Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Nikolai Molvik

Data augmentation using GANs in EEG-based biometric systems

Master's thesis in Cybernetics and Robotics
Supervisor: Professor Marta Molinas
Co-supervisor: Luis Alfredo Moctezuma
December 2021

NTNU
Norwegian University of
Science and Technology

Nikolai Molvik

Data augmentation using GANs in EEG-based biometric systems

Master's thesis in Cybernetics and Robotics
Supervisor: Professor Marta Molinas
Co-supervisor: Luis Alfredo Moctezuma
December 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Acknowledgements

This thesis is the last work I submit as a part of my Master of Science degree in Cybernetics and Robotics at The Norwegian University of Science and Technology, NTNU.

First, I would like to thank my co-supervisor Luis Alfredo Moctezuma for help during the work and writing of my specialization project and masters thesis. It's been really helpful. Also, thank you to my supervisor professor Marta Molinas.

To mamma, pappa, Lisa, Birte and Mille - the last years would not be possible without you. Thank you for all your support.

Thank you to all my study mates in MØkom. Studying with you has been a pleasure. It's been hard, but we made it together.

The months since March 12th, when the COVID-19 pandemic started to affect our lives in Trondheim have been special. Anyway, this was no problem, because everyday, in Elgeseter Gate 13, I could spend time with good friends. Thank you to Gard, Gustav, Herman and Magne. I will miss the Saturday mornings with Nytt på Nytt and too strong coffee.

I would also like to thank Hogne for feed back and discussions during writing of this thesis.

Acknowledgements - Norwegian

Denne masteroppgaven er det siste arbeidet jeg leverer som en del av mastergraden min i Kybernetikk og Robotikk ved Norges teknisk-naturvitenskapelige universitet, NTNU.

Først ønsker jeg å takke med-veileder Luis Alfredo Moctezuma for faglig hjelp og oppfølging under arbeidet med prosjekt og masteroppgaven min. Det har vært nyttig å ha noen diskutere med. Jeg ønsker også å takke veilederen min professor Marta Molinas.

Mamma, pappa, Lisa, Birte og Mille - de siste årene hadde ikke vært gjennomførbare uten dere. Takk for all støtte dere har gitt meg. “Dåkker e gull å ha.”

Jeg ønsker også å takke studievennene mine i MØkom. Det å studere med dere har vært en lek. Ikke fordi det har vært lett, men fordi vi, sammen, har hatt utallige gode øyeblikk, og, sammen har kommet oss gjennom dette.

Tiden siden 12. mars 2020, da korona-pandemien begynte å påvirke livene våre i Trondheim har vært spesiell. Likevel har det gått veldig fint, på grunn av jeg hver dag, i Elgeseter Gate 13, har kunnet være med uvurderlige venner. Takk til Gard, Gustav, Herman og Magne. Jeg kommer til å savne lørdagsmorgenene med Nytt på Nytt og (til tider for sterk) kaffe i koppen.

Jeg ønsker også å takke Hogne for tilbakemeldinger og diskusjon i sammenheng med skriving av denne oppgaven.

Abstract

Brain waves, recorded using electroencephalography(EEG) have a variety of applications. Historically EEG has mainly been used in medicine, but applying state of the art technology, applications such as EEG-based biometric systems are enabled, and have shown promising results. Long calibration times for adding new users to such a system is a challenge that needs to be investigated. A reduced calibration time would make the system more user friendly and could mitigate errors in the EEG data, such as those caused by the user being distracted during data collection. A reduction in calibration time can be achieved by using data augmentation techniques. This consists of generating new data to artificially increase the amount of collected data. Generative Adversarial Networks(GANs) have shown the ability to generate high quality EEG data, and in this thesis GANs are used to generate EEG-data which mimics specific subjects in order to reduce calibration time.

A state of the art biometric system is implemented, as well as a data augmentation procedure using Deep Convolutional GANs (DCGANs). The implemented methods are tested using the Max Planck Institut Leipzig Mind-Brain-Body Dataset(LEMON), a dataset containing EEG-recordings of subjects in resting state.

The EEG data was pre-processed to reduce noise and cut into instances of 1 second. The feature extraction applied Empirical Mode Decomposition (EMD), where the two most relevant components were selected. For each instance the Instantaneous energy, Teager energy, Higuchi fractal dimension and Petrosian fractal dimension were calculated, resulting in a feature vector. For each subject a GAN was trained with a training set. There was also trained a Local Outlier Factor One Class Classifier (LOF OCC) for each subject using the training set, with and without data augmentation. The biometric system was evaluated using the metrics True acceptance rate (TAR) and True rejection rate (TRR).

When training the biometric system with 40 seconds of EEG-data a TAR/TRR of $0.926/0.619$ was achieved. When augmenting the dataset, such that the biometric system was trained on 40 seconds of real EEG-data + 360 seconds of generated EEG-data, TAR/TRR was decreased to $0.915/0.618$. This shows that by appending generated EEG-data to the training set, the system performance is decreased.

Based on the results, it was concluded that the used GAN configuration does not provide EEG-data similar enough to the EEG-data of each subject. The result of this is that the biometric system does not benefit from the data augmentation. Possible suspects for these lackluster results are the low amount of training data and a GAN configuration not permitting stable training of the GANs. These were deemed to be the most important challenges to work with in future iterations of data augmentation using GANs in EEG-based biometric systems.

Abstract - Norwegian

Hjernebølger, målt med electroencephalography(EEG) har mange applikasjoner. Historisk sett har EEG hovedsaklig blitt brukt i medisin, men ved bruk av state of the art teknologi, kan EEG brukes i biometriske systemer, noe som har vist lovende resultater. Lang kalibreringstid for å legge til nye brukere i slike systemer krever videre forskning. Å redusere kalibreringstiden kan gjøre systemet mer brukervennlig, og det kan redusere feil i EEG-dataen som forekommer av at brukeren blir distrahert under data-innhenting. Redusert kalibreringstid kan oppnås med data augmentation-teknikker. Dette består av å generere ny data, for å få en kunstig økt mengde innsamlet data. Generative Adversarial Networks(GANs) har vist at de kan generere høy-kvalitets EEG-data, og i denne masteroppgaven brukes GANs til å generere EEG-data som etterligner spesifikke brukere, for å oppnå en redusert kalibreringstid.

Et state of the art biometrisk system er implementert, i tillegg til en data augmentation prosedyre baser på Deep Convolutional GANs (DCGANs). Metodene er tested med Max Planck Institut Leipzig Mind-Brain-Body Dataset(LEMON), et dataset som inneholder EEG-målinger av personer i resting state.

EEG-dataen ble pre-prosessert for å redusere bråk i signalene, og delt opp i instanser på 1 sekund. Feature extraction var basert på Empirical Mode Decomposition (EMD), og de to mest relevante komponentene ble valgt. For hver instans ble Instantaneous energy, Teagerenergy, Higuchi fractal dimension og Petrosian fractal dimension kalkulert, noe som resulterte i en feature vector. For hver bruker ble et GAN trent med treningssettet. Det ble også trent en Local Outlier Factor One Class Classifier (LOF OCC) for hver bruker med treningssettet, både med og uten data augmentation. Det biometriske systemet ble evaluert med metrikkene True acceptance rate (TAR) og True rejection rate (TRR).

Når det biometriske systemet ble trent på 40 sekunder EEG-data, oppnådde systemet en TAR/TRR på $0.926/0.619$. Når treningssettet gjennomgikk data augmentation, slik at det biometriske systemet ble trent på 40 sekunder ekte EEG-data og 360 sekunder generert EEG-data, ble TAR/TRR redusert til $0.915/0.618$. Dette viser at å legge til generert EEG-data til treningssettet reduserer systemets ytelse.

Basert på resultatene, ble det konkludert med at den brukte GAN-konfigurasjonen ikke genererte EEG-data lik nok EEG-dataen til hver bruker. Resultatet ble at det biometriske systemets ytelse ikke ble forbedret med data augmentation. Mulige mistenkte for disse resultatene er den lave mengden treningsdata og at GAN-konfigurasjonen ikke tillot stabil trening av GANen. Disse ble ansett som de viktigste utfordringene å jobbe med i videre iterasjoner til data augmentation ved bruk av GANs i EEG-baserte biometriske systemer.

Table of Contents

Acknowledgements i

Acknowledgements - Norwegian ii

Abstract iii

Abstract - Norwegian iv

1 Introduction 1
1.1 Background . . . 2

2 Theory 3
2.1 Machine learning . . . 3
2.2 Local Outlier Factor . . . 4
2.3 Artificial Neural networks . . . 4
2.3.1 Training of neural networks . . . 5
2.3.2 Layers . . . 6
2.4 Generative adversarial networks . . . 7
2.4.1 Training of GANs . . . 8
2.4.2 Modifications to stabilize GAN training . . . 8
2.4.3 Evaluation of GANs . . . 9
2.4.4 Deep concolutional GAN . . . 10
2.5 Filtering . . . 10

3 Materials and methods 13
3.1 Dataset . . . 13
3.2 Pre-processing . . . 14
3.2.1 Filtering . . . 14
3.2.2 Down sampling . . . 14
3.2.3 Cutting to instances . . . 14
3.2.4 Normalization . . . 15
3.2.5 Pre-processing layout . . . 15
3.3 Feature extraction . . . 16
3.3.1 Decomposition . . . 16

3.3.2 IMF selection . . . 16
3.3.3 Feature calculation . . . 16
3.3.4 Feature extraction layout . . . 17
3.4 Classification . . . 17
3.5 Data augmentation using GANs . . . 17
3.6 System layout . . . 20
3.7 Evaluation of GANs using GAN-train and GAN-test . . . 21
3.8 Software and hardware used . . . 21

4 Experiments and results **23**
4.1 Finding a baseline . . . 23
4.2 Inspection of real instances . . . 27
4.3 Generation of instances . . . 28
4.3.1 DCGAN . . . 28
4.3.2 Tuning DCGAN hyperparameters . . . 30
4.4 Subject identification with augmented data set . . . 32

5 Discussion **35**
5.1 Discussion . . . 35

6 Conclusion **39**
6.1 Conclusion . . . 39
6.2 Future work . . . 40

1 Introduction

Biometric systems enable automatic identification of subjects - which can be used in countless practical applications. These systems use biometric traits, i.e. something that can be measured from the body to perform identification. They eliminate the need for remembering passwords or bringing key cards - the user simply need to be present and present the biometric trait to some sensor. Biometric systems based on fingerprints or facial recognition are well researched [1, 2] and have for a while been available in commercial products as one of the most popular biometrics. Recently, biometric systems based on brain waves captured though electroencephalography(EEG) have shown to be a strong competitor to the well established biometrics, and are in some ways superior to its alternatives [3].

For a subject to be added to the biometric system, the system needs to be presented the biometric trait of the specific subject. In this way, the system can learn the patterns in the biometric measurements of the subject. For users of smart phones with finger print scanners this is a well known process, where the user scan its finger a few times before being able to unlock their phone using the biometric trait, i.e. their fingerprint. This is a process called calibration, where training data for a machine learning(ML) algorithm is being collected. In a biometric system it is preferred, based on comfort for the user and other practical aspects to keep the calibration time to a minimum. Unfortunately, its is essential to collect enough data during calibration to ensure a biometric system which is able to identify subjects with high precision. Gathering data require time, hence a trade off between gathering enough data and keeping a low calibration time needs to be found. There is a need for reducing calibration time in state of the art EEG based biometric systems, which was addressed for future work in [4].

The results of deep learning have the last years showed great results in many fields of research. One of these fields is data generation, where generative models such at generative adversarial networks(GANs) have showed promising results. These generative models have been used to generate data to be used for training ML systems. This is a kind of data augmentation, which is a technique for increasing the amount of training data available. GANs have previously been used for generating EEG-data, which have been used to do data augmentation to reduce calibration time in brain computer interfaces [5], for various tasks. However, to the authors knowledge, they have never been used to do data augmentation in EEG-based biometric systems.

1.1 Background

In [6], the specialization project delivered June of 2021, the same author as this thesis implemented a EEG-based biometric system. The goal of the project was finding a suitable configuration for identifying subjects in the EEGMIDB dataset[7], and reducing calibration time using a data augmentation technique. The EEG-data was pre-processed using Common Average Reference(CAR), cut into 1 second long instances and decomposed using Empirical Mode Decompostion (EMD). Four features were calculated for each Intrinsic Mode function(IMF) in each channel. Instances were generated through combining IMFs from different instances to create new instances. The generated instances most similar to the average real instance for the corresponding subject was selected to be a part of the training set. A Local Outlier Factor One Class Classifier (LOF OCC) was trained for each subject, and the system was tested with different quantities of both real and generated instances. It was demonstrated that the performance of the biometric system benefited from more training data. The system was evaluated using the metrics True acceptance rate (TAR) and True rejection rate (TRR). The generated instances were not able to improve the performance of the system in the case of the optimal configuration, but, was able to improve TAR/TRR by $0.162/0.036$ in the case of a sub optimal configuration. The best TAR/TRR achieved by the biometric system was $0.875/0.945$, using a sub set of the available EEG-channels and no generated instances for training.

The work in this thesis is a continuation of the described project. A new biometric system is implemented with improvements found in literature, and evaluated using another dataset. The calibration time is intended reduced using a new data augmentation technique, based on GANs for EEG data generation. Note that this is a thesis focusing on GAN-based data augmentation, hence the development and optimization of system performance of the biometric system itself will receive limited attention. A research question was formed and will be intended answered during this masters thesis.

Can a GAN-based data augmentation technique be used to reduce the calibration time for adding new subjects in EEG-based biometric systems?

2 Theory

In this chapter the theory relevant to the thesis is presented.

2.1 Machine learning

ML is a sub-field of artificial intelligence(AI), focused on designing algorithms that automatically can learn to perform some task [8]. These algorithms can do tasks like regression, classification, automatic control and more, exploiting the underlying patterns in presented data. ML models are trained, a process where data is used to help the model find the underlying patterns in the data. The goal of a ML model is to generalize, which is achieved if the underlying pattern in the data is found and the model can do its designated task on unseen data with satisfactory results. If the model only have learned to "remember" the presented data and does not perform well on unseen data, it is said that the model is overfitted.

ML consist of three sub fields - supervised learning, unsupervised learning and reinforcement learning. Depending on the task and data available one of these approaches of ML has to be chosen when doing a ML project.

ML models are defined by its parameters and hyper parameters. The parameters are variables that are learned during training. These are traditionally not manually tuned by the developers, but found applying some training method, often based on optimization as the parameter space can be large and highly complex. Hyper parameters are parameters not learned during training, but are variables defining the model and controlling its learning. These are traditionally manually set by the developer. Similar to the parameters the hyper parameter space can be large and complex as well, hence automatic methods for finding and evaluating hyper parameters can be used.

Essential for ML is the dataset. The dataset is traditionally divided into three parts, which all have their own specified task - train, validation and test. The training set is used to train the model, i.e. find the parameters. During training the validation set can be used to give the developer insight into the performance of the model. While evaluating model performance with the validation set the hyper parameters may be tuned, such that the best model for the task may be found. The test set is used for evaluating the model performance, after the model is trained and the hyper parameters are found. It is important that training

and hyper parameter tunining is stopped after the system is evaluated using test set. This ensures that the model is not overfitted to the presented data, which would lead to a naively high estimate of model performance.

2.2 Local Outlier Factor

This section, containing theory about the LOF algorithm was originally a part of the specialization project[6], but is copied, modified and extended as a part of this thesis.

Local Outlier Factor(LOF) is an unsupervised ML algorithm used for outlier detection [9]. The classification is done calculating the local density of a data sample. The local density of the data is compared with the local density of the $n_{neighbours}$ closest neighbours in input space, which is used to calculate the LOF of each sample. What classification the classifier gives for different LOF-values can be seen in eq. (2.1).

$$\begin{aligned} &LOF \sim 1 \rightarrow \text{Density similar to inliers, should be a inlier} \\ &LOF < 1 \rightarrow \text{Inlier} \\ &LOF > 1 \rightarrow \text{Outlier} \end{aligned} \tag{2.1}$$

For calculating the LOF of a data sample several algorithms can be used, popular candidates are the Brute, Kd Tree and Ball Tree.

LOF can be used as a OCC, where it is trained on data from a class, and all data not belonging to the respective class are considered outliers. By using the one-vs-all approach such a classifier can even be extended to do multi class classification. This requires a LOF model for each class, and the data to be classified need to be evaluated once per LOF.

2.3 Artificial Neural networks

Artificial Neural networks(ANNs) are a kind of ML algorithms which is inspired by the biological brain. Brains contain large networks of neurons, which communicate with nearby neurons through synapses. In ANNs each neuron is modeled as a unit with inputs and outputs, and have parameters. Each neuron sum their inputs and pass it through some non-linear function (called an activation function), passing the output to the next neuron. With networks of neurons complex functions can be approximated, such that a variety of ML tasks are enabled. As the ANNs in many applications can become deep, which means that they consist of many layers of neurons, the research field of ANNs is often called deep learning.

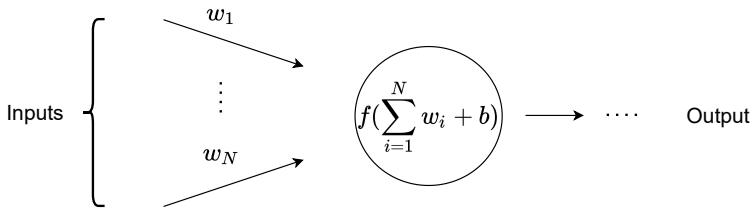


Figure 2.1: ANN Neuron.

A model of a neuron is is shown in fig. 2.1, where w are the weights, b is the bias and f is the activation function. w and b are the parameters associated with this neuron, i.e. the parameters which are learned during training. The inputs come from the neurons in the previous layer, and the output is sent to the next layer.

2.3.1 Training of neural networks

When training a ANN a loss function is defined, and minimized using training data. For a regression problem the mean squared error(MSE) is a popular loss function, as defined in eq. (2.2).

$$MSE = \frac{1}{N}\sum_{i=0}^{N}(\hat{y}_i - y_i)^2 \tag{2.2}$$

Where N is the number of samples, $\hat{y}_i$ is the predicted value for sample i and y_i is the target value for sample i.

The MSE loss is designed such that predictions of the model get better when minimized.

If the network is being trained for a classification problem, another loss function need to be applied. Cross entropy is a popular choice, and in the case of binary classification the binary cross entropy(BCE) may be used, eq. (2.3).

$$BCE = \frac{1}{N}\sum_{i=0}^{N} -(y_i log(p_i) + (1 - y_i) log(1 - p_i))) \tag{2.3}$$

Where N is the number of samples, y_i has a value in $\{0, 1\}$ depending on the class of cample i, and p_i is the probability that sample i belong to class 1. It can be observed that if sample i belong to class 1 ($y_i = 1$), only $y_i log(p_i)$ will be active as the other term is cancelled out, and vice verca.

The prediction $\hat{y}_i$ in eq. (2.2) and p_i in eq. (2.3) are functions of the parameters in the ANN, and is what we are able to change during training to improve model performance. This is done through finding the gradient in parameter-space using back propagation, and gradient

descent. Gradient descent is an iterative optimization algorithm. In an implementation, this means setting N to the number of samples in the training set, finding the gradient, and updating the parameters of the model iteratively. Each time all the training data is looped over and the gradient is calculated is called an epoch. Using this algorithm the parameters in the model are updated once per epoch.

As ANNs require large amounts of training data (large values for N), gradient descent may give problems with memory in practice. This is the reason for using mini batch stochastic gradient descent(mini batch SGD) [10]. The training set is split into mini batches, and the gradient is found for each of these. This can be seen as an estimation of the total gradient, and the parameters in the model can be updated after every mini batch, or after the whole training set is looped through. Using SGD the parameters can be updated many times per epoch, leading to faster convergence in model performance both in time and per epoch.

A state of the art modification to mini batch SGD is the Adam optimizer [11]. Adam combines adaptive momentum and Root Mean Square Propagation to calculate the gradient. Input parameters to this algorithm are β_1, β_2 and learning rate.

2.3.2 Layers

ANNs are organized in sequential layers, which consist of neurons. All ANNs contain a input and a output layer, and layers between which are called hidden layers. Each layer has a width, which refers to the number of neurons in the respective layer, and the network has a depth which refer to the number of layers in the network. As the width and depth of a ANN increase, its number of parameters increase as well. Its also called that this increase the models capacity.

Layers can have a variety of different properties. Here different layers and properties relevant to layers are explained:

- Fully connected layers - in a fully connected layer each neuron has a connection to each neuron in the following layer.
- Convolutionals layer - in a convolutional layer each neuron is only connected to a subset of neurons in the following layer. The convolution operation is a set of filters sliding over the input, performing a local filter operation, where the hyper-parameter $kernel\ size$ decide the size of the filter, and $stride$ decide the how much the filters should shift after each operation. The $padding$ hyper-parameter decide if the layer should append a padding to the input. All these parameters affect output dimensions. What operation the filter actually perform depend on the parameters, such that the filter operation is learned during training. Each filter is applied over the whole spatial input space, such that a reduced number of parameters is used, and thus need to be learned. The convolutional layer has a $depth$, which correspond to how many filters the convolutional layer contain.
- Transposed convolutional layers - a layer upsampling dimensionality through local operations. It is often called a deconvolutional layer, even though it is not a deconvolution operation.

- Batch normalization layers - a layer which perform a nomalization operation.

- Pooling layer - pooling layers are used to down sample the dimensionality in the layers, and can perform operations such as max/min/avg of the different inputs, and output a single value.

- Activation functions - neurons contain activation functions, which is the way to introduce non-linnearity to the network. As demonstrated in fig. 2.1 the input for the neurons are summed, and passed through a function $f(.)$, which is the activation function. The activation function may be a linnear function as well ($f(x) = ax + b$, where parameters a, b are learned), but note that sequencial linnear layers may be reduced to a single linnear layer. Example of popular activation functions are Sigmoid, Hyperbolic Tangent(TanH) and Leaky Rectified Linnear unit(leaky ReLu).

2.4 Generative adversarial networks

Generative adversarial networks(GANs) are a subclass of neural networks used for generating data [12]. If a stable and well functioning GAN is achieved, the generated data is from from the same distribution as the presented training data, from now on called p_{data}. GANs have shown great capabilities in generation of images, but have also proved useful in other research fields. Later publications have modified the GAN, and extended it to be able to generate data from multiple classes, such at the conditional GAN [13]. The original GAN consists of two ANNs:

- Generator(G) - The generator generate data, and take a random noise vector z as input, and output the generated data. The generated samples should converge to be from distribution p_{data}.

- Discriminator(D) - The discriminator tries to detect whether a presented data is a fake sample generated by G, or a real sample from the training data. This network is a binary classifier.

The original GAN arcitecture consist of ReLU and Sigmoid activation functions in G, and Maxout/max activation functions in D. The structure of the GAN is demonstrated in fig. 2.2, where the dotted line represent the output of D which is used to update the parameters in G and D using backpropagation.

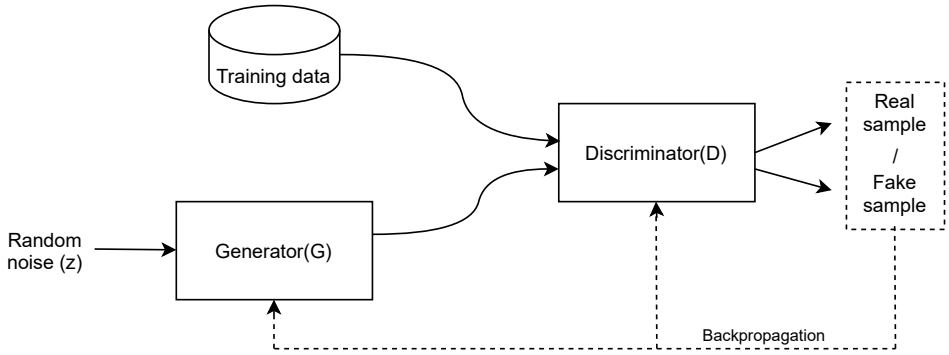


Figure 2.2: GAN structure.

2.4.1 Training of GANs

The two ANNs in the GAN play a min-max game during training, where the G tries to generate samples D classify as 'real', and D is served both fake and real samples, and tries to classify if the presented sample if real or fake. The optimization problem for training the original GAN, published by Ian Goodfellow in 2014 is presented in [12].

$$min_G max_D V(D,G) = E_{x\sim p_{data}(x)} log D(x) + E_{z\sim p_{data}(z)} log(1 - D(G(z)) \quad (2.4)$$

Where x is a real sample, p_{data} is the theoretical distribution x is sampled from, $D(\cdot)$ is the probability that $\cdot$ is a 'real' sample, z is a random noise vector and $G(z)$ is a generated sample. Note that this loss function is similar to the BCE loss function described in section 2.3.1, and consist of two terms. The first serves the purpose to train D in its classification task, and the second to train G in its generation task.

As the networks in the GAN are trained simultaneously, and G is not trained directly on the training data - the training procedure is highly fragile. It is important that their performance is increased gradually and one of the networks does not get better than the other too fast, which will lead to unstable training and no convergence towards the desired results. It is considered hard to find this equilibrium. GAN training is considered a open research question - there are many pitfalls.

Problems such as mode collapse may occur, leading to the whole space of z, called the latent space, being mapped to a limited span of outputs generated by G. This leads to G generating the same or similar data, despite different z noise vectors applied to the input.

2.4.2 Modifications to stabilize GAN training

Two selected modifications to stabilize GAN training are covered.

One sided label smoothing. Deep learning classifiers some times tend to assign a too confident probability that a sample belong to some class [14]. Because of this, it may be advantageous to encourage the classifier to classify softer probabilities. This can be achieved by changing the target values [15], such that the probability of a sample being of some class is 0.8 or 0.9 instead of 1. D in the GAN is a classifier, hence this trick may be applied in GAN training.

Two Time-Scale Update Rule(TTUR). In [16] it is mathematically argued that choosing different learning rate for G and D may contribute to stabilize the training of GANs, a method called TTUR. The method was tested using several types of GANs and data sets, improving the training procedure.

2.4.3 Evaluation of GANs

One of the main challenges related to training a GAN, is evaluating the results. In training of for example an ANN for regression, the loss function is a direct measure of how well the model perform. In the case of the GAN, the loss function is not a measure of how close the generated data distribution is to p_{data}, which complicate the evaluation of the results. This has lead to visual inspection of generated samples being a popular approach for evaluating quality in the case of image generation, but it is both subjective and time consuming. In the case of generation of other types of data, visual inspection may be less effective and feasible. The training of the GAN can be supervised through the loss of G and D, but it does not necessarily give information about the quality of the generated data. A healthy G and D loss should converge, where G it the end of training estimate p_{data} well, and D is not able correctly classify between the training data and the generated data.

In the case of GANs generating data from several classes, the metrics GAN-train and GAN-test may be applied for evaluating the generated data of a GAN [17]. GAN-train is a metric computed through training a classifier for classifying between the classes using generated data, and evaluate the classifier with a evaluation set. If the GAN is not able to estimate p_{data} for each class perfectly, the validation score of the classifier trained on generated data will be lower than the classifier trained on real data. This may be due to several reasons. First of all, the GAN may have failed in estimating p_{data} for the classes, such that the classifier is trained on a totally different distribution than what it is evaluated on. It can also be due to the GAN only estimating part of p_{data} for each class, such that the diversity in generated data is reduced compared to the training data, which may be because of mode collapse. It may also be because of the GAN mixing between the classes. However, if GAN-train is close to validation accuracy, it shows that the quality of the generated data is good, and that it captures the diversity in the classes in the training set. The other score, GAN-test is calculated through training the same classifier using the original training set, and evaluated using generated data. If the GAN-test is higher than the validation score of the same classifier, it is a sign that the GAN is overfitted, such that it only reconstruct training data. However, if it is significantly lower, it is a sign that that the GAN have poorly estimated p_{data} of each class, and the generated data is of low quality.

2.4.4 Deep concolutional GAN

The Deep Convolutional GAN(DCGAN) is a implementation of a GAN, first published in 2016 [18]. The authors combined the concepts of deep convolutional ANNs, which had shown great abilities in image classification with the idea of generating images with GANs. The result was DCGAN, which is a GAN using convolutional layers in both G and D. The authors of DCGAN inctroduced several tricks for constructing a better architecture and training - enabling a better estimation of p_{data}, and more stable training.

DCGAN uses no fully connected layers, in contrast to the original GAN. In the generator the ReLu activation function is used, with TanH at the output layer. In the discriminator the LeakyReLu is used. Batch nomalization layers are applied, contributing to more stable training as it makes sure that the output of the layer is close the the non-linnear region of the ReLu and LeakyReLu layers. During training the mini-batch SGD with Adam optimizer (section 2.4.1) was used, with a batch size of 128. The best results were achieved when normalizing all input values to zero mean and unit variance.

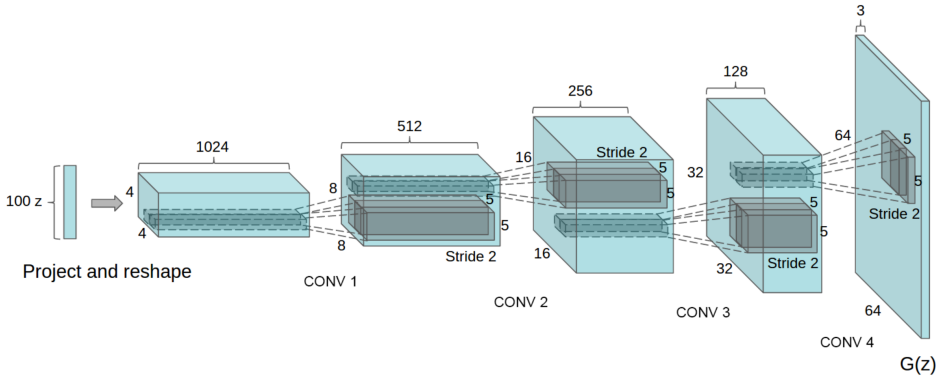


Figure 2.3: Structure of G in original DCGAN. Figure taken from [18].

Figure demonstrating the structure of G in the original DCGAN. It can be observed that the output of this generator is a 3-channel image of $64 \cdot 64$ resolution, hence its shape is $3 \cdot 64 \cdot 64$.

2.5 Filtering

For removing undesired content (e.g. noise) of a signal, filters can be applied.

Filters can be applied to remove a range of frequencies, and let frequencies unde, over or in certain bands pass through. Filters have the disadvantage of affecting the phase of the signal. By filtering in both directions with the same filter the phase is not affected. Such filters are called zero phase. Such a filter can be applied to a signal using eq. (2.5).

$$y(z) = H(z^{-1})H(z)x(z) \tag{2.5}$$

Where $y(z)$ is the z-transform of the output, $H(z)$ is the transfer function of the filter in z-domain and $x(z)$ is the z-transform of the input signal to be filtered.

Filters can be designed using the Butterworth filter [19]. The transfer function of a generalized Butterworth filter in Laplace domain is shown in eq. (2.6).

$$H(s) = G_0 \prod_{k=1}^{n} \frac{\omega_c}{s - s_k} \tag{2.6}$$

Where the G_0 is the gain at zero frequency, n is the order of the filter, ω_c is the cut-off frequency and $(s - s_k)$ is a Butterworth polynomial. Note that for this filter to fit into eq. (2.5) the filter need to be discretized.

These filters can be designed to be both low-pass, high-pass, band-stop and band-pass filters.

3 Materials and methods

In this chapter the methods used in the thesis work is presented and connected to relevant theory.

3.1 Dataset

It was searched for a well suited dataset of EEG for the experiments in this thesis. Several factors were kept in mind when considering datasets:

- Longer recordings of EEG - GANs and deep learning models in general are known for required high amounts of training data. Because of this, it is beneficial with longer EEG-recordings to train the GANs for data augmentation.
- Resting state paradigm - EEG captured through resting state paradigm is used in [6] project and other studies for subject identification, hence the selected data set should be using the same paradigm to be able to use the same techniques(EMD,feats, classificatin schedule etc,).
- Big pool of subjects - The biometric system, including the data augmentation technique should be tested using as many subjects as possible. The effect of including many subjects in the biometric system is interesting for a real world product.
- Multiple sessions - There may be dependencies between the instances if they are from the same session. If the data is captured in several sessions, there may be more data with less (possible) dependencies between the samples. This can result in a dataset more representative for the subject, leading to better trained classifiers and a more realistic estimate of system performance.

It was concluded to use the Max Planck Institut Leipzig Mind-Brain-Body Dataset (LEMON). LEMON consist of MRI and EEG recordings of 228 healthy subjects. The EEG recordings followed the resting state paradigm, and the scans we're divided into 16 blocks of 1 minute, sampled at $2500Hz$, recorded in one session. During 8 of these blocks the eyes were open, and 8 eyes closed. In both cases the subjects were instructed to remain awake, and in the eyes open case the subjects focused on a low contrast grey background. The dataset offers the raw EEG recordings, and pre-processed EEG recordings with ICA. Data

related to gender, age, blood samples, urine drug tests and more is included in the dataset as well. The EEG-electrodes were placed according to the 10-10 system [20], with 61 channels. In this report only the raw EEG data with eyes open were used.

This dataset contain long EEG recordings (8 min eyes open and eyes closed per subject), in the correct paradigm which is resting state. It contain many subjects, but all data is from the same session.

3.2 Pre-processing

The raw EEG-measurements from LEMON was pre-processed before it was passed to the rest of the system.

3.2.1 Filtering

First, the data was bandpass filtered at 1-40Hz.This was done using a Butterworth filter of order 16, covered in section 2.5, with backward-forward filtering for ensuring zero phase.

The signal was then filtered using CAR, covered in[6]. This filtering technique is effective to remove noise affecting all channels at the same time.

The goal of the filtering stage of pre-processing is to remove any noise from the signal, such as power line noise etc., and hence increase the SNR of the signal.

3.2.2 Down sampling

As the signal was sampled at 2500 Hz, the signal was down-sampled to $250Hz$. From Nyquist sampling theorem [21], this creates a limitation, and the down-sampled signal will only contain frequencies up to $\frac{250Hz}{2} = 125Hz$.

The signal is down-sampled to reduce the amount of data, as this sample rate is more similar to the sample rate used in other biometric systems [4].

3.2.3 Cutting to instances

As the raw signal consist of 8 1-minute long measurements, there was a need to break the EEG-recordings into smaller units. It was decided to make each unit of length $1s$. These 1s long units are from now on called instances. As sampling frequency is $250Hz$, each instance is of 250 datapoints, each channel.

It was decided to make the instances of 1 second length as this earlier is used in the biometric systems [4].

3.2.4 Normalization

The instances were normalized to be zero mean, with unit variance. This was done in the original DCGAN [18], and contributes to stabilization of the GAN-training, as it makes sure that the values are around the non-linear region of the ReLu activation function. As the classifiers are trained on a mix of real and generated instances, and classifiers evaluated using real instances, it is important that all instances are normalized such that they come from the same distribution as the generated instances from the GANs.

3.2.5 Pre-processing layout

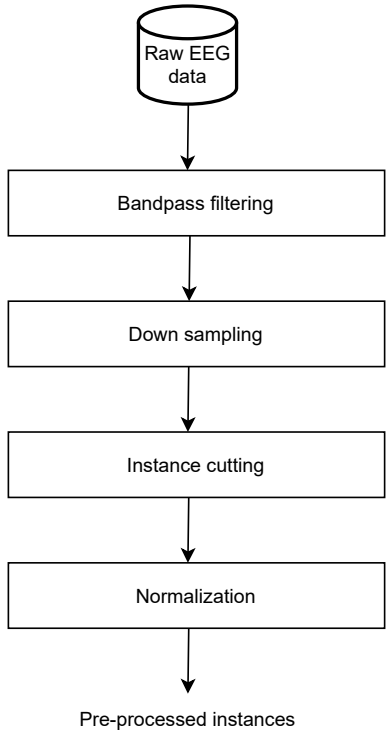


Figure 3.1: Flow chart of pre-processing schedule.

The pre-processing schedule is demonstrated in fig. 3.1. The raw EEG data is filtered using backward-forward filtering and CAR, down sampled to $250Hz$ and normalized such that the instances are zero mean and have unit variance.

3.3 Feature extraction

The pre-processed instances were passed through feature extraction, resulting in a feature vector for each instance.

3.3.1 Decomposition

The channels in all instances were decomposed to its IMFs using EMD algorithm, covered in [6].

This is a process which earlier have been used in biometric systems [4], and have proved to be effective to ectract the subject dependent features.

3.3.2 IMF selection

Some IMFs provided by EMD of a signal may contain limited information due to numerical errors [22]. The two IMFs most similar to the original (pre-processed) instance was selected, calculated according to eq. (3.1).

$$min(\sum_{\forall c} MD(\alpha_c, \beta_c^0), ..., \sum_{\forall c} MD(\alpha_c, \beta_c^n)) \tag{3.1}$$

For a instance α, which is decomposed into IMFs $\beta^0, ..., \beta^n$, MD is the Minkowski Distance function defined in [6] and c are the EEG-channels.

This procedure is earlier used in EEG based biometric systems [23], and is a effective way to reduce dimensionality and remove non-interesting information in the instances, which is due to numerical errors.

3.3.3 Feature calculation

For each instance a feature vector was calculated. The four features, Instantaneous energy, Teager energy, Higuchi fractal dimension, Petrosian fractal dimension, described in [6] was calculated for each channel and IMF, such that a instance was represented in feature space with a vector of length $2 \cdot N_{channels} \cdot 4$, where $N_{channels}$ is the number of channels in the instance.

These features have previously been used in EEG-based biometric systems using EMD, hence they have shown to contain subject dependent information [23].

3.3.4 Feature extraction layout

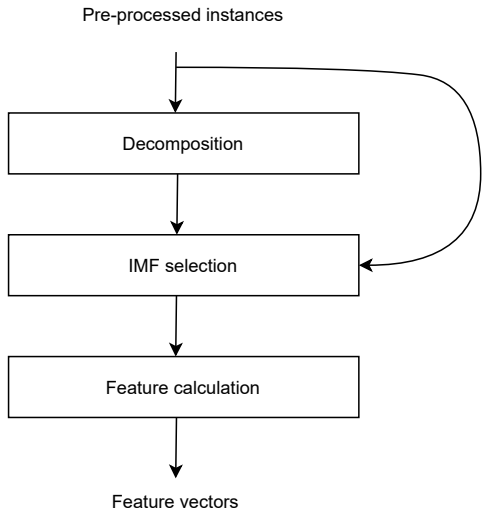


Figure 3.2: Flow chart of feature extraction schedule.

The feature extraction schedule is demonstrated in fig. 3.2. The instances are pre-processed as described in fig. 3.1, before they are sent to feature extraction. In the feature extraction each instance in decomposed using EMD, before the two IMFs are selected, comparing the Minkowski distance between each IMF and the original signal. After IMFs are selected, 4 features are calculated for each IMF in each channel, and concatenated to form a feature vector representing each instance.

3.4 Classification

The classification was performed with LOF OCCs, as described in [6]. A model was trained for every subject, using the training data for that respective subject.

The system was evaluated using the TAR and TRR metrics described in [6].

3.5 Data augmentation using GANs

The DCGAN (section 2.4.4) was used, with the motivation to generate artificial instances. These generated instances can, if similar enough to the instances to a specific subject, be used to augment the training set of the LOF OCCs described in section 3.4 and improve performance. As the DCGAN originally generate images of different dimensions than the EEG-data, the architecture of the networks had to be modified.

Originally the data generated by DCGAN contain a dimension for color-channel, and two dimensions for the pixels as visualized in fig. 2.3. The instances were decided to be interpreted as images with one color-channel, and the rows of the image as EEG-channel time series. As the dataset contain 61 channels and 1 second instances down-sampled to $250Hz$ were used, this led to an "image" of shape $1 \cdot 61 \cdot 250$. The DCGAN implementation contained the parameters ngf and ndf which can be used to modify the depth of the convolutional layers of of G and D respectively. The noise vector z is of length 100.

A implementation of DCGAN was found and modified to fit this exact application. G was forced to output the described shape through tuning the convolution layer's hyperparameters. G's hyper parameter kernel size and stride was tuned, such that given z of length 100, the output dimension of the generated instance was $1 \cdot 61 \cdot 250$. The structure of the tuned network can be seen in table 3.1. A pooling layer with fixed output dimension was added as a last hidden layer in D to ensure a single output number, as it is to perform binary classification., seen in table 3.2.

Generator		
Layer number	Layer	Hyper paramerers
1	Transpose convolutional layer	Depth=$ngf \cdot 8$, Kernel_size= (4,4), stride = (2,2), padding = 0
2	Batch normalization	
3	ReLu	
4	Transpose convolutional layer	Depth=$ngf \cdot 4$,Kernel_size= (4,6), stride = (1,4), padding = 1
5	Batch normalization	
6	ReLu	
7	Transpose convolutional layer	Depth=$ngf \cdot 2$, Kernel_size= (4,4), stride = (2,4), padding = 1
8	Batch normalization	
9	ReLu	
10	Transpose convolutional layer	Depth=ngf, Kernel_size= (4,4), stride = (2,2), padding = 1
11	Batch normalization	
12	ReLu	
13	Transpose convolutional layer	Depth=1, Kernel_size= (6,6), stride = (3,2), padding = 1
14	Tangens hyperbolicus	

Table 3.1: Generator structure

Discriminator		
Layer number	Layer	Hyper paramerers
1	Convolutional layer	Depth=ndf, Kernel_size= (4,4), stride = (2,2), padding = 1
2	Leaky ReLu	
3	Convolutional layer	Depth=$ndf \cdot 2$, Kernel_size= (4,4), stride = (2,2), padding = 1
4	Batch normalization	
5	Leaky ReLu	
6	Convolutional layer	Depth=$ndf \cdot 4$, Kernel_size= (4,4), stride = (2,2), padding = 1
7	Batch normalization	
8	Leaky ReLu	
9	Convolutional layer	Depth=$ndf \cdot 8$, Kernel_size= (4,4), stride = (2,2), padding = 1
10	Batch normalization	
11	Leaky ReLu	
12	Pooling layer	Average pooling
13	Convolutional layer	Depth=1, Kernel_size= (4,4), stride = (1,1), padding = 0
14	Sigmoid	

Table 3.2: Discriminator structure

It can be observed that the activation function in tables 3.1 and 3.2 contain the activation function as a layer, which is not compatible with the model of a neuron in fig. 2.1. This means that there is no activation function applied in the convolutional layers, and the layer operations are executed sequentially.

For each subject, a DCGAN was trained using the instances from that specific subject, such that it could generate instances from that specific subject. DCGAN have earlier been used to generate EEG instances in brain computer interfaces with promising results [5], which was the reason for choosing this exact type of GAN.

It was considered to develop some selection procedure for the generated instances. By selecting the instances most similar to real instances, a higher quality data augmentation could be achieved. An implementation of this could for example be based on comparing the Power Spectral Density (PSD) of generated instances with PSD of the real instances. However, it was found that many instances are similar, such that this could have selected only similar instances. This could be fixed by selecting instances based one one more criteria, such that a variety of different instances was selected. This could be implemented by for example calculating the cross correlation between the generated instances, and select the most different ones. However, due to limitations on time for working with the thesis, this was not implemented.

3.6 System layout

The whole system with the procedures from sections 3.2, 3.3 and 3.5 will be summarized.

In fig. 3.3 the training schedule of the system is demonstrated in a flow chart. Note that it is only the training set that go through this procedure. Also note that this is the training of the LOF OCC models, and that the GAN is both trained and generate instances in this flow chart.

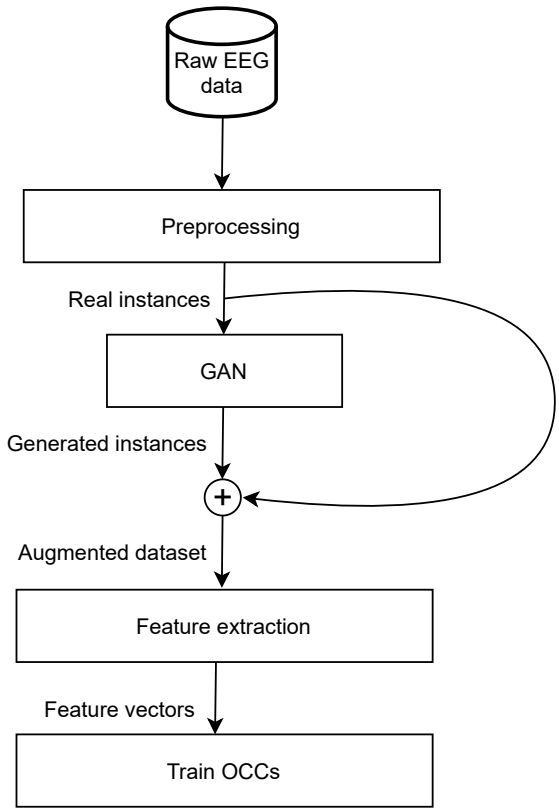


Figure 3.3: Flow chart of training schedule.

In fig. 3.4 the validation schedule is demonstrated in a flow chart. Note that only the validation and test set go through this procedure, and the the instances are classified using the OCCs trained in the training schedule fig. 3.3. The output of the validation is TAR and TRR which are the metric used to evaluate system performance.

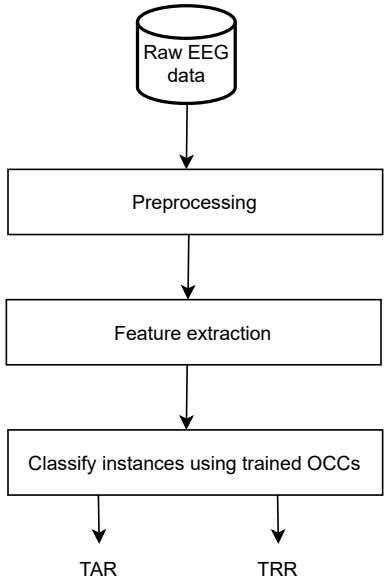


Figure 3.4: Flow chart of validation schedule.

3.7 Evaluation of GANs using GAN-train and GAN-test

In addition to evaluating the instances generated by the GAN with visual inspection, and their ability to increase TAR and TRR in the biometric system, they are evaluated using GAN-train and GAN-test which, described in section 2.4.3. These metrics are made for single GANs producing multiple classes of data. In the case of this thesis, several GANs are trained, generating one class(subject) each, i.e. instances from each subjects. In the sense of this evaluation measure, the single class-generating GANs are considered as one multi class generating GAN. As each GAN only is trained using instances from its respective subject, a low GAN-train can not be due to the GAN mixing between the classes. As the biometric system is a variety of a classifier it self, it is the classifier to be used, and it will be evaluated using TAR and TRR as described in [6].

3.8 Software and hardware used

All implementation in this thesis is done in Python. The data was processed using MNE and SciPy [24, 25]. The DCGAN was implemented in, and provided by Pytorch [26]. The plots are produced in Matplotlib [27]. The computations were run locally on a personal computer and on the IDUN cluster provided by NTNU [28].

4 Experiments and results

The results from the experiments conducted with the methods presented in chapter 3 are presented in this chapter.

The experiments were started with finding a suitable configuration for the biometric system for selected data set, to form a baseline for comparative purposes. A configuration for the GANs to generate artificial instances was then intended found, and tested using TAR and TRR from the biometric system, as well as GAN-train and GAN-test.

For each subject, the dataset was split into a train, validation and test set respectively of 400/40/40 instances.

4.1 Finding a baseline

It was decided to first find a suitable configuration for the biometric system with the chosen dataset and no data augmentation. Some subjects contained less EEG data than 8 minutes, these were discarded. Note that if nothing else is specified, the system is trained on all 400 instances in the training set per subject.

As described in section 3.1, LEMON dataset consist of 218 subjects and 61 channels, where 134 of these contained 8 minutes or more EEG resting state data. First, the performance using all channels and subjects were evaluated, best number of neighbours in the LOF classifiers was found to be 2.

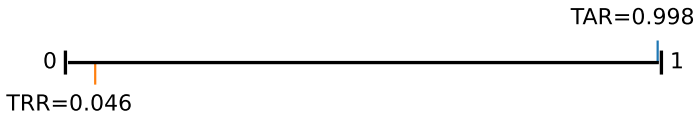


Figure 4.1: TAR and TRR of the biometric system using all channels and subjects, $n_{neighbours} = 2$. Validated using validation set.

From the performance in fig. 4.1, it's clear that the biometric system was not able to correctly perform subject identification. Most subjects are accepted by the models which leads to a high TAR and low TRR. As shown in [6], the performance of biometric systems based on similar methods are highly dependent on both hyper parameters, whats channels are used, and may also be variation in performance between the different subjects. It was decided to investigate if using a subset of channels could improve performance.

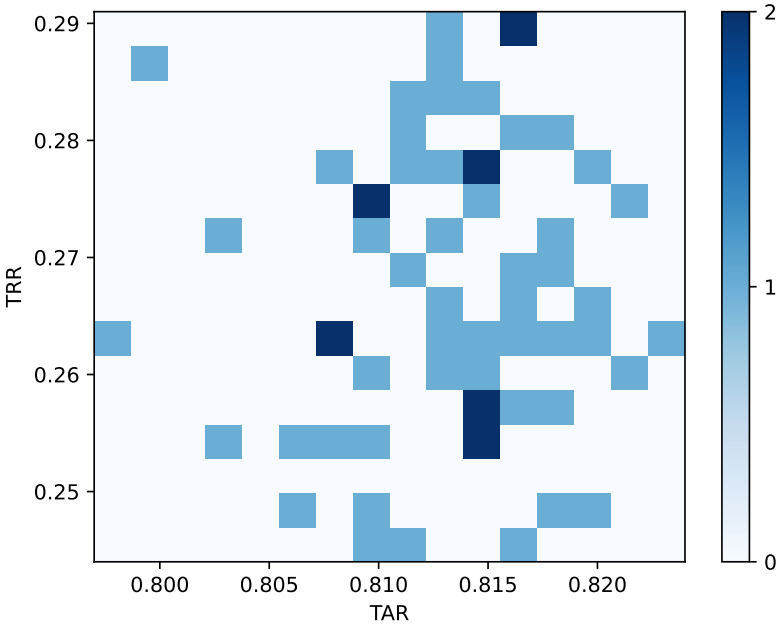


Figure 4.2: Historgram showing the TAR TRR of the biometric system performing subject identification, one channel at a time. Validated using validation set.

The performance of the biometric system was evaluated using one channel at a time for all channels, shown in fig. 4.2. It can be observed that for the TRR was higher for all evaluations with one channel, than with all channels(fig. 4.1). This was at the expense of a lower TAR. Through trial and error it was intended to find a combination of channels which could give better performance than the best configurations with one channel, but it was concluded that channel 13 alone gave the best performance, which can be seen in fig. 4.3. In this case $n_{neighbours} = 1$ gave best performance.

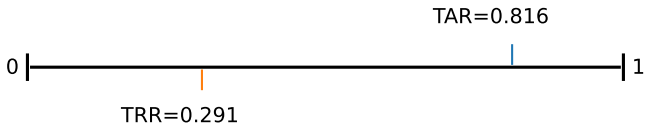


Figure 4.3: TAR and TRR of the biometric system using channel 13 and all subjects. $n_{neighbours} = 1$. Validated using validation set.

When evaluating the performance for each subject, it was found that there was a significant spread in performance for each model. During this evaluation the TAR for a specific subjected was calculated from its OCC models' ability to accept the instances from the subject it self. The TRR was calculated from the models ability to reject instances from other subjects.

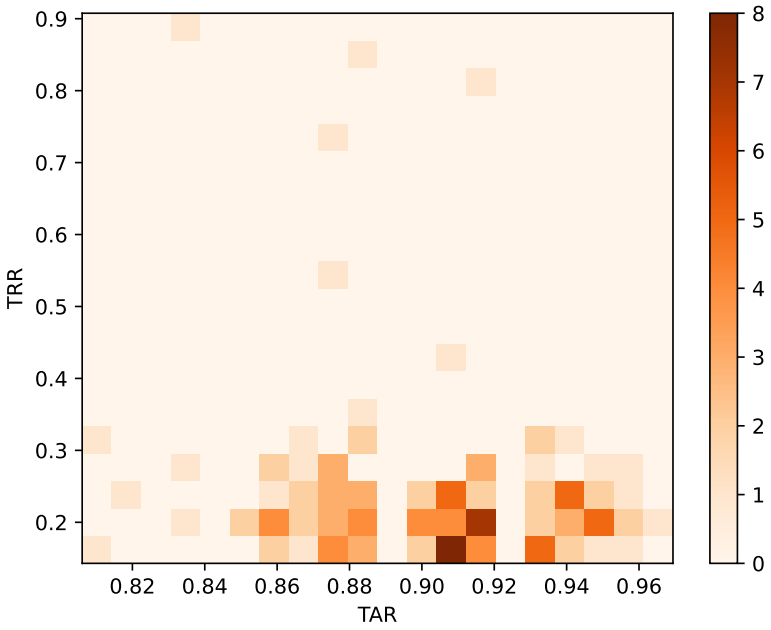


Figure 4.4: Histogram showing the TAR TRR of the biometric system performing subject identification using only channel 13, evaluated one subject at a time. $n_{neighbours} = 1$. Validated using validation set.

It can be observed that the spread in TRR across subjects is big. The models from most of the subjects had $TRR \leq 0.2$, hence they had not learned the features of the subjects - and accepted most instances regardless of whos' brainwaves they came from. However, some models were able to reject significantly more subjects, with a TRR as high as 0.9. It was at this stage intended to select more than 2 IMFs in the IMF selection stage described in section 3.3.2, with no improvement in accuracy.

All subjects with $TRR \geq 0.3$ were selected. The TRR limit was set to have a decent amount of subjects to continue with, but not including too many bad performing subjects. This subset consisted of 14 subjects in total. The age and gender of the selected subjects was investigated. No pattern in gender nor age was found when comparing the group of selected subjects with all subjects in the dataset.

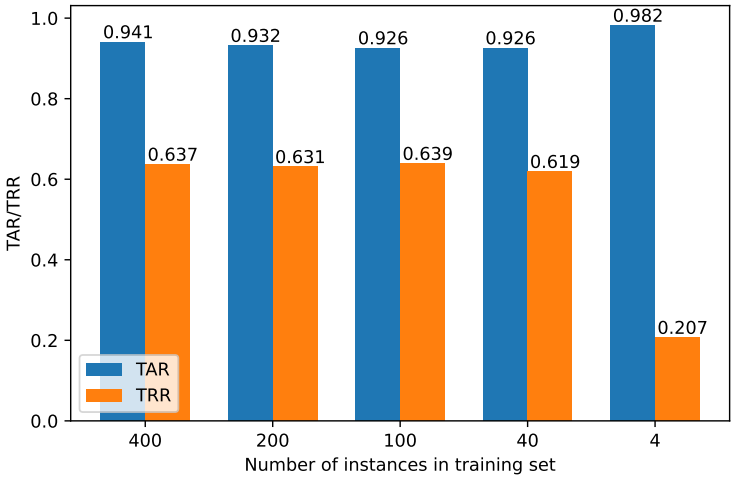


Figure 4.5: TAR and TRR of the biometric system using channel 13 and a subset of subjects, training models on different amounts of training instances. $n_{neighbours} = 5$. Validated using validation set.

Performance when training the selected subjects using different amounts of training instances can be seen in fig. 4.5. In this experiment the optimal value of $n_{neighbours}$ was found to be 5. It can be seen that for 400, 200, 100, 40 training instances the performance is similar, but the subject identification collapse and accept most instances when trained on 4 instances. Model performance does not benefit significantly from being trained on more than 40 training instances per subject. As this configuration give the best performance using the LEMON dataset, these results will be used as the baseline in further subject identification experiments where generated instances also will be included in the training set.

4.2 Inspection of real instances

The instances were inspected, as a preparation to evaluate generated instances by the GANs. As mentioned in section 2.4.3, visual inspection is a normal approach to evaluate GAN-generated images. Evaluating brain waves using visual inspection is not as straight forward, hence some insight into the look and properties of the real instances prove useful. Only channel 13 was inspected in this experiment, as its the only selected channel in section 4.1.

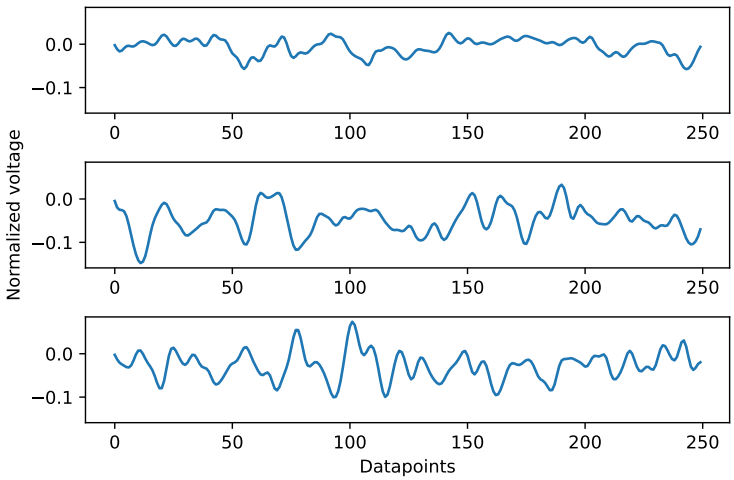


Figure 4.6: Channel 13 from three instances selected from three randomly selected subjects.

The normalization during preprocessing was done with respect to all the channels in the instance, which explain why the the mean of the time series corresponding to each channel not necessarily is 0.

Three different looking instances are shown in fig. 4.6. The instances originate from three random subjects in the sub-set of subjects found in section 4.1, and all are channel 13. It can be seen that the instances are centered around or close to 0. The oscillations in the instances are different, where especially the first instance stand out, with oscillations with less energy than the two other.

4.3 Generation of instances

In this experiment a suitable configuration is intended found for the GANs. They are evaluated through inspection of G and D loss, and inspection of the generated instances.

The instances used to train the GAN included all 61 channels. Only channel 13 is inspected and used in subject identification, due to the decision to use only this channel in section 4.1.

The instances generated by the GANs will be called generated instances, while the the instances originating from actual EEG measurements are called real instances. When the generated instances are included the training set the training set will be referred to as a augmented data set.

4.3.1 DCGAN

It was intended to use a DCGAN as similar to the original DCGAN as possibly. The layers were modified for producing correct output dimensions as described in section 3.5. A DCGAN was trained for each subject, and a random subject was chosen for further inspection. The DCGAN was tested with different batch sizes, and it was found from evaluation of G and D loss, that batch size = 128 gave apparently stable training.

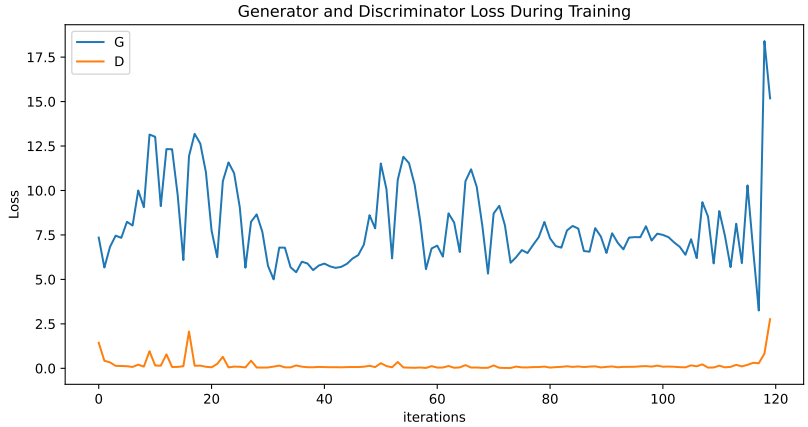


Figure 4.7: G and D loss over iterations for randomly selected subject. Batch size = 128, 30 epochs

As seen in fig. 4.7 there is little development in the loss of the discriminator which is stable close to zero. The G loss oscillates.

10 generated instances are plotted in fig. 4.8. During inspection of the generated instances, it is found that the generated instances look highly similar, a reason to believe that G

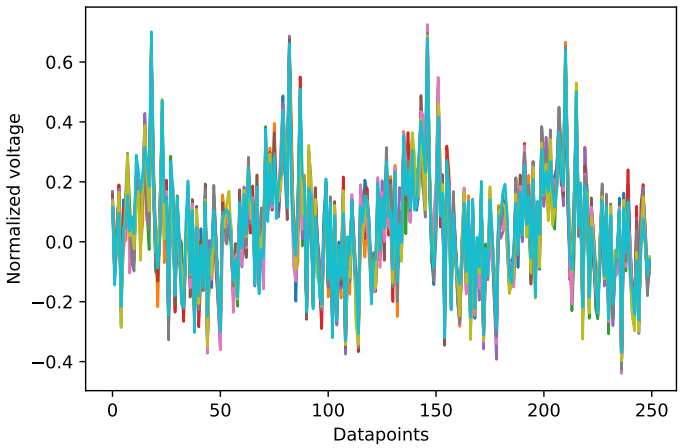


Figure 4.8: Channel 13 from 10 generated instances by GAN trained using original parameters.

is subject to mode collapse, a problem described in section 2.4.3. It can be seen that many instances have the same shape with the same slow oscillation, and peaks at the same datapoint. It was confirmed through inspection that all subjects suffered from the same issue, and it was concluded to discard the GAN configuration without further analysis. It was decided to tune hyper parameters with the goal of stabilizing training.

4.3.2 Tuning DCGAN hyperparameters

The hyperparameters of the DCGAN was tuned for producing a stable DCGAN without mode collapse. The same problem as above happened to many configurations.

The training of a tuned and selected GANs loss is shown in fig. 4.9.

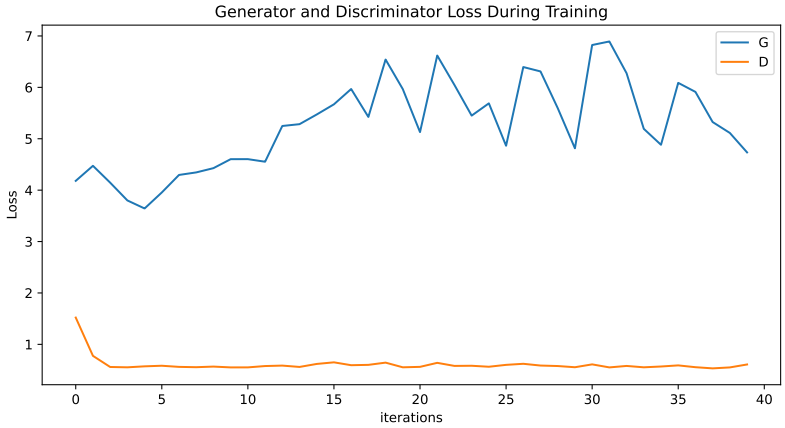


Figure 4.9: G and D loss og tuned DCGAN over iterations for randomly selected subject.

This GAN had the following configuration:

- Capacity of both neworks was reduced, changing parameters ngf and ndf from 64 to 32.
- One sided label smoothing was applied, reducing the label for real instances from 1 to 0.8.
- TTUR was applied, such that G had a learning rate of 0.0001 and D a learning rate of 0.0003
- The batch size was 128 and the GAN was trained for 10 epochs.

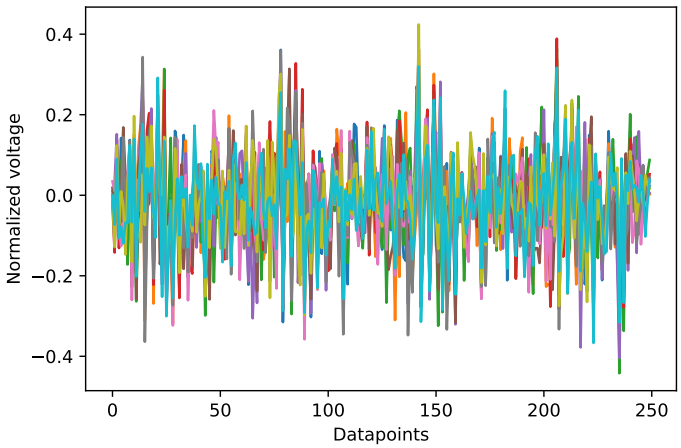


Figure 4.10: Channel 13 from 10 generated instances by tuned GAN.

The generated instances are plotted in fig. 4.10. Through visual inspection, it can be seen that these still is similarities between the instances, as in fig. 4.8, but slightly less. It can be seen that many instances have peaks at the same data point, and that they more or less oscillate in the same phase. From visual inspection, it was concluded that this was the configuration which gave most diverse instances. As of this, it was concluded to test this configuration with subject identification.

4.4 Subject identification with augmented data set

Using the GAN configuration from section 4.3.2, artificial instances were generated, and used to augment the training set.

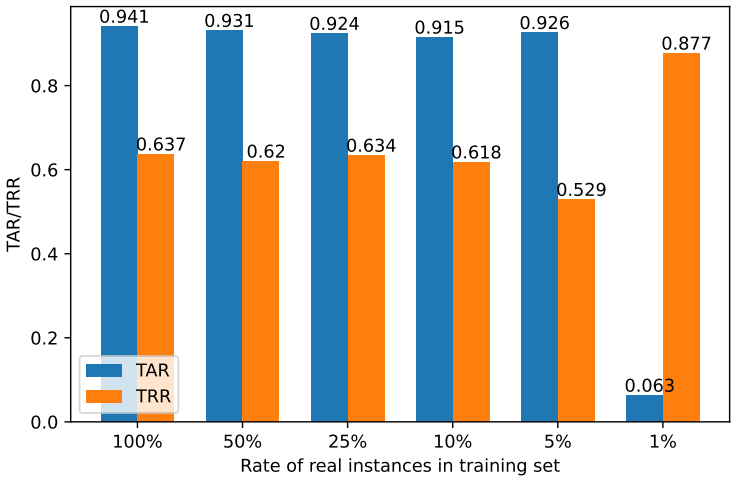


Figure 4.11: TAR and TRR of the biometric system performing subject identification using only channel 13 and 400 training instances. Validated using test set.

The biometric system was trained with different augmented training sets and evaluated in fig. 4.11. All evaluations contained 400 training instances, but the relationship between real and generated instances varied, as shown on x-axis. The performance of the system using the same amount of real training instances(but without generated instances) can be seen in fig. 4.5. It can be observed TAR and TRR is relatively identical and is not significantly affected for 100% 50% 25% and 10% real instances. For 5% real instances the models is less able to reject intruders, and for 1% for models collapse and reject almost all instances, resulting in a low TAR and high TRR. As the system performance is declining as the number of real instances is reduced and number of generated instances is increased, it does not seem like the the generated instances are sufficiently similar to the subjects' instances. This kind of evaluation was performed for many GAN configurations, with similar results.

The system was evaluated using GAN-train and GAN-test, described in section 2.4.3.

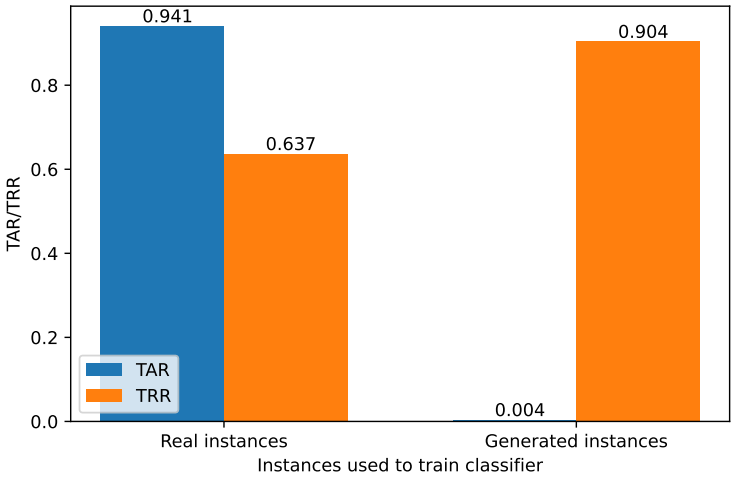


Figure 4.12: GANs tested using GAN-train method.

In fig. 4.12 the system is trained exclusively using real instances and exclusively generated instances, known as the GAN-train. Both are evaluated using the evaluation set, which of course consist of real instances. It can be observed that trained on generated instances the models are significantly less able to recognize the instances from its subject than trained on real instances.

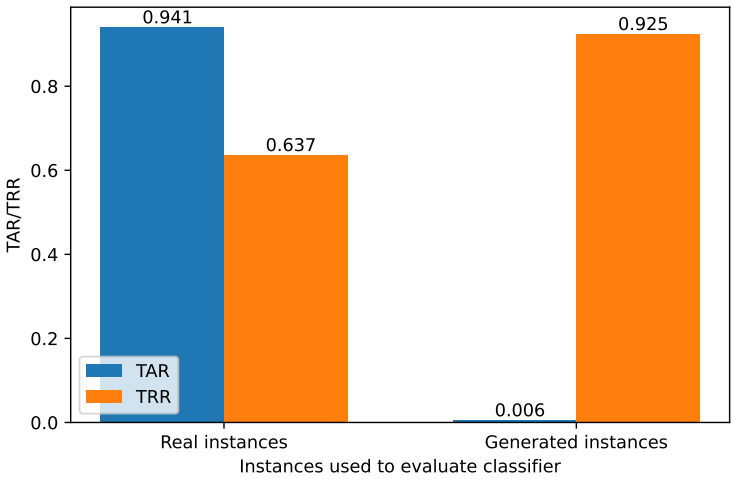


Figure 4.13: GANs tested using GAN-test method.

In fig. 4.13 the system is trained on real instances, but evaluated using exclusively instances and exclusively generated instances, known as the GAN-test. It can be observed

that the system evaluated using generated instance, is less able to recognise what instance belong to what subject, compared with when evaluated using real instances. s most instances are rejected.

5 Discussion

5.1 Discussion

It was found that the biometric system, using methods described in chapter 2 performed better when using a single EEG-channel and the subset of subjects found in section 4.1. It was intended found a combination of EEG-channels which could yield better performance, but such a combination was not found, which was unexpected. This show that the LOF OCCs could not exploit the extra subject dependent patterns present in multiple channels(if these patterns were present) in this case. Only 14 out of the subjects in the data set was able to achieve $TAR > 0.3$, which also was was a lower performance than expected. Most OCCs accepted most instances(despite being from other subjects), resulting in a low TRR. It was intended to find configurations which gave the system a higher TRR, but this was not successful. It was concluded to select the subjects with $TRR > 0.3$, and continue experiments using them.

In a real world product, when subjects are added to a biometric system, it is not feasible to perform a subject selection. Despite this, it was decided to select the best performing subjects as it was hard to find a dataset fulfilling the criteria presented in section 3.1, especially considering length of EEG measurements. The length of the EEG measurements per subject was considered essential as GANs require a lot of training data.

It's hard to pinpoint why the subject identification did not work for so many of the subjects. The resting state paradigm is confirmed to give higher performance with similar methods in other studies [4, 6]. There may be some reason related to the equipment used to acquire the EEG measurements. There may have been noise present in the data outside of the $1 - 40Hz$ band, which did not affect all channels at the same time hence it would not have been removed in the filtering stage of pre-processing. There could also be other artifacts present in the signal, which could be removed with for example Independent Component Analysis [29]. The system was tested through selecting various numbers of IMFs in IMF selection stage, but this did not yield better performance. As of this, it was concluded that it is not likely that the necessary information was lost in IMFs not selected.

It was investigated what quantity of training data was needed for training the biometric system. As each instance correspond to 1 second of EEG-recordings, the number of training instances correspond to number of seconds for collecting data from a subject. It can

be seen in fig. 4.5 that the system is dependent on sufficient training data, and as little as 4 seconds of data collection is not enough. However, the system does not benefit significantly from using a lot more training instances. For example, the improvement in TAR and TRR of training system with 100 seconds of EEG-recordings over 40 seconds, is not as big as 40 seconds over 4 seconds.

Given that the subject dependent information actually is present in the instances, system performance could have been improved through using a more complex classifier. Utilizing multi class classifiers, such as an ANN, more complex patterns in the data could have been exploited to distinguish between the subjects. However, such a classifier would introduce challenges. What quantity of training data is necessary to achieve acceptable system performance? Would the classifier need to re-train using instances from all subjects each time a subject is added, and how much time and compute would this require? The use of LOF OCCs have big advantages related to these questions. The LOF OCCs don't require much training data, and each model is only trained on the data from one model. This makes adding and removing subjects from the biometric system trivial, as it's an operation only involving the LOF OCC corresponding to the subject.

It was challenging to find a configuration for the DCGAN which did not suffer from mode collapse. This lead to many generated instances being highly similar. The generators convolutional layers' kernel size and stride was changed, such that the output was of the correct dimensions. The kernel size in the direction along the channel time-series was increased for some of the layers, and the stride was reduced along the same axis. The intention behind this was that there is dependency from datasample to datasample in the same channel (as the datasamples represent the voltage potential at one specific location), hence filters in the convolutional layers filtering "along" the channel time-series make sense. As the convolutional operations are local, it also implies that there is more dependency between channels that are close (in index), as the first row in the instances is channel with index 0 and so on until channel 61. The "local" effect in the first layers is spread out deeper in the generator. In the real EEG data there may be little dependency between channels which are close in index, as the dependency between channels is more related to localization on the head than nearby indices. This may lead to lower quality of the generated EEG data. The hyper parameters in GANs are sensible such that these changes may have affected the stability of GAN training as well. Using pooling layers in convolutional ANNs is a known technique [30], such that adding an average pooling layer to the discriminator should not introduce instability to training.

Mode collapse of the GANs was identified using visual inspection of the generated instances. As channel 13 was selected as the only channel to use in the biometric system, only this channel was inspected in the generated instances. Using only one channel significantly reduced dimensionality of the instances, this made visual inspection easier. Several instances was plotted on top of each other, which enabled noticing patterns in the data. However, as visual inspection is highly subjective, and comparing time series visually is not trivial, this method of considering mode collapse is sub optimal. Anyway, it is possible to see the that the generated instances suffer from mode collapse, and are more diverse

when visually comparing with the real instances in section 4.2. It was decided not to evaluate the instances generated by GANs suffering from mode collapse using the biometric system (with TAR and TRR), as they would have generated a strictly limited amount of new instances to perform data augmentation. For the data augmentation to be successful a variety of different instances would need to be added to the training set of the biometric system.

The GAN hyper parameters were tuned, and several tricks to stabilize GAN training was implemented. As the parameters ngf and ndf were halved, the depth of the convolutional layers were halved, reducing the capacity of the networks significantly. The results of these changes was that the generated instances, evaluated using visual inspection, did not suffer from mode collapse. It was observed that there was more spread in the "behaviour" of the instances, even though there still was similarities such as peaks a the same datapoint for many of the instances. The generated instances were less diverse than the real instances, which can be seen if visually comparing with section 4.2. The generated instances was used to augment the training data of the biometric system, with different rates between real an generated instances in fig. 4.11. As more generated instances replaced the real instances, system performance was reduced. This show that the generated instances does not contain the same properties as the real instances, as the models to not learn their subjects features to the same extent. If the generated instances were of high enough quality, the system performance should not decrease with more generated instances in training set. In the case of 1% of training set consisting of real instances (which mean the training set consist of 4 real instances and 396 generated instances), the TRR is increased. This is because the models reject almost all instances in evaluation, hence the models have not learned the features of each subject.

Comparing system performance when no artificial instances are included in training set (fig. 4.5) with the system performance with augmented training set, it can be concluded that, for a given amount of real training data, it is better to not add generated instances. For example, training the system with 40 real instances (fig. 4.5) give a TAR/TRR of $0.926/0.619$. However, training the system with 40 real instances and 360 generated instances which was done in fig. 4.11 (10% of 400 is 40) result in a TAR/TRR of $0.915/0.618$. This demonstrate that the generated instances actually reduce system performance, and from a TAR/TRR point of view it would be better not to include generated instances in the training set of the LOF OCCs.

Note that the GANs was trained using a training set of 400 real instances, in all attempts in fig. 4.11. This means that if the training set to train the biometric system were based on only n seconds of EEG recordings per subject, and the rest of the instances are generated(400-n), the GANs are still trained on 400 instances/400 seconds of EEG recordings. This was a simplification done to not have to train a GAN for all runs in fig. 4.11. It would be interesting to see how the behaviour of the generated instances are changed as the amount of instances to train the GANs change as well.

It was intended to evaluate the generated instances using the GAN-train and GAN-test metrics. These metrics were made for multi class generating GANs, and to be evaluated

using a multi class classifier. In the case of this thesis, several GANs generate a class each (i.e. instances from a specific subject), and the testing is performed with the biometric system, which is not a traditional multi class classifier, but with TAR and TRR. From GAN-train and GAN-test it was concluded that the generated instances are either not from distribution p_{data} of each respective subject, or represent a strictly limited part of p_{data} of each respective subject.

A reason for the GANs failing to estimate p_{data} for each subject, may be the lack of training data. The quantity of training data for training neural networks, especially with deep architectures, is known to be large. In [31] techniques are proposed to train GANs using datasets of limited size, and the author claim to be able to generate high quality images "using only a few thousand training images". This thesis is not about image generation, but considering that state of the art impressive low size dataset for training GANs is 2000, using 400 instances to train a GAN seem too low. This show that the chosen approach, which consist training a GAN with 400 instances for each subject may have been too optimistic, there is not enough data to optimize over the large space of parameters.

Using the suggested methods in a biometric system, adding a subject require sequential data collection with an EEG head-cap, pre-processing, training of a GAN, generation of instances, feature extraction for all instances and in the end training of a LOF OCC. It is not performed any experiments to measure the time these steps take. This could have been found, but an educated guess based on experience after developing the system, say that the training of the GAN is what consume most time, and is highly dependent on the number of epochs. Using other data augmentation techniques which does not included training such complex models with many parameters could be faster and hence reduce the time for computations when adding a new subject.

6 Conclusion

6.1 Conclusion

The main objective of this work was to generate EEG data similar enough to the EEG data recorded from a specific subject, such that data augmentation on a biometric system could be performed. This would reduce calibration time, such that the system could be calibrated with less EEG data, and the user could spend less time on EEG data acquisition. A research question was formed in chapter 1, and the answer to the question will now be concluded.

Can a GAN-based data augmentation technique be used to reduce the calibration time for adding new subjects in EEG-based biometric systems?

In the experiments in this report, the training set of a EEG based biometric system was augmented using instances generated by GANs. It was found that that when an increasing part of the instances in the training set was generated instances, the system performance decreased. As of this, it is concluded that the GAN configuration developed in this thesis, can not be used for data augmentation in combination with the LEMON dataset and the developed biometric system.

GANs have in previous studies shown the ability to generate high quality EEG data for data augmentation in the field of brain computer interfaces. However, it is unclear if the GANs are able to generate data with the properties separating the subject - which is necessary if they are to be used for data augmentation in biometric systems. No studies trying to do this is found, hence this thesis is a first attempt to this exact task.

The reasons for the GANs failing the data augmentation was discussed in chapter 5. The tuned parameters in the GAN convolutional layers which may affected training stability and the amount of training data for the GAN are believed to be the two most important reasons for the data augmentation to fail.

6.2 Future work

The configuration for the GANs should be further investigated. Other approaches for generating data of correct dimensions should be found, as it is known that the hyper parameters in GANs are sensible to tuning, which may lead to unstable training. It may be experimented with other types of GANs, such as Wasserstein GAN [32], which is claimed to give more stable training, and get rid of problems like mode collapse, which indeed was a challenge in the work of this thesis.

Several approaches could be chosen to deal with the lack of training data for the GAN. Some data augmentation technique could be used to increase size of the training set, but to not over complicate the whole system, it could be advantageous to test such a data augmentation approach directly on the training set to the biometric system instead of on the training set for the GAN. A conditional GAN [13] could be used with subject-conditioning, such that one GAN is used for all subjects. This would increase size of training set for the GAN, as one model would be train on the instances of all subjects combined. This could improve the data augmentation, as generation of high quality instances could be enabled. However, real time related issues could be introduced. Would the GAN need to be re-trained from scratch each time a new subject is added? This could drastically increase time for adding a subject, especially if the biometric system, including the GAN training routine was to be run on a embedded system with little computational resources. Transfer learning is another approach which could be used. This would mean using a GAN, pre-trained to generate "general" instances as a foundation. It would then be trained on the instances from a specific subject to learn to generate instances from that specific subject. This could require less training instances. However, such a pre-trained GAN need to be found, and the resources available for generating EEG instances is limited.

Less complex data augmentation techniques are published in the field of brain computer interfaces. Some of these are as simple as adding noise to the EEG signal [33], or create instances using a sliding window such that more instances are created [34]. Such methods for data augmentation could contribute to reduce calibration time for adding new subjects, and could be used in combination with a GAN approach, both to augment the training set of the GAN and the biometric system.

There are many parameters affecting performance of the system as a whole. This include parameters such as what channels are used, how many IMFs are selected, the hyper parameters in the GAN, the hyper parameters in classification process and more. The values for these are mainly found through trial and error, which has its limitations as it takes a lot of time. It can also be hard for the person analyzing results to see complex patterns between parameters and results. As of this, a more structured approach could be applied to find more suitable parameters in all stages of the system. This could for example be solved as a optimization problem where the system performance of the biometric system is optimized, as done in [4] using the genetic algorithm NSGA-III to find the optimal set of channels. It could also be done as a traditional hyper parameter search, for example using grid search [35]. However, some of the parameters may affect the way the instances are processed, complicating the implementation of such an algorithm.

Bibliography

[1] Fadzilah Ahmad and Dzulkifli Mohamad. A review on fingerprint classification techniques. In *2009 International Conference on Computer Technology and Development*, volume 2, pages 411–415. IEEE, 2009.

[2] Paramjit Kaur, Kewal Krishan, Suresh K Sharma, and Tanuj Kanchan. Facial-recognition algorithms: A literature review. *Medicine, Science and the Law*, 60(2):131–139, 2020.

[3] Hui-Ling Chan, Po-Chih Kuo, Chia-Yi Cheng, and Yong-Sheng Chen. Challenges and future perspectives on electroencephalogram-based biometrics in person recognition. *Frontiers in neuroinformatics*, 12:66, 2018.

[4] Luis Alfredo Moctezuma and Marta Molinas. Towards a minimal eeg channel array for a biometric system using resting-state and a genetic algorithm for channel selection. *Scientific RepoRtS*, 10(1):1–14, 2020.

[5] Fatemeh Fahimi, Strahinja Dosen, Kai Keng Ang, Natalie Mrachacz-Kersting, and Cuntai Guan. Generative adversarial networks-based data augmentation for brain-computer interface. *IEEE transactions on neural networks and learning systems*, 2020.

[6] Nikolai Molvik. Data augmentation for subject identfication in a eeg-based biometric system. *Specialization project June 2021, NTNU*, https://drive.google.com/file/d/1qksHD4TpKQr8NDc2emnL0amJDyQLAXCS/view?usp=sharing.

[7] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. Ch. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation*, 2000 (June 13).

[8] Pierre Lison. An introduction to machine learning. *Language Technology Group (LTG)*, 1(35):1–35, 2015.

[9] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000.

[10] F Rosenblatt. A probabilistic model for information storage and organization in the brain. *Artificial Intelligence: Critical Concepts*, 2(6):398, 2000.

[11] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2014.

[13] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[14] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[15] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29:2234–2242, 2016.

[16] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[17] Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari. How good is my gan? In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 213–229, 2018.

[18] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[19] Ivan W Selesnick and C Sidney Burrus. Generalized digital butterworth filter design. *IEEE Transactions on signal processing*, 46(6):1688–1694, 1998.

[20] Oriano Mecarelli. Electrode placement systems and montages. In *Clinical Electroencephalography*, pages 35–52. Springer, 2019.

[21] Luc Lévesque. Nyquist sampling theorem: understanding the illusion of a spinning wheel captured with a video camera. *Physics Education*, 49(6):697, 2014.

[22] Gabriel Rilling, Patrick Flandrin, Paulo Goncalves, et al. On empirical mode decomposition and its algorithms. In *IEEE-EURASIP workshop on nonlinear signal and image processing*, volume 3, pages 8–11. Citeseer, 2003.

[23] Luis Alfredo Moctezuma and Marta Molinas. Eeg-based subjects identification based on biometrics of imagined speech using emd. In *International Conference on Brain Informatics*, pages 458–467. Springer, 2018.

[24] Alexandre Gramfort, Martin Luessi, Eric Larson, Denis A. Engemann, Daniel Strohmeier, Christian Brodbeck, Roman Goj, Mainak Jas, Teon Brooks, Lauri Parkkonen, and Matti S. Hämäläinen. MEG and EEG data analysis with MNE-Python. *Frontiers in Neuroscience*, 7(267):1–13, 2013.

[25] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[26] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[27] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[28] Magnus Själander, Magnus Jahre, Gunnar Tufte, and Nico Reissmann. EPIC: An energy-efficient, high-performance GPGPU computing research infrastructure, 2019.

[29] Izabela Rejer and Pawel Górski. Benefits of ica in the case of a few channel eeg. In *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 7434–7437. IEEE, 2015.

[30] Sukarna Barua, Sarah Monazam Erfani, and James Bailey. Fcc-gan: A fully connected and convolutional net architecture for gans. *arXiv preprint arXiv:1905.02417*, 2019.

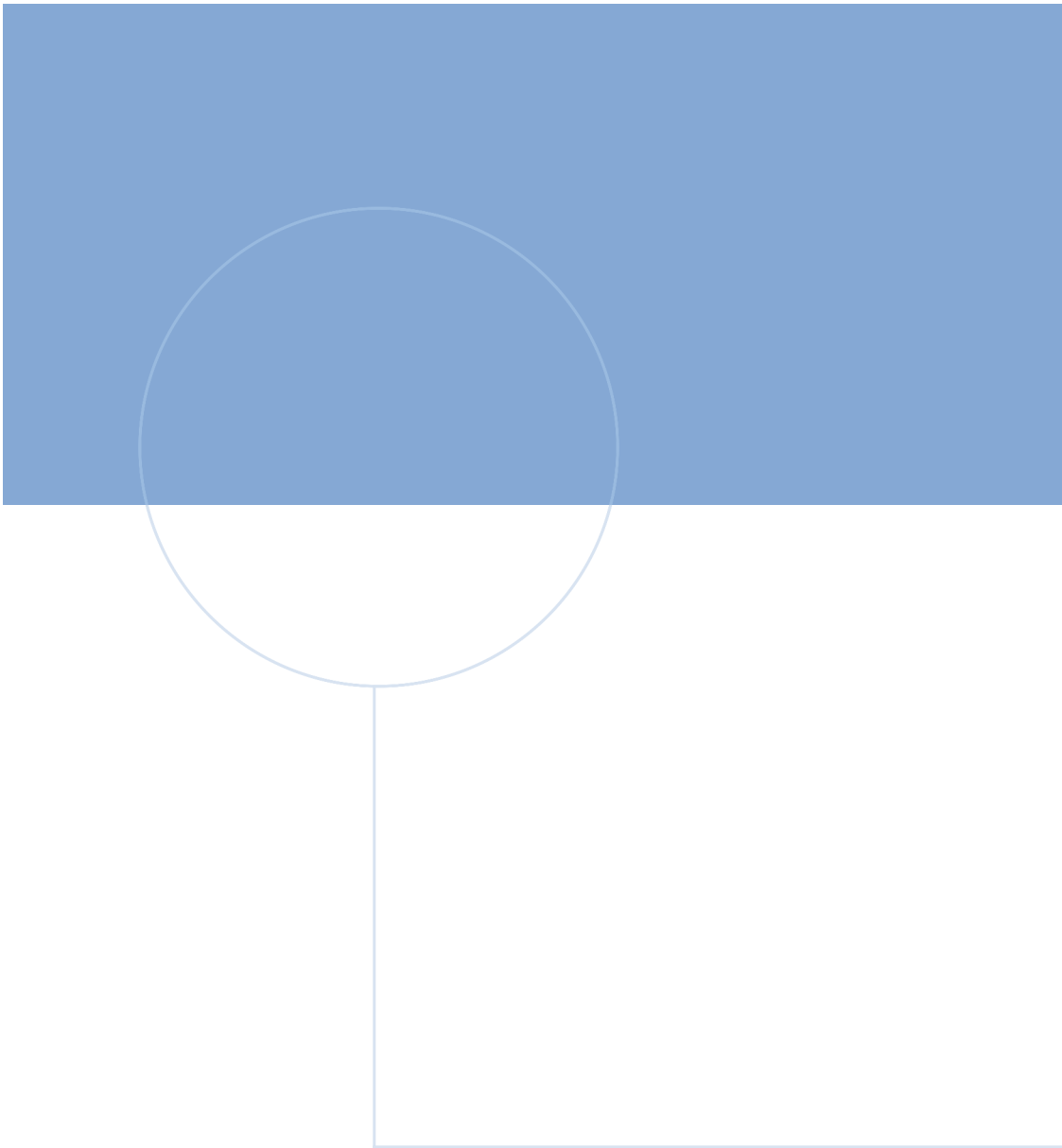
[31] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *arXiv preprint arXiv:2006.06676*, 2020.

[32] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR, 06–11 Aug 2017.

[33] Fang Wang, Sheng-hua Zhong, Jianfeng Peng, Jianmin Jiang, and Yan Liu. Data augmentation for eeg-based emotion recognition with deep convolutional neural networks. In *International Conference on Multimedia Modeling*, pages 82–93. Springer, 2018.

[34] Elnaz Lashgari, Dehua Liang, and Uri Maoz. Data augmentation for deep-learning-based electroencephalography. *Journal of Neuroscience Methods*, page 108885, 2020.

[35] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24, 2011.



NTNU

Norwegian University of
Science and Technology