ABSTRACT

CHOO, SANGHYUN. Improving Generalization Performance of Deep Learning for Brain-Computer Interface: Effects of Data Augmentation and Batch Size Strategy. (Under the direction of Dr. Chang S. Nam).

In motor imagery (MI)-based brain-computer interface (BCI), diverse deep learning (DL) architectures have been developed to decode the related electroencephalogram (EEG) signals. Among the various DL models, convolutional neural networks (CNNs) have contributed significantly to the performance improvement for classifying the MI EEG signals. However, most of the studies have not paid attention to how to train the networks well for high generalization performance of the classifiers despite the high possibility of overfitting due to a small number of EEG training datasets. In this study, we focused on improving the generalization performance of the CNNs using regularization methods. In particular, we developed a data augmentation (DA) methodology using a generative model to enhance the generalization performance of CNNs for MI BCI. We also investigated the impacts of a crucial hyperparameter on the generalization performance. This dissertation consists of two main chapters, 1) EEG DA to Improve Classification Performance of MI BCI using Conditional Generative Adversarial Networks (cGANs) and 2) Effects of Batch Size and Its Strategy on Generalizability of CNNs on MI BCI.

In the first chapter, we focused on DA of MI EEG signals. In general, machine learning (ML)/DL-based classifiers require a large dataset for training to build reliable and accurate models. However, collecting large enough EEG datasets is difficult due to intra-/inter-subject variabilities and experimental costs. This leads to the data scarcity problem, which causes overfitting issues to training samples, resulting in reducing generalization performance. To solve the EEG data scarcity problem and improve the performance of the EEG classifiers, we proposed a novel EEG DA framework using cGANs. An experimental study was implemented with two public EEG datasets

for MI tasks to validate the effectiveness of the proposed EEG DA method for the classifiers. To evaluate the proposed cGAN-based DA method, we tested six EEG classifiers for the experiment, including traditional ML algorithms and state-of-the-art CNNs for MI EEG signals with three existing EEG DA methods. Experimental results showed that most DA methods with proper DA proportion in the training dataset had higher classification performances than without DA. Moreover, applying the proposed DA method showed superior classification performance improvement to the other DA methods. This shows that the proposed method is a promising EEG DA method for enhancing the performances of the EEG classifiers in MI-based BCIs.

In the second chapter, we investigated the effects of batch size and its strategy on the generalization performance of CNNs. The existing studies for the CNNs have validated the effectiveness of their classifiers by overlooking the importance of batch size without elaborating on their settings. This can cause errors in verifying the models' effectiveness. To emphasize the importance of batch size in model validation and comparison, an experimental study was implemented with two public BCI datasets in which we considered four CNN classifiers, one general batch size strategy, and four different batch size strategies for the investigation. Experimental results showed that the best batch size and the strategy to get the highest classification performance varied depending on the CNN classifiers. Our findings suggest that they need to be carefully considered and specified to show the effectiveness of the developed models objectively.

Improving Generalization Performance of Deep Learning for Brain-Computer Interface: Effects of Data Augmentation and Batch Size Strategy

by
Sanghyun Choo

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Industrial Engineering

Raleigh, North Carolina
2022

APPROVED BY:

Chang S. Nam Committee Chair	Kevin Flores
Yunan Liu	Sara Shashaani
Emily Hector	

DEDICATION

This dissertation is dedicated to my dad, Daeho Choo. Thank you for your unconditional love, sacrifice, and support throughout my life.

BIOGRAPHY

Sanghyun Choo is a Ph.D. candidate in the Edward P. Fitts Department of Industrial and Systems Engineering at North Carolina State University. He earned his Bachelor's and Master's degrees in Industrial Engineering from Kumoh National Institute of Technology, Korea in 2016 and 2018, respectively. His research interests lie in developing, improving, and applying AI models for various application domains, including structured and unstructured datasets. During his Ph.D., he has mainly focused on developing regularization and interpretation methods for deep learning to enhance the generalization performance and model interpretation in the brain-computer interface.

ACKNOWLEDGMENTS

First and foremost, I would like to truly thank my advisor Dr. Chang S. Nam for his limitless support and guidance during my years at North Carolina State University. I thank him for giving me many opportunities to become an independent researcher and for helping me achieve my goal.

I would also like to thank each member of my committee, Dr. Kevin Flores, Dr. Yunan Liu, Dr. Sara Shashaani, and Dr. Emily Hector for being on my dissertation committee and providing me with their valuable feedback and suggestions. Additionally, I would like to thank Dr. Daniel Gruehn for serving as a Graduate School Representative for my dissertation defense.

I want to acknowledge and thank all of my folks, friends, and colleagues, especially Zachary Traylor and Jiali Huang who helped and encouraged me. I am also grateful to my friend Yunsoo Ha for always encouraging me to pursue my dreams.

Most importantly, I would like to thank my family. I especially want to thank my dad who always believed, encouraged, and cheered me up throughout my studies.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF CONTRIBUTING PUBLICATIONS

Peer-reviewed Journal Articles

Choo, S., & Nam, C. S. (2022). Detecting human trust calibration in automation: A convolutional neural network approach. *IEEE Transactions on Human-Machine Systems*.

Choo, S., Park, H., Jung, J., Flores, K., & Nam, C. S. (under review). EEG data augmentation to improve classification performance of motor imagery BCI using conditional generative adversarial networks.

Choo, S., Park, H., Kim, S., Park, D., Jung, J., Lee, S., & Nam, C. S. (under review). Effectiveness of multi-task deep learning framework for EEG-based emotion and context recognition.

Choo, S., & Nam, C. S. (under review). Effects of batch size and its strategy on generalizability of convolutional neural networks on motor imagery BCI.

Peer-reviewed Conference Proceedings

Traylor, Z., Houk, M., **Choo, S.**, & Nam, C. S. (accepted). Deep learning classification of human emotional state: The search for a robust and interpretable model. In *Proceedings of the Human Factors and Ergonomics Society's 66th Annual Meeting*. SAGE Publications.

Choo, S., Ghasemi, Y., Jeong, H., & Nam, C. S. (2021). Effects of multi-task learning for deep learning on prediction performance of EEG-based cognitive state recognition. In *Proceedings of IIE Annual Conference*, 334-339. Institute of Industrial and Systems Engineers (IISE).

Choo, S., & Nam, C. S. (2020). Emotion recognition with a CNN using functional connectivity-based EEG features. In *Proceedings of the Human Factors and Ergonomics Society's 64th Annual Meeting*. SAGE Publications.

Choo, S., & Nam, C. S. (2020). DCGAN based EEG data augmentation in cognitive state

recognition. In *Proceedings of IIE Annual Conference*, 1-6. Institute of Industrial and Systems Engineers (IISE).

Choo, S., Sanders, N., Kim, N., Kim, W., & Nam, C. S. (2019). Detecting human trust calibration in automation: A deep learning approach. In *Proceedings of the Human Factors and Ergonomics Society's 63rd Annual Meeting* (Vol. 63, No. 1, pp. 88-90). SAGE Publications.

Book Chapters

Choo, S., & Nam, C. S. (2022). Interactive reinforcement learning and error-related potential classification for implicit feedback. *Human-Centered Artificial Intelligence: Research and Applications* (pp. 127-143). Elsevier.

Choo, S., & Nam, C.S. (2020). Deep learning techniques in neuroergonomics. *Neuroergonomics: Principles and Practices* (pp. 115-138). Springer.

LIST OF ACRONYMS

AAB	algorithm-based adaptive batch size
AB	adaptive batch size
AUC	area under curve
BCI	brain-computer interface
CNN	convolutional neural network
CNNs	convolutional neural networks
cDCGAN	conditional deep convolutional generative adversarial network
cGAN	conditional generative adversarial network
cGANs	conditional generative adversarial networks
cWGAN	conditional Wasserstein generative adversarial network
cWGAN-GP	conditional Wasserstein generative adversarial network with gradient penalty
CSP	common spatial pattern
DA	data augmentation
DBN	deep Boltzmann machine
DCGAN	deep convolutional generative adversarial network
DL	deep learning
EEG	electroencephalogram
EEG-TCNet	EEG temporal convolutional network
ESTCNN	EEG-based spatial-temporal convolutional neural network
FB	fixed batch size
GAN	generative adversarial network

GP	gradient penalty
HAB	heuristic-based adaptive batch size
HS-CNN	hybrid-scale convolutional neural network
JSD	Jensen–Shannon divergence
LB	large batch size
LDA	linear discriminant analysis
LR	logistic regression
MCNN	multi-layer convolutional neural network
MI	motor imagery
ML	machine learning
MMCNN	multi-branch multi-scale convolutional neural network
MSNN	multi-scale neural network
NA	noise addition
RG	Riemannian geometry
RS	resampling
SB	small batch size
SGD	stochastic gradient descent
SMR	sensorimotor rhythm
SVM	support vector machine
VAE	variational autoencoder
WGAN	Wasserstein generative adversarial network
WGAN-GP	Wasserstein generative adversarial network with gradient penalty

CHAPTER 1

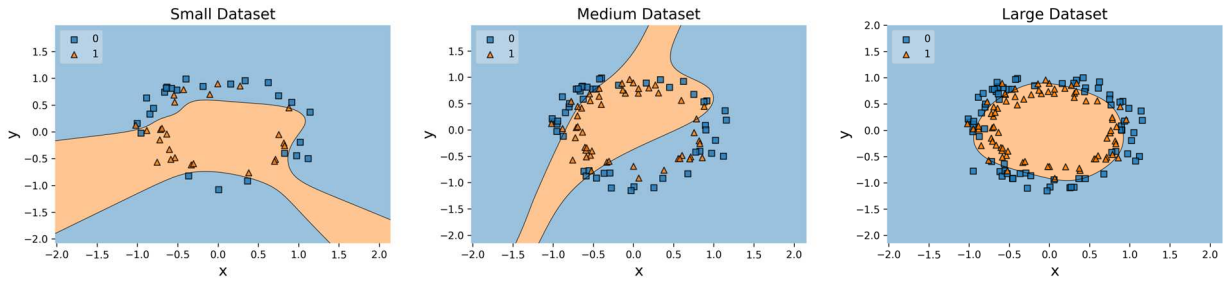
EEG Data Augmentation to Improve Classification Performance of Motor Imagery BCI using Conditional Generative Adversarial Networks

1.1. Introduction

The brain-computer interface (BCI) provides a promising communication method between a human brain and a computer (Zhang et al., 2018). The primary function of BCI is to decode neural signal patterns of a user for specific mental tasks and then translate the neural responses into a computer command (Zhang et al., 2012). BCI systems allow users to control external devices (e.g., wheelchair (Huang et al., 2012) and robots (Vidaurre et al., 2016)), especially those with disabilities, to help interact with external environments or to restore their abilities (Ang et al., 2015; Chaudhary et al., 2016). Currently, electroencephalogram (EEG) has been most widely adopted for BCI due to its high temporal resolution, non-invasiveness, portability, and adaptability (Remsik et al., 2016).

Sensorimotor rhythm (SMR) is one of the most frequently used neural patterns for BCI development (Pfurtscheller et al., 2006). SMR is induced by performing motor imagery (MI) tasks (e.g., the imagination of body movement) and can be measured through EEG signals in the sensory cortex area (Pfurtscheller & Lopes da Silva, 1999). To decode the SMR from the EEG signals, machine learning (ML) and deep learning (DL) techniques can be employed (Kang & Choi, 2014; Zhang et al., 2021). These techniques have been applied in many areas successfully with a large amount of dataset such as image processing (Rawat & Wang, 2017; Guo et al., 2021), natural language processing (Jiang et al., 2021; Young et al., 2018), and speech recognition (Stewart et al., 2014; Agarwalla & Sarma, 2016).

However, unlike these domains, it is tough to collect a large EEG dataset due to the following reasons: (1) EEG experiments cannot last for a long time since participants might feel uncomfortable from the long wear of EEG cab. (2) There is difficulty in getting fully utilizable EEG signals because the signals are exposed to many noises and artifacts (e.g., lights and blinking). (3) Because of the nonstationary of the EEG, the brain signals are different from session to session (intra-subject variability) and subject to subject (inter-subject variability) (Shenoy et al., 2006). EEG data scarcity from these factors can cause overfitting problems in ML and DL. This is because classifiers trained on a small dataset are more likely to see non-existent patterns, resulting in a high error on a test dataset (unseen data).



(a) **(b)** **(c)**

Figure 1.1. Example of effects of DA on the classifier.

Simple multi-layer perceptron and oversampling were used for the classifier and DA, respectively. The blue area represents decision boundaries for class 0 (square), whereas the orange indicates decision boundaries for class 1 (triangle). (a) shows a decision boundary for the classifier with a small dataset. (b) shows the decision boundary by augmented datasets with the previous small dataset. (c) shows the decision boundary close to the ground truth with enough DA.

To overcome the overfitting problem by the data scarcity, data augmentation (DA) approaches have been attracted as a promising and effective way since they can directly handle the data deficiency problem of EEG. Figure 1.1 shows graphically why DA can be effective in overcoming the overfitting problem by the data scarcity. In this dataset, we assume the ground

truth for the classification decision boundary is a circle. Figure 1.1 (a) represents a decision boundary for classification by a small dataset. This shows that class 1 is heavily distributed in a particular area, resulting in biased decision boundaries. Suppose we can generate additional data samples from a well-trained model that can represent the true distribution of the dataset. In that case, the synthetic data samples can help not to overfit in training. By augmenting the training samples, we can mitigate the overfitting (see Figure 1.1 (b)). With enough DA, we would be able to get a decision boundary close to the ground truth (see Figure 1.1 (c)).

Studies have shown that DA methods to augment EEG data, such as Gaussian noise addition and generative models, can improve the performance of EEG classifiers by avoiding overfitting by EEG data scarcity (Wang et al., 2018; Choo & Nam, 2020; Luo et al., 2020). In particular, generative adversarial networks (GANs; Goodfellow et al., 2014) have received considerable attention because of their powerful ability to generate artificial data mimicking real EEG data (Hartmann et al., 2018; Luo & Lu, 2018; Zhang & Liu, 2018; Aznan et al., 2019; Gan et al., 2019; Choo & Nam, 2020; Luo et al., 2020; Panwar et al., 2020; Fahimi et al., 2021). Since a GAN-based single channel EEG generation method using a Wasserstein GAN (WGAN) with gradient penalty (GP) was first introduced (Hartmann et al., 2018), various GAN-based methods of augmenting EEG data have been proposed and shown to improve the performance of the models, including deep convolutional GAN (DCGAN; Aznan et al., 2019; Choo & Nam, 2020), conditional GAN (cGAN; Luo & Lu, 2018), condition DCGAN (cDCGAN; Zhang & Liu, 2018; Fahimi et al., 2021), and conditional WGAN with the gradient penalty (cWGAN-GP; Luo & Lu, 2018; Gan et al., 2019; Panwar et al., 2020).

However, it is also true that these studies have several limitations when augmenting training EEG samples. First, Gaussian noise addition can lead to learning patterns unrelated to

training samples because it does not reflect the distribution of EEG signals (Wang et al., 2018). Second, studies overlooked the imbalanced class of the generated EEG samples caused by mode collapse of basic GANs (Hartmann et al., 2018; Aznan et al., 2019; Choo & Nam, 2020). Third, Jensen–Shannon divergence (JSD)-based GANs frequently face vanishing gradients, resulting in training failure of the generator (Zhang & Liu, 2018; Aznan et al., 2019; Choo & Nam, 2020; Fahimi et al., 2021). Fourth, most studies augmented features of the EEG signals, rather than EEG data itself, before analyzing them with their classifiers (Luo & Lu, 2018; Zhang & Liu, 2018; Gan et al., 2019; Choo & Nam, 2020). Fifth, only one study (Fahimi et al., 2021) provided network architectures for the GAN to generate raw MI EEG signals. However, their architecture works on only predefined numbers of channels and time points by them.

To address these issues, we propose a novel EEG DA framework using the cGAN family (cGAN, cWGAN, and cWGAN-GP) to improve the classification performance of EEG classifiers for MI tasks. The main novelty of the proposed model is to provide the new network architectures of the cGAN family specialized for MI EEG signals. The proposed cGANs, especially cWGAN-GP, could successfully overcome problems of gradient vanishing and mode failure of the basic GANs by using Wasserstein distance and condition of MI EEG class. Also, we provide a pseudo code to implement the cGAN family for the EEG DA, which was not given in the previous studies. The code helps to implement it more easily for researchers interested in the EEG DA using cGAN family. In addition, with the proposed DA framework, this study systematically investigates the effects of synthetic EEG data proportion in training data on EEG classifiers. This study addressed three main research questions: (1) Is the proposed DA method more effective than other DA methods? (2) Is the EEG DA method effective in improving the performance of MI EEG

classifiers? (3) What is the best proportion of synthetic EEG data in the training dataset for the MI EEG classifiers to improve their performance?

1.2. Related Work

1.2.1. EEG Data Augmentation

DA increases the number of training datasets by appending new artificial datasets manipulated from existing training datasets (Um et al., 2017). DA has mainly been used to resolve two main problems in DL domains – data scarcity and overfitting. For example, in computer vision applications, augmenting data can include the use of translations, rotations, cropping, and flipping because the researcher expects these perturbations not to affect the distribution of the training data (Perez-Benitez et al., 2018). But instead, the hypothesis is that the augmented training points lie within the same distribution as the non-augmented training set. Also, many studies have shown that DA improves the accuracy and stability of the ML and the DL by making more generalized models (Rashid & Louis, 2019; Shorten & Khoshgoftaar, 2019; Luo et al., 2020; Zhang et al., 2021). In particular, it can be very effective for improving the performance of the DL since it requires many training data to train a huge number of parameters by a learning algorithm. Due to the effectiveness of the DA, it can be a promising method for EEG classifiers to improve their performance since collecting large amounts of EEG data is very expensive by its cost and tedious, time-consuming tasks. There have been a few EEG DA methods, such as noise addition and generative models.

The noise addition method mainly uses Gaussian white noise to augment EEG training data. For EEG signals, the assumption is that the classifier should be robust with respect to added gaussian noise, and hence applying gaussian noise augmentation can provide samples from the

underlying non-augmented EEG data distribution. The noise addition is appropriate for augmenting EEG signals due to the strong randomness and non-stationarity of the EEG data (Wang et al., 2018). Users select the parameters such as mean and standard deviation. Given the parameters, the Gaussian noise is added to EEG input features. The users can increase the artificial EEG data as much as they want based on the Gaussian distribution. Compared to other EEG DA methods such as overlapping time windows and generative models, the noise addition method is straightforward to use because it requires only Gaussian distribution for noise. Studies showed that this approach increased the accuracy and robustness of DL classifiers compared to the model without EEG DA (Yin & Zhang, 2017a, 2017b; Wei et al., 2018; Pei et al., 2021). However, it might lead to poor quality of generated EEG signals since it could generate the artificial EEG signals that are almost similar to the existing training data or beyond the characteristics of the EEG signals depending on noise intensity.

In contrast to DA methods that rely on possibly inaccurate assumptions about the insensitivity of the classifier to user-defined augmentations, generative models can reflect the true distribution of the training dataset to generate new artificial datasets, which can make the classifiers to be invariant to the generated samples. They aim at learning and approximating the distribution of the training dataset. There are many generative models such as deep Boltzmann machine (DBM; Salakhutdinov & Hinton, 2009), variational autoencoder (VAE; Doersch, 2016), and GAN (Goodfellow et al., 2014). GAN models have been commonly used for DA since they can capture key features from various datasets to generate artificial datasets (Goodfellow et al., 2014). Also, GAN-based DA models have been developed in EEG classification domains to improve the performance of EEG classifiers (Luo & Lu, 2018; Zhang & Liu, 2018; Aznan et al., 2019; Luo et al., 2020; Panwar et al., 2020; Fahimi et al., 2021; Pei et al., 2021; Zhang et al., 2021).

These studies show the possibility of GAN as a DA method to improve the EEG classifiers. However, several studies have overlooked the imbalanced class of the EEG data caused by mode collapse of basic GANs (Hartmann et al., 2018; Aznan et al., 2019; Choo & Nam, 2020; Fahimi et al., 2021). Also, most studies augmented features of the EEG signals, rather than raw EEG signals, before analyzing them with their classifiers (Luo & Lu, 2018; Zhang & Liu, 2018; Gan et al., 2019; Choo & Nam, 2020). Furthermore, to our best knowledge, only one study (Fahimi et al., 2021) has developed a GAN model to augment multi-channel MI EEG signals to improve MI EEG classifiers. However, there are mainly three limitations of their GAN model. (1) Their GAN model has training instability due to JSD-based generator training (Karras et al., 2017; Kodali et al., 2017), leading to training failure. (2) Their model did not consider mode failure from EEG data classes. This can cause severe data imbalance when generating artificial EEG data, resulting in overfitting specific classes. (3) Their network architectures for the generator and discriminator for the GAN can only be applied to the predefined numbers of channels and time points from them. It cannot be applied to a different number of channels and time points. In addition to such limitations of EEG DA methods, little study has been done on the effectiveness of the EEG DA methods for improving diverse EEG classifiers by systematically exploring DA methods and synthetic data proportion in the training dataset.

1.2.2. GANs

GANs are composed of two opposing main networks (discriminator D and generator G) employing each other to be trained (Goodfellow et al., 2014). The discriminator D is trained to discriminate real input data from fake input data generated by the generator G by estimating the probability of being real data. The generator uses a latent noise $\boldsymbol{z}$ as input and generates fake data

samples to fool the discriminator *D*. Given the characteristics of the two networks, a two-player minimax game occurs where the discriminator *D* tries to maximize discrimination between the real samples and the fake samples, and the generator *G* aims at fooling the discriminator *D* by minimizing the distribution of the real data and the distribution of the generated data. The minimax game of the original GAN is represented with the loss function $L^{GAN}(D, G)$ as:

$$\min_G \max_D L^{GAN}(D, G) = \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_r}[logD(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim\mathbb{P}_z}[\log(1 - D(G(\boldsymbol{z})))] \quad (1.1)$$

where $\boldsymbol{x}$ is a real data, $\boldsymbol{z}$ is a noise used as input for the generator *G*, $\mathbb{P}_r$ is the real data distribution, $\mathbb{P}_z$ is the prior noise distribution such as the uniform distribution, $D(\boldsymbol{x})$ is the true data distribution, and $G(\boldsymbol{z})$ is the fake data generated from noise $\boldsymbol{z}$.

1.2.2.1. Conditional GAN

The basic GAN cannot reflect class labels in generating artificial data. Namely, it may cause the imbalanced data problem by generating specific classes more frequently. To overcome this problem and provide more stable training, conditional GAN (cGAN) was developed by conditioning the generator and the discriminator for exploited classes (Mirza & Osindero, 2014). The problem of cGAN is defined as:

$$\begin{aligned}\min_G \max_D L^{cGAN}(D, G) \\ = \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_r,\boldsymbol{y}\sim\mathbb{P}_y}[logD(\boldsymbol{x}|\boldsymbol{y})] \\ + \mathbb{E}_{\boldsymbol{z}\sim\mathbb{P}_z,\boldsymbol{y}\sim\mathbb{P}_y}[\log(1 - D(G(\boldsymbol{z})|\boldsymbol{y}))]\end{aligned} \quad (1.2)$$

where $\boldsymbol{y}$ is an auxiliary label from the true class distribution $\mathbb{P}_y$.

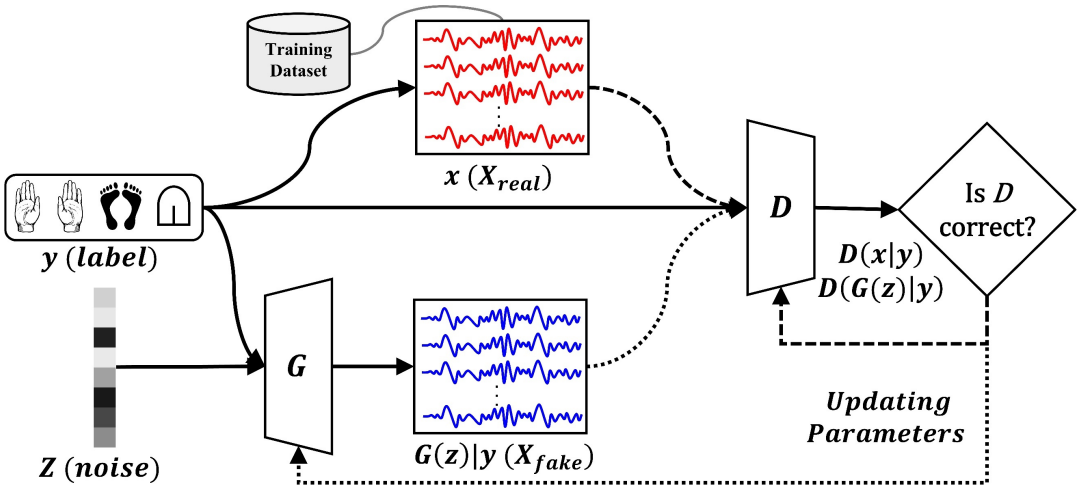


Figure 1.2. cGAN structure.

As shown in Figure 1.2, the auxiliary information for the label $\boldsymbol{y}$ is fed into both the discriminator *D* and the generator *G* by being concatenated. Given the label $\boldsymbol{y}$ and the noise $\boldsymbol{z}$, the goal of the generator *G* is to deceive the discriminator *D* by generating fake data $G(\boldsymbol{z})|\boldsymbol{y}\ (X_{fake})$. On the other hand, the discriminator *D* aims at distinguishing well whether the input data is real data $\boldsymbol{x}\ (X_{real})$ or fake data X_{fake} given the specific label $\boldsymbol{y}$. If $D(\cdot\,|\boldsymbol{y})$ is 1, the discriminator *D* classifies the input to the real. Otherwise, it is classified to the fake.

1.2.2.2. WGAN and WGAN-GP

The original GAN aims at minimizing the Jensen–Shannon divergence (JSD) between the real data distribution $\mathbb{P}_r$ and the fake data distribution $\mathbb{P}_g$ to train the model (Goodfellow et al., 2014). To overcome the gradient vanishing problem that is caused by the JSD-based generator training, a training method using Wasserstein distance, which has a smoother gradient everywhere, was proposed to minimize the distance of the distributions instead of adding just noise (Arjovsky et al., 2017). The Wasserstein distance can be simplified using the Kantorovich-Rubinstein duality

with an 1-Lipschitz function since the distance is highly intractable to be used by itself (Arjovsky et al., 2017). The simplified Wasserstein distance between the real data distribution $\mathbb{P}_r$ and the fake distribution $\mathbb{P}_g$ is represented as:

$$W(\mathbb{P}_r, \mathbb{P}_g) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_r}[f(\boldsymbol{x})] - \mathbb{E}_{\tilde{\boldsymbol{x}}\sim\mathbb{P}_g}[f(\tilde{\boldsymbol{x}})], \tag{1.3}$$

where $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ are real and fake samples following with $\mathbb{P}_r$ and $\mathbb{P}_g$, respectively. And sup is the upper bound, and $f(x)$ is the 1-Lipschitz function satisfying $|f(x_1) - f(x_2)| \leq |x_1 - x_2|$.

In WGAN, the discriminator is also called a critic. Given formula (1.3), the critic maximizes the Wasserstein distance and the generator *G* maximizes $\mathbb{E}_{\tilde{\boldsymbol{x}}\sim\mathbb{P}_g}[D(\tilde{\boldsymbol{x}})]$. They showed the generator can learn when the critic performs well. However, to enforce the constraint $\|f\|_L \leq 1$, $D(x)$ has to be a 1-Lipschitz function and the weight values of the critic should be clipped within an interval [-*c*, *c*]. Because of the Lipschitz constraint, if the hyperparameter *c* is not set correctly, the model might lead low quality and not converge (Arjovsky et al., 2017). To overcome the sensitivity to the hyperparameter *c* of the WGAN, a gradient penalty for the WGAN was proposed for the so-called WGAN-GP rather than clipping the weight to enforce the Lipschitz constraint (Gulrajani et al., 2017). The minimax game of the WGAN-GP is represented with the loss function of the critic $L^{WGAN-GP}(D, G)$ as:

$$\begin{aligned} \min_G \max_D L^{WGAN-GP}(D, G) \\ &= \mathbb{E}_{\tilde{\boldsymbol{x}}\sim\mathbb{P}_g}[D(\tilde{\boldsymbol{x}})] - \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_r}[D(\boldsymbol{x})] \\ &+ \lambda \mathbb{E}_{\hat{\boldsymbol{x}}\sim\mathbb{P}_{\hat{x}}}[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2], \end{aligned} \tag{1.4}$$

where the first two terms in the right-hand side of the equation represent the original critic loss, and the last term is the gradient penalty used to satisfy a differentiable function *f* (see the details in Gulrajani et al. (2017)), and $\hat{\boldsymbol{x}} = t\boldsymbol{x} + (1-t)\tilde{\boldsymbol{x}}$ with $0 \leq t \leq 1$, i.e., $\hat{\boldsymbol{x}}$ is sampled from $\tilde{\boldsymbol{x}}$ and

$\boldsymbol{x}$ with t uniformly sampled between 0 and 1 (i.e. $t \sim U[0,1]$). λ is called the gradient penalty coefficient. If λ is high, the penalty term can have more dominant effect on the loss function than the original critic loss. If λ is low, the Lipschitz continuity is not enforced enough.

Despite such attempts to stabilize the GANs, it is difficult to reach an equilibrium state that satisfies the learning of critic (discriminator) and generator. Also, the network structure of the critic (discriminator) is important since it is heavily involved in training the GANs (Creswell et al., 2018; Hong et al., 2019). In that sense, it is essential to select the proper network structure for successful training of the GANs. Given these difficulties, the GANs may cause poor performance of EEG classifiers by augmenting artificial data generated by the poorly trained GANs.

1.3. Methods

1.3.1. cGANs

The minimax game of the original cGAN is presented in formula (1.2). Based on formula (1.2) and (1.4), the problem of cWGAN can be represented as:

$$\min_G \max_D L^{cWGAN}(D,G) = \mathbb{E}_{\tilde{\boldsymbol{x}}\sim\mathbb{P}_g,\boldsymbol{y}\sim\mathbb{P}_{\boldsymbol{y}}^{\boldsymbol{train}}}[D(\tilde{\boldsymbol{x}}|\boldsymbol{y})] - \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_{\boldsymbol{r}}^{\boldsymbol{train}},\boldsymbol{y}\sim\mathbb{P}_{\boldsymbol{y}}^{\boldsymbol{train}}}[D(\boldsymbol{x}|\boldsymbol{y})], \tag{1.5}$$

where $\tilde{\boldsymbol{x}}$ is a fake data that follows the fake data distribution $\mathbb{P}_g$, $\boldsymbol{x}$ is a real training EEG sample from the distribution $\mathbb{P}_{\boldsymbol{r}}^{\boldsymbol{train}}$, and $\boldsymbol{y}$ is the class label that follows the real training label distribution $\mathbb{P}_{\boldsymbol{y}}^{\boldsymbol{train}}$.

Similarly, the game of cWGAN-GP is also defined with the gradient penalty as:

$$\min_G \max_D L^{cWGAN-GP}(D,G) = L^{cWGAN}(D,G) + \lambda\mathbb{E}_{\hat{\boldsymbol{x}}\sim\mathbb{P}_{\hat{\boldsymbol{x}}},\boldsymbol{y}\sim\mathbb{P}_{\boldsymbol{y}}^{\boldsymbol{train}}}\left[\left(\left\|\nabla_{\hat{\boldsymbol{x}}|\boldsymbol{y}}D(\hat{\boldsymbol{x}}|\boldsymbol{y})\right\|_2 - 1\right)^2\right], \tag{1.6}$$

where λ is the gradient penalty coefficient that can control the trade-off between the original loss of the cWGAN and the GP, and $\hat{\boldsymbol{x}}$ given the class label $\boldsymbol{y}$ is blended from a real data $\boldsymbol{x}_r^{train}$ and a generated data $\tilde{\boldsymbol{x}}$ with a uniform sample t in $U[0,1]$ as:

$$\hat{\boldsymbol{x}}|\boldsymbol{y} = t \cdot \boldsymbol{x}_r^{train}|\boldsymbol{y} + (1-t) \cdot \tilde{\boldsymbol{x}}|\boldsymbol{y}. \tag{1.7}$$

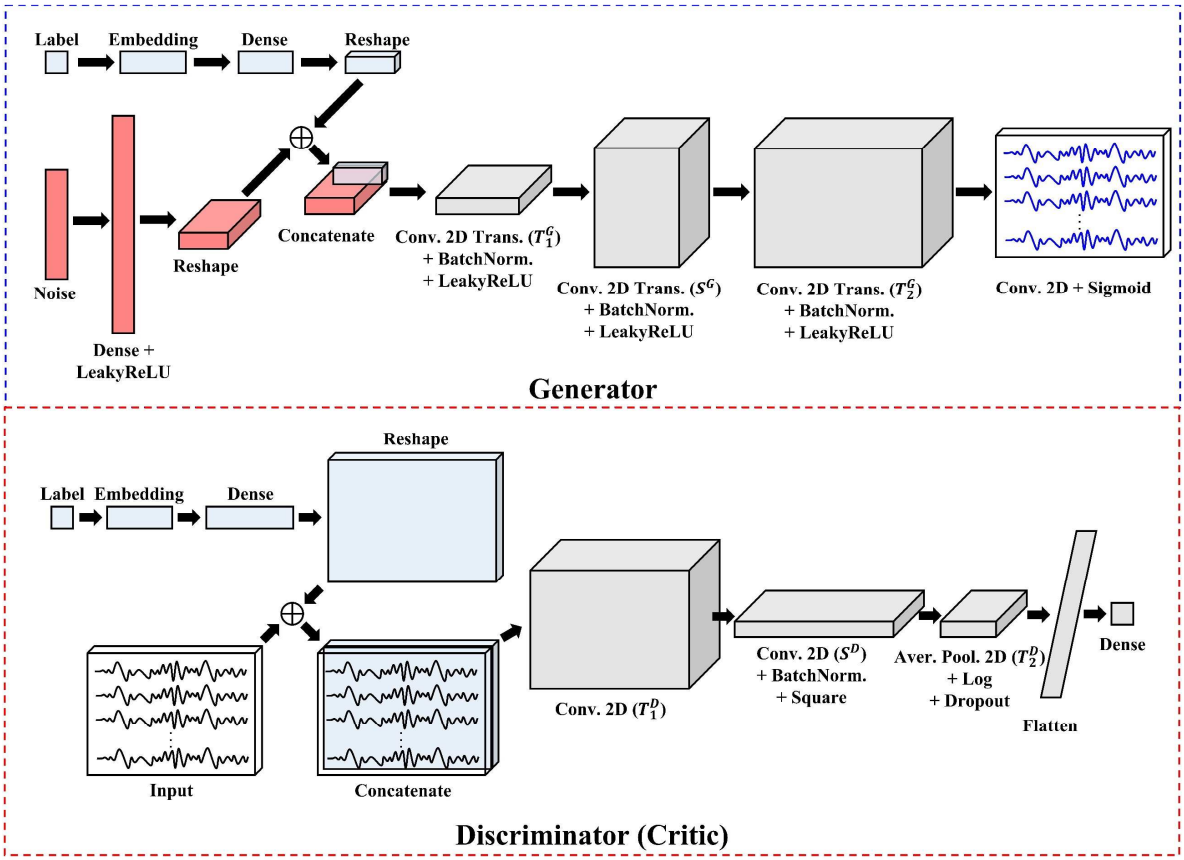


Figure 1.3. Proposed network architecture of generator and discriminator for cGANs.

Table 1.1. The architecture of the proposed cGAN.

Model	Module	Layer/ Operation	Kernel (Stride) Size	Input Size	Output Size
Gener.	Label Emb.	Embedding	-	y	$1 \times n_{eb}$
		Dense	-	$1 \times n_{eb}$	$1 \times (n_{tp}/8)$
		Reshape	-	$1 \times (n_{tp}/8)$	$1 \times (n_{tp}/8) \times 1$
	Noise Encoder	Dense +LeakyReLu	-	n_{ns}	$n_{tp} \cdot n_f^{G,T_1}/8$
		Reshape	-	$n_{tp} \cdot n_f^{G,T_1}/8$	$1 \times (n_{tp}/8) \times n_f^{G,T_1}$
	Trans. Conv.	Concatenate	-	$1 \times (n_{tp}/8) \times 1$ $1 \times (n_{tp}/8) \times n_f^{G,T_1}$	$1 \times (n_{tp}/8) \times (n_f^{G,T_1} + 1)$
		Conv2DTrans (T_1^G) +BatchNorm +LeakyReLU	$(1 \times s_{T_1}^G)$	$1 \times (n_{tp}/8) \times (n_f^{G,T_1} + 1)$	$1 \times (n_{tp} \cdot s_{T_1}^G/8) \times n_f^{G,T_1}$
		Conv2DTrans (S^G) +BatchNorm +LeakyReLU	$(s_S^G \times 1)$	$1 \times (n_{tp} \cdot s_{T_1}^G/8) \times n_f^{G,T_1}$	$s_S^G \times (n_{tp} \cdot s_{T_1}^G/8) \times n_f^{G,S}$
		Conv2DTrans (T_2^G) +BatchNorm +LeakyReLU	$(1 \times s_{T_2}^G)$	$s_S^G \times (n_{tp} \cdot s_{T_1}^G/8) \times n_f^{G,S}$	$s_S^G \times (n_{tp} \cdot s_{T_1}^G \cdot s_{T_2}^G/8) \times n_f^{G,T_2}$
		Conv2D +Sigmoid	1×1 (1×1)	$s_S^G \times (n_{tp} \cdot s_{T_1}^G \cdot s_{T_2}^G/8) \times n_f^{G,T_2}$	$n_{ch} \times n_{tp} \times 1$
Discr.	Label Emb.	Embedding	-	y	$1 \times n_{eb}$
		Dense	-	$1 \times n_{eb}$	$1 \times (n_{ch} \cdot n_{tp})$
		Reshape	-	$1 \times (n_{ch} \cdot n_{tp})$	$n_{ch} \times n_{tp} \times 1$
	EEG Decoder	Concatenate	-	$n_{ch} \times n_{tp} \times 1$ $n_{ch} \times n_{tp} \times 1$	$n_{ch} \times n_{tp} \times 2$
		Conv2D(T_1^D)	$1 \times k_{T_1}^D$ $(1 \times s_{T_1}^D)$	$n_{ch} \times n_{tp} \times 2$	$n_{ch} \times n_{tp} \times n_f^{D,T_1}$
		Conv2D(S^D) +BatchNorm +Square	$k_S^D \times 1$ $(1 \times s_S^D)$	$n_{ch} \times n_{tp} \times n_f^{D,T_1}$	$1 \times n_{tp} \times n_f^{D,T_1}$
		AvgPool2D(T_2^D) +Log+Dropout	$1 \times P_{T_2}^D$	$1 \times n_{tp} \times n_f^{D,T_1}$	$1 \times floor\left(\frac{n_{tp} - p_{T_2}^D}{S_{T_2}^D} + 1\right) \times n_f^{D,T_1}$
		Flatten	-	$1 \times floor\left(\frac{n_{tp} - p_{T_2}^D}{S_{T_2}^D} + 1\right) \times n_f^{D,T_1}$	$floor\left(\frac{n_{tp} - p_{T_2}^D}{S_{T_2}^D} + 1\right) \times n_f^{D,T_1}$
		Dense	-	$floor\left(\frac{n_{tp} - p_{T_2}^D}{S_{T_2}^D} + 1\right) \times n_f^{D,T_1}$	1

1.3.2. Network Architectures for Generator and Discriminator

Figure 1.3 visually summarizes the proposed network architectures of the generator and the discriminator (critic) for cGANs. On the other hand, Table 1.1 shows the architectures with kernel (stride), input, and output sizes depending on layers. In this table, n_{ch}, n_{tp}, n_{eb}, and n_{ns} represent the size of EEG channel, time point, embedding node, and noise, respectively. y indicates class label. T_1, T_2, and S are the first, the second temporal filtering layers, and the spatial filtering layer, respectively. Finally, s, k, p, and n_f represent stride size, kernel size, pooling size, and number of filters. We were inspired by the network structure of the ShallowConvNet (Schirrmeister et al., 2017) that uses two temporal filtering layers and one spatial filtering layer to extract EEG features. This is because the use of separate filters for spatial and temporal dimensions can make the network to be computationally more efficient, as long as it is accurate to assume independence of the dimensions along which the filters are consecutively convolved over (Chollet, 2017). For the generator, it is assumed that the size of the target n_{tp} is a multiple of 8 to reduce the complexity of the model. As in the generator, the class label (y) is passed through an embedding layer to map it to a unique element vector. Then, it is fed into the fully connected layer with a linear activation followed by reshaping to match the output size with the output by the noise input. Given the size of latent space, the noise is passed through a dense layer with a LeakyReLU activation followed by reshaping layer. The reshaped two inputs (class label and noise) are concatenated to create a noise tensor including information of class label. This merged tensor is pass through two temporal filtering layers (T_1^G and T_S^G) and one spatial filtering layer (S^G), which the layers include batch normalization and LeakyReLU activation all. Finally, it is fed into a convolutional layer with sigmoid activation to generate the same data structure for the preprocessed EEG signals. As in the discriminator (critic), the network structure is similar to the

one of the ShallowConvNet (Schirrmeister et al., 2017). But, like the network structure of the generator, it includes class label networks and concatenation layer to merge the input and class information. Also, the final dense layer is modified as a binary classification layer that can play a role in classifying real and fake data in the cGANs. All other structure and configurations were kept as the ShallowConvNet (for details, see Schirrmeister et al. (2017)).

Algorithm 1.1. Minibatch training of cWGAN-GP

Input: batch size *m*, the maximum number of iterations of critic, n_{critic}, real EEG training data distribution ($\mathbb{P}_{\boldsymbol{r}}^{train}, \mathbb{P}_{\boldsymbol{y}}^{train}$), and prior noise distribution $\mathbb{P}_{\boldsymbol{z}}$

Output: trained generator G^*

Initialize critic $D(\boldsymbol{x}; \boldsymbol{\theta_D})$ and generator $G(\boldsymbol{z}; \boldsymbol{\theta_G})$ with their initial parameters $\boldsymbol{\theta_D^0}$ and $\boldsymbol{\theta_G^0}$
while Eq. (6) have not converged **do**
  for k = 1 **to** n_{critic} **do**
    Sample a batch of real data, $\left\{(\boldsymbol{x_r}^{(i)}, \boldsymbol{y}^{(i)})\right\}_{i=1}^{m} \sim (\mathbb{P}_{\boldsymbol{r}}^{train}, \mathbb{P}_{\boldsymbol{y}}^{train})$
    Sample a batch of noises, $\{\boldsymbol{z}^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_z$
    Sample a batch of blending values, $\{t^{(i)}\}_{i=1}^{m} \sim U[0,1]$
    for i = 1 **to** m **do**
      Generate a sample, $\tilde{\boldsymbol{x}} \leftarrow G(\boldsymbol{z}^{(i)}, \boldsymbol{y}^{(i)}; \boldsymbol{\theta_G})$
      Blend real data and artificial data, $\hat{\boldsymbol{x}}|\boldsymbol{y}^{(i)} \leftarrow t^{(i)} \cdot \boldsymbol{x_r}^{(i)}|\boldsymbol{y}^{(i)} + (1 - t^{(i)}) \cdot \tilde{\boldsymbol{x}}|\boldsymbol{y}^{(i)}$ # Eq. (1.7)
      Calculate the loss, $L^{(i)} \leftarrow D(\tilde{\boldsymbol{x}}|\boldsymbol{y}^{(i)}; \boldsymbol{\theta_D}) - D(\boldsymbol{x_r}^{(i)}|\boldsymbol{y}^{(i)}; \boldsymbol{\theta_D}) + \lambda \left(\left\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}}|\boldsymbol{y}^{(i)})\right\|_2 - 1\right)^2$ #Eq. (1.6)
    end for
    $\boldsymbol{\theta_D} \leftarrow \text{Adam}(\nabla_{\boldsymbol{\theta_D}} \frac{1}{m} \sum_{i=1}^{m} L^{(i)}, \boldsymbol{\theta_D})$
    Update the critic $D(\boldsymbol{x}|\boldsymbol{y}; \boldsymbol{\theta_D})$ with updated parameters $\boldsymbol{\theta_D}$
  end for
  Sample a batch of real label, $\{\boldsymbol{y}^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_{\boldsymbol{y}}^{train}$
  Sample a batch of noises, $\{\boldsymbol{z}^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_z$
  $\boldsymbol{\theta_G} \leftarrow \text{Adam}(\nabla_{\boldsymbol{\theta_G}} \frac{1}{m} \sum_{i=1}^{m} -D(G(\boldsymbol{z}^{(i)}, \boldsymbol{y}^{(i)})), \boldsymbol{\theta_G})$
  Update the generator $G(\boldsymbol{z}, \boldsymbol{y}; \boldsymbol{\theta_G})$ with updated parameters $\boldsymbol{\theta_G}$
end while
return $G(\boldsymbol{z}, \boldsymbol{y})$

1.3.3. Training Algorithm

The minibatch training algorithm for the cWGAN-GP is described in Algorithm 1.1. The algorithm uses the Adam optimization (Kingma & Ba, 2014) to minimize the loss of the critic (discriminator) and the generator. Given the Algorithm 1.1, a generator G is trained to fit EEG training data based on cWGAN-GP.

In Algorithm 1.1, it starts with initializing critic $D(\boldsymbol{x}; \boldsymbol{\theta_D})$ and generator $G(\boldsymbol{z}; \boldsymbol{\theta_G})$ with their initial parameters $\boldsymbol{\theta_D^0}$ and $\boldsymbol{\theta_G^0}$ (line 1). Real data, noises, and blending values are sampled as much as the batch size m based on the predefined distributions (line 3-6). Given these samples, each loss for the batch is computed with real data, artificial data, and blended data samples (line 7-10). With the Adam optimizer and the mean loss from the batch, the parameters of the critic $\boldsymbol{\theta_D}$ is updated (line 12-13). This updating process for the critic repeats n_{critic} times. After updating the parameters of the critic $\boldsymbol{\theta_D}$, real label data and noises are sampled with the batch size m given the distributions. With the samples, the mean loss is calculated and the parameters of the generator $\boldsymbol{\theta_G}$ is trained with the Adam optimizer (line 15-18). The whole training processes proceed until the loss (Eq. (1.6)) converges. The cGAN-based artificial data can be generated by the generator G^* that was trained through Algorithm 1.1 as:

$$\boldsymbol{x_{cGAN}^{DA}} | \boldsymbol{y} \sim G^*(\boldsymbol{z}, \boldsymbol{y}; \boldsymbol{\theta_G^*}), \tag{1.8}$$

The algorithm for training cWGAN-GP can be converted easily to other cGAN family such as cGAN and cWGAN by replacing the loss function in line 10 with their loss functions.

1.4. Experimental Settings

1.4.1. Datasets and Preprocessing

Two public benchmarking datasets were used to evaluate the proposed DA method because most studies to develop EEG classifiers for MI BCI have utilized them for the validation (Schirrmeister et al., 2017; Lawhern et al., 2018; Zhao et al., 2019; Amin et al., 2019; Dai et al., 2020; Jia et al., 2021; Ko et al., 2021).

Dataset 1 (BCI Competition IV Dataset IIa; Tangermann et al., 2012***):*** The dataset includes a total of four classes imagined movements (left hand, right hand, feet, and tongue) from nine subjects. There were 144 trials per class per subject. The EEG signals were recorded with 22 Ag/AgCl electrodes measured at positions of the international 10-20 system. The EEG signals were originally sampled at 250 Hz and bandpass-filtered between 0.5 Hz and 100 Hz (for details, see http://www.bbci.de/competition/iv/).

Dataset 2 (BCI Competition III Dataset IVa; Dornhege et al., 2004***):*** The dataset contains two classes of MI (right hand and foot) from five subjects. There were 140 trials per class per subject. 118 EEG channels were measured at positions of the international 10-20 system. The signals were originally sampled at 100 Hz and bandpass-filtered between 0.05 and 200 Hz (for details, see http://www.bbci.de/competition/iii/).

The EEG signals were resampled to 128 Hz for dataset 1. We minimally preprocessed the datasets, applying a third-order Butterworth filter in the 4-40 Hz frequency bands to minimize eye movement artifacts. The channel-wise min-max normalization was performed. Suppose $\boldsymbol{s_{norm}} \in \mathbb{R}^{C \times P}$ denote the normalized EEG signal, where C and P are the numbers of channels and time samples, respectively. We epoched the preprocessed data $\boldsymbol{X_r}$ at (0.5, 2.5) seconds post-cue onset (the same window epochs used in Lotte (2015) and Lawhern et al. (2018)) from the normalized

EEG signals $\boldsymbol{s_{norm}}$ to extract both training and test datasets. We only used this time range for both training and testing the datasets. We merged both training and test dataset for each dataset to conduct experiments. We then split the merged dataset for each into reorganized training data $\boldsymbol{X_r^{train}}$ and test data $\boldsymbol{X_r^{test}}$ depending on cross-validation. Also, the corresponding labels of the training and test data are denoted by $\boldsymbol{Y_r^{train}}$ and $\boldsymbol{Y_r^{test}}$.

1.4.2. EEG DA Methods

Resampling (RS): RS represents augmenting training data of EEG by sampling the new data $\boldsymbol{x_{RS}^{DA}}$ from the real EEG training data $\boldsymbol{X_r^{train}}$. It is simply defined as:

$$\boldsymbol{x_{RS}^{DA}} \in \boldsymbol{X_r^{train}} \tag{1.9}$$

This method was rarely explored to augment EEG data. However, in the experiment we will investigate the effect of this method for the CNN-based EEG classifiers since it is much simpler than all other DA methods.

Noise addition (NA): NA is a powerful and simple way to escalate the amount of EEG training samples due to randomness and non-stationarity of the EEG data. The generation of the noise and the NA to the EEG signals are defined as, respectively:

$$z^{noise} \sim N(\mu, \sigma^2) \tag{1.10}$$

$$\boldsymbol{x_{NA}^{DA}} = \langle x_{NA,k}^{DA} \rangle \tag{1.11}$$

where
$$x_{NA,k}^{DA} = \begin{cases} 1, & x_{r,k}^{train} + z^{noise} > 1 \\ 0, & x_{r,k}^{train} + z^{noise} < 0 \\ x_{r,k}^{train} + z^{noise}, & otherwise \end{cases} \tag{1.12}$$

and $$\boldsymbol{x_r^{train}} = \langle x_{r,k}^{train} \rangle \in \boldsymbol{X_r^{train}}. \quad (1.13)$$

z^{noise} is a noise that follows the Gaussian distribution $N(\mu, \sigma^2)$ with mean value μ and standard deviation σ. $x_{r,k}^{train}$ is the *k*-th element of a real EEG training data $\boldsymbol{x_r^{train}}$ sampled from $\boldsymbol{X_r^{train}}$, and $x_{NA,k}^{train}$ is the truncated (clipped) value to [0, 1] after adding z^{noise} to the $x_{r,k}^{train}$. Finally, we can obtain $\boldsymbol{x_{NA}^{DA}}$, which the generated sample from $\boldsymbol{x_r^{train}}$ through NA.

We considered six cases for Gaussian noise ($\mu = 0.5; \sigma = 0.001, 0.01, 0.02, 0.1, 0.2, 0.5$) for augmenting EEG training datasets in the experiment. The mean value of the Gaussian noise was set to 0.5 since the range of the normalized EEG signals was [0, 1]. The standard deviation of the Gaussian noise was set to 0.001, 0.01, 0.02, 0.1, 0.2, and 0.5 to investigate the effect of different noise intensity (the same noise intensity used in Wang et al. (2018)).

cDCGAN (Fahimi et al., 2021)***:*** We implemented the cDCGAN model for DA (Fahimi et al., 2021) and customized it for our experiments. Their proposed model is for subject-independent classification which conditioned on the subjects not EEG class labels. Thus, their model is a general GAN model for subject-dependent classification. Also, they predefined the number of channels and the time points for their study, which are not same as our validation datasets. Given this sense, we customized their GAN model for our experiments. For the generator, we set the number of nodes for the initial two fully connected layer (first layer, second layer) as (100, 1408) and (200, 5900) for dataset 1 and dataset 2, respectively. The outputs are reshaped to (1, 64, 22) and (1, 50, 118) for each dataset, respectively. Then, the reshaped output is fed into two convolution layers with a kernel size of 5 and filters as many as the number of EEG channels for datasets. For the discriminator, we used as many filters as the number of EEG channels for the first convolution layer whereas we used twice as many filters as the number of the channels for the

second convolution layer. Besides above mentioned, all structures and configurations followed their model.

cGAN family (proposed): We built three cGANs (cGAN, cWGAN, and cWGAN-GP) based on Section 1.3. All used parameter values for the cGANs were listed in Table 1.2: for input and output sizes of each layer for both tested datasets, refer to Table A.1 in Appendix A.

Table 1.2. Parameter settings of cGANs for datasets.

Group	Hyperparameter	Description	Dataset 1	Dataset 2
Predefined by Data	y	Size of class label	4	2
	n_{eb}	Size of embedding node	50	50
	n_{ch}	Size of EEG channel	22	118
	n_{tp}	Size of time point	256	200
	n_{ns}	Size of noise vector	100	100
Generator	$n_f^{G,T_1}, n_f^{G,T_2}, n_f^{G,S}$	Number of filters for T_1, T_2, and S layers in G	40	40
	s_S^G	Stride size of S^G layer	22	118
	$s_{T_1}^G$	Stride size of T_1^G layer	4	4
	$s_{T_2}^G$	Stride size of T_2^G layer	2	2
	k_S^G	Kernel size of S^G layer	4	6
	$k_{T_1}^G$	Kernel size of T_1^G layer	13	13
	$k_{T_2}^G$	Kernel size of T_2^G layer	35	35
Discriminator (Critic)	$n_f^{D,T_1}, n_f^{D,T_2}, n_f^{D,S}$	Number of filters for T_1, T_2, and S layers in D	40	40
	s_S^D	Stride size of S^D layer	1	1
	$s_{T_1}^D$	Stride size of T_1^D layer	13	13
	$s_{T_2}^D$	Stride size of T_2^D layer	7	7
	k_S^D	Kernel size of S^D layer	22	22
	$k_{T_1}^D$	Kernel size of T_1^D layer	13	13
	$p_{T_2}^D$	Kernel size of T_2^D layer	35	35
	n_{critic}	Number of iterations to update critic	5	5
Layers for LeakyReLU and Batch Normalization	α for LeakyReLU	Negative slope coefficient	0.2	0.2
	Epsilon for batch norm.	Small float added to variance to avoid dividing by zero	10^{-5}	10^{-5}
	Moment for batch norm.	Momentum for the moving average	0.1	0.1
cWGANs	λ	Gradient penalty coefficient for cWGAN-GP	10	10
	c	Clipping value for cWGAN	0.1	0.1

1.4.3. EEG Classifiers

To evaluate the effectiveness of the DA methods, we utilized diverse MI EEG classifiers including traditional methods and state-of-the-art DL-based methods.

CSP + LDA / SVM: We built a common spatial pattern (CSP), which is most typically used to extract MI EEG features associated with spatial filters (Blankertz et al., 2008), with two traditional machine learning techniques (linear discriminant analysis (LDA) and support vector machine (SVM)). To select the best number of components to decompose EEG signals for the CSP and the best hyperparameters for the LDA and the SVM, 5-fold cross-validation with a grid search was performed in the training dataset. For the CSP, ten components (from 1 to 10 with 1 scale) were considered. In the LDA, we considered three solvers (singular value decomposition, least squares, and eigenvalue decomposition) and 101 shrinkage values (from 0 to 1 with a 0.01 scale). For the SVM, four regularization parameters (0.1, 1, 10, and 100), four kernel functions (linear, polynomial, radial basis, and sigmoid), and four kernel coefficients (0.001, 0.01, 0.1, and 1) were considered.

xDAWN + RS + LR: We built a combination of xDAWN spatial filtering (Rivet et al., 2009) and Riemannian geometry (Barachantt et al., 2012; Barachant & Congedo, 2014) with ElasticNet with logistic regression (LR) as an additional conventional EEG classifier, which is winning approach for the Kaggle BCI competition (see for details for code and implementation https://github.com/alexandrebarachant/bci-challenge-ner-2015). We also performed 5-fold cross-validation with the grid search to get the best hyperparameters for xDAWN, the ElasticNet with the LR. For xDAWN, ten filters per class (from 1 to 10 with scale 1) were considered. In the ElasticNet with the LR, we considered eleven L1 ratios (from 0 to 1 with 0.1 scale).

EEGNet / ShallowConvNet / DeepConvNet: We implemented EEGNet (Lawhern et al., 2018), a compact CNN structure for EEG classification. This model is an end-to-end network that can automatically extract/train temporal and spatial EEG features and classify them (see Lawhern et al. (2018) for more details). We also implemented ShallowConvNet and DeepConvNet, developed by Schirrmeister et al. (2017) for EEG classification. They are also types of the end-to-end network, including automatically temporal and spatial EEG feature extractions. However, their network architectures are larger than EEGNet, so that they have more trainable parameters. Also, DeepConvNet has deeper network architecture than ShallowConvNet (see Schirrmeister et al. (2017) for more details). We only modified the number of EEG channels and the time points for those DL models to fit the preprocessed EEG signals for our experiments. The remaining hyperparameters were set with the default settings selected by them.

1.4.4. Synthetic Data Proportion of DA

Let synthetic data proportion in training dataset (p_{tr}) be the number of synthetic data by DA (n_{DA}) divided by the sum of the number of training data (n_{tr}) and n_{DA}. Also, we assume that n_{DA} is equal to n_{tr} when $p_{tr} = 1$ that represents we use only synthetic dataset to train the classifier with the same amount of original training dataset. Then, the number of synthetic data can be derived by:

$$n_{DA} = \begin{cases} \frac{p_{tr} n_{tr}}{1 - p_{tr}}, & 0 \leq p_{tr} < 1 \\ n_{tr}, & p_{tr} = 1 \end{cases}. \tag{1.14}$$

A total of eleven cases for the proportion (0, 0.1, 0.2, …, 1) were considered. $p_{tr} = 0$ represents non-DA whereas $p_{tr} = 1$ represents only synthetic data.

1.4.5. Validation

We performed subject-dependent classification with 5-fold cross-validation depending on EEG classifiers, DA methods, and synthetic data proportion in the training dataset to assess each DA method's performances and the effects of the synthetic data proportion on the performance of the classifiers. The extracted mean performances by the cross-validation took the mean for all subjects and used these values as result analysis. Accuracy and area under curve (AUC) were used as performance metrics for dataset 1 (4-class) and dataset 2 (2-class), respectively. We used the Adam optimizer (Kingma & Ba, 2014) to train all DL models with default settings, a minibatch size of 16, and 500 epochs. For the conventional EEG classifiers, only the best models were selected by cross-validation to report the results. Also, we reported only the best results for the NA. All experiments were performed with the following hardware specifications: CPU (Intel i-7 8750H), GPU (NVIDIA GeForce RTX 2080 with Max-Q Design), and memory (32GB).

1.5. Results

1.5.1. Best DA Method for EEG Classifiers

In this section, we describe the details of the best DA techniques for MI EEG classifiers in terms of each dataset as shown in Table 1.3. In dataset 1, cWGAN-GP was the best DA method for all MI EEG classifiers (LDA: 0.648 ± 0.177, SVM: 0.639 ± 0.175, LR: 0.761 ± 0.101, EEGNet: 0.820 ± 0.111, ShallowConvNet: 0.780 ± 0.136, and DeepConvNet: 0.748 ± 0.108), which had the highest classification accuracy for each classifier. In the dataset 2, cWGAN was the best DA method for DeepConvNet (0.883 ± 0.109), which had the highest AUC value. Besides DeepConvNet, cWGAN-GP was the best DA model for the other classifiers (LDA: 0.935 ± 0.075, SVM: 0.937 ± 0.081, LR: 0.716 ± 0.109, EEGNet: 0.941 ± 0.059, and

ShallowConvNet: 0.955 ± 0.078). In addition, the proposed cGAN family for MI EEG DA showed higher performances for the classifiers than the ones by the cDCGAN.

Table 1.3. Mean performances of MI EEG classifiers applying DA methods with the best DA and the worst DA for each dataset.

The parentheses represent the standard deviation.

Dataset	Classifier	DA Methods							Best DA		Worst DA	
		Non-DA	RS	NA	cDCGAN	cGAN	cWGAN	cWGAN-GP	Method	DA rate	Method	DA rate
Dataset 1 (Acc.)	LDA	0.606 (0.173)	0.635 (0.173)	0.609 (0.173)	0.614 (0.181)	0.621 (0.176)	0.629 (0.182)	0.648 (0.177)	cWGAN-GP	0.1	Non-DA	0
	SVM	0.608 (0.174)	0.615 (0.162)	0.568 (0.139)	0.594 (0.166)	0.616 (0.170)	0.615 (0.172)	0.639 (0.175)	cWGAN-GP	0.3	NA	0.1
	LR	0.700 (0.106)	0.735 (0.098)	0.735 (0.092)	0.719 (0.096)	0.734 (0.091)	0.744 (0.096)	0.761 (0.101)	cWGAN-GP	0.2	Non-DA	0
	EEG.	0.740 (0.120)	0.802 (0.108)	0.789 (0.111)	0.804 (0.100)	0.810 (0.103)	0.819 (0.103)	0.820 (0.111)	cWGAN-GP	0.8	Non-DA	0
	Shall.	0.721 (0.140)	0.767 (0.124)	0.743 (0.120)	0.750 (0.120)	0.760 (0.128)	0.771 (0.127)	0.780 (0.136)	cWGAN-GP	0.6	Non-DA	0
	Deep.	0.619 (0.134)	0.735 (0.103)	0.712 (0.094)	0.731 (0.089)	0.737 (0.098)	0.744 (0.095)	0.748 (0.108)	cWGAN-GP	0.4	Non-DA	0
Dataset 2 (AUC)	LDA	0.910 (0.081)	0.919 (0.075)	0.885 (0.064)	0.922 (0.078)	0.929 (0.078)	0.932 (0.073)	0.935 (0.075)	cWGAN-GP	0.5	NA	0.7
	SVM	0.901 (0.086)	0.908 (0.079)	0.867 (0.087)	0.914 (0.075)	0.919 (0.081)	0.926 (0.080)	0.937 (0.081)	cWGAN-GP	0.4	NA	0.9
	LR	0.680 (0.103)	0.715 (0.086)	0.667 (0.105)	0.693 (0.099)	0.694 (0.109)	0.711 (0.102)	0.716 (0.109)	cWGAN-GP	0.9	NA	0.8
	EEG.	0.806 (0.033)	0.919 (0.061)	0.877 (0.061)	0.932 (0.077)	0.935 (0.073)	0.934 (0.066)	0.941 (0.059)	cWGAN-GP	0.6	Non-DA	0
	Shall.	0.895 (0.135)	0.944 (0.081)	0.900 (0.062)	0.938 (0.082)	0.947 (0.074)	0.947 (0.076)	0.955 (0.078)	cWGAN-GP	0.6	Non-DA	0
	Deep.	0.662 (0.195)	0.874 (0.110)	0.820 (0.121)	0.877 (0.109)	0.879 (0.110)	0.892 (0.099)	0.883 (0.109)	cWGAN	0.9	Non-DA	0

1.5.2. Performance Comparison of EEG Classifiers by non-DA and DA

In order to show the effectiveness of the DA methods to improve the performance of the MI EEG classifiers, we compare the performances of the classifiers by non-DA with the ones by the worst DA and the best DA methods for each classifier in terms of the datasets. Based on Table 1.1, we extracted the classification performances of the MI EEG classifiers depending on non-DA, the worst DA, and the best DA methods as shown in Figure 1.4. NA was selected as the worst DA

method for SVM in dataset 1 and for LDA, SVM, and LR in dataset 2, which had the lower classification performances compared to the ones by non-DA. Besides that, all DA methods showed improved classification performances compared to the ones by non-DA. In particular, the best DA (cWGAN-GP) improved the performance of DeepConvNet for dataset 2 by 33% relative to the one of non-DA.

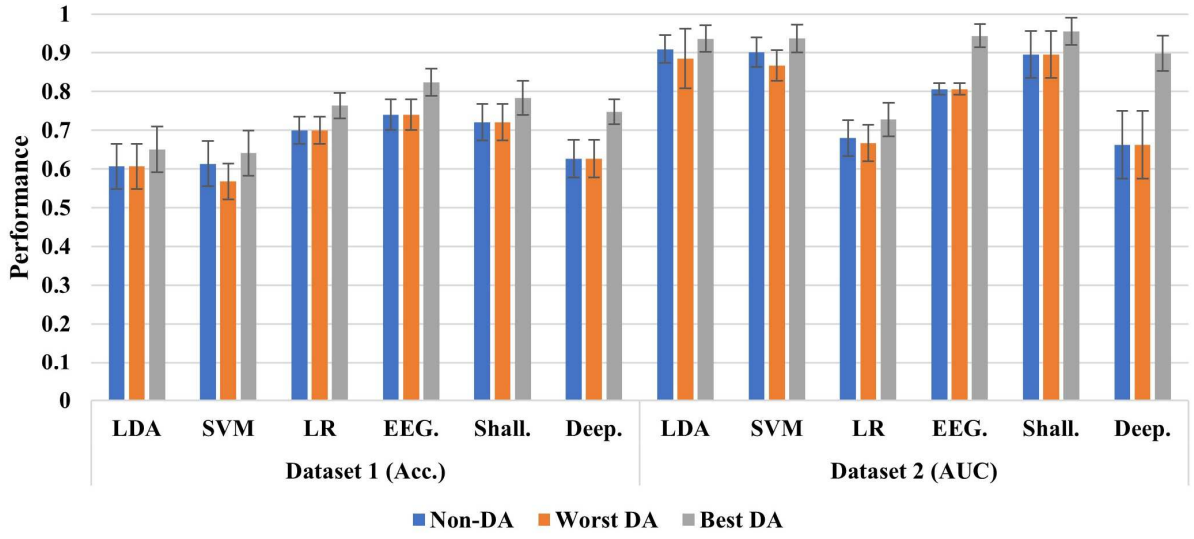
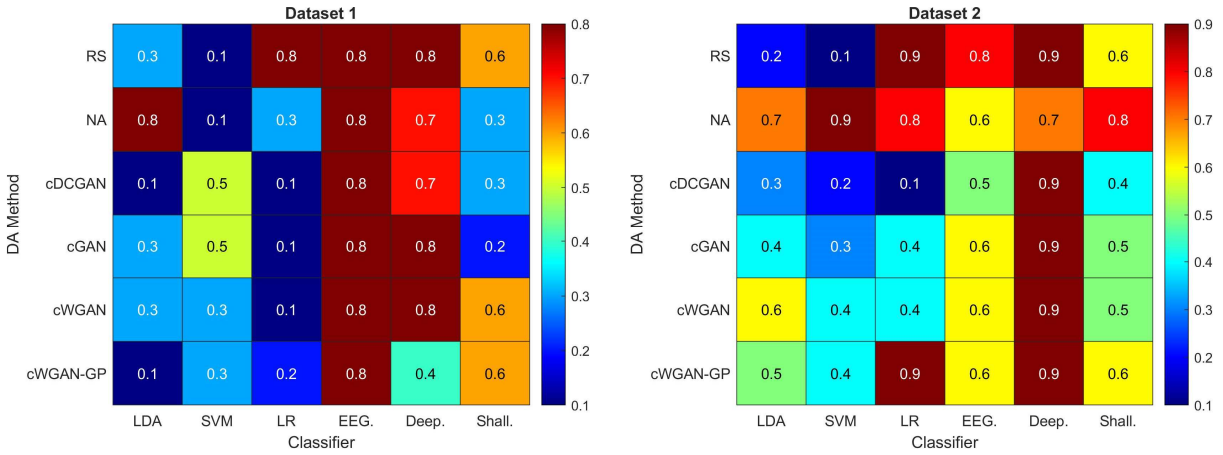


Figure 1.4. Performances of EEG classifiers for datasets depending on non-DA, worst DA, and best DA.

The plots show the mean performances and the standard errors for the classifiers. Blue, orange, and gray colors represent the performance results of the classifiers using non-DA, worst DA, and best DA, respectively.

1.5.3. Best DA Proportion for EEG Classifiers

The best DA proportions in the training dataset of the DA methods were extracted as shown in Figure 1.5. Depending on classifiers, DA methods, and datasets, the best synthetic data proportion varied. However, for the traditional MLs (LDA, SVM, and LR), the best proportion was less than 0.5 generally. On the other hand, for the DLs (EEGNet, ShallowConvNet, and DeepConvNet), the best proportion was greater than 0.5 in general.



(a) Dataset 1 **(b) Dataset 2**

Figure 1.5. Heatmap of best synthetic data proportion in training dataset depending on MI EEG classifiers, DA methods, and datasets.

Colors and their numbers represent the best proportion.

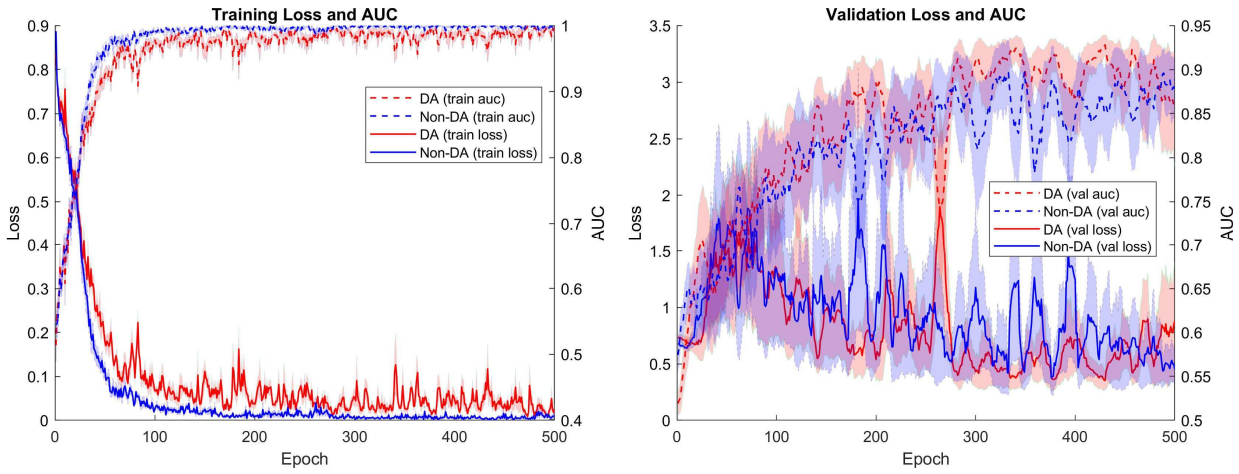
1.6. Discussion

1.6.1. Effectiveness of DA for EEG Classifiers

DA has been applied to improve the performance of the MLs and the DLs in many areas, such as image classification (Perez & Wang, 2017; Shorten & Khoshgoftaar, 2019) and cognitive states recognition using EEG features (Luo & Lu, 2018; Wang et al., 2018; Zhang & Liu, 2018; Gan et al., 2019; Luo et al., 2019; Al-Saegh et al., 2021a). Like those studies, we validated that DA could improve MI EEG classifiers using the MLs and the DLs through the experiments. This is because DA can contribute to new information for unseen data that can serve as regularization.

When using the DA methods, the DLs' performance improvement was much more significant than the traditional MLs. It is common sense that the performance of the DLs is constantly increasing with the amount of data, whereas one of the MLs is stagnant (Najafabadi et al., 2015). The DLs require much more data than the traditional MLs to accurately be trained since they include tons of parameters (Najafabadi et al., 2015). By taking large enough data through the

DA methods, the DLs might be trained well and show the higher performance than the traditional MLs. Therefore, the DA methods can be more effective for the DLs to improve their performance.



(a) Training **(b) Validation**

Figure 1.6. Training and validation performances of EEGNet classifier for subject 2 from dataset 2.

In addition, we investigated how the DA affects the DL-based EEG classifier through the performances of its training and validation as shown in Figure 1.6. For the figure, we used subject 2 data from dataset 2, EEGNet, and cWGAN-GP with 0.6 DA proportion. From the subject data, we used 80% training set and 20% validation set and extracted the training performances and the validation with 30 macro-replications. We observed that even if the use of the DA methods had more training loss and smaller training AUC by using the DA method, it had less validation loss and more validation AUC then the non-DA approach. This shows that the DA method is effective in avoiding the model's overfitting problem as a regularizer.

1.6.2. Effectiveness of the Proposed DA Method

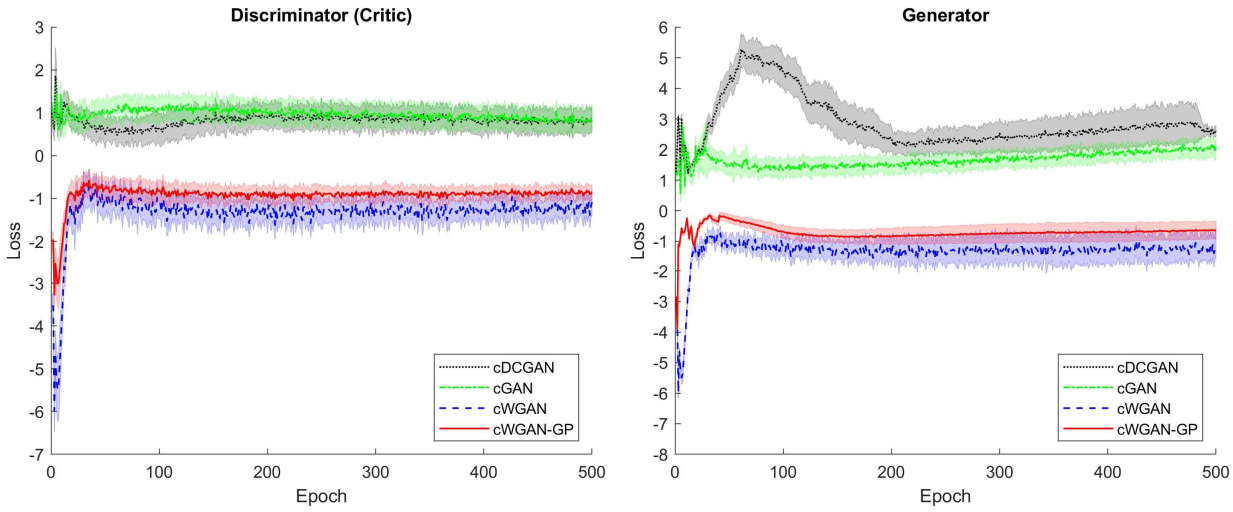
Among the DA techniques, cWGAN-GP was the best DA method for most EEG classifiers. This is because it can be given more stable training than cGAN and cWGAN by applying a gradient penalty (Gulrajani et al., 2017). The stable training would produce a better fit for the training datasets of EEG.

Other DA methods also showed their effectiveness for improving EEG classifiers as compared to the classification without DA. Despite the effectiveness of the existing DA methods, they still have several limitations. First, the RS method augments the training datasets by sampling from its data. This is a straightforward way to augment training datasets, but it might lead to an overfitting problem that can degrade the performance of the classifiers. This is because RS focuses on only the selected training datasets without exploring other possible datasets that might affect test performance (Bischl et al., 2012).

Second, the NA method adds Gaussian noise into the training datasets. Generally, it uses a parameter (σ) to adjust the noise rate. If the parameter is low in value, the generated datasets are similar to the original training datasets. On the contrary, if the parameter is high in value, it generates far different datasets than the original training datasets. Therefore, it is vital to determine the optimal parameter that can help to improve the classifiers (Wang et al., 2018). However, this DA method uses Gaussian distribution to create noise, but EEG signals have strong randomness and non-stationarity (Wang et al., 2018). In this respect, in strictly using Gaussian distribution, it is challenging to generate artificial EEG signals that retain their properties just only by using Gaussian distribution.

Third, the existing GAN model (cDCGAN) for MI EEG signals does not consider label data of the training datasets. It can cause the data imbalance problem from mode failure, which

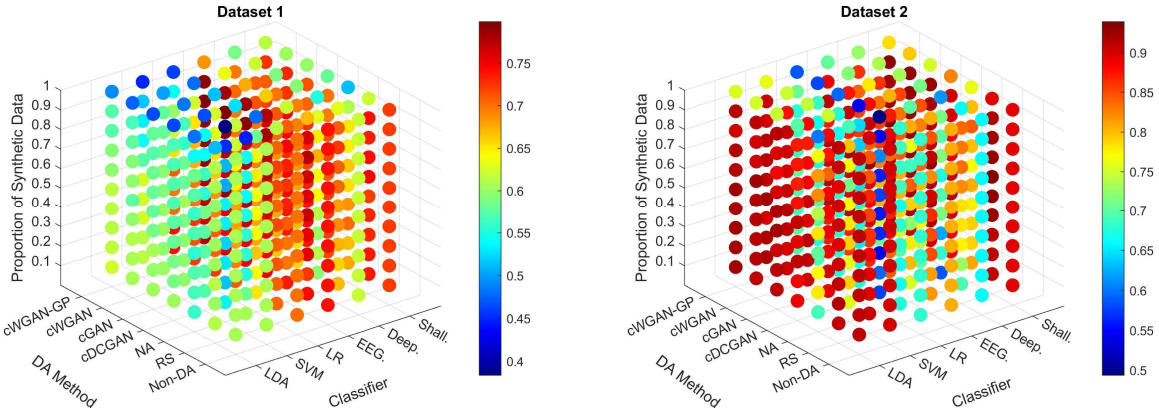
can cause overfitting problems (Mirza & Osindero, 2014). Also, training failure of a generator frequently occurs when using JSD for GAN model (Arjovsky et al., 2017). Figure 1.7 shows training losses of cDCGAN, cGAN, cWGAN, and cWGAN-GP for discriminator (critic) and generator. We observed that the proposed cGAN had more stable training (low variance) and lower loss in the generator than the cDCGAN. In the generator loss, their losses were not converged. However, cWGAN and cWGAN-GP showed more stable training with convergence than them in both discriminator and generator. In particular, cWGAN-GP had lower losses than cWGAN. These results show that our proposed model is more effective than the existing GAN model (cDCGAN) for MI EEG signals. Also, among the proposed method, cWGAN-GP can be the most promising DA method for MI EEG signals.



(a) Discriminator **(b) Generator**

Figure 1.7. Training losses for cGAN models in terms of discriminator and generator.

For this plot, subject 3 data from dataset 1 was used. A total of 30 macro-replication experiments were performed.



(a) Dataset 1 **(b) Dataset 2**

Figure 1.8. Sensitivity of synthetic data proportion from DA methods for EEG classifiers.

The colors for (a) and (b) represent accuracy and AUC, respectively.

1.6.3. Sensitivity Analysis of DA Proportion for EEG Classifiers

The synthetic data proportion in the training dataset is a vital hyperparameter when applying the DA method to the EEG classifiers. This is because it directly affects the performance of the classifiers depending on how much data it will be augmented. Figure 1.8 shows the performances of EEG classifiers by DA methods and DA proportions. It shows that even if the same DA method is used to augment, the performance of the classifiers varies according to the DA proportion. Therefore, setting an appropriate DA proportion is essential for the maximum performance improvement of the classifiers through the DA methods. As we confirmed in Section 1.5.3, the best performance was mainly shown in less than 0.5 of the conventional MLs whereas greater than 0.5 for the DLs. These proportion patterns show the ML and the DL properties, as we discussed in Section 1.6.1. In general, the performance of the ML according to data size is stagnant at some point, but the DL can improve its performance constantly with it (Najafabadi et al., 2015). Therefore, the MLs prefer a relatively smaller DA proportion to regularize the models, whereas

the DLs require a higher DA proportion to update lots of weight parameters for higher generalization.

1.6.4. Limitations and Future Directions

In this study, we showed the effectiveness of the proposed cGAN techniques by applying them to the EEG classifiers. We validated the proposed DA from the experimental results by comparing it with other DA techniques in terms of classification performances (accuracy for dataset 1 and AUC for dataset 2) without the quality check of the generated EEG signals. In general, it is very difficult to evaluate whether they are qualitatively or quantitatively well-generated or synthesized EEG signals. In computer vision, images can be evaluated well quantitatively and qualitatively. For example, users can assess the generated images directly. Also, the artificial images can be assessed quantitatively with numerical scores such as inception score (IS; Salimans et al., 2016) and the Frechet inception distance (FID; Heusel et al., 2017). In the EEG domain, there have been several studies to evaluate the qualities of the EEG signals using visual information (Hartmann et al., 2018; Luo & Lu, 2018; Aznan et al., 2019; Panwar et al., 2020), numerical scores (Hartmann et al., 2018; Luo & Lu, 2018), and improved model performance by the generated EEG data (Zhang & Liu, 2018). However, it is still impossible to visually evaluate the generated high-dimensional EEG signals' qualities (Fahimi et al., 2021). Also, numerical scores such as IS and FID were used to evaluate the qualities of the generated EEG signals (Hartmann et al., 2018; Luo & Lu, 2018), but there were no consensus indices to guarantee the generated EEG qualities. If the performance of the used EEG classifiers for those scores is not high enough, the results of the scores are not reliable. Given those limitations, we used the improved classification performance of the EEG classifiers to evaluate the qualities of the

generated EEG signals. This is because well-generated EEG signals will improve the performances of the classifiers by providing more training samples reflecting EEG properties and diversities. However, such improved performance-based methods cannot explain exactly what characteristics of the generated data helped improve the performance of the classifiers. Therefore, new quantitative or qualitative methods for the generated EEG signals are required to interpret the signals.

1.7. Conclusion

In this study, we proposed a novel MI EEG DA framework using cGAN to enhance the performances of the MI EEG classifiers, including conventional MLs and state-of-the-art DLs. The proposed framework generates artificial EEG signals conditioned on a specific class and has the same data structure of training input. An experimental study was implemented on two public EEG datasets with MI classification tasks depending on EEG classifiers, DA techniques, and DA proportion in the training dataset to validate the proposed DA method. The performances of the EEG classifiers were enhanced in most of the implemented DA conditions. A superior performance improvement for the MI EEG classifiers by the proposed cGAN was confirmed in the experiment compared to other DA methods such as RS, NA, and cDCGAN. This shows that EEG DA effectively improves the MI EEG classifiers, and the proposed EEG DA method is promising for performance improvement of the EEG classifiers in the MI-based BCI domain.

The proposed EEG DA can improve the MI EEG classifiers by supplementing insufficient EEG data amount with synthetic data, which is new and diverse but following the characteristics of the EEG training data. In addition, it can be utilized for simulations of EEG experiments which are highly expensive and time-consuming in the real world.

CHAPTER 2

Effects of Batch Size and Its Strategy on Generalizability of Convolutional Neural Networks on Motor Imagery BCI

2.1. Introduction

Brain-computer interface (BCI) is a promising technology that enables the communication between a user and an external device like a computer by decoding neural patterns from the brain signals. These are typically recorded with a noninvasive electroencephalogram (EEG) and translating them into commands for some device (Graimann et al., 2010; Zhang et al., 2012; Zhang et al., 2018). For BCI development, motor imagery (MI) is one promising approach, which induces sensorimotor rhythm (SMR) by imagining the motion of a body part without any physical movement (Pfurtscheller & Lopes da Silva, 1999; Pfurtscheller et al., 2006). This BCI paradigm has been widely utilized in a variety of applications (e.g., motor rehabilitation (Cho et al., 2018a), prosthesis control (Cho et al., 2018b), and wheelchair control (Xiong et al., 2019)).

There are two approaches to decoding the MI EEG signals: conventional machine learning (ML) and deep learning (DL) approach. In the ML approach, it includes the following stages to build EEG decoders (Lawhern et al., 2018): (1) EEG signal acquisition, (2) signal processing (e.g., band-pass filter and artifact removal), (3) feature representation learning, (4) classifier learning, and (5) feedback. Studies have shown the promising classification performances based on the previous processes (Blankertz et al., 2008; Zhang et al., 2018), but it requires specialized domain knowledge for MI EEG signals (e.g., common spatial pattern (CSP; Blankertz et al., 2008) or its variants (Ang et al., 2012; Suk & Lee, 2013)). On the other hand, DL-based EEG classifiers (Schirrmeister et al., 2017; Lawhern et al., 2018; Ingolfsson et al., 2020; Martinez et al., 2020; Ko et al., 2021) have been attractive due to their ability to reduce the data preprocessing and remove

the procedure of hand-crafted MI EEG feature extraction. These DL-based EEG classifiers are composed of end-to-end pipelines, which automates feature extraction/selection and classification by optimizing them jointly. Among the various DL-based EEG classification models, convolutional neural networks (CNNs; Schirrmeister et al., 2017; Lawhern et al., 2018; Amin et al., 2019; Zhang et al., 2019; Dai et al., 2020; Ingolfsson et al., 2020; Jia et al., 2021; Ko et al., 2021) have received the most attention due to their remarkable performance and ability to extract diverse EEG feature information (e.g., temporal, spectral, and spatial features) using their unique network architectures.

Many hyperparameters in the CNNs need to be adjusted to train them well since they are closely associated with the classification performance regarding the MI EEG signals. Among them, the batch size is one of the primary hyperparameters that need to be tuned (Ioffe & Szegedy, 2015) and directly involves the models' performance (Keskar et al., 2016; Devarakonda et al., 2017; Kandel & Castelli, 2020). The batch size represents the number of EEG samples used in every epoch or iteration for the training and can be either fixed or adaptive when training the CNN. In the fixed batch size (FB), the network can take a long time to achieve training convergence and not reach better performance if we set it too high (Kandel & Castelli, 2020). Whereas if it is too low, it is difficult to achieve acceptable performance since the network fluctuates back and forth (Kandel & Castelli, 2020). The adaptive batch size (AB) has been devised to overcome the limitations of the FB. In the AB, the batch size is adaptively set based on specific criteria (e.g., heuristics (Devarakonda et al., 2017) and inner product/orthogonality tests (Bollapragada et al., 2018)). The central intuition of the AB is to explore an ample search space by using a small batch size (SB) at the beginning of training and then achieve the training convergence with a large batch

size (LB) in the second half. It can allow us to achieve fast convergence and acceptable model performance relatively compared to the FB.

However, although the batch size and its strategy can significantly affect the generalization performance of the CNNs for decoding MI EEG signals, the previous studies (Schirrmeister et al., 2017; Lawhern et al., 2018; Amin et al., 2019; Zhang et al., 2019; Dai et al., 2020; Jia et al., 2021) have overlooked the importance of them in validating their models' effectiveness by not specifying the batch size settings. Also, most studies regarding the effects of batch size on classifier performance (Bengio, 2012; Keskar et al., 2016; Radiuk, 2018; Kandel & Castelli, 2020) have focused on image datasets, including a large number of training samples as compared to EEG datasets. In addition, there is still a lack of study on the effects of the batch size strategy on the generalization performance for the CNN-based MI EEG classifiers.

In this study, we investigate the effects of batch size and its strategy on the generalization performance of the CNNs for the MI BCI. Multi-scale fixed batch sizes and four different batch size strategies are used to assess the impacts. Four different CNN architectures for the MI BCI are used with the batch size settings to assess the impacts. Also, two public BCI competition datasets are adopted for the performance comparison of the CNNs. This experimental study emphasizes the importance of the batch size and strategy to validate the effectiveness of the developed CNNs. Our study addresses the following research questions (RQs):

RQ-1. What is the best batch size for the CNNs to get high generalization performance?

RQ-2. What is the most effective batch size strategy for achieving high generalization performance of the CNNs?

RQ-3. What is the most effective CNN model for decoding MI EEG signals?

The main contributions of this study are as follows:

- We investigate the effects of batch size and its strategy on the generalization performance of CNNs, especially for MI BCI. To the best of our knowledge, this work is the first experimental study on the impacts in the BCI domain, which emphasizes the importance of the batch size and strategy when validating the effectiveness of newly developed CNN models for classifying MI EEG signals.
- To our best knowledge, this study is the first to utilize small datasets for exploring the effects of the batch size and the strategy on the performance of a DL model. The previous related studies have adopted large datasets (e.g., image and speech signals) compared to the EEG datasets. Our study provides empirical results and findings for the effects of both the batch size and strategy on the generalization performance of the CNN models in small datasets, which will help confirm if the previous findings are consistent.
- We provide practical guidelines to select the batch size and strategy for the CNNs in achieving high classification performance for MI BCI generally. Also, we recommend a CNN model for MI BCI. The guidelines and the recommended CNN model will help the BCI practitioners select the batch size strategy and CNN classifier in their experiments and applications.

2.2. Background

2.2.1. Mini-batch Gradient Descent

Training deep neural networks is a non-convex optimization problem. It can be represented mathematically as follows:

$$\min_{x \in \mathbb{R}^d} f(\theta) \triangleq \frac{1}{N} \sum_{i=1}^{N} f_i(\theta), \tag{2.1}$$

where f_i is a loss function of the neural networks for data point $i \in \{1,2, \dots, N\}$ with d dimension and θ is the weight vector updated in the optimization process. To find optimal weight parameters that minimize the loss function, which is also called training of the neural networks, stochastic gradient descent (SGD; Bottou, 1998; Sutskever et al., 2013) and its variants (e.g., Adam (Kingma & Ba, 2014)) are used in general. These methods have the following updating rules of the gradient descent form iteratively minimizing the loss function f:

$$\theta_{k+1} = \theta_k - \alpha_k \left(\frac{1}{|S_k|} \sum_{i \in S_k} \nabla f_i(\theta_k) \right), \tag{2.2}$$

where $S_k \subset \{1,2, \dots, M\}$ is the batch sampled from the training dataset and α_k is the step size at iteration k in the training loop. When the sampled batch S_k is M and 1, it is called batch gradient descent and SGD, respectively. When the sampled batch size S_k is greater than 1 and less than M, it is called mini-batch gradient descent. However, mini-batch gradient descent is usually referred to as SGD since it is a special case of SGD. The effectiveness of mini-batch gradient descent has been validated practically (Graves et al., 2013; Mnih et al., 2013; Simonyan & Zisserman, 2014) and theoretically (Ge et al., 2015; Hardt et al., 2016; Lee et al., 2016; Bottou et al., 2018) in DL society.

2.2.2. Effects of Batch Size and Its Strategy on Generalization Performance of Deep Learning

Batch size is one of the primary hyperparameters that need to be adjusted for training DL models. The DL models can become more robust and avoid bad local minima using a well-tuned batch size. Many researchers have studied the effects of batch size and its strategy on DL performance. In the FB strategy, using LB to train DL models leads to an accurate estimate of the gradient, high computation cost per training iteration, and high availability of parallelism (Masters & Luschi, 2018). In contrast, SB leads to a noisy estimate of the gradient, low computation cost per training iteration, and low availability of parallelism (Masters & Luschi, 2018). Also, if we train the DL models with SB, it tends to converge to a flat minimizer which varies slightly in a relatively large neighborhood of the minimizer (Keskar et al., 2016). On the other hand, when we train them with LB, it tends to converge to a sharp minimizer which varies sharply (Keskar et al., 2016). According to Keskar et al. (2016), the flat minimizers tend to generalize better than the sharp minima since it is less sensitive to the loss function changes. In addition, many theoretical (Ge et al., 2015; Hardt et al., 2016; Keskar et al., 2016; Lee et al., 2016; Bottou et al., 2018) and empirical studies (Graves et al., 2013; Mnih et al., 2013; Simonyan & Zisserman, 2014; Masters & Luschi, 2018; Kandel & Castelli, 2020) have been shown that SB was more effective to train the DL models to get higher generalization performance than the LB. The AB strategy initially uses SB and then gradually increases the size. Doing so has a similar effect on the learning rate decay and reduces the number of parameters to be turned during the training phase (Smith et al., 2017). Also, empirical and theoretical studies have shown that it is effective to better the DL performance as compared to the FB (Byrd et al., 2012; Friedlander & Schmidt, 2012; Balles et al., 2017; De et al., 2017; Devarakonda et al., 2017; Smith et al., 2017; Bottou et al., 2018). There are mainly two approaches for the AB to train the DL models, i.e., (1) heuristic-based adaptive batch

size (HAB; Devarakonda et al., 2017) (2) algorithm-based adaptive batch size (AAB; Bollapragada et al., 2018). HAB can be easily applied without significant changes in the existing training algorithms by gradually increasing the batch size at every iteration or epoch. However, it does not use any information generated from training (e.g., gradient direction) which can lead to a bad local minimum. On the other hand, AAB can be globally convergent on nonconvex function by using inner product test for guaranteeing descent direction of gradient and orthogonality test that can avoid wrong gradient direction to find optimum (Bollapragada et al., 2018). Both inner product (2.3) and orthogonality (2.4) tests are mathematically defined, respectively, as follows:

$$\frac{\sum_{i\in S_k}\left(\nabla f_i(\theta_k)^T\nabla f_{S_k}(\theta_k)-\left\|\nabla f_{S_k}(\theta_k)\right\|^2\right)^2}{|S_k|\cdot(|S_k|-1)}\leq \eta^2\left\|\nabla f_{S_k}(\theta_k)\right\|^4, \tag{2.3}$$

$$\frac{\sum_{i\in S_k}\left\|\nabla f_i(\theta_k)-\frac{\nabla f_i(\theta_k)^T\nabla f_{S_k}(\theta_k)}{\left\|\nabla f_{S_k}(\theta_k)\right\|^2}\nabla f_{S_k}(\theta_k)\right\|}{|S_k|\cdot(|S_k|-1)}\leq \nu^2\left\|\nabla f_{S_k}(\theta_k)\right\|^2, \tag{2.4}$$

where the set $S_k \subset \{1,2,\dots\}$ indexes certain data point i and $\nabla f_{S_k}(\theta_k)$ indicates the gradients of the parameters given the set. η and ν represent certain positive constants.

2.2.3. Convolutional Neural Networks for MI BCI

Several CNN classifiers have been developed for MI BCI. They use minimally preprocessed EEG signals as input and then extract/select EEG features, including temporal, spatial, and spectral information based on unique layers (e.g., convolution and pooling). Then, the EEG features are classified into specific cognitive states with fully connected layers and softmax function. These processes are automated in the CNNs and optimized jointly to minimize their loss function in training EEG samples.

For instance, Schirrmeister et al. (2017) developed ShallowConvNet and DeepConvNet, which were first noticed for their reliable performance compared to the conventional EEG classifiers. Using conventional convolution and pooling operations, they extracted temporal-spatial features by making several temporal and spatial filtering layers. However, their models had many weight parameters that needed to be trained, causing an overfitting problem due to high model complexity. To address this issue, Lawhern et al. (2018) introduced EEGNet, which uses depthwise and separable convolution layers with more compact CNN architecture than ShallowConvNet and DeepConvNet. EEGNet also attempted to extract temporal-spatial features and showed its reliable classification performance in various EEG paradigms (e.g., MI and error-related potential) compared to the conventional EEG classifiers, ShallowConvNet, and DeepConvNet.

Zhang et al. (2019) suggested a Multi-Branch 3D CNN to classify MI EEG signals. It utilized a 3-D tensor to represent EEG input signals that the first two dimensions represent spatial information, and the last dimension represents temporal information. This 3-D input tensor used three branches to extract various spatial-temporal EEG features and classified the EEG signals by merging the features from the branches.

Amin et al. (2019) developed a Multi-layer CNN (MCNN), which utilized model fusion from the individual CNN models, including convolution, pooling, and fully connected layers. It utilized four individual CNN models with varied layer depths to extract spatial-temporal EEG features. The last layer concatenated all EEG features from the CNN models for the classification task.

Dai et al. (2020) proposed a Hybrid-Scale CNN (HS-CNN) for classifying MI EEG signals. Their model utilized three different branches to extract spectral information from input EEG

signals. Each branch had the same network structure, including two convolutional layers for temporal and spatial features and pooling and flattening layers for dimension reduction. All extracted features from each branch were concatenated with a fully connected layer, and then the merged features were connected with another fully connected layer for the classification task.

Ingolfsson et al. (2020) introduced an accurate Temporal Convolutional Network called EEG-TCNet for MI BCIs. They applied causal convolutions, dilated convolutions, and residual blocks with temporal, depthwise, and separable convolutions, which are similar to the operations of EEGNet. Given the layers for the EEG feature extractions, the fully-connected layer with softmax function was adopted in the last layer for the decoding task. Their proposed model required few trainable parameters to reduce memory footprint and computational complexity.

Jia et al. (2021) proposed a Multi-branch Multi-scale Convolutional Neural Network (MMCNN) to consider subject and time differences which could cause the best classification models. Their model was composed of an inception block to address different convolution kernel sizes, a residual block to avoid network degradation, and a squeeze and excitation block to adaptively capture attentive features.

Ko et al. (2021) developed a Multi-Scale Neural Network (MSNN) for EEG classification tasks for diverse paradigms. The main difference from the previous studies for CNN-based EEG classifiers was that their CNN architecture could extract temporal-spectral-spatial information only using a single network. This allows us to have a more compact CNN network to extract important EEG features than the others requiring several branches for spectral information.

These CNN classifiers argue that their proposed models effectively decode MI EEG signals. However, most studies have overlooked the importance of batch size and its strategy. They did not specify their settings given the optimizer and learning rate, as shown in Table 2.1. As we

mentioned before, the batch size and its strategy are crucial hyperparameters that directly affect the generalization performance of the CNN.

Table 2.1. Batch size and its strategy of CNN studies for MI BCI.

Author (Year)	Model Name	Optimizer	Learning Rate	Batch Size Strategy	Batch Size
Schirmeister et al. (2017)	ShallowConvNet, DeepConvNet	Adam	-	-	-
Lawhern et al. (2018)	EEGNet	Adam	0.001	-	-
Zhao et al. (2019)	Multi-Branch 3D CNN	Adam	0.001	-	-
Amin et al. (2019)	MCNN	Adam	-	-	-
Dai et al. (2020)	HS-CNN	SGD	Decay	-	-
Ingolfsson et al. (2020)	EEG-TCNet	Adam	0.001	FB	64
Jia et al. (2021)	MMCNN	Adam	0.0001	-	-
Ko et al. (2021)	MSNN	Adam	Decay	FB	16

Therefore, the effectiveness of their proposed CNNs cannot be validated objectively without specific information about their settings. Also, many studies have investigated the effects of the batch size and the strategy on the DL performance based on a large number of datasets like images (Bengio, 2012; Keskar et al., 2016; Devarakonda et al., 2017; Masters & Luschi, 2018; Radiuk, 2018). For a consistent conclusion regarding their findings, the small number of datasets also needs to be considered. To address them, we explore how the generalization performance of the CNNs for the EEG datasets can be differed depending on the batch size and the strategy settings.

2.3. Methods

2.3.1. Datasets and Preprocessing

In this study, we used two different publicly available BCI competition datasets to investigate the effects of the batch size and its strategy on the generalization performance of CNNs for MI BCI.

Dataset 1 (BCI Competition III-IVa; Dornhege et al., 2004***):*** This dataset contained two balanced classes of MI (right hand and foot) for five subjects. There was a total of 280 trials for each subject. 118 EEG channels were used at positions of the international 10-20 system. The EEG signals were sampled at 100 Hz and bandpass-filtered between 0.05 Hz and 200 Hz (for details, see http://www.bbci.de/competition/iii/).

Dataset 2 (BCI Competition IV-IIa; Tangermann et al., 2012***):*** In this dataset, four balanced classes of imagined movements (left hand, right hand, feet, and tongue) were included for nine subjects. There was a total of 576 trials for each subject. 22 Ag/AgCl electrodes were employed to measure the EEG signals at the positions of the international 10-20 system. The signals were sampled at 250 Hz and bandpass-filtered between 0.5 Hz and 100 Hz (for details, see http://www.bbci.de/competition/iv/).

Preprocessing: We minimally preprocessed both datasets for the classification tasks by the CNNs. For only dataset 1, the EEG signals were resampled to 128 Hz. Also, we selected 21 channels (FC5, FC3, FC1, FCz, FC2, FC4, FC6, C5, C3, C1, Cz, C2, C4, C6, CP5, CP3, CP1, CPz, CP2, CP4, and CP6) for dataset 1 known to be effective for MI tasks (Al-Saegh et al., 2021b). Each EEG trial was epoched between 0.5 and 2.5 seconds after post-cue onset for both dataset 1 and 2. A third-order Butterworth filter in the 4-40 Hz was applied to the trials. We applied channel-wise Gaussian normalization.

2.3.2. Experimental Settings

2.3.2.1. Batch Size and Its Strategy

In our work, we adopted two batch size strategies, i.e., FB and AB, to evaluate the generalization performance of CNNs to classify MI EEG signals. FB uses a constant value for the batch size during whole model training, whereas AB starts from an initial batch size and increases the batch size gradually based on a specific criterion. As shown in Figure 2.1, the criteria can be based on heuristics (for HAB) and inner product/orthogonality tests (for AAB) which increase the batch size whenever the tests are not satisfied.

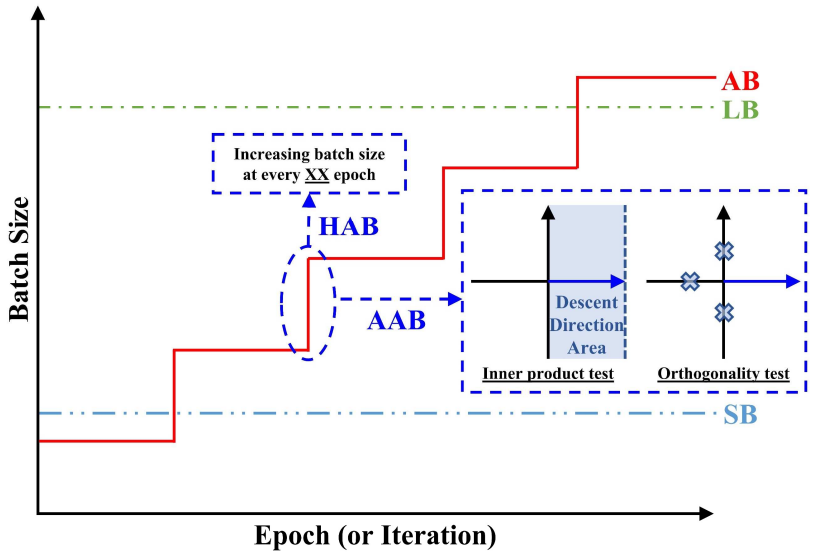


Figure 2.1. Batch size strategy.

FB Strategy: In the FB, we used six different batch sizes for both datasets (dataset 1: 2, 6, 10, 14, 18, 22; dataset 2: 4, 12, 20, 28, 36, 44). For dataset 1, batch sizes of 6 (about 3% of training samples) and 22 (about 10%) were selected as SB and LB, respectively. On the other hand, batch

sizes of 20 (about 5% of training samples) and 44 (about 10%) were chosen for SB and LB, respectively, in dataset 2.

Algorithm 2.1. HAB-based Mini-Batch Gradient Descent

Inputs: Learning Rate α, Batch Size B, Size of Training Samples M, Batch Size Increase Criteria R, Stochastic Objective Loss Function with Parameters θ $f(\theta)$
Output: Trained CNN Classifier C^*

1: $k \leftarrow 0$ (Initialize gradient update step k)
2: **Initialize** weight parameter θ_0
3: **while** θ_k not converged do
4: **for** t = 1, 2, 3, … do (*epoch)
5: **if** R mod t == 0
6: $B \leftarrow B \times 2$
7: **end if**
8: **for** i = 1, 2, 3, …, floor(M/B)
9: $s = i \times B$; $q = s + B$
10: $\theta_{k+1} = \theta_k - \alpha_k \left(\frac{1}{B}\sum_{j=s}^{q} \nabla f_j(\theta_k)\right)$
11: $k \leftarrow k + 1$
12: **end for**
13: **end for**
14: **end while**
15: **return** C^*

AB Strategy: In the AB, we adopted two ABs, i.e., HAB called AdaBatch (Devarakonda et al., 2018) and AAB (Bollapragada et al., 2018). For the HAB, initial batch sizes started from the number of classes for each dataset. The batch sizes were doubled at every 20 epoch, which was the same strategy with Devarakonda et al. (2018) until the training processes were converged. We also provide pseudocode for HAB, which was not given in Devarakonda et al. (2018), to help practitioners implement it (see Algorithm 2.1). In Algorithm 2.1, it starts with initializing update step k for the gradient and weight parameter θ for the model (lines 1-2). (lines 3-14) In every epoch, the parameter θ_k is updated using mini-batch gradient descent with the given batch size B

until the whole training samples are used. But the batch size B is doubled at every R epoch. The updating process is continued until θ_k converges.

Algorithm 2.2. AAB-based Mini-Batch Gradient Descent

Inputs: Learning Rate α, Batch Size B, Size of Training Samples M, Stochastic Objective Loss Function with Parameters θ $f(\theta)$
Outputs: Trained CNN Classifier C^*

$k \leftarrow 0$ (Initialize gradient update step k)
Initialize weight parameter θ_0
while θ_k not converged do
    for t = 1, 2, 3, … do (*epoch)
        Initialize training samples $S \subseteq \{1,2,\dots,M\}$
        Sample $S_B \subseteq S$ with sample size B
        while S is not empty or $|S| < B$ **do**
            Compute $\nabla f_{S_B}(\theta_B) = \frac{1}{|S_B|}\sum_{i \in S_B} \nabla f_i(\theta_B)$
            while conditions (2.3) or (2.4) are not satisfied **do**
                $B \leftarrow B \times 2$
                Sample $S_B \subseteq S$ with updated B
                Compute $\nabla f_{S_B}(\theta_B)$
            end while
            $k \leftarrow k + 1$
            $\theta_{k+1} = \theta_k - \alpha_k \nabla f_{S_B}(\theta_B)$
            $S \leftarrow S \setminus S_B$
        end while
    end for
end while
return C^*

The initial batch size settings for the AAB were the same as for the HAB. We doubled the batch sizes whenever either the inner product or orthogonality tests were not satisfied. The inner product test is applied to guarantee the descent direction of the current gradient for the loss function. In contrast, the orthogonality test prevents going in the wrong direction, causing the model's performance degradation (Bollapragada et al., 2018). We utilized default hyperparameter settings for the HAB, which were recommended by Bollapragada et al. (2018). Also, we provide

a pseudo code to implement AAB as shown in Algorithm 2.2. In Algorithm 2.2, it also starts with the initialization of gradient update step k and weight parameter θ for a CNN classifier (lines 1-2). In every epoch, it initializes training samples S and samples training data with batch size B. (lines 3-20) Given the samples, the mini-batch gradients of θ are calculated. Based on the gradients, if the conditions (2.3) or (2.4) are not satisfied, the batch size B is doubled. Then, the training data is sampled again with the updated B. The batch size B is increased until the conditions are satisfied. The mini-batch gradient descent is performed with the selected batch size B. Then, the used batch samples are removed from S. This updating process is repeated until θ_k converges.

2.3.2.2. CNN Architectures

We adopted four different CNN-based EEG classifiers, i.e., EEG-TCNet (Ingolfsson et al., 2020), EEGNet (Lawhern et al., 2018), ShallowConvNet (Schirrmeister et al., 2017), and DeepConvNet (Schirrmeister et al., 2017), for the numerical experiment. This is because DeepConvNet (Schirrmeister et al., 2017), ShallowConvNet (Schirrmeister et al., 2017), and EEGNet (Lawhern et al., 2018) are the most popular models for MI BCI and have been used as the baselines to show the effectiveness of the newly developed CNNs. EEG-TCNet (Ingolfsson et al., 2020) has shown its effectiveness in classifying MI EEG signals compared to the ShllowConvNet and the EEGNet. Also, we selected these CNN models for the experiment since they are reproducible due to open sources to implement them correctly without any errors from scratch.

2.3.2.3. Optimizer and Learning Rate

We utilized the Adam optimizer (Kingma & Ba, 2015) at a learning rate of 0.001, which was the same setting as the adopted CNN models for training their models (Schirrmeister et al., 2017; Lawhern et al., 2018; Ingolfsson et al., 2020).

2.3.2.4. Performance Metric and Experimental Tools

For the performance metric of both datasets, accuracy was used because they were all balanced classes. We performed subject-dependent with 5×5-fold cross-validation to evaluate the classification accuracy. We also reported the mean classification accuracy by averaging all results across the subjects. To conduct the experiment, the following software and hardware were employed: Tensorflow and Keras, CPU (Intel i7-10870H), GPU (NVIDIA GeForce RTX 3080), and RAM (32GB).

2.4. Results

2.4.1. Effects of the FBs on the Model Performance

Tables 2.2 and 2.3 summarize the mean accuracy of CNN-based EEG classifiers by different fixed batch sizes on datasets 1 and 2, respectively. The tables include four CNN-based EEG classifiers (EEG-TCNet, EEGNet, ShallowConvNet, DeepConvNet) and six different fixed batch sizes (Table 2.2: 2, 6, 10, 14, 18, 22; Table 2.3: 4, 12, 20, 28, 36, 44). We also report the best batch size indicating the highest mean accuracy among the used batch sizes for each CNN classifier as well as the best CNN classifier for each batch size. In addition, we show the best classifier with the best batch size having the highest mean classification accuracy as compared to the other combinations. As shown in Table 2.2, the mean accuracy for each CNN classifier varied depending

on the batch size settings. In the view of the classifier, the best batch sizes for EEG-TCNet, EEGNet, ShallowConvNet, and DeepConvNet were 22 (*M* = 72.7%, *SD* = 11.9%), 6 (89.1%, 8.3%), 18 (87.6%, 8.6%), and 14 (77.6%, 17.8%), respectively. In the aspect of the batch sizes, EEGNet was the best classifier for the batch size 2 (85.3%, 9.9%), 6 (89.1%, 8.3%), 10 (88.7%, 9.3%), 18 (87.9%, 7.8%), and 22 (88.5%, 8.9%) whereas ShallowConvNet was the best on 14 (86.7%, 9.3%). Among all combinations, EEGNet (batch size = 6) had the highest mean classification accuracy (89.1%, 8.3%) than the others. Similarly, the mean accuracy for the CNN classifiers on dataset 2 varied depending on the batch sizes as shown in Table 2.3. The best batch sizes for EEG-TCNet, EEGNet, ShallowConvNet, and DeepConvNet were 20 (56.8%, 8.8%), 20 (77.2%, 10.5%), 20 (73.3%, 11.5%), and 12 (64.4%, 12.1%), respectively. In all batch sizes, EEGNet showed higher mean accuracy than the other CNN classifiers. EEGNet (batch size = 20) had the highest mean accuracy (77.2%, 10.5%) as compared to the other combinations.

Table 2.2. Mean classification accuracy (%) of CNN classifiers for FBs on the dataset 1.

(Note: the parentheses represent the standard deviations, but the last cell indicates the best combination of the classifier and the batch size having the highest mean classification accuracy.)

EEG Classifier	Batch Size						Best Batch Size
	2	6	10	14	18	22	
EEG-TCNet	52.9 (16.3)	60.7 (9.8)	65.9 (17.1)	52.6 (14.7)	70.3 (13.3)	72.7 (11.9)	22
EEGNet	85.3 (9.9)	89.1 (8.3)	88.7 (9.3)	86.0 (15.1)	87.9 (7.8)	88.5 (8.9)	6
ShallowConvNet	79.8 (10.6)	86.3 (9.0)	86.6 (9.5)	86.7 (9.3)	87.6 (8.6)	85.1 (12.5)	18
DeepConvNet	65.4 (14.2)	73.6 (18.4)	71.9 (18.5)	77.6 (17.8)	72.8 (18.6)	71.8 (18.0)	14
Best Classifier	EEG.	EEG.	Shall.	EEG.	EEG.	EEG.	EEG. (6)

Table 2.3. Table 3. Mean classification accuracy (%) of CNN classifiers for FBs on the dataset 2.

(Note: the parentheses represent the standard deviations, but the last cell indicates the best combination of the classifier and the batch size having the highest mean classification accuracy.)

EEG Classifier	Batch Size						Best Batch Size
	4	12	20	28	36	44	
EEG-TCNet	53.2 (9.1)	54.7 (11.3)	56.8 (8.8)	54.3 (10.0)	54.8 (9.0)	55.8 (9.8)	20
EEGNet	72.3 (10.6)	74.5 (11.5)	77.2 (10.5)	73.9 (12.2)	75.3 (10.6)	74.3 (11.5)	20
ShallowConvNet	70.7 (9.8)	71.1 (13.3)	73.3 (11.5)	68.8 (14.5)	73.3 (9.9)	72.9 (11.8)	20
DeepConvNet	62.1 (10.4)	64.4 (12.1)	59.8 (14.9)	50.9 (20.3)	48.7 (17.3)	42.9 (13.2)	12
Best Classifier	EEG.	EEG.	EEG.	EEG.	EEG.	EEG.	EEG. (20)

Table 2.4. Mean classification accuracy (%) of CNN classifiers depending on the batch size strategy.

The parentheses represent the standard deviations, but the last cell in each dataset column indicates the best combination of the classifier and the batch size strategy having the highest mean classification accuracy.

EEG Classifier	Dataset 1 (2-class)					Dataset 2 (4-class)				
	Batch Size Strategy				Best Strategy	Batch Size Strategy				Best Strategy
	SB	LB	HAB	AAB		SB	LB	HAB	AAB	
EEG-TCNet	72.7 (9.8)	60.7 (11.9)	63.4 (11.9)	69.2 (6.4)	SB	56.8 (8.8)	55.8 (9.8)	57.4 (10.4)	56.1 (9.6)	HAB
EEGNet	89.1 (8.3)	88.5 (8.9)	88.7 (9.6)	89.6 (9.5)	AAB	77.2 (10.5)	74.3 (11.5)	73.2 (10.7)	72.1 (11.1)	SB
ShallowConvNet	86.3 (9.0)	85.1 (12.5)	86.5 (9.1)	86.1 (9.2)	HAB	73.7 (11.5)	72.9 (11.8)	73.3 (11.5)	70.8 (13.0)	SB
DeepConvNet	73.6 (18.4)	71.8 (18.0)	68.1 (16.7)	70.4 (18.5)	SB	59.8 (14.9)	42.9 (13.2)	55.2 (10.7)	60.1 (8.2)	AAB
Best Classifier	EEG.	EEG.	EEG.	EEG.	EEG. (AAB)	EEG.	EEG.	Shall.	EEG.	EEG. (SB)

2.4.2. Effects of the Batch Size Strategy on the Model Performance

Table 2.4 summarizes the mean accuracy of CNN-based EEG classifiers depending on the batch size strategy for both datasets 1 and 2. The table covers the mean accuracy of the CNN classifiers according to four different batch size strategies (SB, LB, HAB, and AAB). Also, we show the best batch size strategies, the best CNN classifiers, and the best combinations having the highest mean performance in both datasets. For dataset 1, the best batch size strategies for EEG-

TCNet, EEGNet, ShallowConvNet, and DeepConvNet were SB (*M* = 72.7%, *SD* = 9.8%), AAB (89.6%, 9.5%), HAB (86.5%, 9.1%), and SB (73.6%, 18.4%), respectively. The best CNN classifier for SB (89.1%, 8.3%), LB (88.5%, 8.9%), HAB (88.7%, 9.6%), and AAB (89.6%, 9.5%) was EEGNet. Among all combinations, EEGNet using AAB showed the highest mean accuracy (89.6%, 9.5%). In dataset 2, HAB and AAB were the best batch sizes for EEG-TCNet (57.4%, 10.4%) and DeepConvNet (60.1%, 8.2%), respectively, whereas SB was the best for EEGNet (77.2%, 10.5%) and ShallowConvNet (73.7%, 11.5%). EEGNet was the best CNN classifier for three batch size strategies, i.e., SB (77.2%, 10.5%), LB (74.3%, 11.5%), and AAB (72.1%, 11.1%). ShallowConvNet was the best for HAB (73.3%, 11.5%). The best combination showing the highest mean accuracy was EEGNet using SB (77.2%, 10.5%).

2.5. Discussion

2.5.1. Importance of Batch Size Setting

As we confirmed in the previous result, the best batch size and strategy for each CNN classifier were different. Also, the best classifier could differ depending on which batch size or strategy was used to train the model. Therefore, we need to recognize the importance of the batch size setting and mention specifically how to set the batch size and strategy for the training to validate the effectiveness and superiority of proposed CNNs compared to the other models objectively. Otherwise, it would be possible to derive incorrect results from performance comparisons. For instance, Figure 2.2 shows the minimum and maximum classification accuracy of the CNNs based on investigated batch sizes and strategies. In dataset 1, the maximum accuracy of ShallowConvNet (*M* = 87.6%, *SD* = 8.6%, *Batch Size* = 18) showed the higher performance than the minimum accuracy of EEGNet (*M* = 85.3%, *SD* = 9.9%, *Batch Size* = 2). However, the

maximum accuracy of EEGNet had a higher performance (*M* = 89.6%, *SD* = 9.5%, *Batch Size Strategy* = AAB) than the one of ShallowConvNet. In other words, the result of the model validation might vary depending on what batch size settings are used to train the model. So, the previous studies (Schirrmeister et al., 2017; Lawhern et al., 2018; Amin et al., 2019; Zhang et al., 2019; Dai et al., 2020; Jia et al., 2021) that did not indicate information about the batch size setting are required to mention this in detail for their models' evaluation and comparison. In addition, we can significantly improve the classification performance of the CNNs by selecting the proper batch size settings compared to the worst cases, as shown in Figure 2.2.

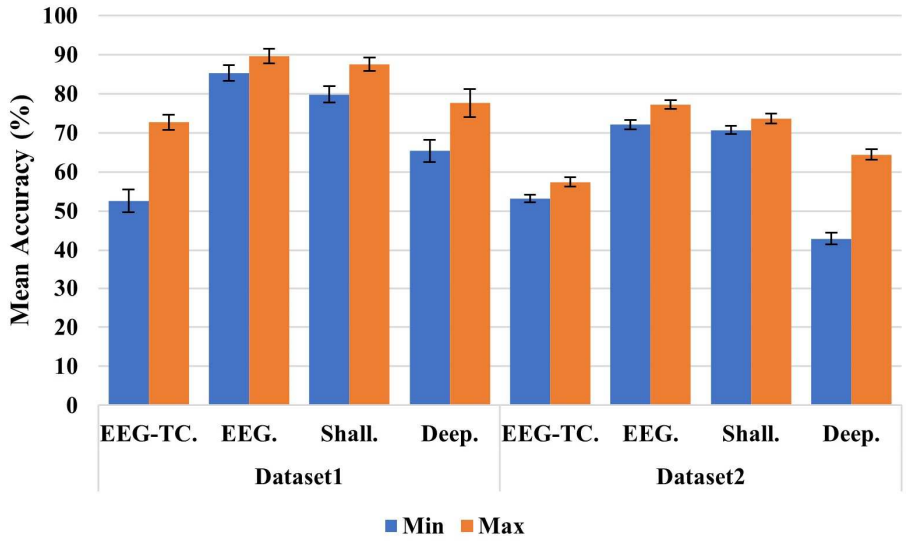
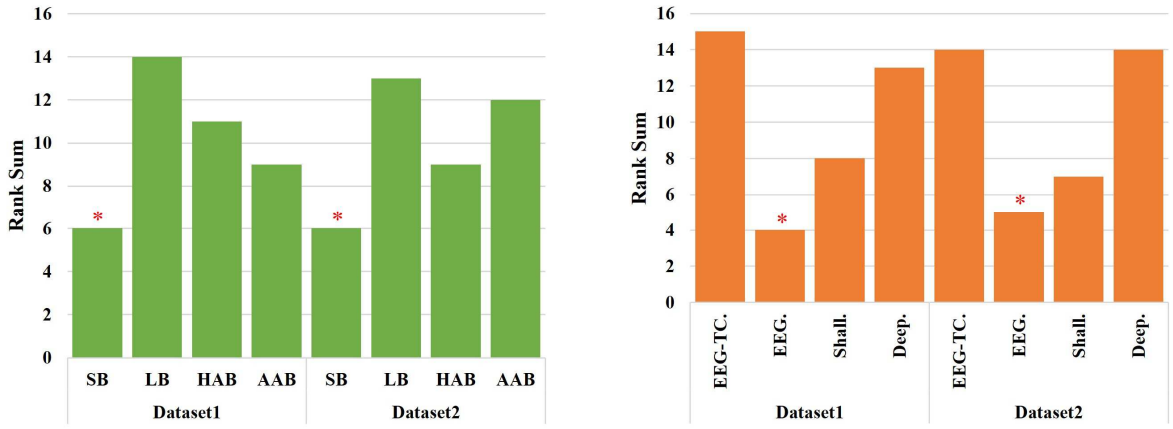


Figure 2.2. Mean classification accuracy and standard error of CNN classifiers for the datasets in terms of the minimum and the maximum.



(a) Rank-sum for batch size strategy **(b) Rank-sum for CNNs**

Figure 2.3. Results of rank-sum for batch size strategy and CNNs.

2.5.2. Best Batch Size Strategy and Best CNN Classifier

In section 2.4.2, we reported the best batch size strategies for each CNN classifier and the best CNNs for each strategy in both datasets. We performed rank-sum analysis to comprehensively derive the best batch size strategy and the best CNN classifier in the datasets. In each dimension (CNN classifier and batch size strategy), the ranks were calculated and summed for both datasets. Intuitively, the lower rank-sum represents, the better one. As shown in Figure 2.3 (a), SB was a better batch size strategy than the others for both datasets. This represents that SB can achieve better test accuracy than using other comparison strategies for the model training in general. SB tends to converge to a flat minimizer that is not sensitive to the function change from the training to the test (Keskar et al., 2016). Therefore, SB is an excellent strategy to achieve the high generalization performance of the classifiers.

In contrast, LB tends to converge to a sharp minimizer insensitive to the function change, which is not good for the generalization performance (Keskar et al., 2016). Also, even though HAB and AAB utilize SB at the beginning of the training, they increase the batch size gradually

after that. They might become a state of LB without sufficient exploration for a solution. They might get a sharp minimizer for the training loss function. With this in mind, we recommend using SB to train the CNN-based EEG classifiers.

In the classifier view, EEGNet was better than the other classifiers for both datasets (see Figure 2.3 (b)). Also, in section 2.4.2, we confirmed that EEGNet using AAB and EEGNet using SB were the best classifiers for the datasets. These results differed from the previous study's results (Ingolfsson et al., 2020). Their proposed model EEG-TCNet can get higher classification performance for decoding MI EEG signals than ShallowConvNet and EEGNet. However, EEG-TCNet could not achieve higher performance than the others in our experimental settings. Given our analysis, EEGNet is recommended to decode MI EEG signals for the practitioners. Also, we emphasize that researchers need to carefully consider diverse batch size strategies to train their proposed models and show their effectiveness objectively.

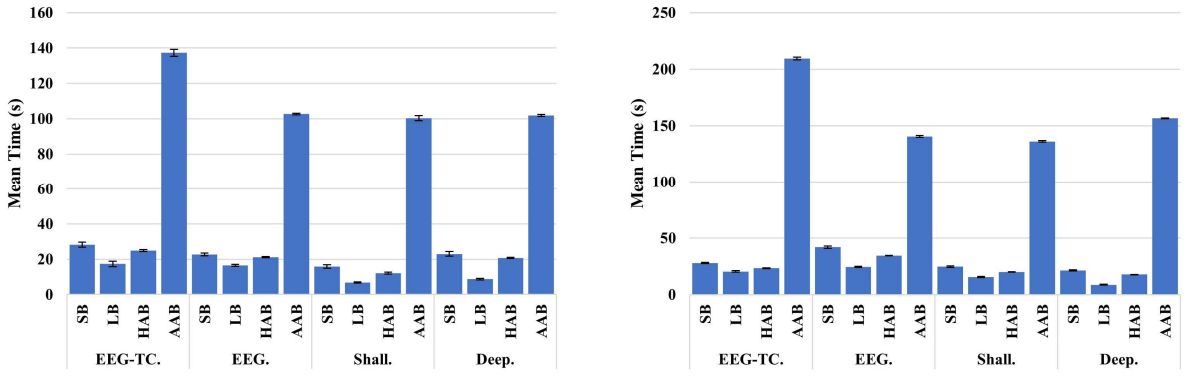


Figure 2.4. Comparison of computational time cost by different batch size strategies for the CNNs.

2.5.3. Computational Cost for Batch Size Strategies

To assess the computational efficiency of the batch size strategies, we further implemented an experimental comparison of computational time to train the CNN classifiers with them for both datasets (see Figure 2.4). All the batch size strategies with the CNN classifiers were implemented given the datasets with Python on the same computer settings (CPU, GPU, and RAM specs), and the computational time was averaged across subjects. In general, LB showed the least computational cost because it can estimate the gradient for the training accurately, which can require small numbers of iterations for the training convergence (Masters & Luschi, 2018). In contrast, SB showed a higher computational cost than LB since it requires significant iterations due to the noisy estimate of the gradient (Masters & Luschi, 2018). The computational cost for HAB was less than SB and greater than LB. This is because it increases the batch size for every specific epoch, although it starts from SB. The strategy requiring the highest computational cost was AAB since it should calculate all gradient information additionally to decide whether to increase the batch size or not in the current iteration. As shown in Table 2.4 (Dataset 1 – EEGNet using AAB), it could achieve higher model performance than using the others occasionally. However, the computation is excessively required compared to other strategies. Thus, we recommend using SB for the training since it has the minimum rank-sum and reasonable computational cost.

2.5.4. Reason for Higher Generalization Performance

We further investigated the relationship between the generalization performance of the classifiers and the sharpness of the minimizers for them. Sharpness can be used to explain why a model's generalization performance can be better than the other model. For instance, Keskar et al.

(2016) revealed that SB is a better option to train the DL model than LB since SB can have a lower sharpness value of a minimizer from the training than LB. Mathematically, the sharpness measure (Keskar et al., 2016) of training loss function g at data x can be defined as follows:

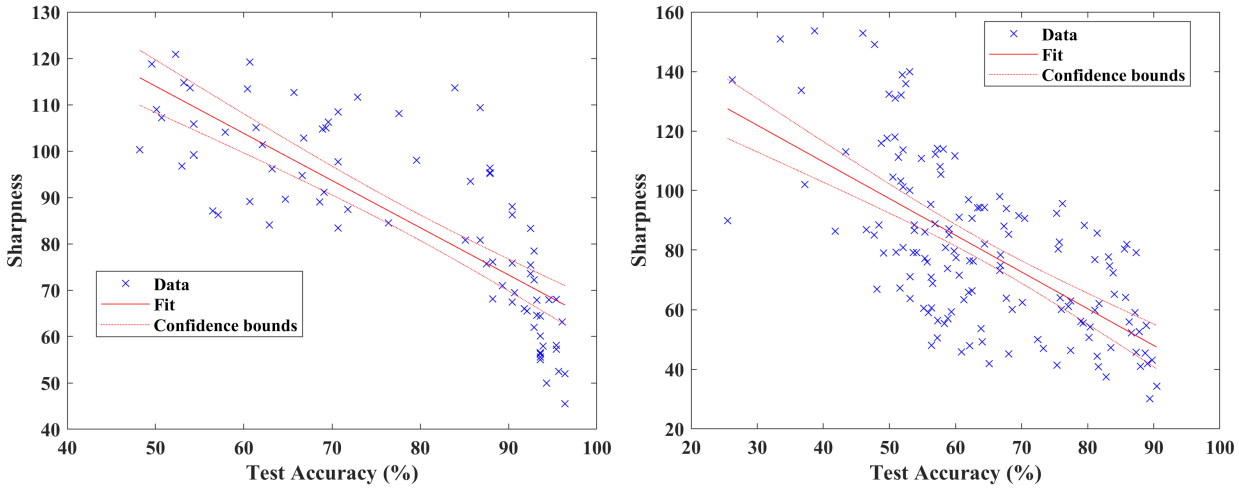
$$\phi_{x,g}(\epsilon, A) = \frac{\left(\max_{y \in \mathcal{C}_\epsilon} g(x + Ay)\right) - g(x)}{1 + g(x)} \times 100, \tag{2.5}$$

where $A \in \mathbb{R}^{n \times p}$ indicate a trick matrix to solve the problem whose columns are generated randomly, $\mathcal{C}_\epsilon$ denotes a box around the optimal solution for g and define as follows:

$$\begin{aligned} \mathcal{C}_\epsilon = \{ z \in \mathbb{R}^p : -\epsilon(|(A^+ x)_i| + 1) \leq z_i \leq \epsilon(|(A^+ x)_i| + 1), \\ \forall i \in \{1,2,..,p\} \}, \end{aligned} \tag{2.6}$$

where ϵ represents a control value for the box size and A^+ indicates the pseudo-inverse of A (see Keskar et al. (2016) for more details). We used L-BFGS-B (Byrd et al., 1995) with 15 iterations and 10^{-4} for ϵ to solve this problem for all experiments.

We performed Pearson's correlation analysis in terms of the generalization performance and the sharpness. Given the datasets, we performed the analysis by extracting all sharpness values for the classifiers based on the batch size strategies and then averaging the values across the subjects. In dataset 1, the generalization performance and the sharpness values were found to be moderately negatively correlated, $r(78) = .631$, $p < .001$. In dataset 2, they were moderately negatively correlated, $r(144) = .416$, $p < .001$ (see Figure 2.5 for the results of the correlation analysis). Therefore, we can say that a model or a batch size strategy is likely better than the others is because it can find a flatter minimizer.



(a) Dataset 1 **(b) Dataset 2**

Figure 2.5. Results of correlation analysis between the sharpness and the generalization performance of the CNNs for datasets.

2.5.5. Limitations and Extension

Our current experimental study only tested the performance of four CNN classifiers (EEG-TCNet, EEGNet, ShallowConvNet, DeepConvNet) using raw EEG signals for MI-based BCI. Thus, our findings (e.g., SB is a better batch size strategy than the others to train the CNNs for raw EEG-based MI BCI.) would not be generalized because we might discover different results from the other CNNs. We will explore more diverse CNN models in our next studies to generalize the findings.

Our experimental study adopted two public EEG datasets for MI-based BCI. The results of our experiments lacked statistical evidence due to the small number of subjects provided in the datasets. Thus, many other MI datasets (e.g., Kaya et al. (2018) and Lee et al. (2019)), which have the larger number of subject data, could be used in the numerical experiments to get the statistical evidence. Furthermore, datasets for other BCI paradigms (e.g., event-related potential (Cao et al., 2019), steady-state visually evoked potential (Fernandez-Fraga et al., 2018)) might provide

consistent results as our findings like the effectiveness of SB and EEGNet. Similar to the previous limitation by limited CNN classifiers, investigating additional large MI datasets and BCI paradigms for the experiments would enhance the generalizability of our findings. Thus, investigating the performance of the CNN classifiers based on other large MI datasets and different BCI paradigms is worthy of a future study.

Our study focused on the effects of batch size and its strategy on the generalization performance of CNN-based EEG classifiers. However, many other hyperparameters (e.g., learning rate, types of activation function) could also affect the generalization performance. For instance, the learning rate is one of the essential hyperparameters for selecting step length for an optimizer (Goodfellow et al., 2016). If the learning rate is too large, it can lead to the model converging too quickly to a suboptimum. This effect is very similar to training the model with SB. On the other hand, it can cause the searching process to get stuck if the rate is too low. This is close to the effect of LB. Also, the decaying learning rate has a similar effect on the model training by the AB (Smith et al., 2017). Thus, it is worth further investigating how the other hyperparameters impact the generalization performance of the CNNs.

2.6. Conclusion

In this study, we have investigated the effects of batch size and its strategy on the generalization performance of raw EEG-based CNN classifiers for MI-based BCI. We implemented numerical experiments on two public datasets for MI-based BCI using various FBs, batch size strategies, and CNN classifiers for raw EEG signals. Our results and findings showed that (1) SB is encouraged for training the CNNs to decode MI EEG signals, (2) EEGNet is more effective CNN architectures for the classification of the signals than the other comparison

classifiers, (3) batch size settings should be explicitly stated for the objective model validation and comparison. Beyond that, our study also provides a practical guideline for BCI researchers and practitioners to select the settings of the batch size and the strategy for the CNNs.

CHAPTER 3

General Discussion and Conclusion

This dissertation proposed one approach of EEG DA for MI-based BCI to improve the generalization performance of classifiers and investigated the effects of batch size and its strategy on the generalization performance of CNNs for MI-based BCI.

In the first chapter, we developed an EEG DA framework using cGANs to reduce overfitting to the training datasets and enhance the generalization performance of classifiers for MI-based BCI. For the stable training of the cGANs, we adopted Wasserstein distance for loss function and GP. Also, we proposed new network architectures for the generator and the discriminator of the cGANs, which can reflect on any EEG data structures. The numerical experiments showed that the proposed DA method with the proper DA proportion could improve the generalization performance of the classifiers, including conventional MLs and CNNs, more than the existing EEG DA methods.

In the second chapter, we explored the impacts of batch size and its strategy on the generalizability of CNNs on MI-based BCI to emphasize their importance for model validation. We implemented an experimental study for the investigation in which we adopted one general group of batch sizes and four different batch size strategies to train the CNN classifiers for decoding MI EEG signals. The experimental results showed that the best batch size and strategy for the highest generalization performance differed depending on the CNN classifiers. Depending on the settings of the batch size and strategy, the validation results for the model comparisons could be changed.

Both studies in this dissertation are associated with improving the generalization performance for EEG classifiers, especially CNNs, on MI-based BCI. The studies contribute to

reducing adverse situations in real-world BCI applications by enhancing the performance of the MI-based BCIs given the improved classifiers. Also, our findings from the studies provide practical guidelines to address training data scarcity, set batch size for the training of the CNNs, and select a CNN model for BCI practitioners and researchers as follows:

- To address the data scarcity problem in the EEG dataset, any DA methods can be applied to enhance the generalization performance of EEG classifiers. For the conventional MLs, relatively small amounts of DA proportions are recommended, whereas large amounts of the proportions are suggested for the CNNs to update a lot of weight parameters.
- When it comes to training the CNNs, we recommend using the SB strategy. In general, it can be a better strategy to get a high generalization performance with reasonable computational costs. However, if high-performance computing is accessible, we also suggest applying the AAB strategy for the training because it can find a better solution and requires high computational costs.
- Among the investigated CNN models, we recommend adopting EEGNet to decode raw MI EEG signals. It can achieve a higher classification accuracy than other comparison CNNs in general.

REFERENCES

Agarwalla, S., & Sarma, K. K. (2016). Machine learning based sample extraction for automatic speech recognition using dialectal Assamese speech. *Neural Networks*, *78*, 97–111.

Al-Saegh, A., Dawwd, S. A., & Abdul-Jabbar, J. M. (2021a). CutCat : An augmentation method for EEG classification. *Neural Networks*, *141*, 433–443.

Al-Saegh, A., Dawwd, S. A., & Abdul-Jabbar, J. M. (2021b). Deep learning for motor imagery EEG-based classification: A review. *Biomedical Signal Processing and Control*, *63*, 102172.

Amin, S. U., Alsulaiman, M., Muhammad, G., Mekhtiche, M. A., & Hossain, M. S. (2019). Deep Learning for EEG motor imagery classification based on multi-layer CNNs feature fusion. *Future Generation Computer Systems*, *101*, 542–554.

Ang, K. K., Chin, Z. Y., Wang, C., Guan, C., & Zhang, H. (2012). Filter bank common spatial pattern algorithm on BCI competition IV datasets 2a and 2b. *Frontiers in Neuroscience*, *6*, 1–9.

Ang, K. K., Chua, K. S. G., Phua, K. S., Wang, C., Chin, Z. Y., Kuah, C. W. K., … Guan, C. (2015). A Randomized Controlled Trial of EEG-Based Motor Imagery Brain-Computer Interface Robotic Rehabilitation for Stroke. *Clinical EEG and Neuroscience*, *46*(4), 310–320.

Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein GAN. *arXiv preprint arXiv: 1701.07875*.

Balles, L., Romero, J., & Hennig, P. (2017). Coupling adaptive batch sizes with learning rates. *Uncertainty in Artificial Intelligence - Proceedings of the 33rd Conference, UAI 2017*.

Barachant, A., Bonnet, S., Congedo, M., & Jutten, C. (2012). Multiclass brain-computer interface classification by Riemannian geometry. *IEEE Transactions on Biomedical Engineering*, *59*(4), 920–928.

Barachant, A., & Congedo, M. (2014). A Plug & Play P300 BCI Using Information Geometry. arXiv preprint arXiv:1409.0107.

Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade* (pp. 437-478). Springer, Berlin, Heidelberg.

Bischl, B., Mersmann, O., Trautmann, H., & Weihs, C. (2012). Resampling methods for meta-model validation with recommendations for evolutionary computation. *Evolutionary Computation*, *20*(2), 249–275.

Blankertz, B., Tomioka, R., Lemm, S., Kawanabe, M., & Muller, K. R. (2007). Optimizing spatial filters for robust EEG single-trial analysis. *IEEE Signal processing magazine*, *25*(1), 41-56.

Bollapragada, R., Byrd, R., & Nocedal, J. (2018). Adaptive sampling strategies for stochastic optimization. *SIAM Journal on Optimization*, *28*(4), 3312–3343.

Bottou, L. (1998). Online learning and stochastic approximations. *On-line learning in neural networks*, *17*(9), 142.

Bottou, L., Curtis, F. E., & Nocedal, J. (2018). Optimization methods for large-scale machine learning. *SIAM Review*, *60*(2), 223–311.

Byrd, R. H., Chin, G. M., Nocedal, J., & Wu, Y. (2012). Sample size selection in optimization methods for machine learning. *Mathematical Programming*, *134*(1), 127–155.

Byrd, R. H., Lu, P., Nocedal, J., & Zhu, C. (1995). A limited memory algorithm for bound constrained optimization. *SIAM Journal on scientific computing*, *16*(5), 1190-1208.

Cao, Z., Chuang, C. H., King, J. K., & Lin, C. T. (2019). Multi-channel EEG recordings during a sustained-attention driving task. *Scientific Data*, *6*(1), 1-8.

Chaudhary, U., Birbaumer, N., & Ramos-Murguialday, A. (2016). Brain-computer interfaces for communication and rehabilitation. *Nature Reviews Neurology*, *12*(9), 513–525.

Cho, W., Heilinger, A., Ortner, R., Murovec, N., Xu, R., Swift, J., ... & Guger, C. (2018a, October). Motor rehabilitation for hemiparetic stroke patients using a brain-computer interface method. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (pp. 1001-1005). IEEE.

Cho, J. H., Jeong, J. H., Shim, K. H., Kim, D. J., & Lee, S. W. (2018b, October). Classification of hand motions within EEG signals for non-invasive BCI-based robot hand control. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (pp. 515-518). IEEE.

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1251-1258).

Choo, S., & Nam, C. S. (2020). DCGAN Based EEG Data Augmentation in Cognitive State Recognition. In *IIE Annual Conference. Proceedings* (pp. 1-6). Institute of Industrial and Systems Engineers (IISE).

Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., & Bharath, A. A. (2018). Generative Adversarial Networks: An Overview. *IEEE Signal Processing Magazine*, *35*(1), 53–65.

Dai, G., Zhou, J., Huang, J., & Wang, N. (2020). HS-CNN: A CNN with hybrid convolution scale for EEG motor imagery classification. *Journal of Neural Engineering*, *17*(1).

De, S., Yadav, A., Jacobs, D., & Goldstein, T. (2017, April). Automated inference with adaptive batches. In *Artificial Intelligence and Statistics* (pp. 1504-1513). PMLR.

Devarakonda, A., Naumov, M., & Garland, M. (2017). Adabatch: Adaptive batch sizes for training deep neural networks. *arXiv preprint arXiv:1712.02029*.

Doersch, C. (2016). Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*.

Dornhege, G., Blankertz, B., Curio, G., & Müller, K. R. (2004). Boosting bit rates in noninvasive EEG single-trial classifications by feature combination and multiclass paradigms. *IEEE Transactions on Biomedical Engineering*, *51*(6), 993–1002.

Fahimi, F., Dosen, S., Ang, K. K., Mrachacz-Kersting, N., & Guan, C. (2021). Generative Adversarial Networks-Based Data Augmentation for Brain-Computer Interface. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(9), 4039–4051.

Fernandez-Fraga, S. M., Aceves-Fernandez, M. A., Pedraza-Ortega, J. C., & Tovar-Arriaga, S. (2018). Feature extraction of EEG signal upon BCI systems based on steady-state visual evoked potentials using the ant colony optimization algorithm. *Discrete Dynamics in Nature and Society*, *2018*.

Friedlander, M. P., & Schmidt, M. (2012). Hybrid deterministic-stochastic methods for data fitting. *SIAM Journal on Scientific Computing*, *34*(3), A1380-A1405.

Gan, L., Liu, W., Luo, Y., Wu, X., & Lu, B. L. (2019, December). A Cross-Culture Study on Multimodal Emotion Recognition Using Deep Learning. In *International Conference on Neural Information Processing* (pp. 670-680). Springer, Cham.

Ge, R., Huang, F., Jin, C., & Yuan, Y. (2015, June). Escaping from saddle points—online stochastic gradient for tensor decomposition. In *Conference on learning theory* (pp. 797-842). PMLR.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, *27*.

Graimann, B., Allison, B., & Pfurtscheller, G. (2010). Brain–computer interfaces: A gentle introduction. In *Brain-computer interfaces* (pp. 1-27). Springer, Berlin, Heidelberg.

Graves, A., Mohamed, A. R., & Hinton, G. (2013, May). Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing* (pp. 6645-6649). IEEE.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. C. (2017). Improved training of wasserstein gans. *Advances in neural information processing systems*, *30*.

Guo, N., Gu, K., Qiao, J., & Bi, J. (2021). Improved deep CNNs based on Nonlinear Hybrid Attention Module for image classification. *Neural Networks*, *140*, 158-166.

Hardt, M., Recht, B., & Singer, Y. (2016, June). Train faster, generalize better: Stability of stochastic gradient descent. In *International conference on machine learning* (pp. 1225-1234). PMLR.

Hartmann, K. G., Schirrmeister, R. T., & Ball, T. (2018). EEG-GAN: Generative adversarial networks for electroencephalograhic (EEG) brain signals. *arXiv preprint arXiv:1806.01875*.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, *30*.

Hong, Y., Hwang, U., Yoo, J., & Yoon, S. (2019). How generative adversarial networks and their variants work: An overview. *ACM Computing Surveys*, *52*(1), 1-43.

Huang, D., Qian, K., Fei, D. Y., Jia, W., Chen, X., & Bai, O. (2012). Electroencephalography (EEG)-based brain–computer interface (BCI): A 2-D virtual wheelchair control based on event-related desynchronization/synchronization and state control. *IEEE transactions on Neural Systems and Rehabilitation engineering*, *20*(3), 379-388.

Ingolfsson, T. M., Hersche, M., Wang, X., Kobayashi, N., Cavigelli, L., & Benini, L. (2020, October). EEG-TCNet: An accurate temporal convolutional network for embedded motor-imagery brain–machine interfaces. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (pp. 2958-2965). IEEE.

Ioffe, S., & Szegedy, C. (2015, June). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning* (pp. 448-456). PMLR.

Jia, Z., Lin, Y., Wang, J., Yang, K., Liu, T., & Zhang, X. (2021, September). MMCNN: A multi-branch multi-scale convolutional neural network for motor imagery classification. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (pp. 736-751). Springer, Cham.

Jiang, D., Ren, H., Cai, Y., Xu, J., Liu, Y., & Leung, H. fung. (2021). Candidate region aware nested named entity recognition. *Neural Networks*, *142*, 340–350.

Kandel, I., & Castelli, M. (2020). The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT Express*, *6*(4), 312–315.

Kang, H., & Choi, S. (2014). Bayesian common spatial patterns for multi-subject EEG classification. *Neural Networks*, *57*, 39–50.

Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2017). Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*.

Kaya, M., Binli, M. K., Ozbay, E., Yanar, H., & Mishchenko, Y. (2018). A large electroencephalographic motor imagery dataset for electroencephalographic brain computer interfaces. *Scientific data*, *5*(1), 1-16.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., & Tang, P. T. P. (2016). On large-

batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ko, W., Jeon, E., Jeong, S., & Suk, H. I. (2021). Multi-scale neural network for EEG representation learning in BCI. *IEEE Computational Intelligence Magazine*, *16*(2), 31-45.

Kodali, N., Abernethy, J., Hays, J., & Kira, Z. (2017). On convergence and stability of GANs. *arXiv preprint arXiv:1705.07215*.

Lawhern, V. J., Solon, A. J., Waytowich, N. R., Gordon, S. M., Hung, C. P., & Lance, B. J. (2018). EEGNet: A compact convolutional neural network for EEG-based brain-computer interfaces. *Journal of Neural Engineering*, *15*(5), 1–30.

Lee, J. D., Simchowitz, M., Jordan, M. I., & Recht, B. (2016, June). Gradient descent only converges to minimizers. In *Conference on learning theory* (pp. 1246-1257). PMLR.

Lee, M. H., Kwon, O. Y., Kim, Y. J., Kim, H. K., Lee, Y. E., Williamson, J., ... & Lee, S. W. (2019). EEG dataset and OpenBMI toolbox for three BCI paradigms: an investigation into BCI illiteracy. *GigaScience*, *8*(5), giz002.

Luo, Y., & Lu, B. L. (2018, July). EEG data augmentation for emotion recognition using a conditional Wasserstein GAN. In *2018 40th annual international conference of the IEEE engineering in medicine and biology society (EMBC)* (pp. 2535-2538). IEEE.

Luo, Y., Zhu, L. Z., & Lu, B. L. (2019, July). A GAN-based data augmentation method for multimodal emotion recognition. In *International Symposium on Neural Networks* (pp. 141-150). Springer, Cham.

Luo, Y., Zhu, L. Z., Wan, Z. Y., & Lu, B. L. (2020). Data augmentation for enhancing EEG-based

emotion recognition with deep generative models. *Journal of Neural Engineering*, *17*(5).

Zhang, X., Yao, L., Wang, X., Monaghan, J., Mcalpine, D., & Zhang, Y. (2020). A survey on deep learning based brain computer interface: Recent advances and new frontiers. *arXiv preprint arXiv:1905.04149*.

Masters, D., & Luschi, C. (2018). Revisiting small batch training for deep neural networks. *arXiv preprint arXiv:1804.07612*.

Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Najafabadi, M. M., Villanustre, F., Khoshgoftaar, T. M., Seliya, N., Wald, R., & Muharemagic, E. (2015). Deep learning applications and challenges in big data analytics. *Journal of Big Data*, *2*(1), 1.

Aznan, N. K. N., Atapour-Abarghouei, A., Bonner, S., Connolly, J. D., Al Moubayed, N., & Breckon, T. P. (2019, July). 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)* (pp. 1-8). IEEE.

Panwar, S., Rad, P., Jung, T. P., & Huang, Y. (2020). Modeling EEG data distribution with a wasserstein generative adversarial network to predict RSVP events. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *28*(8), 1720-1730.

Pei, Y., Luo, Z., Yan, Y., Yan, H., Jiang, J., Li, W., ... & Yin, E. (2021). Data Augmentation: Using Channel-Level Recombination to Improve Classification Performance for Motor Imagery EEG. *Frontiers in Human Neuroscience*, *15*, 113.

Perez-Benitez, J. L., Perez-Benitez, J. A., & Espina-Hernandez, J. H. (2018, February). Development of a brain computer interface interface using multi-frequency visual stimulation and deep neural networks. In *2018 International Conference on Electronics, Communications and Computers (CONIELECOMP)* (pp. 18-24). IEEE.

Perez, L., & Wang, J. (2017). The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*.

Pfurtscheller, G., Brunner, C., Schlögl, A., & Lopes da Silva, F. H. (2006). Mu rhythm (de)synchronization and EEG single-trial classification of different motor imagery tasks. *NeuroImage*, *31*(1), 153–159.

Pfurtscheller, Gert, & Lopes da Silva, F. H. (1999). Event-related EEG/MEG synchronization and desynchronization: basic principles. *Clinical Neurophysiology*, *110*(11), 1842–1857.

Radiuk, P. M. (2018). Impact of training set batch size on the performance of convolutional neural networks for diverse datasets. *Information Technology and Management Science*, *20*(1), 20–24.

Rashid, K. M., & Louis, J. (2019). Times-series data augmentation and deep learning for construction equipment activity recognition. *Advanced Engineering Informatics*, *42*, 100944.

Rawat, W., & Wang, Z. (2017). Deep convolutional neural networks for image classification: A comprehensive review. *Neural Computation*, *29*(9), 2352-2449.

Remsik, A., Young, B., Vermilyea, R., Kiekoefer, L., Abrams, J., Elmore, S. E., … Williams, J. (2016). A review of the progression and future implications of brain–computer interface therapies for restoration of distal upper extremity motor function after stroke. *Expert Review of Medical Devices*, *13*(5), 445–454.

Rivet, B., Souloumiac, A., Attina, V., & Gibert, G. (2009). xDAWN algorithm to enhance evoked

potentials: application to brain–computer interface. *IEEE Transactions on Biomedical Engineering*, *56*(8), 2035-2043.

Salakhutdinov, R., & Hinton, G. (2009). Deep Boltzmann machines. *Journal of Machine Learning Research*, *5*(3), 448–455.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., & Chen, X. (2016). Improved techniques for training GANs. *Advances in Neural Information Processing Systems*, 2234–2242.

Schirrmeister, R. T., Springenberg, J. T., Fiederer, L. D. J., Glasstetter, M., Eggensperger, K., Tangermann, M., … Ball, T. (2017). Deep learning with convolutional neural networks for EEG decoding and visualization. *Human Brain Mapping*, *38*(11), 5391–5420.

Shenoy, P., Krauledat, M., Blankertz, B., Rao, R. P. N., & Müller, K. R. (2006). Towards adaptive classification for BCI. *Journal of Neural Engineering*, *3*(1).

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, *6*(1).

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Smith, S. L., Kindermans, P. J., Ying, C., & Le, Q. V. (2017). Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*.

Stewart, D., Seymour, R., Pass, A., & Ming, J. (2014). Robust audio-visual speech recognition under noisy audio-video conditions. *IEEE Transactions on Cybernetics*, *44*(2), 175–184.

Suk, H. Il, & Lee, S. W. (2013). A novel bayesian framework for discriminative feature extraction in brain-computer interfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(2), 286–299.

Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013, May). On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning* (pp. 1139-1147). PMLR.

Tangermann, M., Müller, K. R., Aertsen, A., Birbaumer, N., Braun, C., Brunner, C., ... & Blankertz, B. (2012). Review of the BCI competition IV. *Frontiers in neuroscience*, 55.

Um, T. T., Pfister, F. M., Pichler, D., Endo, S., Lang, M., Hirche, S., ... & Kulić, D. (2017, November). Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks. In *Proceedings of the 19th ACM international conference on multimodal interaction* (pp. 216-220).

Vidaurre, C., Klauer, C., Schauer, T., Ramos-Murguialday, A., & Müller, K. R. (2016). EEG-based BCI for the linear control of an upper-limb neuroprosthesis. *Medical Engineering and Physics*, *38*(11), 1195–1204.

Wang, F., Zhong, S. H., Peng, J., Jiang, J., & Liu, Y. (2018, February). Data augmentation for eeg-based emotion recognition with deep convolutional neural networks. In *International conference on multimedia modeling* (pp. 82-93). Springer, Cham.

Wei, X., Zhou, L., Chen, Z., Zhang, L., & Zhou, Y. (2018). Automatic seizure detection using three-dimensional CNN based on multi-channel EEG. *BMC Medical Informatics and Decision Making*, *18*(5), 111.

Xiong, M., Hotter, R., Nadin, D., Patel, J., Tartakovsky, S., Wang, Y., ... & Zhen, A. (2019, October). A low-cost, semi-autonomous wheelchair controlled by motor imagery and jaw muscle activation. In *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)* (pp. 2180-2185). IEEE.

Yin, Z., & Zhang, J. (2017a). Cross-session classification of mental workload levels using EEG

and an adaptive deep learning model. *Biomedical Signal Processing and Control*, *33*, 30–47.

Yin, Z., & Zhang, J. (2017b). Cross-subject recognition of operator functional states via EEG and switching deep belief networks with adaptive weights. *Neurocomputing*, *260*, 349–366.

Young, T., Hazarika, D., Poria, S., & Cambria, E. (2018). Recent trends in deep learning based natural language processing. *IEEE Computational Intelligence Magazine*, *13*(3), 55-75.

Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2021). Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, *64*(3), 107-115.

Zhang, K., Robinson, N., Lee, S., & Guan, C. (2021). Adaptive transfer learning for EEG motor imagery classification with deep Convolutional Neural Network. *Neural Networks*, *136*, 1–10.

Zhang, Q., & Liu, Y. (2018). Improving brain computer interface performance by data augmentation with conditional deep convolutional generative adversarial networks. *arXiv preprint arXiv:1806.07108*.

Zhao, X., Zhang, H., Zhu, G., You, F., Kuang, S., & Sun, L. (2019). A multi-branch 3D convolutional neural network for EEG-based motor imagery classification. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *27*(10), 2164-2177.

Zhang, Y., Nam, C. S., Zhou, G., Jin, J., Wang, X., & Cichocki, A. (2018). Temporally constrained sparse group spatial patterns for motor imagery BCI. *IEEE Transactions on Cybernetics*, *49*(9), 3322–3332.

Zhang, Y., Zhao, Q., Jin, J., Wang, X., & Cichocki, A. (2012). A novel BCI based on ERP components sensitive to configural processing of human faces. *Journal of Neural Engineering*, *9*(2).

APPENDIX

Appendix A: Input and output sizes for the proposed cGANs

Table A.1. Input and output sizes of each layer for the proposed cGANs depending on dataset.

Model	Module	Layer/Operation	Dataset1			Dataset2		
			Kernel (Stride) Size	Input Size	Output Size	Kernel (Stride) Size	Input Size	Output Size
Gener.	Label Emb.	Embedding	-	4	1×50	-	2	1×50
		Dense	-	1×50	1×32	-	1×50	1×25
		Reshape	-	1×32	1×32×1	-	1×25	1×25×1
	Noise Encoder	Dense +LeakyReLu	-	100	1280	-	100	1000
		Reshape	-	1280	1×32×40	-	1000	1×25×40
	Trans. Conv.	Concatenate	-	1×32×1 1×32×40	1×32×41	-	1×25×1 1×25×40	1×25×41
		Conv2DTrans(T_1^G) +BatchNorm +LeakyReLU	(1×4)	1×32×41	1×128×40	(1×4)	1×25×41	1×100×41
		Conv2DTrans(S^G) +BatchNorm +LeakyReLU	(22×1)	1×128×40	22×128×40	(118×1)	1×100×41	118×100×40
		Conv2DTrans(T_2^G) +BatchNorm +LeakyReLU	(1×2)	22×128×40	22×256×40	(1×2)	118×100×40	118×200×40
		Conv2D +Sigmoid	1×1 (1×1)	22×256×40	22×256×1	1×1 (1×1)	118×200×40	118×200×1
Discr.	Label Emb.	Embedding	-	4	1×50	-	2	1×50
		Dense	-	1×50	1×5632	-	1×50	1×23600
		Reshape	-	1×5632	22×256×1	-	1×23600	118×200×1
	EEG Decoder	Concatenate	-	22×256×1 22×256×1	22×256×2	-	118×200×1 118×200×1	118×200×2
		Conv2D(T_1^D)	1×13 (1×13)	22×256×2	22×256×40	1×13 (1×13)	118×200×2	118×200×40
		Conv2D(S^D) +BatchNorm +Square	22×1 (1×1)	22×256×40	1×256×40	22×1 (1×1)	118×200×40	1×200×40
		AvgPool2D(T_2^D) +Log+Dropout	1x35	1×256×40	1x32x40	1x35	1×200×40	1x24x40
		Flatten	-	1x32x40	1280	-	1x24x40	960
		Dense	-	1280	1	-	960	1