEG-based brain-computer interfaces (eBCIs). Addressing the challenges of high variability and low signal-to-noise ratio inherent in EEG data, various GAN architectures were evaluated, with a focus on the Wasserstein GAN with gradient penalty (WGAN-GP) for its ability to stabilise training and ensure diverse data generation. The developed conditional WGAN effectively replicated key EEG signal features, demonstrating the model's capability to capture both general and channel-specific patterns across different sessions and subjects. However, maintaining the fidelity of the generated signals consistently posed significant challenges. The findings underscore the promise of using conditional WGANs in synthetic EEG data generation, highlighting both achievements and limitations, and suggesting avenues for future research to optimise and expand the utility of conditional WGANs in non-invasive neurotechnology applications.

Contents

Section 1

Introduction

1.1 Project objectives and scope

This project aims to deploy a combination of the promising techniques and methods discussed in the literature review to generate synthetic EEG data for MI tasks. This objective is achieved through the implementation of a GAN conditioned with the CSP features of each trial of the original dataset. The high variability of EEG data is one of the key challenges to the implementation of eBCIs, particularly when attempting to classify data the classification algorithm hasn't been trained on. Good performance in cross-session classification is necessary for implementation of eBCIs in patients' day-to-day life, where conditions are naturally even more variable than in the lab. CSP features may provide a means to highlight the differentiating spatial characteristics of this variable signal. A number of different GAN architectures were evaluated during the project development phase, with a Wasserstein GAN with gradient penalty found to produce the best results.

While the eventual goal of synthetic EEG data generation is to improve the classification accuracy in eBCIs through improving data availability, the optimisation of a GAN to achieve this overarching objective and demonstration of classification performance improvements were not possible within the timeframe of this project. Nonetheless, it is demonstrated that a conditional GAN developed and tested over only a few months can achieve promising results, producing EEG signals which reproduce many of the characteristic features of the original data. Directions for future work are also identified.

Section 2

Literature Review

Decoding brain signal to translate into function has been a commendable pursuit of scientists and researchers ever since we learned to record brain activity. Brain computer interfaces (BCI) provide a means of bi-directional communication between human brain and external devices. Be it using brain signal to operate a tablet [1], type messages by imagining [2] or control mobility vehicles [3], BCIs provide a promising means for greatly improving patients' life quality. Intracortical BCIs using multielectrode arrays permanently implanted in the patient's head have provided a means to reconnect to the world for patients with locked-in syndrome [1] or movement to people affected by motor neuron diseases (MNDs) [3]. But these devices require intricate surgery which comes with its own risks and can sometimes lead to problems associated with infection and tissue scarring in the medium to long term. Other problems, such as signal degradation and hardware failure, incur further inconvenience and dangers, not to mention the limitations imposed on the patient having a sensitive device permanently attached to their skull. Alternative, less invasive data collection techniques have therefore been of interest to BCI researchers.

2.1 Use of EEG data in BCIs

Electroencephalography (EEG) based BCIs (eBCIs) use electrodes placed on the scalp to acquire electrical signal from brain activity without the need to rely on the muscular system and no implantation required. This non-invasive nature along with high temporal resolution and relative cost-effectiveness makes eBCI a popular choice among other brain imaging techniques for many BCI applications [4].

While easier to use and less invasive, eBCIs come with their own set of challenges, largely owing to the complexity and low signal-to-noise ratio (SNR) of EEG. eBCIs work by measuring the electrical activity generated by firing of the neurons and translating these signals

into commands to control external devices or applications. Whereas signal acquisition is easier for EEG, processing and decoding EEG data to translate into action still constitutes a formidable challenge. The non-stationarity of EEG data means the data recorded from session to session and subject to subject has very high variability in distribution. Changes in signal are commonly caused by slight differences in the alignment of electrodes on the scalp, but also by fluctuations in the subject's mood, thoughts, surrounding environment and many other factors. The combination of these complications makes training the classification algorithms within eBCI a particularly difficult task. The collection of a typical training dataset requires hundreds of trials over a number of sessions and days. These long and tedious sessions of performing repetitive tasks can result in fatigue and frustration in able-bodied subjects at the best of times, let alone patients with disabilities and limitations.

Among the various paradigms, Motor Imagery (MI) BCIs are considered the most prominent. Motor Imagery is the cognitive task of imagining the movement of a limb without actually moving it. Imagining or intending a movement show similar neural structures and physiological correlates [5]. Therefore, the neuronal activity observed during imagining performing a task can be translated into action by decoding the signal. During the training phase of these BCIs, the user is directed by way a visual or auditory cue, to imagine the limb movement and perform the desired action, often grasping an object or simply moving a hand. MI-BCIs have been used in a number of applications such as prosthetic limbs, controlling wheelchairs and neurorehabilitation, making them one of the most impactful technologies for restoring functions for people with lost limbs or neurodegenerative disorders.

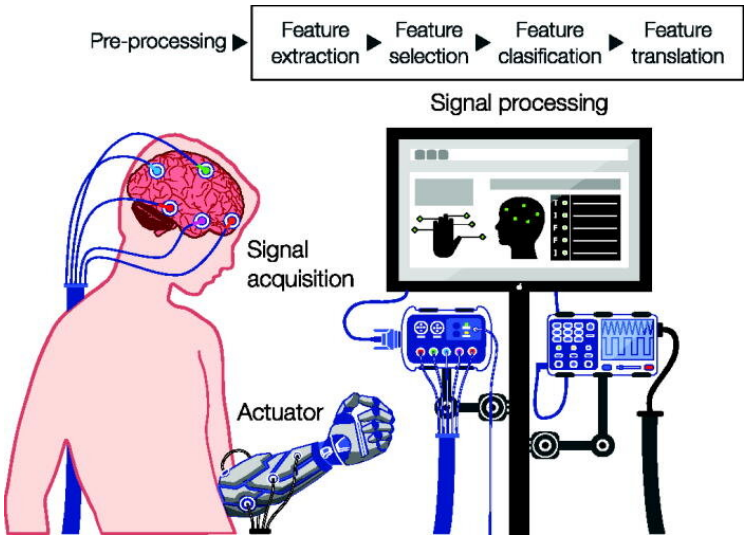

Figure 2.1: The architecture of an EEG-based brain–computer interface [4].

2.2 Challenges of Motor Imagery eBCI development

Similarly to eBCIs in general, MI-eBCIs face a number of challenges that can be broken down into a few categories [6]; low SNR, overlapped brain activity, input formulation, inter-subject/session & cross-subject/session variability and limited datasets.

Given the utility of eBCIs, addressing these limitations and challenges has been an active area of research over the last two decades. A number of approaches have been proposed addressing different problems with varying results. Some researchers have proposed approaching cross-session adaptation as a domain adaptation problem (or transfer learning) [7] and using subject-specific channel selection and channel-specific transformations [8] to address the variability problem. Others have suggested the SNR could be improved using advanced machine learning techniques [9].

Among all challenges, the lack of reliable data for training classification algorithms in eBCIs remains an important limiting factor. While the initial barrier to data collection is significantly lower than for more invasive methods, the acquired data is often inconsistent due to the long training sessions and natural variability, which translates to poor offline training of the classification algorithms resulting in poor performance in online use.

A number of techniques have been used to overcome the difficulties related with processing and decoding EEG data such as noise addition, signal amplification and filtering variation. None have proved to be sufficient in dramatically improving the performance of these devices in practical use. One of the more promising avenues has been data augmentation which has been shown to improve classification accuracy of many algorithms [10]. Data augmentation techniques within image data for example include colour and/or feature space augmentation and mixing images etc.

2.3 Generative Adversarial Networks (GANs)

Due to the developments in deep learning in recent years, more advanced techniques have been used to augment datasets to improve quality and/or diversity of the data. In domains such as biomedical imaging, data augmentation can help in increasing the size of difficult to obtain datasets. One of the most prominent data augmentation techniques involves the use of Generative Adversarial Networks (GANs). GANs, as originally proposed by Ian Goodfellow in 2014 [11], operate through the simultaneous training of two multi-layer perceptrons: a generator (G) and a discriminator (D). The generator aims to create data that is indistinguishable from real data, thereby "fooling" the discriminator. It does this by maximising the probability that the discriminator incorrectly classifies the generated data as real, expressed as $\max_G \log D(G(z))$, where $G(z)$ is the data generated from noise

input z, and $D(G(z))$ is the probability that the discriminator assigns to the generated data being real.

The discriminator, on the other hand, seeks to accurately distinguish between real data and fake data generated by the generator. Its objective is expressed as $\max_D[\log D(x) + \log(1 - D(G(z)))]$, where x is real data and $D(x)$ is the probability assigned by the discriminator to x being real.

The combined objective function for a GAN is a min-max game, which can be written as:

$$\begin{aligned} \min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_x}[\log D(x)] \\ + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \end{aligned} \tag{2.1}$$

$V(D, G)$ is the value function that both the discriminator and generator aim to optimise, where p_x is the distribution of real data, and $p_z(z)$ is the generator's noise distribution. Through this adversarial training process, the generator improves its ability to produce data that closely mimics real data, effectively "fooling" the discriminator with greater consistency over time.

Though the main principle behind GANs is relatively simple, they are notoriously difficult to train. The non-convex optimisation problem often leads to instability [12, 13], where the model parameters oscillate, cycle, or diverge rather than converge to a stable solution. This is complicated by the fact that the Hessian of the loss function, a matrix representing the curvature via second-order partial derivatives, becomes indefinite in GANs. Consequently, the optimal solution often involves finding a saddle point rather than a local minimum [12]. The generator frequently learns to produce a limited variety of outputs—sometimes even a single output—rather than a diverse range of data resembling the training set, a problem often referred to as "mode collapse," where the generator collapses to a state of generating the same point or a few points repeatedly. Achieving the delicate balance required to train both the generator and discriminator simultaneously is usually challenging. More often than not, the discriminator learns to distinguish between real and fake data very quickly, overpowering the generator and leading to a vanishing gradient where the discriminator fails to provide useful gradients for the generator to "learn" from. GANs are also highly sensitive to the choice of hyperparameters such as learning rates and batch sizes, as well as the architecture of both networks. Slight changes in these parameters can lead to vastly different training outcomes.

2.4 GANs in BCIs

In the domain of eBCIs, GANs have presented promising solutions to a number of problems. Brophy et al. [9] proposed a novel approach for denoising EEG signal. They trained a GAN consisting of a Long Short Term Memory (LSTM) network as the generator and a 1-dimensional Convolutional Neural Network (CNN) as the discriminator, to map noisy signal to clean EEG signal, effectively removing contamination introduced by muscle movement and electrical interference. Aznan et al. [14] employed a neural-based GAN and Variational Auto-Encoder (VAE) to successfully generate synthetic EEG signal vectors that improved classification performance. The result was particularly significant in cross-subject generalisation, demonstrating over a 35% improvement. Luo et al. [15] presented an EEG signal reconstruction method using a GAN with Wasserstein distance (WGAN). Their approach tackled the high cost and complexity of EEG hardware by enabling the reconstruction of high-quality EEG signals from lower quality ones.

As evidenced by extensive literature, GANs hold promise in a number of avenues. However, the training of GANs, particularly ensuring the quality and stability of generated data, presents inherent challenges. Stability in training is often addressed by enforcing a Lipschitz constraint, which limits how rapidly functions within the network can change, helping prevent issues like exploding or vanishing gradients.

A function $f : X \to Y$ is Lipschitz continuous if, for a constant $L \geq 0$ and all points x_1, x_2 in the domain X, the following condition holds:

$$\|f(x_1) - f(x_2)\| \leq L\|x_1 - x_2\| \tag{2.2}$$

In neural networks generally, enforcing this constraint helps stabilise the training process.

Traditional GANs measure the similarity between the distribution of generated data and real data using the Jensen-Shannon divergence (JSD), a symmetric and smoothed version of the Kullback-Leibler divergence (KLD);

$$JSD(P\|Q) = \frac{1}{2}KLD(P\|M) + \frac{1}{2}KLD(Q\|M) \tag{2.3}$$

$$\text{where } M = \frac{1}{2}(P + Q) \tag{2.4}$$

is the average of the two distributions, and $KLD(P||Q)$ is the Kullback-Leibler divergence between P and Q, defined as

$$KLD(P\|Q) = \sum_x P(x) \log\left(\frac{P(x)}{Q(x)}\right) \tag{2.5}$$

As discussed, the discriminator in a GAN is trained to maximise the log-likelihood for estimating whether the given sample is real or "fake" and the generator is trained to minimise the log-likelihood of the discriminator being correct. In the original GAN architecture, JSD is used as part of the loss function to measure the distance between the original data and the generated data. As is often the case, the discriminator becomes too good too quickly, which results in JSD approaching its maximum value since the two distributions are easily distinguishable. The discriminator's outputs for the generated data become close to zero, indicating high confidence, leading to a large negative JSD (log of a value close to zero) and a very small gradient for updating the generator's weights, a problem known as "vanishing" gradients. The vanishing gradient in vanilla GANs leads to training instability and eventually mode collapse where the generator produces a limited type of outputs.

2.4.1 WGAN and Gradient Penalty

To overcome these fundamental issues with GAN training, Arjovsky et al. [16] proposed "Wasserstein GAN" , using Wasserstein distance (or earth mover's distance), as a loss function for enforcing the Lipschitz constraint which offers more stable and reliable training dynamic between the two networks. The Wasserstein distance is another measure of distance between two probability distributions; the minimum cost of transporting "mass" in converting one distribution into the other. For two probability distributions P and Q defined on space X, the Wasserstein distance is defined as

$$W(P,Q) = \inf_{\gamma \in \Pi(P,Q)} \mathbb{E}_{(x,y)\sim\gamma}[\|x-y\|] \tag{2.6}$$

where $\Pi(P,Q)$ denotes the set of all joint distributions $\gamma(x,y)$ whose marginals are respectively P and Q. The expectation $E_{(x,y)\sim\gamma}[\|x-y\|]$ computes the average distance the mass must be transported to transform P into Q. Since the infimum is generally intractable [16] and cannot be computed directly, the Kantorovich-Rubinstein duality is used, which provides a way to compute the Wasserstein distance using a parameterised family of functions:

$$W(P,Q) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x\sim P}[f(x)] - \mathbb{E}_{x\sim Q}[f(x)] \tag{2.7}$$

where the supremum is taken over all 1-Lipschitz functions $f : X \to \mathbb{R}$, which are functions whose gradients do not exceed 1. The WGAN's discriminator, instead called "critic", is

designed to approximate this supremum.

The critic effectively becomes a function that measures the Wasserstein distance, instead of classifying inputs as real or fake, scoring them with the goal of satisfying the Lipschitz constraint. To enforce the Lipschitz condition, the original WGAN paper proposed clipping the weights of the critic to a fixed box (e.g., [-0.01, 0.01]) after each gradient update.

Weight clipping can be a rather crude way to enforce the Lipschtiz constraint, leading to convergence on suboptimal solutions [16]. Clipping the weights can lead to extreme gradients when the weights are at the boundary of the clipping range, causing optimisation to bounce around the boundaries, resulting in erratic updates and training instability. Moreover, choosing the correct range for clipping weights can be challenging. Clipping the weights to a small fixed range can lead to underuse of the critic's capacity, i.e. enforcing the weights to lie within a very narrow range, makes the critic's function overly simplistic which limits its ability to learn complex patterns in the data. Conversely, too wide a range, makes the critic's function too complex and violates the Lipschitz constraint, potentially leading to unstable and divergent behaviour.

Gradient Penalty To address these issues, Gulrajani et al. [17] proposed penalising the norm of the gradient of the critic with respect to its inputs. This gradient penalty is added to the loss function to enforce the Lipschitz constraint, instead of weight clipping, which was shown to promote more stable training.

Given real data x and generated data $\hat{x}$, an interpolated sample $\tilde{x}$ is constructed as:

$$\tilde{x} = \epsilon x + (1 - \epsilon)\hat{x} \tag{2.8}$$

where $\epsilon \sim U[0, 1]$.

The gradient penalty (GP) is then computed as:

$$GP = \lambda \left(\|\nabla_{\tilde{x}} D(\tilde{x})\|_2 - 1 \right)^2 \tag{2.9}$$

where λ is the penalty coefficient, $\nabla_{\tilde{x}}$ denotes the gradient with respect to $\tilde{x}$, $D(\tilde{x})$ is the discriminator's output for $\tilde{x}$, and $\|\cdot\|_2$ represents the L2 norm.

The loss function for WGAN-GP is updated as

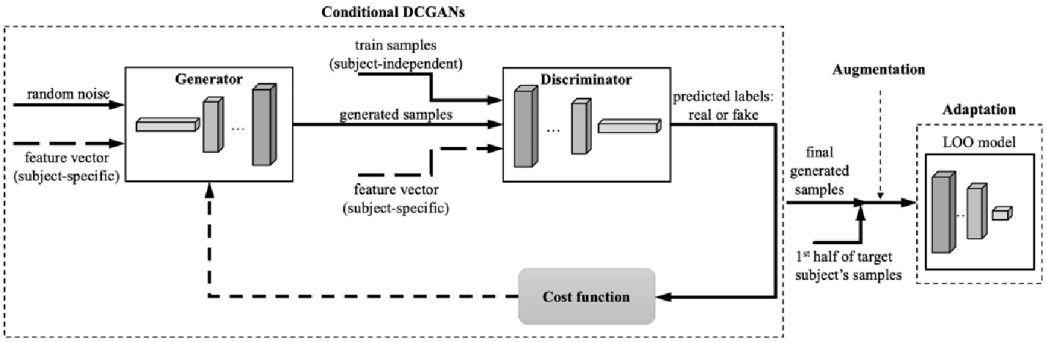

Figure 2.2: DCGAN architecture

$$\mathcal{L} = \underbrace{\mathbb{E}_{x\sim\mathbb{P}_r}[D(x)] - \mathbb{E}_{\tilde{x}\sim\mathbb{P}_g}[D(\tilde{x})]}_{\text{Critic's Wasserstein loss}} + \underbrace{\lambda\,\mathbb{E}_{\hat{x}\sim\mathbb{P}_{\hat{x}}}\left[\left(\|\nabla_{\hat{x}}D(\hat{x})\|_2 - 1\right)^2\right]}_{\text{Gradient penalty}} \tag{2.10}$$

where $\mathbb{P}_r$ is the real data distribution, $\mathbb{P}_g$ is the generator's data distribution, D is the critic, $\tilde{x}$ are samples from the generator, $\hat{x}$ are sampled uniformly along straight lines between pairs of points sampled from the real data distribution $\mathbb{P}_r$ and the generator's data distribution $\mathbb{P}_g$, λ is the penalty coefficient (a hyperparameter), $\nabla_{\hat{x}}D(\hat{x})$ is the gradient of the critic's output with respect to $\hat{x}$.

The gradient penalty term enforces the Lipschitz constraint by penalising the deviation of the gradient norm of the critic from 1, thus avoiding the exploding or vanishing gradient problems. The stable training means that the critic can be trained optimally, providing more useful losses to the generator which in turn gets better training. [paper] showed that using the proposed method, the generator and the discriminator no longer needed "balanced capacity". The better critic provided higher quality gradients to be used to train the generator. The authors reported seeing no evidence of mode collapse in WGAN during experimentation.

2.4.2 CGAN

Augmenting MI data using GANs with convolutional layers (CGANs) has been a popular research area. Habashi et al. [18] addressed the challenge of limited data availability by using CGANs to generate synthetic EEG spectrum images, which were then used to augment

the training data for CNN classifiers. Their method demonstrated that the synthetic EEG images possess temporal, spectral, and spatial characteristics akin to actual EEG data. By incorporating these synthetic images into the training dataset, the classification accuracy of MI tasks was enhanced by 2.5%, achieving an average accuracy of 76.71%. Although the study showed some improvement in the classification accuracy on the BCI Competition IV dataset, since this is the only dataset used in the study and given the variability of EEG data, the generalisability of the proposed approach remains unknown. The authors used EEG spectrum images converted to gray-scale, to limit computational cost. Combining the three channels into one introduces the potential loss of spatial information and the gray-scale conversion means loss of phase information and power distribution across bands. Finally, the inherent trade-off between time and frequency resolution in Short Time Fourier Transform (STFT), used to convert EEG segments to 2D spectrum images, could result in the loss of finer spectral details, given the rapid changes in EEG signal.

2.4.3 CS-GAN

Cross-subject variability poses another important challenge in eBCIs due to the previously discussed non-stationarity of EEG. Song et al. introduced CS-GAN in [19], aiming to highlight the key temporal and spatial characteristics in the variable EEG data. To achieve this, they incorporated Common Spatial Patterns (CSP) features in the GAN architecture to better differentiate between motor imagery tasks, thus enhancing cross-subject classification accuracy. CSP, widely used in BCIs for extracting salient signal features, computes the signal covariance matrices for two distinct tasks, finding spatial filters that maximise variance for one task while minimising it for the other. The process effectively emphasises the class-discriminative signal characteristics, improving accuracy of classification algorithms in BCI applications.

CSP Let X_A and X_B be the EEG data matrices for conditions A and B. The covariance matrix for each condition is given by

$$\begin{aligned} C_A &= \frac{1}{N_A - 1} X_A X_A^T \\ C_B &= \frac{1}{N_B - 1} X_B X_B^T \end{aligned} \tag{2.11}$$

where N_A and N_B are the number of time samples in each condition and X_A^T and X_B^T are the transposes of X_A and X_B respectively. An average of the individual convariances adjusted by the respective sample size then gives the composite matrix

$$C_{\text{composite}} = \frac{N_A \cdot C_A + N_B \cdot C_B}{N_A + N_B} \tag{2.12}$$

Eigenvalue decomposition finds the set of eigenvectors and eigenvalues that represent the principal components of the combined EEG data

$$C_{\text{composite}} = U\Lambda U^T \tag{2.13}$$

The eigenvectors that correspond to the largest (and smallest) eigenvalues are the spatial filters which project the EEG data from the original space to a new space where the variance between A and B is maximised (or minimised). Applying these filters to the EEG data gives a new time series X with maximally discriminative conditions. The log-variance of this time series is the feature vector extracted for each channel.

$$features = \log\left(\text{var}(X_{\text{new}})\right) \tag{2.14}$$

Song et al. integrated CSP into the GAN framework [19], with CSP-derived spatial filters informing the discriminator as part of the input, ensuring that the generated synthetic EEG maintains essential spatial features. Since CSP features are usually used to classify two classes, for their multi-classification task, the authors adapted a modified one-versus-rest (OVR) strategy by posing the problem as multiple bi-classification tasks between one class and all remaining classes. Their results showed improved classification accuracy, demonstrating CSP GAN's potential in reducing BCI system calibration needs. With adaptive training and sufficient augmented data, on cross-subject classification task, their method yielded an improvement of 15.85% over leave-one-out (LOO) test and 8.57% over adaptation of 100 original samples on the BCIC IV dataset 2a. However, the study's reliance on BCI competition dataset and the risk of losing detailed signal information through their processing approach highlights the need for broader validation and optimisation.

2.4.4 DCGAN

The training phase of eBCI usually takes place in the lab under perfect conditions where the subject can maintain the focus required for the duration of training. In real-world use of the said BCI though, this is unlikely to be the case. Fahimi et al. [20] argued this contributes to the performance mismatch often seen in eBCIs when put to test in real world applications. In their study [20], they proposed an end-to-end Deep Convolutional GAN (DCGAN) conditioned on features extracted from the real data. Conditional GAN (cGAN), first introduced by Mirza and Osindero in 2014 [21], adds a condition in the GAN architecture (e.g. class labels as in the case of the original study) to guide the generation process and produce specific types of outputs. In this modified network, the generator G takes conditional information (labels or features) along with random noise and generates data corresponding to the given condition. The objective function (1) for cGAN is updated

as

$$\min_G \max_D V(G,D) = \mathbb{E}_{x\sim p_x}[\log(D(x|y))] + \mathbb{E}_{z\sim p_z}[\log(1-D(G(z,y)|y))] \tag{2.15}$$

and the updated loss functions are

$$Loss_D = -\log D(x_r|y) - \log(1-D(x_g|y)) \tag{2.16}$$

$$Loss_G = -\log D(x_g|y) \tag{2.17}$$

where $E_{x\sim p_x}$ is the expected log probability that x belongs to p_x given y and $E_{x\sim p_z}$ is the expected log probability that the discriminator correctly identifies generated data $G(z,y)$ given y.

Fahimi et al. [20] trained an end-to-end DCNN to classify the EEG data into MI and rest under two conditions, focused or diverted. The network was trained on data from all but the target subject and then used to process a subset of the data from the target subject (LOO). The output from the first fully connected layer, which formed the basis of classification, served as the feature vector for the target subject in their GANs. Two separate DCGANs were trained using the features from each condition as part of the generator and the discriminator's input, conditional part of the cGAN.

Using LOO, it is reported, they achieved baseline accuracies of 73.04% for diverted attention and 80.09% for focused attention conditions. Using the data generated by the DCGANs to augment the dataset achieved a significant improvement in accuracy. For diverted attention, the study reported an improvement of 7.32% (reaching 80.36%), and 5.45% (reaching 85.54%) for focused attention.

While the results are promising and provide a strong foundation for further development, some potential points of concern must be noted. First and foremost, using a black-box approach to extract features to be used as the condition for the cGAN, presents a fundamental interpretability issue. Lack of deeper insight into the learned features could hinder further research and development. The study set out to propose a method to improve cross-subject classification to aid adaptation of BCIs, tailoring the system without an understanding of the salient features, may thus prove counterproductive.

Section 3

Methodology

3.1 GAN architectures

GAN The "vanilla" GAN model served as a foundational architecture with 11 generator and 7 discriminator linear layers followed by batch normalisation layers. The generator expands the noise vector through a series (5) of linear transformations, starting at (noise dimension + features dimension) nodes, each followed by LeakyReLU activations, until reaching 1024 input nodes and channels x time steps output nodes, followed by a Tanh activation. The discriminator, starts with (channels × time steps × features) input nodes and 512 output nodes, reducing at each layer until reaching 256 input and 1 output node for the final layer. A sigmoid activation function is applied at the end to force the output to take values between 0 and 1.

Wasserstein GAN with Gradient Penalty (WGAN-GP) The WGAN-GP featuring 16 generator and 7 discriminator layers, with linear and batch normalisation layers

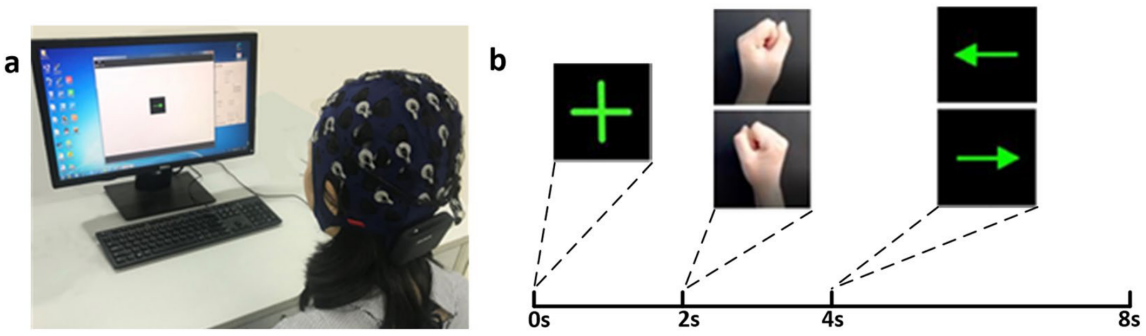

Figure 3.1: Data acquisition scenario. (a) The experimental scene, (b) Motor imagery experiment procedure.

was developed to stabilise the training. The layer count was increased in the generator to enable the learning of more complex patterns. The generator's input of the noise vector and CSP features, is passed through multiple fully connected layers with LeakyReLU activations before finishing with a Tanh function. The critic is implemented without the final sigmoid activation function since, unlike in standard GANs, the output does not require normalisation in the WGAN paradigm. This is because the "realness" of the data is evaluated based on Wasserstein distance, rather than being assigned a value between 0 and 1. The critic's performance was also significantly stabilised by a gradient penalty regulated by lambda value ($\lambda = 0.1 - 1$).

DC-GAN The deep convolutional GAN architecture used in this project consisted of 11 generator and 16 discriminator layers, including 1D convolutional, linear, and upsampling layers, with batch normalisation to capture the spatial and temporal features within the EEG data. Although 2D convolutional layers are more common in GAN architectures than 1D, this is because usually processed data takes form of images. 1D convolutional layers are sufficient to process time series data such as EEG. Dropout layers were introduced in the discriminator for regularisation. The architecture employed both ReLU and LeakyReLU activations, with the generator utilising a Tanh function and the discriminator using both Tanh and sigmoid functions to finalise their outputs.

DC-WGAN-GP The most complex model, the DC-WGAN-GP, is built upon the DC-GAN architecture with convolutional, upsampling, and batch normalisation layers in both its 11-layer generator and 16-layer discriminator. Wasserstein loss with gradient penalty helped improve the training dynamics.

Activation Functions Considering the subtle yet crucial variations in EEG signal and review of the existing literature, LeakyReLU was chosen as the activation function. The slightly negative slope (0.02-0.2) helped with convergence in most cases. Some initial experimentation with ReLU did not yield satisfactory results, hence LeakyReLU with varying slope values was used for fine tuning in most cases.

Considering that the normalised EEG data falls between $[-1, 1]$, for the output layers of the generator, the hyperbolic tangent function (tanh) was used in all generators. This once again was based on the available literature and some experimentation at early stages.

Table 1 provides as overview of the architectures.

Table 3.1: GAN Architectures

Name	**Types of Layers**	**Number of Layers (G/D)**	**Activation Functions**	**Loss Functions**	**Gradient Penalty**
GAN	Linear, BatchNorm	11/7	LeakyReLU, Sigmoid, Tanh	Binary Cross-Entropy	-
WGAN-GP	Linear, BatchNorm	16/7	LeakyReLU, Tanh	Wasserstein with gradient penalty	$\lambda = 0.1 - 1$
DC-GAN	Conv1d, Linear, BatchNorm, Upsample, Flatten	11/16	ReLU, LeakyReLU, Sigmoid, Tanh	Binary Cross-Entropy	-
DC-WGAN-GP	Conv1d, Linear, BatchNorm, Upsample, Flatten	11/16	LeakyReLU, Tanh	Wasserstein with gradient penalty	$\lambda = 0.1 - 1$

3.2 Dataset

Description The dataset used for this project was collected by Ma et al. [22] who set out to fulfil the need for such datasets for MI BCI research on cross-session and cross-subject variability in EEG data. They kindly made the dataset available under Open Access for public use. The dataset was published in Nature and is available for download via Figshare [23].

Experiment Design The dataset includes EEG recordings from 25 healthy subjects (12 females and 13 males, aged 20-24), collected over 5 independent sessions spanning 4-5 days. Each session originally included 100 trials (a small proportion of these was removed due to "exceptions", with no further clarification provided) of left-hand and right-hand MI tasks, amounting to a total of 12,500 (11,988 after removal) trials across the dataset.

The experiment focused on two types of motor imagery tasks: imagining left-hand and right-hand grasping movements. These tasks are commonly used in BCI research due to their distinct neural activation patterns [24]. Each trial began with a fixation cross displayed on the monitor to alert the subject and ensure their focus. Subsequently, an arrow indicating left or right appeared on the screen to inform the subjects about the type of hand movement they were to imagine. Following the cue, subjects were required to vividly imagine the movement of the specified hand grasping an object. No actual physical move-

ment was performed. Each imagery task lasted for 7.5 seconds per trial.

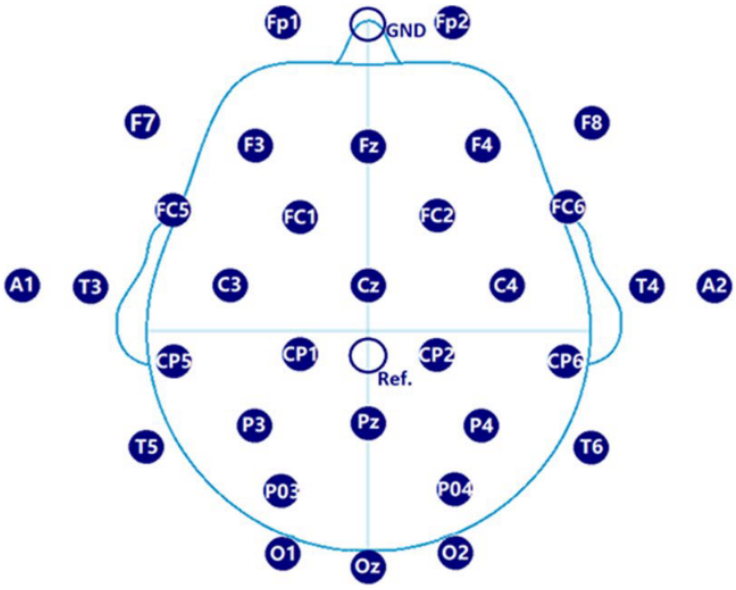

Figure 3.2: EEG cap electrode distribution. [22]

Data Acquisition and Preprocessing The data were captured using a 32-channel Ag/AgCl electrode cap, using the international 10-10 system placement (see Figure 3.2). The recording apparatus included a wireless amplifier capable of real-time impedance monitoring, with electrode impedances maintained below 20 $K\Omega$. The EEG signals were sampled at 250 Hz, with a resolution of 24 bits per sample.

Raw EEG data were filtered using a finite impulse response (FIR) band-pass filter between 0.5 Hz and 40 Hz to exclude frequency components outside the typical EEG spectrum and to reduce noise. Following this, EEG data underwent baseline removal to correct for DC offsets and were then segmented into epochs corresponding to each motor imagery trial.

Bad segments were identified and removed based on amplitude thresholds exceeding 100 μV, as automatically marked by the EEGLAB toolbox in MATLAB. Further visual inspection by experienced researchers ensured the exclusion of any artefact-contaminated trials, such as those caused by eye blinks or muscle movements. Continuous EEG data were segmented according to the trials, each corresponding to the specific MI tasks.

The preprocessed data for each session were saved in '.edf' (European data format)

format, while trial labels (left-hand, right-hand imagery) were saved in '.mat' files [22].

Utilisation in Model Training The authors provided the raw as well as preprocessed data to allow researchers to use the dataset directly. Denoising and preprocessing EEG data was deemed beyond the scope of this project given the time constraint and domain specific expertise required, hence the professionally preprocessed dataset was used to maximise the value of the work completed here.

Channel Selection For MI tasks, the electrodes overlying or near the motor cortex are of particular importance. As such, only channels Cz, C3 and C4 were used for synthetic data generation due to their relevance for MI tasks [25] to reduce dimensionality. Channels C3 and C4 are located over the left and right hemisphere of the brain, respectively, above the motor cortex. C3 captures activity when imagining movement on the right side of the body (right hand in this case) and C4 captures the left side (left hand in this case). Cz is located over the midline at the central top of the head and is relevant for capturing bilateral motor functions.

Section 4

Implementation

4.1 System specifications

The experiments conducted in this project were performed on a MacBook equipped with the M2 Pro processor, with Metal Performance Shaders (MPS) support. The software environment was anchored by Python 3.11, with the following library support.

- PyTorch: Used for building and training the neural network models and support for tensor operations on GPU.
- Python MNE: EEG data processing and analysis
- Scikit-learn (SKlearn): Implementing machine learning algorithms, model fitting, data splitting, and performance evaluation
- NumPy: The foundational package for numerical computation
- Matplotlib: Generating visualisations of data and results
- SciPy: Advanced calculations and to support additional functionality in signal processing not covered by NumPy.
- Pandas: (Some) data manipulation and analysis.

4.2 Python Implementation

This section provides an overview of some of the main parts of the pipeline and architecture in Python.

Data Preprocess First of all, data_preprocess.py module loads and prepares the dataset for analysis and feeding to the rest of the GAN architecture. The functions are outlined below:

get_data() loads the EEG data and labels for the given subject and session from the .mat file. This is where the desired channels are selected (Cz, C3, an C4) and returned with the corresponding labels.

create_raw() takes the data, labels, and metadata information (info) to create a 'Raw' object that represents the raw EEG data in MNE. It transposes and reshapes the data array, creating an events array from the labels, and setting up the channel names and types. It sets the EEG montage (the layout of EEG electrode positions) to a standard 10-05 system and returns the raw data object along with the events array.

```
def create_raw(data, labels, info, freq=250):
    labels = np.ravel(labels)
    ch_names = info.ch_names
    si, sj, sk = data.shape
    da = data.transpose(1, 0, 2)
    da = da.reshape(sj, si * sk)
    llen = data.shape[0]
    event = np.zeros((llen, 3))
for i in range(llen):
    event[i, 0] = i * sk
    event[i, 2] = labels[i]
event = event.astype(int)

info = mne.create_info(ch_names=ch_names, ch_types="eeg", sfreq=freq)
raw = mne.io.RawArray(da, info)  # create raw object
montage = mne.channels.make_standard_montage('standard_1005')
raw.set_montage(montage)

return raw, event
```

mnebandFilter() performs bandpass filtering on the data. It creates a 'Raw' object by calling create_raw(), sets up event IDs for epoching the data and then creates Epochs by applying the bandpass filter using the specified low and highfrequency cutoffs. This filtered epoch data is returned.

```
def mnebandFilter(data, labels, lowfreq, highfreq, freq=250):
    info = mne.create_info(ch_names=ch_names, ch_types='eeg', sfreq=freq)
    raw, event = create_raw(data, labels, info, freq)
```

```
    raw.filter(lowfreq, highfreq, fir_design='firwin')
    epochs = mne.Epochs(raw, event, event_id, 0, 4 - 0.004, baseline=None, preload=True
    train_data = epochs.get_data()
    return train_data
```

Please note, these functions were provided by dataset's authors for the LDA classification pipeline but were adapted to work with the current setup.

CSP feature extraction $extract\_csp\_features()$ uses the CSP algorithm to extract features from the EEG data. Given the MI task this experiment was designed around, the frequency bands $3Hz35Hz$ are chosen as the filters due to their relevance to the task, the most relevant being $\mu 48 - 12Hz$ and $\beta 13 - 30Hz$. [26]

```
def extract_csp_features(sub_id, test_session, data_path):
    # Load and preprocess the data
    data, labels = get_data(sub_id, test_session, data_path)
    data = mnebandFilter(data, labels, 3, 35)
    print('Data shape: ', data.shape)

# Initialize CSP and extract features
csp = CSP(n_components=10, reg=None, log=False, norm_trace=False)
csp_features = csp.fit_transform(data, labels)

return csp_features
```

GAN architecture The *Generator* class from *GAN.py* defines the generator in the architecture. The constructor takes $noise_dim$, $feature_dim$, $channles$ and $time_steps$ as arguments which respectively determine the size of the random noise vector, dimensions of the feature vector (3 for 3 channels in this case), channels (3 relevant EEG channels) and the number of time steps in each output (1000 in all cases due to the length of the MI task sessions). The architecture is defined sequentially (nn.Sequential) as a series of linear layers and activation functions. The first layer (nn.Linear(noise_dim + feature_dim, 128)) takes the concatenated noise and feature vectors as input and upscales to a higher-dimensional space. Leaky ReLU activation (nn.LeakyReLU) and batch normalisation (nn.BatchNorm1d) are applied after each linear transformation respectively. The upscaling continues through larger linear layers until reaching the final output size (channels * time_steps). The final activation function, (nn.Tanh) scales the output to be between -1 and 1.

$initialise\_weights()$ initialises the weights of the linear layers using Xavier normal initialisation. Biases are initialised to zero if they are present.

The $forward()$ the noise and feature vectors are concatenated along the feature dimension, and the combined vector is passed through the model. The output of the model is then reshaped to the required output shape.

```
class Generator(nn.Module):
def __init__(self, noise_dim=100, feature_dim=3, channels=3, time_steps=1000):
    super(Generator, self).__init__()
    self.channels = channels
    self.time_steps = time_steps

    self.model = nn.Sequential(

        nn.Linear(noise_dim + feature_dim, 128),
        nn.LeakyReLU(0.02, inplace=True),
        nn.BatchNorm1d(128),

        nn.Linear(128, 256),
        nn.LeakyReLU(0.02, inplace=True),
        nn.BatchNorm1d(256),

        nn.Linear(256, 512),
        nn.LeakyReLU(0.02, inplace=True),
        nn.BatchNorm1d(512),

        nn.Linear(512, 1024),
        nn.LeakyReLU(0.02, inplace=True),
        nn.BatchNorm1d(1024),

        nn.Linear(1024, channels * time_steps),
        nn.Tanh())
def initialize_weights(self):
    for m in self.modules():
        if isinstance(m, nn.Linear):
            nn.init.xavier_normal_(m.weight)
            if m.bias is not None:
                nn.init.zeros_(m.bias)

def forward(self, noise, features):
    x = torch.cat((noise, features), dim=1)
    output = self.model(x)
    return output.view(-1, self.channels, self.time_steps)
```

Section 5

Development

This project investigates the potential of Conditional Generative Adversarial Networks (CGANs) to address key challenges in non-invasive EEG-based brain-computer interfaces. Due to the inherent complexity and low SNR of EEG data, decoding these signals accurately remains a formidable challenge. The aim is to build upon the work on CGANs to generate synthetic EEG data with "enhanced" CSP features, to improve classification in eBCIs. This approach is anticipated to enhance the performance of BCIs by improving the classification accuracy between motor imagery tasks in BCIs applications. To that end, ideally, the generated EEG data should exhibit high temporal fidelity and consistency, reflecting the dynamic changes in brain activity over time. Additionally, the synthetic data should closely mimic the statistical properties of real EEG recordings, such as amplitude distribution and spectral characteristics, ensuring that it is both realistic and varied enough to effectively train machine learning models without introducing bias or overfitting.

CSP helps to extract discriminative spatial patterns from EEG signals that are relevant to different tasks (or mental states) and may also capture the characteristics of the individual response in a given trial, providing a conditioned GAN with valuable information. The hypothesis posits that by generating synthetic data conditioned on CSP features, the GAN can effectively capture and reproduce the spatial patterns relevant to the MI tasks under consideration. As a result, the synthetic data generated by the GAN would exhibit similar spatial characteristics to the original EEG data, making it easier for classification algorithms to distinguish between the two MI tasks with the ultimate goal of improving the classification performance in eBCIs. While the latter may not be possible within the scope of this project, an initial evaluation of the classification performance with augmented data will be performed.

5.1 Experimentation with GAN architectures

The initial experiments were conducted using a simple GAN with fully connected layers (GAN.py). As anticipated from the literature, the stability of this model was low, and it failed to converge, leading to mostly unrecognisable outputs. Trial and error to modify hyperparameters such as learning rates$(0.0001 - 0.2)$, the first moment $(\beta_1 = 0.1 - 0.9)$ for the ADAM optimiser, as well as different training strategies and increased model complexity did not yield significant improvements. The discriminator consistently overpowered the generator, likely owing to the complexity and high dimensionality of the original data. Nevertheless, these efforts provided valuable insights into the GAN training process.

Subsequent experiments lead to the implementation of Wasserstein loss with weight

Figure 5.1: GAN training losses. With balanced training between the two networks, BCE Loss and learning rate of 0.0002 (both) for ADAM.

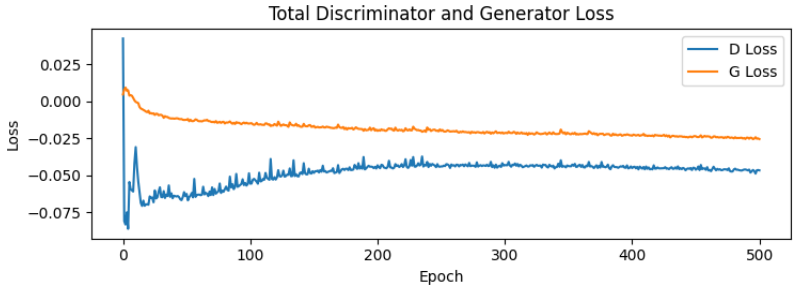

Figure 5.2: WGAN with weight clipping ($c = 0.5$), total training losses.The discriminator was trained 5x more often, BCE Loss, and learning rate of 0.0002 (both) for RMSProp ($\alpha = 0.9$).

clipping. This modification showed slight improvements in the stability of the training process but still failed to achieve consistent convergence. Given the impact of weight clipping on the discriminator's ability to guide the generator, a large range of values for the clipping parameter c, $(0.1-1)$ did not help balance the training dynamics. Following the

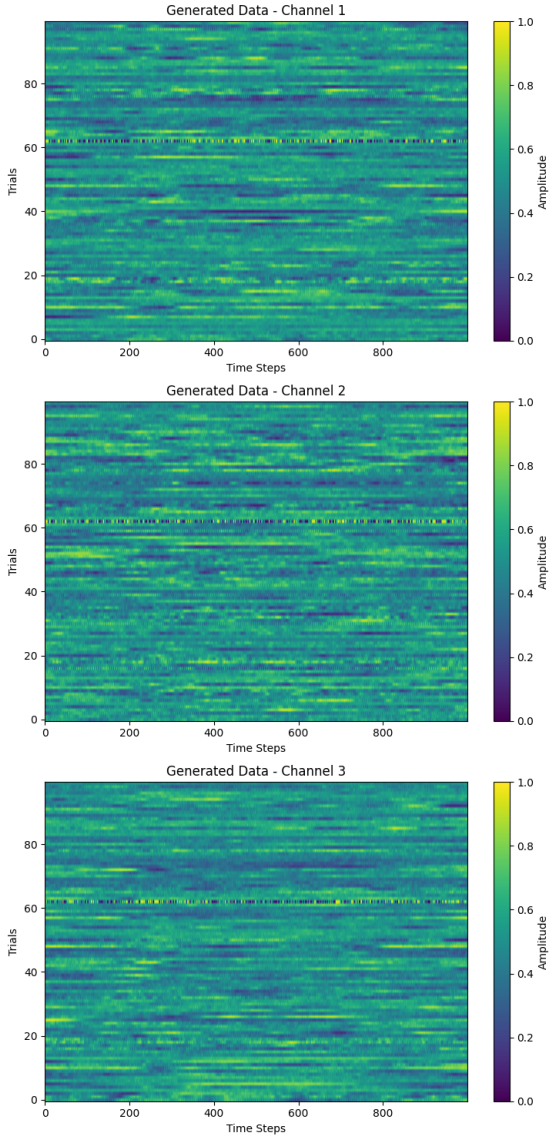

Figure 5.3: Unlikely similar patterns at the same time point and trial for all channels.

recommendation for RMSProp (Root Mean Square Propagation) optimiser in the original WGAN paper [16], this was chosen as the loss function, varying learning rates $(0.01 - 0.2)$ and $\alpha = (0.5 - 0.999)$ helped to stabilise the training but the model still consistently "converged" to less optimal solutions. The overall losses were now on a much smaller scale, hinting towards a step in the right direction for further development, data produced by the generator at this stage lacked variety. More crucially, the generated samples showed

peculiar behaviour. Fig 5.3 shows a sample taken at epoch 400 during training, notice the activity around trial 61/62 for all three channels. Given the task being carried out by the subjects, at any given trial, the subject would imagine either left hand or right hand movement, leading to different patterns being captured by each channel. The generated sample shows very similar patterns around this point, particularly the peaks around 700ms and 950ms.

To improve training and enhance stability further, a gradient penalty to the discriminator's loss was introduced to the WGAN architecture (WGAN.py). The implementation (detailed in section x) led to noticeable improvements in training stability and allowed the model to begin converging. The decision was made to switch to also switch to ADAM optimiser as per finding of Gulrajani in the original WGAN paper [17] and early experiments showing improved convergence (Fig 8). Same learning rate (0.0002) but different β_1 for ADAM optimiser, 0.2 for generator and 0.9 for discriminator provided a good balance between the two networks' training.

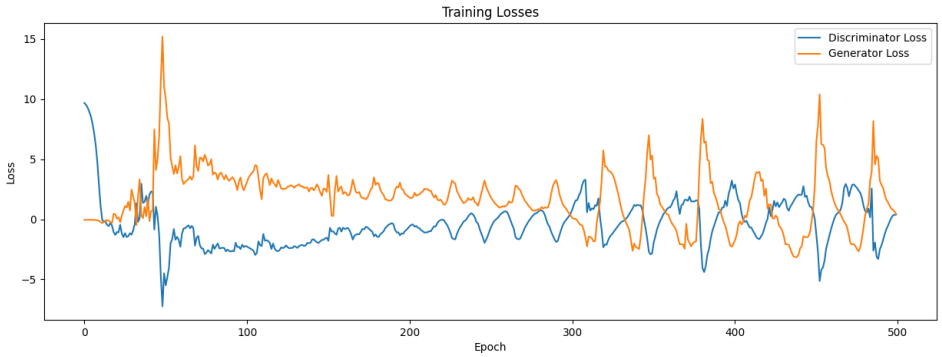

Figure 5.4: GP ($\lambda = 1$) and ADAM ($lr = 0.0002, \beta_1 = 0.2$ and 0.9 for generator and discriminator respectively) improved training dynamics showing balance between the two networks.

Further adaptations involved the incorporation of convolutional layers into the WGAN. The network was conditioned with features along the channel dimension. However, the convergence issues persisted, and the model still failed to produce outputs that were of practical use, lacking any spatial or temporal details.

An example of the EEG data is provided in Fig 5.5, plotted as a heat-map, for comparison with the generated data during training of the DCGAN with BCE loss in Fig 5.7.

The experimentation then shifted towards testing a Deep Convolutional GAN with Wasserstein loss and gradient penalty (DCWGAN-GP). While this model initially showed promise in terms of slow convergence, it ultimately did not generate usable data samples. The generated data lacked any detail (see Fig 5.8 even after long training sessions and experimentation with various hyperparameter values. Moreover, likely due to architecture

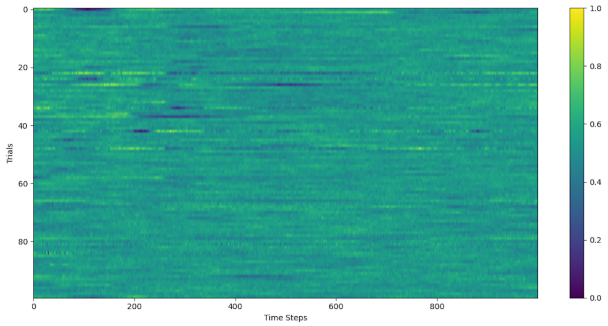

Figure 5.5: Recordings from C4 for subject 1 session 1 plotted as a heat-map with trials on the y-axis, time steps on the x-axis and colour representative of amplitude. Consistent patterns across trials can be be observed .

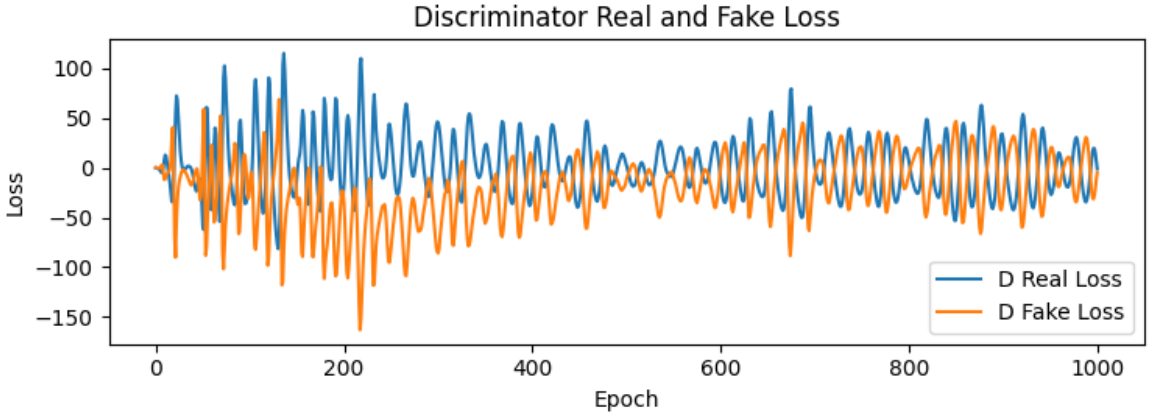

Figure 5.6: D's losses for original vs generated data, both oscillating around a small value. Notable that both losses trend together without one dominating significantly over the other.

design and gradient penalty, the model was the slowest of all with training for 500 epochs taking anything up to 3 hours even with GPU support.

After extensive testing and modifications across various models and configurations, the focus returned to the WGAN with a gradient penalty, which had shown the most consistent and promising results. Fig 5.9 shows losses for both G and D starting to converge around 600 epochs as well as decreasing magnitude of oscillations, indicative of training stability. D's losses oscillating around a value with a slight downward trend. G's losses tend to oscillate less and stabilise quicker but slight upward trend in later stages. Overall, the losses show good balance. Lack of sharp dips or sustained very low losses hint towards a lack of mode collapse experienced in earlier models, but required further examination of the generated samples. Fig 5.10 further shows that D's losses on real and fake data converge and stabilise over time, with no clear signs of overpowering. The close but distinct

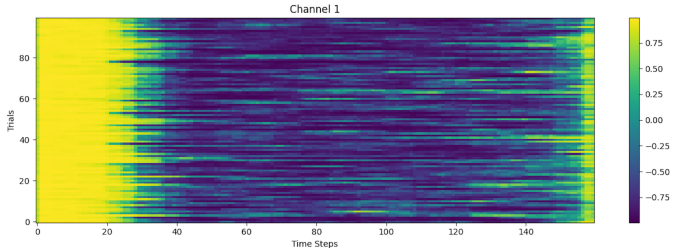

Figure 5.7: A random sample of the generated data for C4 showing no resemblance with the original data.

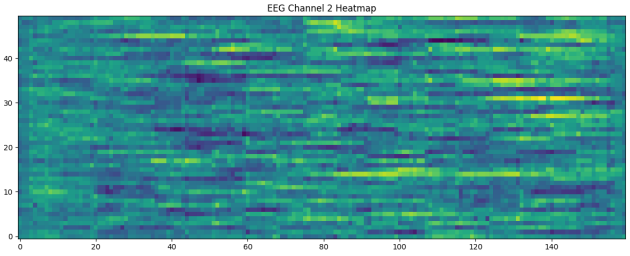

Figure 5.8: Channel C3 sampled after 500 epochs training. DCWGAN-GP failed to capture any details in the data

magnitudes of these losses suggest the discriminator effectively distinguishing between the two while allowing the generator to continue improving. The oscillations indicate active learning without evidence of the discriminator becoming too strong, suggesting a healthy balance in the adversarial training process. This model was finally selected for generating EEG data for further analysis.

These experiments underscored the difficulties of training GANs for EEG data synthesis and highlighted the importance of specific architectural and training modifications, such as the use of gradient penalties and Wasserstein loss, in improving the performance and stability of these models.

5.2 WGAN-GP

As described previously, the WGAN architecture consisted of an input layer followed by 7 fully connected layers with LeakyReLU activations and batch normalisation in between. The input layer takes the noise vector of given dimension concatenated with the CSP features vector and outputs to 64 nodes, allowing the network to learn high-level features first. The concatenated input is then sequentially expanded until reaching the last layer with 1024 output nodes followed by Tanh activation.

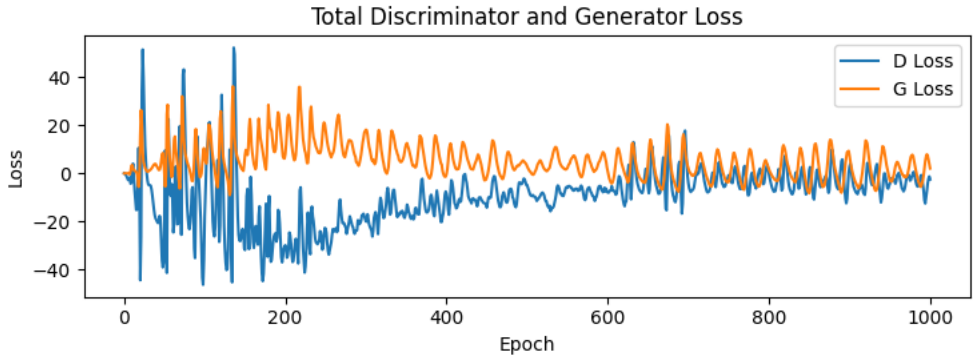

Figure 5.9: Converging trends in D and G losses, reflecting the stable training.

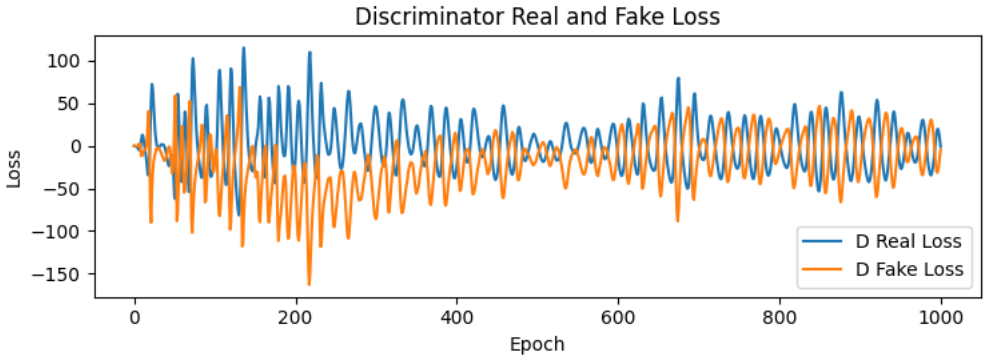

Figure 5.10: D's balanced performance over training epochs, showcasing effective differentiation between real and fake data in training.

The discriminator (or critic) essentially caries out the process in reverse. The flattened input vector is transformed into a 1024-dimensional space and passed through a LeakyReLU activation and then a sequence of transformations until reaching the output layer with 64 nodes. Since the loss is calculated using the Wasserstein distance, no activation is applied to the output of the discriminator.

Gradient penalty implementation A random weight, alpha, is used to interpolate between real and fake data. This interpolated set is fed into the discriminator for evaluation. Gradients of the discriminator's output with respect to these interpolated data are then computed. Gradient penalty is subsequently calculated by squaring the deviation of each gradient's norm from 1 and averaging these values across the batch. This penalty is then used to adjust the discriminator during training.

Xavier initialisation Considering the use of Tanh activation, which saturates, in the generator and vanishing gradients experienced during experimentation, weights of both networks were initialised using the Xavier/Glorot initialisation, to maintain the variance of the activations and gradients across layers during the forward and backward passes. Weights W of a given layer are initialised from a distribution with a variance given by $\text{Var}(W) = \frac{2}{n_{\text{in}}+n_{\text{out}}}$, where n_{in} is the number of neurons feeding into the layer, and n_{out} is the number of neurons the results are fed into. Xavier normal initialisation was used in the network, the weights drawn from a normal distribution $\mathcal{N}(0, \sqrt{\frac{2}{n_{\text{in}}+n_{\text{out}}}})$, balanced the scales of the gradients in both directions.

As suggested in the original WGAN paper [16], an unbalanced training approach resulted in the best results. Training the discriminator as much as 5 to 10 times more often than the generator yielded better results during experimentation before eventually adapting the rate of training of the discriminator to 10 times as often for the final model. Furthermore, to compensate for the discriminator learning too quickly, different beta1 coefficients in the Adam optimiser were adapted for each network, with $\beta_1 = 0.5$ for the generator and $\beta_1 = 0.9$ for the discriminator.

5.3 Data preparation and CSP feature extraction

To reduce the dimensionality and focus on the most relevant parts of the dataset for the given task, only channels Cz, C3 and C4 were extracted to create a subset of the dataset during the data loading process within the pipeline. The capture more of the nuances within each sample, given the aforementioned variability of EEG, the data was normalised per sample rather than globally. Although this seemed to introduce some more apparent noise, the overall quality of the samples extracted during training showed improvement.

A Linear Discriminant Analysis (LDA) classifier achieved accuracy of up to 78% for within session classification using the CSP features, these CSP features were extracted using Python MNE. The log-variance features were fed into the pipeline separately to be used strategically during the training.

Section 6

Results

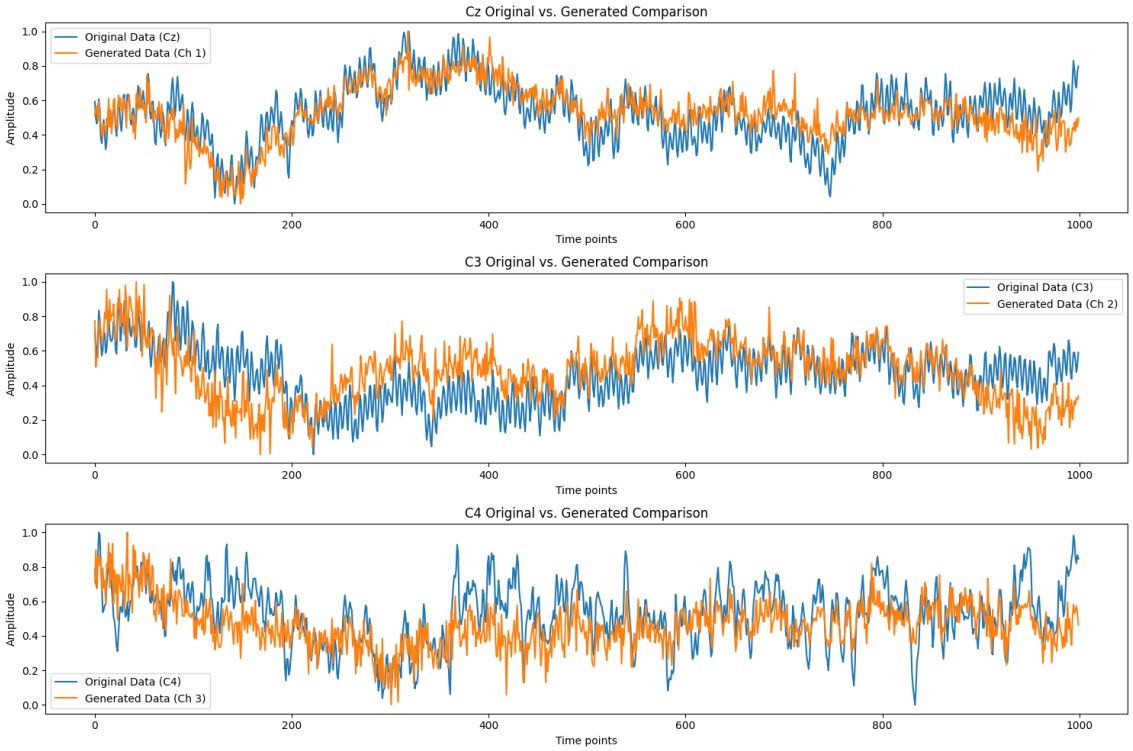

Figure 6.1: Comparison between original and generated data, averaged over sessions for a single subject. Each plot shows the corresponding channels. Barring some lags, the general behaviour is well captured.

The Wasserstein GAN with gradient penalty was trained using the observations form earlier experimentation to choose values for parameters. The data from sessions 1 to 5 for subjects 1 to 20 was used for training. Monitoring the losses during training showed that

the model converged relatively quickly and remained stable throughout training.

6.1 Preservation of patterns in the data

The model was able to capture the general patterns in the data as depicted in Fig 6.1, where the average normalised signals for all trials of real and generated data are plotted for each channel. Both real and generated data follow similar amplitude trends across the three channels, showing the model's ability to capture temporal dynamics of the signal, at least over the population average. The degree of correlation between the two samples is indicative of the model replicating the characteristic patterns of the EEG signal. Notably, the consistency is maintained across different channels, suggesting that the model treats the inherent variability in the signals uniformly without bias toward any particular channel. The Cz (bilateral) and C3 (right hand) channel show a tighter alignment, hinting at channel-specific performance. The C4 channel shows less temporal variability than the original data while still replicating the general behaviour. The model demonstrates the ability for preserving the unique features of individual channels.

Some examples of the generated data were evaluated individually. Using the trained model to generate data for subject 21 session 2 further shows the strength of the model in capturing nuances. Fig 6.2 with channel Cz (labelled channel 0 (original) and 1 (fake) in the plot) shows the generated data with clear resemblance with the original data on the left. While the original data seems more smoothed out, the lack of activity is still captured by the model with small spikes between 400ms and 700ms.

Comparing the same trial for channel C3 (labelled 1 (original) and 2 (fake) in the plot),a similar pattern can be seen where the model has tried to replicate the activity between 0 and 300ms and between 800ms and 1000ms, albeit with more spikes compared to the continuous activity in the original sample. It is worth highlighting that subject 21 was part of the test set rather than the training one, meaning the only subject and session-specific context the GAN was provided was in the form of CSP features. This suggests CSP conditioning can be useful for replicating artefacts present in real data.

(The consistency of the generated data from trial to trial indicates that the WGAN is stable and not prone to mode collapse, where it might produce only a limited range of outputs.The differences in patterns between the different channels are maintained in the generated data, implying that the WGAN has learned channel-specific features of the EEG data.)

Similar patterns were observed for most test subjects with some seeming to be easily captured by the model while other to a lesser degree. Comparing the data from subject

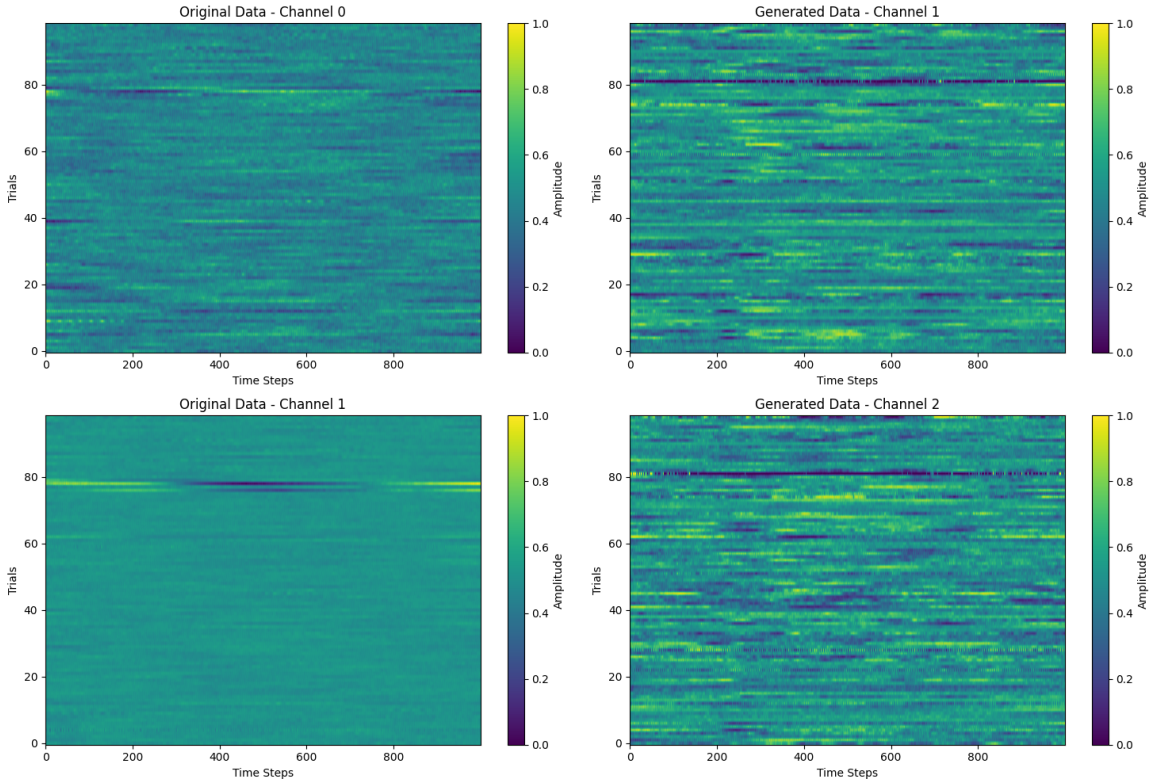

Figure 6.2: Given a noisy training set, some important details are still preserved.

22's session 5 shows the model lacking to replicate the activity in Cz and C3 around 300ms to 400ms for most trials. This particular subject's response to the stimulus (200ms) seems to cause a spike in both channels which the model fails to replicate. Some similar patterns in C3 round the same time frame show that model is still able to predict the onset of the stimulus to some extent.

Once the model reliably produced subject and session specific data with enough details, it was used to generate a dataset using the CSP features for each subject, corresponding to the each subject and session. This subset was then used for further analysis.

Section 7

Evaluation and Discussion

7.1 Analysis of generated data consistency with original data

The model showed promise in being able to reproduce subject and session specific data but with no labels for the MI task for each trial. A semi-supervised learning approach thus had to be adapted. An SVM was trained on real data to predict the labels for the original dataset, which achieved an accuracy of 80%. This trained SVM was then used to predict labels for the generated data to evaluate the alignment of the generated data's label distribution with the original dataset (real labels). Comparing the distribution between left/right hand task for some subjects showed a very similar distribution, with some subjects' prediction being 100%. Table 2 shows the predicted labels for all subjects' session 5 predicted labels for each task compared with the real labels in the original data.

Although not a precise measure, comparing the labels over the entire test set suggests that the model is generally effective in mimicking the label distribution seen in the real, labelled data. This consistency is indicative of a successful transfer of the underlying pattern of MI tasks from the training dataset (sessions 1-4) to the generated data (session 5).

Some of the subject's activity seems to be more easily transferable than others. Subjects 003, 012, 015, 021, and 023 show clear discrepancies. Subject 021 for example, shows quite poor alignment. Further analysis of the subject's original data plotted against the generated sample (Fig 7.1) reveals that this particular subject's data indeed shows more noise/artefacts and variability. The model seems to have overfit to these, leading to the poor generated outcome. Plotting the same subject's data from the training set (session 1) against the test session (session 5) reveals the potential cause of the artefact. The signal from C3 seems to be either very distorted or missing for most trials, the activity from around trial 50 seems to be over represented in the generated sample.

Table 7.1: Comparison of Label Distributions

Subject ID	**Left Hand - O/G**	**Right Hand - O/G**
001	46/46	48/48
002	49/48	46/47
003	50/40	50/60
004	50/52	50/48
005	43/40	46/49
006	49/50	46/45
007	50/52	50/48
008	50/53	50/47
009	50/44	49/55
010	49/49	48/48
011	50/57	50/43
012	36/46	47/37
013	50/60	50/40
014	50/50	50/50
015	48/39	50/59
016	47/47	44/44
017	47/44	48/51
018	50/54	50/46
019	44/46	49/47
020	50/56	50/44
021	47/39	49/57
022	45/50	48/43
023	49/56	48/41
024	50/46	50/54
025	50/54	50/46

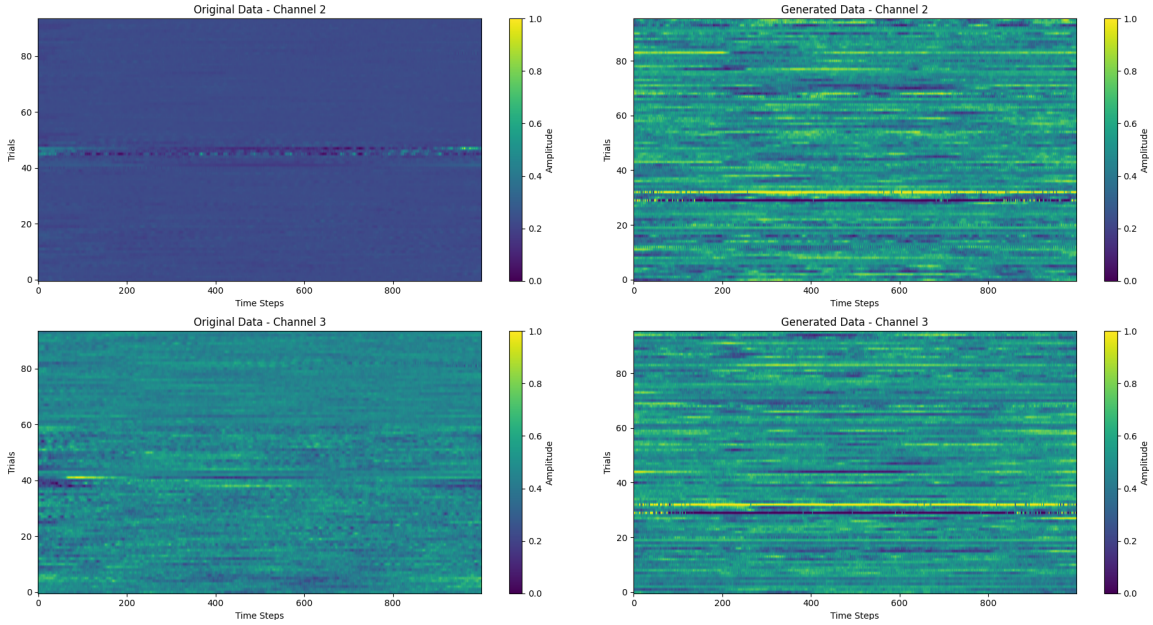

Figure 7.1: Noise and artefacts over represented in the test sample

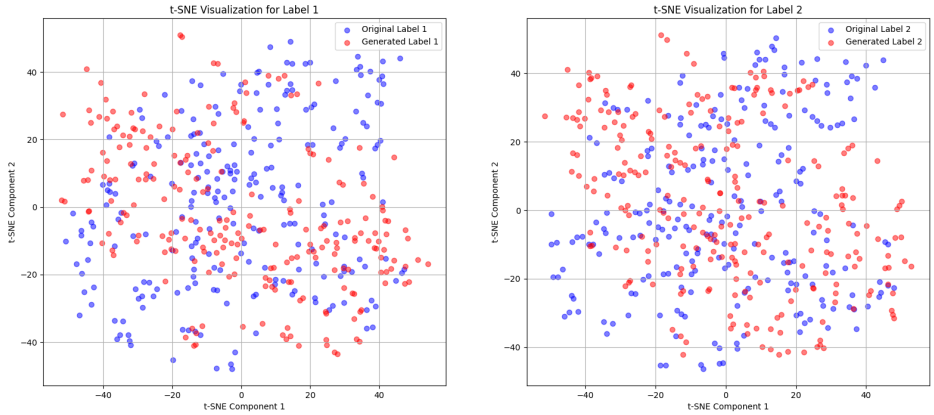

Figure 7.2: t-SNE plot for subject 001 session 5. The generated data (red) shows less variability but the distribution closely resembles the original.

Overall, the model shows clear signs of ability to replicate the nuanced MI task activity for most subjects. To further evaluate the fidelity of the generated data in a high-dimensional context, t-Distributed Stochastic Neighbour Embedding (t-SNE) visualisations were employed. These visualisations offer another method to discern the extent to

which the generative model captures the details within the EEG data.

The t-SNE plots provided insight into the model's performance beyond label accuracy. By embedding both the original and generated EEG data into a two-dimensional plane, visual assessment can be carried out to see how closely the generated data approximates the structure of the original data.

For subjects with well-aligned labels, such as subject 001 (Fig 7.2), the t-SNE plots corroborate this alignment, with generated data points interspersed among the original data points, indicating a successful capture of the underlying MI activity distribution. However, for subjects like 021, where label comparison suggested discrepancies, t-SNE plots offer additional insights. The visual disparity in clusters between the original and generated data for subject 021 suggests the model's limitations in capturing the true signal amidst noise and artefacts, as previously surmised from the label analysis.

Plotting the covariance matrices (Fig 7.3)of the original and generated data reveals the limitations of the model in reproducing the natural variability of EEG data. The aforementioned discrepancies, especially in the case of subject 021, underscore a potential overfitting to idiosyncratic noise and artefacts rather than capturing the neurophysiological patterns of the motor imagery task. The differences in variance and correlation patterns reflect the dynamism and complexity of cerebral activity. For the generated data to have practical applicability, this biological fidelity must be mirrored.
The uniformity in the generated covariance matrices implies a homogenisation of neural

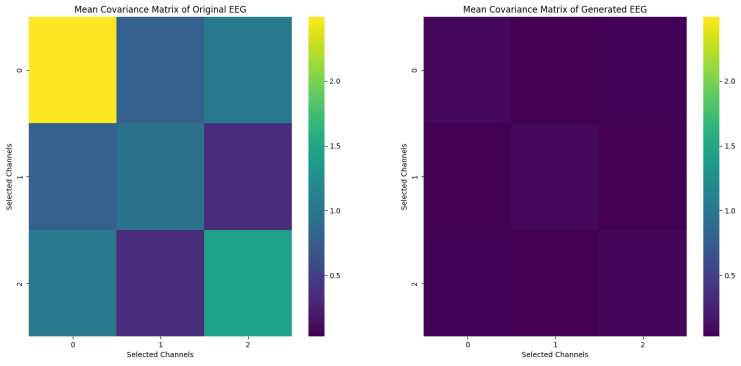

Figure 7.3: Homogenisation of neural signals in generated data.

signals, suggesting model's learning was plateaued and captured an oversimplified essence of the data, stripping away the distinctive features of individual subject responses. The model's state, as evidenced by the covariance matrices, may also point towards a limitation

in the architectural depth.

Measuring and comparing the Inter-trial coherence (ITC) between the two datasets further highlights the limitations of the generated dataset. ITC measures the consistency or phase synchrony of EEG signals across multiple trials at specific time points or frequency bands, useful in studying event-related potential (ERP) (hand movement in this case). Although ITC is a statistical tool, Python MNE provides straightforward and useful implementation which was used to analyse the samples.

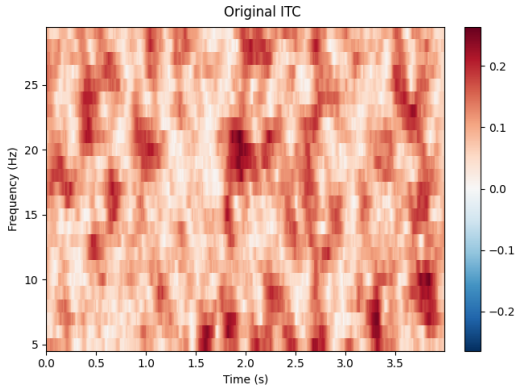

Figure 7.4: ITC Plot for real data

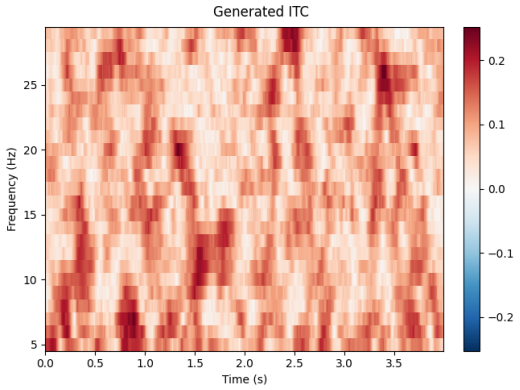

Figure 7.5: ITC Plot for generated data.

As shown in Fig 7.4 and 7.5, while there are some similarities in the patterns of ITC across both samples, quite distinct differences can be observed. The original data shows some frequency bands with higher coherence, particularly in the lower frequencies. The generated data seems to have less distinct bands of high coherence, indicating possible discrepancies in the phase-locking patterns that the model is capturing.

Lastly, the model's lack of capturing ERP can be clearly seen in the model failing to capture the high activity around 200ms (event onset) for most subjects, and example is depicted in Fig 7.6. There seem to be some pattern recognition around 400ms in Cz but not in the other two channels, for the test sample from subject 001 session 5. Similar behaviour can be observed in Fig 7.7, for subject 015 session 5, C3 shows the a lagging behaviour, with the higher activity about 300ms later than the original sample. To further evaluate the quality of the generated data, an SVM was trained on the original data and tested on a subset of the original data and the generated data. The performance on the original data was clearly much better, the performance on the generated data is barely above chance. The findings are presented in Table 7.2.

Considering the data showed signs of relatively successfully replicating activity across channels for various subjects and sessions, it was hypothesised that it may still aid in improving classification. The SVM was then trained on the dataset comprising of data

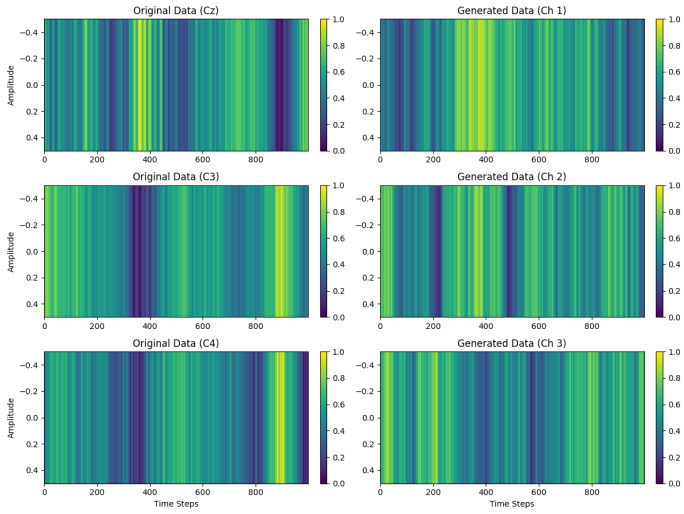

Figure 7.6: Failure to capture ERP in subject 001.

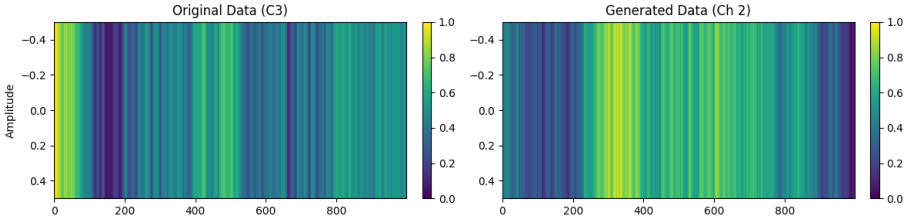

Figure 7.7: Lagging behaviour in channel C3, event onset expected around 200ms after the stimulus at 0ms.

Table 7.2: Comparison of SVM Performance on Original and Generated Data

	Original Data		Generated Data	
Metric	Class 1	Class 2	Class 1	Class 2
Precision	0.56	0.57	0.53	0.49
Recall	0.58	0.54	0.54	0.48
F1-score	0.57	0.56	0.54	0.48
Support	1199	1216	1230	1117
Accuracy	0.56		0.51	
Macro Avg	0.56		0.51	
Weighted Avg	0.56		0.51	

from sessions 1 to 4 for all subjects as well as the generated data for corresponding sessions. On the contrary though, the performance of the SVM this time was in fact inferior to when

trained on just the original data. The model showed slightly better accuracy for class 2 (right hand) compared to just the generated data but remained poor. The model trained on the mixed dataset did show sign of improvement for class 1 recall, moving up to 0.73 compared to 0.58 for the training on original dataset only. Table 7.3 provides detailed findings from the SVM classification report. While further improvement of the generated data was to improve the classification accuracy was not possible within the timeframe of this project, it is possible that a better labelling technique for the generated data or modifications to the GAN architecture could achieve this objective. Other suggestions for future work are provided in Section 8.1.

Table 7.3: SVM Performance on Mixed Dataset

Metric	Class 1	Class 2	Overall
Precision	0.53	0.57	
Recall	0.73	0.35	
F1-score	0.61	0.43	
Support	2404	2427	
Accuracy		0.54	4831
Macro Avg		0.55	4831
Weighted Avg		0.55	4831

Section 8

Conclusion

This project explored the use of conditional Generative Adversarial Networks to generate synthetic EEG data for enhancing the performance of brain-computer interfaces. Through the application of various GAN architectures, particularly the Wasserstein GAN with gradient penalty (WGAN-GP), the aim was to address the significant challenge of EEG data scarcity and variability.

The results suggest that conditioning the GAN with CSP features can be helpful in preserving some of the prominent spatial characteristics of the data. The technique has been demonstrated to highlight some of the high and low intensity patterns in individual trials within a session which remained in the generated data not only for the training but also for the testing datasets. This means the spatial patterns were replicated despite the GAN having never seen the full-dimensional testing data before and only being provided with its CSP features.

Overall, the project contributes valuable insights into the potential and challenges of using conditional GANs for EEG data synthesis. The lessons learned provide a basis for future research in this area, emphasising the need for continued exploration of advanced techniques and architectures to overcome the complexities of EEG data generation for BCIs. The advancements in this field will likely play a crucial role in the development of more robust, adaptable, and effective non-invasive neurotechnologies.

8.1 Future work

The findings indicate that while the fully connected layers used in the initial GAN models laid a foundational groundwork, there remain substantial opportunities for improvement. In particular, the segmentation of data per trial appears to be a promising strategy for enhancing the analysis and labelling accuracy of the generated EEG data. Further in-depth

research in EEG data analysis and signal processing would alleviate the need for unreliable labelling techniques.

Furthermore, the use of Common Spatial Patterns (CSP) features in conditioning the GANs demonstrated initial success in capturing the spatial characteristics of EEG signals. This suggests that further exploration and optimisation of CSP feature integration could significantly improve the fidelity of the generated data.

The use of convolutional layers in GANs is now well-established, and has been demonstrated to successfully capture complex patterns in real data. The poor results reported for the convolutional GAN architectures in this work are therefore likely to be a consequence of suboptimal implementation rather than a fundamental limitation of the technique. The addition of convolutional layers with a more optimal architecture is anticipated to enhance the model's ability to capture spatial and temporal details of the real data. This modification is expected to be a significant direction for future work, aiming to produce more realistic EEG datasets that could better support the needs of BCI applications.

References

[1] P. Nuyujukian, J. Albites Sanabria, J. Saab, C. Pandarinath, B. Jarosiewicz, C. H. Blabe, B. Franco, S. T. Mernoff, E. N. Eskandar, J. D. Simeral, L. R. Hochberg, K. V. Shenoy, J. M. Henderson, Cortical control of a tablet computer by people with paralysis, PLOS ONE 13 (11) (2018) e0204566. doi:10.1371/journal.pone.0204566.

[2] F. R. Willett, D. T. Avansino, L. R. Hochberg, J. M. Henderson, K. V. Shenoy, High-performance brain-to-text communication via handwriting, Nature 593 (7858) (2021) 249–254. doi:10.1038/s41586-021-03506-2.

[3] A. Kumar Chaudhary, V. Gupta, K. Gaurav, T. Kumar Reddy, L. Behera, EEG Control of a Robotic Wheelchair, in: R. Vinjamuri (Ed.), Human-Robot Interaction - Perspectives and Applications, IntechOpen, 2023. doi:10.5772/intechopen.110679.

[4] R. Portillo-Lara, B. Tahirbegi, C. Chapman, J. Goding, R. Green, Mind the gap: State-of-the-art technologies and applications for EEG-based brain–computer interfaces, APL Bioengineering 5 (2021) 031507. doi:10.1063/5.0047237.

[5] M. Jeannerod, The representing brain: Neural correlates of motor intention and imagery, Behavioral and Brain Sciences 17 (2) (1994) 187–202. doi:10.1017/S0140525X00034026.

[6] Z. Khademi, F. Ebrahimi, H. M. Kordy, A review of critical challenges in MI-BCI: From conventional to deep learning methods, Journal of Neuroscience Methods 383 (2023) 109736. doi:10.1016/j.jneumeth.2022.109736.

[7] Z. Miao, X. Zhang, C. Menon, Y. Zheng, M. Zhao, D. Ming, Priming Cross-Session Motor Imagery Classification with A Universal Deep Domain Adaptation Framework (Jul. 2023). arXiv:2202.09559.

[8] G. C. De Melo, G. Castellano, A. Forner-Cordero, A procedure to minimize EEG variability for BCI applications, Biomedical Signal Processing and Control 89 (2024) 105745. doi:10.1016/j.bspc.2023.105745.

[9] E. Brophy, P. Redmond, A. Fleury, M. De Vos, G. Boylan, T. Ward, Denoising EEG Signals for Real-World BCI Applications Using GANs, Frontiers in Neuroergonomics 2 (2022) 805573. doi:10.3389/fnrgo.2021.805573.

[10] C. Shorten, T. M. Khoshgoftaar, A survey on Image Data Augmentation for Deep Learning, Journal of Big Data 6 (1) (2019) 60. doi:10.1186/s40537-019-0197-0.

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial networks, Commun. ACM 63 (11) (2020) 139–144. doi:10.1145/3422622.
URL https://doi.org/10.1145/3422622

[12] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, A. A. Bharath, Generative Adversarial Networks: An Overview, IEEE Signal Processing Magazine 35 (1) (2018) 53–65. arXiv:1710.07035, doi:10.1109/MSP.2017.2765202.

[13] A. Jabbar, X. Li, B. Omar, A Survey on Generative Adversarial Networks: Variants, Applications, and Training, ACM Computing Surveys 54 (8) (2022) 1–49. doi:10.1145/3463475.

[14] N. K. N. Aznan, A. Atapour-Abarghouei, S. Bonner, J. Connolly, N. A. Moubayed, T. Breckon, Simulating Brain Signals: Creating Synthetic EEG Data via Neural-Based Generative Models for Improved SSVEP Classification, in: 2019 International Joint Conference on Neural Networks (IJCNN), 2019, pp. 1–8. arXiv:1901.07429, doi:10.1109/IJCNN.2019.8852227.

[15] T.-j. Luo, Y. Fan, L. Chen, G. Guo, C. Zhou, EEG Signal Reconstruction Using a Generative Adversarial Network With Wasserstein Distance and Temporal-Spatial-Frequency Loss, Frontiers in Neuroinformatics 14 (2020) 15. doi:10.3389/fninf.2020.00015.

[16] M. Arjovsky, S. Chintala, L. Bottou, Wasserstein GAN (Dec. 2017). arXiv:1701.07875.

[17] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, A. Courville, Improved Training of Wasserstein GANs (Dec. 2017). arXiv:1704.00028.

[18] A. G. Habashi, A. M. Azab, S. Eldawlatly, G. M. Aly, Motor Imagery Classification Enhancement using Generative Adversarial Networks for EEG Spectrum Image Generation, in: 2023 IEEE 36th International Symposium on Computer-Based Medical Systems (CBMS), 2023, pp. 354–359. doi:10.1109/CBMS58004.2023.00243.

[19] Y. Song, L. Yang, X. Jia, L. Xie, Common Spatial Generative Adversarial Networks based EEG Data Augmentation for Cross-Subject Brain-Computer Interface (Feb. 2021). arXiv:2102.04456.

[20] F. Fahimi, S. Dosen, K. K. Ang, N. Mrachacz-Kersting, C. Guan, Generative Adversarial Networks-Based Data Augmentation for Brain–Computer Interface, IEEE Transactions on Neural Networks and Learning Systems 32 (9) (2021) 4039–4051. doi:10.1109/TNNLS.2020.3016666.

[21] M. Mirza, S. Osindero, Conditional Generative Adversarial Nets (Nov. 2014). arXiv:1411.1784.

[22] J. Ma, B. Yang, W. Qiu, Y. Li, S. Gao, X. Xia, A large EEG dataset for studying cross-session variability in motor imagery brain-computer interface, Scientific Data 9 (1) (2022) 531. doi:10.1038/s41597-022-01647-1.

[23] M. Jun, B. Yang, W. Qiu, shu$_d$*ataset* (8 2022). doi:10.6084/m9.figshare.19228725.v1.
URL https://figshare.com/articles/software/shu_dataset/19228725

[24] W. H. Lee, E. Kim, H. G. Seo, B.-M. Oh, H. S. Nam, Y. J. Kim, H. H. Lee, M.-G. Kang, S. Kim, M. S. Bang, Target-oriented motor imagery for grasping action: Different characteristics of brain activation between kinesthetic and visual imagery, Scientific Reports 9 (1) (2019) 12770. doi:10.1038/s41598-019-49254-2.

[25] Z. Chen, Y. Wang, Z. Song, Classification of Motor Imagery Electroencephalography Signals Based on Image Processing Method, Sensors (Basel, Switzerland) 21 (14) (Jul. 2021). doi:10.3390/s21144646.

[26] X. Wang, X. Dai, Y. Liu, X. Chen, Q. Hu, R. Hu, M. Li, Motor imagery electroencephalogram classification algorithm based on joint features in the spatial and frequency domains and instance transfer, Frontiers in Human Neuroscience 17 (2023). doi:10.3389/fnhum.2023.1175399.
URL https://www.frontiersin.org/articles/10.3389/fnhum.2023.1175399