Application of GANs to Generate Synthetic DNA Methylation Data to Overcome Class Imbalance

Bachelor's Thesis at the Berlin Institute of Health
to obtain the academic degree
Bachelor of Science

submitted by Anthony Hamilton Fernando

Student ID: 5199958

Freie Universität Berlin
Faculty of Mathematics and Computer Science
Department of Bioinformatics

1st Examiner: *Dr. Naveed Ishaque*
2nd Examiner: *Prof. Dr. Tim Landgraf*
Co-supervisor: *M. Sc. Sebastian Tiesmeyer*

Berlin, May 21, 2024

Acknowledgement

This thesis would not have been possible without the support of many people around me. First of all, I would like to thank Dr. Naveed Ishaque, who was a brilliant supervisor, because he gave me the right ideas to get the best out of this thesis. I am very grateful for his support in general but especially for his rapid responses and his open discussions throughout the whole process. I would also like to express my gratitude to my co-supervisor Sebastian Tiesmeyer, for his patient encouragement and guidance throughout the research phase. I am very grateful for his expertise in deep learning and I would never have come this far without his knowledge. Furthermore, I would like to thank Christiane Weber and Shashwat Sahay for their ideas and discussions about my topic. Additionally, I would like to thank Thore Bürgel and Julius Upmeier zu Belzen for the technical assistance. Moreover, I would like to thank Prof. Dr. Tim Landgraf for his availability of being my second reviewer and giving a clear feedback from the presentation. Last but not least, my thank goes to my parents and my friends especially Janis, Pantea and Shanta for supporting and motivating me the whole time.

Abstract

A major aim of precision brain cancer therapy is to apply the most suitable drugs for any given brain tumour. With the aim to treat the patients with the most appropriate drugs, it is very important to diagnose the precise type of a brain tumour. Nevertheless, not all tumours that are found in the brain are brain tumours. Many of them are often cancer metastasis originating from other tissues. Researchers have shown that DNA methylation, a stable epigenetic mark, is a very good bio-indicator for classifying both cell types and tumour types. Combining this with machine learning has allowed researchers to exploit Random Forest classifiers to successfully classify more than 90 types of different brain tumours based on DNA methylation profiles. However, the classification rate was not high enough for some classes (tumour types) in the dataset because of a class imbalance problem: out of 2801 training samples and 91 brain tumour classes, the largest class had around 143 samples, whereas the smallest class had only 8 samples. This has an effect on the predictive accuracy of the model, where this class imbalance is leading the models to predict against under-represented classes. The aim of this thesis is to overcome this class imbalance problem by implementing and testing an Auxiliary Classifier Generative Adversarial Network (AC-GAN), a class-aware GAN, that tries to generate realistic synthetic methylation data from lowly represented brain tumour types. This thesis also aims to investigate various AC-GAN architectures on DNA methylation data. Thus, different model architectures were implemented as well as evaluated using various criteria including generic loss functions and also domain-specific evaluation metrics such as DNA methylation profiles and nearest neighbour statistics. The results of this thesis have shown that the input data affects the outcome of the model. The selection of fewer but more important features can tremendously improve the model performance. Another finding suggests using less complex model architectures, such as the type of layers and number of nodes in each layer that play an essential role with regards to the computational time, memory requirements and model's performance. It was also recognized that it is not necessary to train the model for a large number of iterations. Based on the promising results, this thesis helps in understanding AC-GAN models and the influence of hyperparameters with the intention of solving the class imbalance problem as well as generating realistic synthetic DNA methylation data.

Zusammenfassung

Ein Hauptziel der Präzisions-Hirntumortherapie besteht darin, die am besten geeigneten Medikamente für ein bestimmten Hirntumor zu finden. Um die Patienten mit den optimalen Medikamenten behandeln zu können, ist es sehr wichtig die spezifische Art eines Hirntumors zu diagnostizieren. Allerdings sind nicht alle Tumore, die im Gehirn gefunden werden, Hirntumore. Bei vielen von ihnen handelt es sich um Krebsmetastasen, die aus anderen Geweben stammen. Forscher haben gezeigt, dass die DNA-Methylierung, eine stabile epigenetische Markierung, einen sehr guten Bioindikator für die Klassifizierung von Zelltypen sowie von Tumortypen darstellt. Durch die Kombination mit maschinellem Lernen konnten Forscher Random-Forest-Klassifikatoren nutzen, um mehr als 90 verschiedene Hirntumorarten auf der Grundlage von DNA-Methylierungsprofilen erfolgreich zu klassifizieren. Allerdings war die Klassifizierungsrate für einige Klassen (Tumortypen) im Datensatz aufgrund eines Class-Imbalance-Problems nicht hoch genug: Von 2801 Trainingsproben und 91 Hirntumorklassen umfasste die größte Klasse etwa 143 Proben, während die kleinste Klasse nur 8 Proben enthielt. Dies wirkte sich auf die Vorhersagegenauigkeit des Modells aus, da diese Class-Imbalance dazu führt, dass die Modelle Vorhersagen für unterrepräsentierte Klassen nicht berücksichtigen. Ziel dieser Arbeit ist es, dieses Class-Imbalance-Problem zu überwinden, indem ein Auxiliary Classifier Generative Adversarial Network (AC-GAN), ein klassenbasiertes GAN, implementiert und getestet wird. Das AC-GAN versucht, realistische synthetische Methylierungsdaten von unterrepräsentierten Hirntumortypen zu generieren. Des Weiteren wurde angestrebt, verschiedene AC-GAN-Architekturen auf DNA-Methylierungsdaten zu untersuchen. Dazu wurden verschiedene Modellarchitekturen implementiert und anhand verschiedener Kriterien evaluiert. Darunter fallen generische Verlustfunktionen sowie domänenspezifische Evaluierungsmetriken wie DNA-Methylierungsprofile und Nearest Neighbour-Statistiken. Die Ergebnisse dieser Arbeit haben gezeigt, dass die Eingabedaten das Ergebnis des Modells beeinflussen. Die Auswahl weniger, aber wichtiger Merkmale, kann die Modellleistung enorm verbessern. Eine weitere Erkenntnis ist die Verwendung weniger komplexer Modellarchitekturen, wie zum Beispiel die Art der Schichten und die Anzahl der Knoten in jeder Schicht, die im Hinblick auf die Rechenzeit, den Speicherbedarf und die Leistung des Modells eine wesentliche Rolle spielen. Es wurde auch erkannt, dass es nicht notwendig ist, das Modell für eine große Anzahl von Iterationen zu trainieren. Auf der Grundlage der vielversprechenden Ergebnisse trägt diese Arbeit dazu bei, AC-GAN-Modelle und den Einfluss von Hyperparametern zu verstehen, mit dem Ziel, das Problem des Class-Imbalance zu lösen und realistische synthetische DNA-Methylierungsdaten zu erzeugen.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation and Problem

Out of all diseases in the world, cancer is one of the most deadly and feared disease in the 21^{st} century [7] [8]. This is particularly true for brain tumours because the brain is the most complex organ in the human body, which consists billions of cells [9]. A brain tumour is highly likely to occur when uncontrollable cell division takes place. This results in having abnormal types of cells around and inside the brain. These abnormal cells are able to manipulate the functionality of the brain activity and destroy healthy cells [9]. It is therefore important to identify accurately the precise type of the brain tumour in the clinic with the aim to administer the most appropriate drug for the patients [10] [5] [11]. However, diagnostics and classifications of brain tumours remain tricky, in part due to the brain being a frequent site for metastasis that are originating from other tissues, e.g. lung or breast [11]. Another cause of miss-classification is that in general brain tumours are very heterogeneous entities [12].

In recent years, DNA methylation has been shown to be a very good biomarker for classifying cell types or tumour types [13] [14]. This could be seen by the work from Capper et al. [5], where they used DNA methylation profiles in combination with a comprehensive machine learning method, called Random Forest classifier, to classify certain brain tumour types. They demonstrated that DNA methylation can accurately distinguish between different brain tumour types with an accuracy of 0.9 [5].

However, there is a training bias due to the class-imbalance problem in the training dataset. Some of the tumour types are significantly overrepresented. For example, out of 2801 training samples and 91 tumour classes, the largest class has around 143 samples, whereas the smallest class has only 8 samples. This has a huge effect on the predictive accuracy of the model, where the lowly represented classes are less considered in the classification process in comparison to the highly represented classes.

This problem can be overcome by generating synthetic data in order to complement the lowly represented tumour classes through the application of Generative Adversarial Networks (GAN). This has been successfully done for single-cell RNA sequencing data by Marouf et al. [15]. The newly generated data can be obtained and used as additional samples in the training data to retrain the reference Random Forest classifier. This thesis expands the work of Marouf et al. [15] by generating new samples for the lowly represented classes by using an Auxiliary Classifier GAN (AC-GAN).

1.2 Objective

The aim of this thesis is to implement different AC-GAN models and to evaluate them on the basis of how good they generate lowly represented tumour classes. Thereby, two questions are to be answered in this process: 1) How many features does a given AC-GAN architecture need, and 2) what type of architecture including layers and nodes does the AC-GAN need to generate lowly represented DNA methylation cancer types.

1.3 Outline

The following chapter introduces the necessary background for this thesis, which is divided into a biological and a computational part. In the biological part, DNA methylation will be introduced in detail, followed by the importance of tumour diagnostics and classification. In the computational part, the class imbalance problem will be defined. Furthermore, GAN and the AC-GAN will be explained, followed by the review and summary of the state of the art. In chapter 3, a brief description of the dataset will be given. Chapter 4 and 6 are the main parts of this thesis. In chapter 4 , the methods and the workflow that have been applied in this thesis will be explained. This include the model architecture, the strategy to overcome overfitting and mode collapse, the evaluation metric to evaluate the performance of the Generator and Discriminator and the hyperparameter optimization. Chapter 5 introduces the applied programming language and the corresponding libraries that have been used in this thesis. In chapter 6, the results will be presented, while chapter 7 will discuss and summarise the findings.

2 Background Information

This chapter introduces the background information needed for understanding the main problem that this thesis is trying to solve. This chapter is divided into a biological and a computational part, while the last section describes a work where GANs have been used for generating biological data.

2.1 Biological Background

This section describes the biological background focusing on DNA methylation as well as tumour diagnostics and classification.

2.1.1 Epigenetic Modification/DNA Methylation

Epigenetic modifications are defined as heritable alterations that do not change the nucleotide sequence in comparison to genetic mutations [16]. In other words, epigenetic information operates at a level higher than genetic information [17]. These modifications are heritable, reversible and have an effect on the transcriptional activity of genes by controlling access to regulatory elements in the genome [18] [19].

One of the best known epigenetic modifications is DNA methylation. It is defined by adding a methyl group at the 5' carbon of the cytosine base and forms into 5-methylcytosine (m5C) [20]. Figure 1 illustrates this modification. The methylation in mammalian DNA is found, when at all, at cytosine–guanosine dinucleotides (CpGs) [20], which are depleted in the mammalian genome [21]. Most of the CpGs are frequently present in high density in regions called CpG islands (CGIs), which are GC-rich regions [20]. CGIs are motifs that overlap promoters, gene bodies and transposable elements [21].

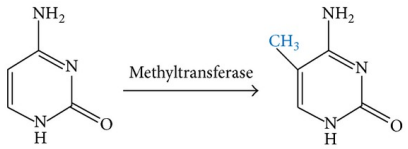


Figure 1: **Structural difference between unmethylated and methylated cytosine.** Left is cytosine that will be modified by DNA methyltransferase and right is a methylated cytosine. The methyl group is added at the 5' carbon site (blue). The figure is taken from Fakhr et al. [1].

The DNA methyltransferases (DNMTs) enzymes play a major role in the aforementioned modification. In mammals, DNMTs have 2 types of roles: (1) de novo DNA methylation and (2) maintenance DNA methylation [21]. The human enzymes, DNMT3A and DNMT3b, are responsible for the modification of the unmethylated DNA double-strand in the de novo case [21]. For the maintenance case, the human enzyme DNMT1 is responsible for the methylation. It methylates the hemimethylated DNA strand during cell division, where only the template DNA strand is methylated and the newly synthesised strand is not [20] [22]. One of the main roles of DNA methylation is gene expression regulation, which occurs when methylated DNA recruits proteins that are involved in gene repression. Additionally, DNA methylation inhibits the binding sites of the DNA for transcriptions factors [23]. In other words, DNA methylation is a major determinant of whether a gene will be transcribed or not [24]. To sum up, the activity of this modification and regulatory elements in the genome is mostly associated with gene silencing [25] [21].

However, this modification can be dysregulated, e.g. when the gene regulatory elements have been incorrectly methylated or unmethylated, it can lead to diseases including cancer [26]. Two classes of genes that play a role in oncogenesis are tumour suppressor genes and proto-oncogenes, respectively [27]. These genes play intrinsic roles in cell cycle regulation, apoptosis, immune evasion and many other processes that are required for the onset of cancer. If the promoters of proto-oncogenes are unmethylated, those oncogenes are highly expressed, resulting in uncontrollable cell growth. On the other hand, if the promoter of a tumour suppressor is methylated, that tumour suppressor can not suppress pro-tumorigenic mechanisms. In that case, these tumour suppressors are lowly expressed, which can lead to cancer [27] [24].

Nevertheless, DNA methylation plays a huge role in medicine [13]. According to Dor and Cedar [13] methylation patterns are determined in a process that continues throughout human's development. To

put it differently, a change in the methylation profile has a correlation with ageing and the development of cancer [13] [14] , which is a good biomarker for tumour diagnostic [13]. The methylated patterns are able to characterise the cell identity, lineage, and classify cell types as well as tumour types [14]. Capper et al. [5] concluded that DNA methylation is a good indicator for distinguishing between different brain tumour types. Figure 2 illustrates the typical bimodal distribution of DNA methylation in brain tumours, where the methylation rate is mainly 0.0 or 1.0 [2]. This is similarly true for most somatic cells.

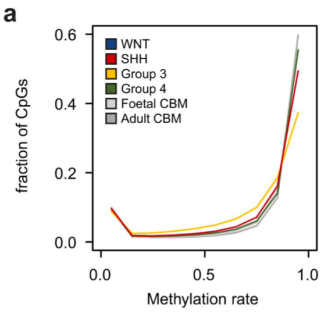


Figure 2: **DNA methylation profile for certain types of brain tumours.** The x- and y-axis describe the methylation rate with respect to the fraction of CpGs. The legend represents different types of brain tumours. The figure is taken from Hovestadt et al. [2] .

2.1.2 Profiling DNA Methylation

Most DNA sequencing and hybridization array assays can not directly detect methylated CpGs. The most common way to identify DNA methylation is by the usage of bisulphite conversion. The chemical substance, called sodium bisulfite, turns all unmethylated cytosine into uracil, while the methylated cytosine stays unchanged. In the next step, the uracyl will be converted into a thymine by using polymerase chain reaction (PCR) [28]. After PCR, the remaining cytosines would be methylated cytosine. With this prepossessing step, DNA methylation profiling can be performed. One method of methylation profiling is to use methylation arrays, such as Illumina Infinium HumanMethylation450 (450K) Bead-Chip array. Of all $\sim$ 4m CpGs in the human genome, the Illumina Infinium HumanMethylation450 (450K) Bead-Chip array covers over 480K CpG sites and targets 96% of CpG islands [29]. This array employs probes/beads, such as Infinium I and Infinium II, that can target CpG sites [3] [29]. Infinium I has two probes/beads for each CpG site. The first probe/bead is designed to locate methylated sites, while the second probe/bead is designed to locate the unmethylated sites [29]. In contrast to Infinium I, Infinium II only uses one probe/bead to differentiate methylated and unmethylated. This is done by using different dye colours (green=methylated and red=unmethylated). Both processes are illustrated in Fig 3.

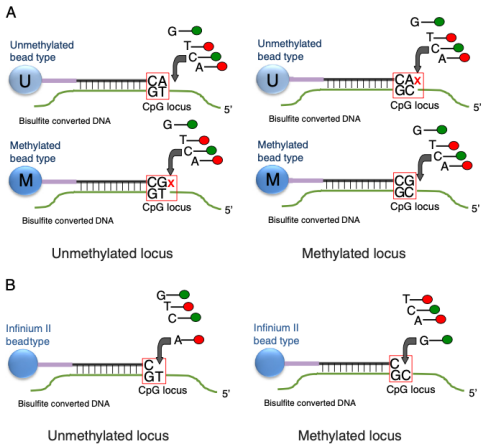


Figure 3: **Infinium Methylation Assay scheme for (A) Infinium I assay and (B) Infinium II assay. A**: Two probes correspond to each CpG locus. One probe for methylated cytosine ("C") and the other probe for unmethylated thymine ("T") state of CpG site. Both probe types for the same CpG locus will incorporate the same type of labeled nucleotide, determined by the base preceding the interrogated ("C") in the CpG locus, and therefore will be detected in the same colour channel. **B**: One probe corresponds to each CpG locus. Each locus will be detected in two colours. Adenine ("A") is always incorporated at unmethylated query site ("T"), and guanin ("G") is incorporated at methylated query site ("C"). Figure and description was taken from Bibikova et al. [3] .

In order to determine the methylation level of one CpG site/position after the above described process, the β-value can be computed as $\beta = M/(M + U + \alpha)$, where α is 100 and $M = log_2(\beta/(1-\beta))$, which is the logit-transformed β-value. The value U and M are unmethylated and methylated respectively [29]. At the end, the derived DNA methylation profile can be analysed bioinformatically. In addition to the methylation array method, there are other methods to determine a methylation profile, such as Whole-Genome Bisulfite sequencing (WGBS) and Reduced-Representation Bisulfite Sequencing (RRBS) [28]. Recently, there have been other sequencing methods that can profile DNA methylation without the need for bisulphite conversions, such as Oxford Nanopore sequencing and Enzymatic Methylation sequencing (EM-seq), however, both of these methods fall outside the scope of this study.

2.1.3 Tumour Diagnostic and Classification

According to Jiao et al. [10], the primary tumour's organ and cell of origin and histopathology are the most valuable factors for the patient clinical behaviour. Even tumours are very heterogeneous and not all tumours that have been found in patients are primary tumour's organ of origin [10] [11]. Often, these tumours are metastatic and originate from other tissues [5]. It turned out that a therapy based on the tumour's cell of origin is more effective than broad-spectrum chemotherapy [10]. Being able to identify the precise type of tumour from the patient means to treat the patient with a more appropriate drug [11]. The current practice for tumour classification in clinics is Immunohistochemistry (IHC). The type and the path of the origin of a tumour can be detected with the usage of this histological method [10]. It is however, a time-intensive process [10]. In addition to that, some tumours are less differentiated, indicating that these cells can not express the cell-type-specific proteins that are needed for the immunohistochemical classification [10]. Recently, tumour diagnostics utilize Next Generation Sequencing (NGS) approaches, including gene expression [30], DNA methylation [5] [31] and whole-genome sequencing data-based classification [10]. In combination with machine learning algorithms, the results and classification for all these types of data have been paramount in realizing precision medicine in the clinic and improving patient outcomes.

2.2 Computational Background

This section describes the computational background. To begin with, the class imbalance problem, which is illustrated in section 2.2.1. Then the Generative Adversarial Network (GAN) and class-aware GANs are introduced in the sections 2.2.2 and 2.2.3 as a solution for the imbalance problem.

2.2.1 Class Imbalance Problem

The class imbalance problem is an important and challenging topic in the field of machine learning [32]. It is defined as having samples from one or more classes that are more highly represented in the dataset in comparison to samples from other classes, which are less available [33]. In our case, each class denotes a single brain tumour type.

Classes with a great number of samples in the dataset are called the majority class and minority otherwise [33]. In terms of classification, the problem is considered to be more drastic when the minority class becomes more important and interesting than the majority class [32]. This is due to the fact that the model will be more biased to the majority classes, e.g. a model may preferentially predict a majority class over a minority class in the cases of uncertainty due to class imbalance during training when model optimization is evaluated using precision and recall metrics. This implies that the majority classes will be more accurately predicted than the minority classes, which indicates having a bad classification model [33]. The class imbalance problem is a significant problem for some machine learning algorithms, such as decision trees and neural networks, assuming that classes have an equal or similar number of samples within each class [32].

A sampling method can be used to overcome the imbalance problem by rebuilding the training data with the aim of having a balanced class-samples distribution for all classes. Sampling is achieved by either under-sampling the majority class or oversampling the minority class [33]. Undersampling is not suitable for our dataset, because there are not enough samples from the minority class. Thus, oversampling is more appropriate, which denotes generating new samples based on existing observations from samples in lowly represented classes. Generative Adversarial Network (GAN) is one algorithm that can be

used to oversample the dataset. For this thesis, the usage of GANs for effective oversampling is employed and explained in the next section.

2.2.2 Generative Adversarial Network (GAN)

In 2014, Goodfellow et al [34] introduced the idea of Generative Adversarial Network (GAN). GAN is a generative model that consists of two different neural networks called Generator (G) and Discriminator (D) [35]. Both models are designed to play a two-player game, in which the Generator G obtains a random noise z as an input and tries to generate realistic data out of z, to be named $G(z)$. On the other hand, the Discriminator D obtains $G(z)$ or a sample x from the training dataset as input [34]. The Discriminator's task is to decide whether the input derives from the training dataset or from the Generator G based on the computed probability [34]. To put it differently, the Discriminator D tries to learn to correctly classify the input samples either as *real* (sample from dataset) or as *generated* (sample from G). To be mentioned, G aims to bluff D by generating data that can not be distinguished (real vs. generated) by D [36]. Fig 4 illustrates the structure of a simple GAN.

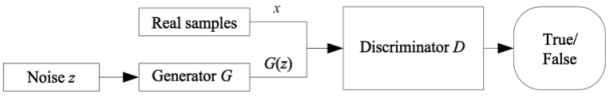


Figure 4: **Simplified architecture of a GAN.** The Figure was taken from Wang et al. [4] .

This two-player game is written in a min-max form and can be mathematically expressed in the following equation:

$$\min_G \max_D (G, D) = E_x \left[\log D(x)\right] + E_z \left[\log(1 - D(G(z)))\right] \tag{1}$$

where the Discriminator tries to maximize the left term, whereas the Generator tries to minimize the right term.

According to Wang et al. [4], GAN models are a vast research topic in the context of artificial intelligence. Especially, since they have a good performance in generating photorealistic object images. Since the aim of the thesis is to generate specific types of samples, class-aware GANs are considered, which will be described in the next section.

2.2.3 Class-aware Generative Adversarial Networks

A GAN can be modified into a class-aware GAN by adding extra information to the model, e.g. class label, from each training's sample [35]. This class-aware GAN generates samples based on the extra-input feature that specifies the class label/type of the output sample. According to Odena et al. [37], GAN's performance will be improved by adding an extra feature to its original structure.

Currently, there are three major class-aware GANs, such are Conditional GAN (cGAN) [35] , Auxiliary Classifier GAN (AC-GAN) [37], and Information Maximizing GAN (InfoGAN) [38]. Each of them have different approach for processing the input and for characterizing the output. Due to the limited scope of the bachelor thesis, only one class-aware GAN could be implemented and tested. In this thesis, AC-GAN has been selected as it is an extension of the cGAN with an additional feature for the label prediction.

AC-GAN has been first introduced in 2016 by Odena et al. [37]. According to Odena et al. [37], AC-GAN generates samples of better quality and stablize the training process. This is achieved by adding a specific cost function and improving the original GAN structure. In comparison to the normal GAN, AC-GAN is class conditional with an auxiliary decoder that can rebuild class labels. In contrast to the GAN's Generator, AC-GAN's Generator accepts two inputs: (1) the latent space and (2) a class label. The output of the AC-GAN's Generator the same as that of a GAN's Generator. The input of the AC-GAN's Discriminator is the same as the input of the normal GAN. To be noted, the AC-GAN's Discriminator

does not receive the class label as input. The AC-GAN's Discriminator outputs the probability prediction of real/generated sample (which is the same as the output obtained from GAN's Discriminator) and the probability distribution over the class labels. To sum it up, the Discriminator has an auxiliary classifier that considers the type of data to be generated. This process can be summarised in the following equation, while Figure 5 shows the structure of an AC-GAN:

$$L_S = E[\log P(S = real|X_{real})] + E[\log P(S = generated|X_{generated})] \tag{2}$$

$$L_C = E[\log P(C = c|X_{real})] + E[\log P(C = c|X_{generated})] \tag{3}$$

where $P(S = real|X_{real})$ is the probability of input S (input sources) given X_{real}. To be noticed that $P(S = generated|X_{generated}) = 1 - P(S = real|X_{generated})$. The aim of the Discriminator is to maximize the total loss $L = L_S + L_C$. Simultaneously, the Generator aims at maximizing the total loss of $L = L_C - L_S$.

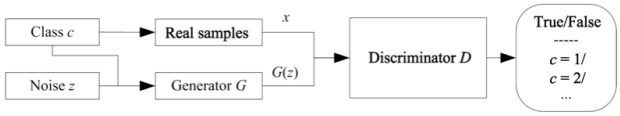


Figure 5: **Simplified architecture of an AC-GAN.** The figure was taken from Wang et al. [4] .

2.3 State of the Art by Marouf et al. 2020

At the present, not much work has been done on generating biologically omics data by the usage of GANs. Nevertheless, Marouf et al. [15] were able to generate single-cell RNA-seq data by using GANs.

The title of this paper is "Realistic in silico generation and augmentation of single-cell RNA-seq data using generative adversarial networks", which was published in 2020. In this paper, two GANs were applied: (1) single-cell generative adversarial neural networks (scGAN) for generating single-cell RNA-seq data and (2) conditional single-cell generative adversarial neural networks (cscGAN) for generating specific single-cell RNA-seq data.

The same annotated dataset was used for both GANs, which is a single-cell RNA-seq data with 68,579 cells/samples. This data is derived from peripheral blood mononuclear cells. The same loss function, Wasserstein loss and model architectures were applied for scGA and cscGAN. Three dense layers with the following order of nodes were considered for the Generator: 256, 512, and 1024. For the Discriminator the order was 1024, 512 and 256. The outermost dense layer of the Discriminator has no activation function, while the remaining layers for both Discriminator and Generator employ Rectified Linear Unit (ReLU) as activation function. At each layer of the Generator, Batch Normalization has been included. For the class-aware part, cscGAN obtains additional information, which is the label (cell types), for each sample/cell to generate type-specific cells/samples. To evaluate the quality of the generated data by scGAN, they applied (1) t-SNE, (2) marker gene correlation, (3) maximum mean discrepancy (MMD), and (4) Random Forest classification performance. On the other hand, except for (3) all other metrics has been used to evaluate cscGAN performance [15].

The scGAN generates scRNA-seq data, while the cscGAN generates scRNA-seq data for specific cell types with good quality. Also, they observed that the cscGAN could generate data from lowly represented classes (16 samples/cells in their case) because they are similar to other classes. In summary, this paper is a good basis to solve the problem which is addressed in this thesis. Certain evaluation methods, such as t-SNE for visualisation of real and generated data as well as Random Forest classifier as a classification evaluation metric, are very helpful. However, the the main problem with our dataset is that it is imbalanced and small in comparison to their dataset. Our dataset is explained in the next chapter in detail.

3 DNA Methylation Training Data

The dataset used in this thesis is from the Capper et al. [5]. This dataset contains 82 DNA methylation brain tumour classes from neuroectodermal and sellar region tumours, which are Central Nervous System (CNS) tumours. In addition, the dataset contains 9 control tissue samples. These 91 sub-family classes can be grouped into 39 class families. These sub-families are demonstrated in Figure 6. Capper et al. [5] divided these sub-family classes into 5 different categories. There are 2801 samples in total, while each sample has 428799 methylation positions/CpG features. Each position has a value between 0 and 1. These values are partitioned into highly, partially and lowly methylated and unmethylated regions. The distribution of methylation values for a single sample shows a peak at 0 and 1.

In this work, similar sub-families are grouped together to a single family on the basis of the feature selection algorithm (see section 4.5). This gives a total of 39 classes, which are presented in Table 1. The last column in this table defines the families that are highly and lowly represented. The first 7 families in this table are considered the highly represented classes, whereas the last 14 families are considered the lowly represented classes. For the remaining of this thesis, families are denoted as classes. In summary, this data set consists of 2801 rows. Each row has 428799 columns and each sample is annotated as one of the 39 classes.

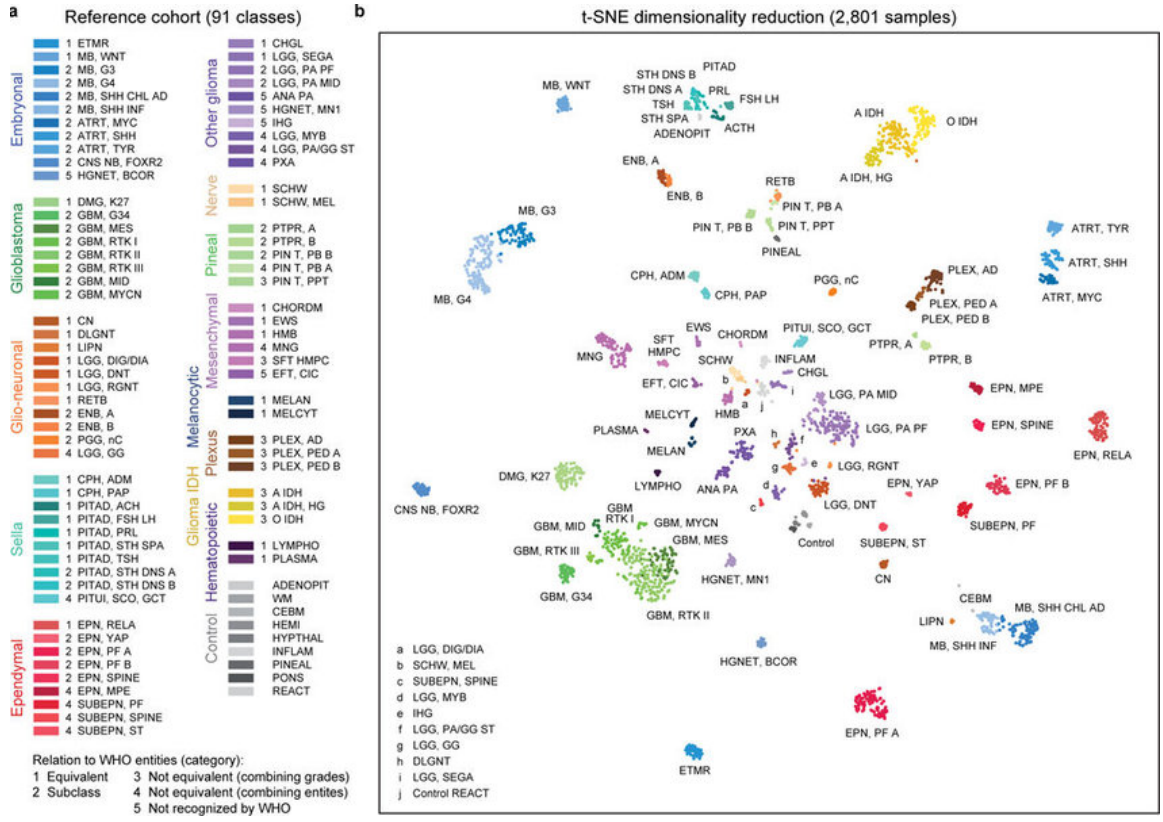


Figure 6: **Overview of the 82 CNS tumour methylation sub-classes and the 9 control tissue methylation classes illustrated at a t-SNE plot for all 2801 samples.** The methylation classes are grouped by histology and colour-coded. Category 1 methylation classes are equivalent to a WHO entity, category 2 methylation classes are a subgroup of a WHO entity, category 3 methylation classes are not equivalent to a unique WHO entity with combining of WHO grades, category 4 methylation classes are not equivalent to a unique WHO entity with combining of WHO entities, and category 5 methylation classes are not recognized as a WHO entity. Figure and description was taken from Capper et al [5].

Family	Sub-family fullname	Sub-family abbreviated	Number of samples
MB=Medulloblastoma	Medulloblastoma, sub-class group 3	MB, G3	77
	Medulloblastoma, sub-class group 4	MB, G4	138
	Medulloblastoma, sub-class sonic hedgehog A (children and adult)	MB, SHH CHL AD	52
	Medulloblastoma, sub-class sonic hedgehog B (infant)	MB, SHH INF	84
	Medulloblastoma wingless	MB, WNT	39
GBM=Glioblastoma	Glioblastoma,sub-class midline	GBM, MID	14
	Glioblastoma, H3.3 G34 mutant	GBM, G34	41
	Glioblastoma, mesenchymal	GBM, MES	56
	Glioblastoma,myelocytomatosis neuroblastoma	GBM, MYCN	16
	Glioblastoma,receptor tyrosine kinase I	GBM, RTK I	64
	Glioblastoma,receptor tyrosine kinase II	GBM, RTK II	143
	Glioblastoma,receptor tyrosine kinase III	GBM, RTK III	13
LGG=Low grade glioma	Low grade glioma, sub-class hemispheric pilocytic astrocytoma and ganglioglioma	LGG, PA/GG ST	24
	Low grade glioma, desmoplastic infantile astrocytoma / ganglioglioma	LGG, DIG/DIA	8
	Low grade glioma, sub-class midline pilocytic astrocytoma	LGG, PA MID	38
	Low grade glioma, subependymal giant cell astrocytoma	LGG, SEGA	21
	Low grade glioma, ganglioglioma	LGG, GG	21
	Low grade glioma, rosette forming glioneuronal tumour	LGG, RGNT	9
	Low grade glioma,myeloblastosis	LGG, MYB	22
	Low grade glioma, sub-class posterior fossa pilocytic astrocytoma	LGG, PA PF	114
	Low grade glioma, dysembryoplastic neuroepithelial tumour	LGG, DNT	44
EPN=Ependymoma	Ependymoma, v-rel avian reticuloendotheliosis viral oncogene homolog A	EPN, RELA	70
	Ependymoma, yes-associated protein	EPN, YAP	11
	Ependymoma, myxopapillary	EPN, MPE	28
	Ependymoma, posterior fossa group A	EPN, PF A	91
	Ependymoma, posterior fossa group B	EPN, PF B	51
	Ependymoma, spinal	EPN, SPINE	27
IDH=Isocitrate dehydrogenase	Isocitrate dehydrogenase glioma, sub-class astrocytoma	A IDH	78
	Isocitrate dehydrogenase glioma, sub-class high grade astrocytoma	A IDH, HG	46
	Isocitrate dehydrogenase glioma, sub-class 1p/19q codeleted oligodendroglioma	O IDH	80
CONTR=Control tissue	Control tissue, pituitary gland anterior lobe	CONTR, ADENOPIT	9
	Control tissue, hypothalamus	CONTR, HYPTHAL	9
	Control tissue, cerebellar hemisphere	CONTR, CEBM	8
	Control tissue, reactive tumour microenvironment	CONTR, REACT	23
	Control tissue, hemispheric cortex	CONTR, HEMI	13
	Control tissue, pons	CONTR, PONS	12
	Control tissue, white matter	CONTR, WM	9
	Control tissue, inflammatory tumour microenvironment	CONTR, INFLAM	24
	Control tissue, pineal gland	CONTR, PINEAL	12
ATRT=Atypical teratoid/rhabdoid tumour	Atypical teratoid/rhabdoid tumour, sub-class tyrosinase	ATRT, TYR	37
	Atypical teratoid/rhabdoid tumour, sub-class myelocytomatosis	ATRT, MYC	29
	Atypical teratoid/rhabdoid tumour, sub-class sonic hedgehog	ATRT, SHH	46
PITAD=Pituitary adenoma	Pituitary adenoma,thyroid-stimulating hormone	PITAD, TSH	10
	Pituitary adenoma, prolactin	PITAD, PRL	8
	Pituitary adenoma,Adrenocorticotropic Hormone	PITAD, ACTH	18
	Pituitary adenoma,Follicle-stimulating hormone/Luteinizing Hormone	PITAD, FSH/LH	21
	Pituitary adenoma,Somatotropes Hormon densely granulated, group A	PITAD, STH DNS A	9
	Pituitary adenoma,Somatotropes Hormon densely granulated, group B	PITAD, STH DNS B	12
	Pituitary adenoma,Somatotropes Hormon sparsely granulated	PITAD, STH SPA	17
MNG=Meningioma	Meningioma	MNG	90
PLEX=Plexus	Plexus tumour, sub-class paediatric A	PLEX, PED A	15
	Plexus tumour, sub-class paediatric B	PLEX, PED B	46
	Plexus tumour, sub-class adult	PLEX, AD	22
DMG=Diffuse midline glioma	Diffuse midline glioma H3 K27M mutant	DMG, K27	78
SUBEPN=Subependymoma	Subependymoma, posterior fossa	SUBEPN, PF	37
	Subependymoma, supratentoria	SUBEPN, ST	19
	Subependymoma, spinal	SUBEPN, SPINE	9
PIN T=Pineoblastoma	Pineoblastoma group A / intracranial retinoblastoma	PIN T, PB A	9
	Pineoblastoma group B	PIN T, PB B	22
	Pineal parenchymal tumour	PIN T, PPT	19
CPH=Craniopharyngioma	Craniopharyngioma, papillary	CPH, PAP	20
	Craniopharyngioma, adamantinomatous	CPH, ADM	25
HGNET=High grade neuroepithelial tumour	CNS high grade neuroepithelial tumour with MN1 alteration	HGNET, MN1	21
	CNS high grade neuroepithelial tumour with BCOR alteration	HGNET, BCOR	23
PXA=Pleomorphic xanthoastrocytoma	(anaplastic) Pleomorphic xanthoastrocytoma	PXA	44
ETMR=Embryonal tumour with multilayered rosettes	Embryonal tumour with multilayered rosettes	ETMR	43
CNS NB=CNS neuroblastoma	CNS neuroblastoma with FOXR2 activation	CNS NB, FOXR2	39
ENB=Esthesioneuroblastoma	Esthesioneuroblastoma, sub-class A	ENB, A	23
	Esthesioneuroblastoma, sub-class B	ENB, B	16
SCHW=Schwannoma	Schwannoma	SCHW	23
	Schwannoma,Melanotic	SCHW, MEL	8
PTPR=Papillary tumour of the pineal region	Papillary tumour of the pineal region group A	PTPR, A	8
	Papillary tumour of the pineal region group B	PTPR, B	22
PITUI= Pituicytoma	Pituicytoma / granular cell tumour / spindle cell oncocytoma	PITUI GCT/SCO	29
HMB=Hemangioblastoma	Hemangioblastoma	HMB	25
ANA PA=Anaplastic pilocytic astrocytoma	Anaplastic pilocytic astrocytoma	ANA PA	21
CN=Central neurocytoma	Central neurocytoma	CN	21
PGG=Paraganglioma	Paraganglioma, spinal non-CIMP	PGG, nC	19
RETB=Retinoblastoma	Retinoblastoma	RETB	19
SFT HMPC=Solitary fibrous tumour / hemangiopericytoma	Solitary fibrous tumour / hemangiopericytoma	SFT HMPC	16
MELCYT=Melanocytoma	Melanocytoma	MELCYT	15
EWS=Ewing sarcoma	Ewing sarcoma	EWS	14
EFT=Ewing sarcoma family tumour	CNS Ewing sarcoma family tumour with CIC alteration	EFT, CIC	13
LYMPHO=Lymphoma	Lymphoma	LYMPHO	13
CHGL=Chordoid glioma	Chordoid glioma of the third ventricle	CHGL	12
MELAN=Melanoma	Melanoma	MELAN	12
IHG=Infantile hemispheric glioma	Infantile hemispheric glioma	IHG	10
LIPN=Liponeurocytoma	Cerebellar liponeurocytoma	LIPN	10
CHORDM= Chordoma	Chordoma	CHORDM	9
DLGNT=Diffuse leptomeningeal glioneuronal tumo	Diffuse leptomeningeal glioneuronal tumour	DLGNT	8
PLASMA=Plasmacytoma	Plasmacytoma	PLASMA	8

Table 1: **Overview of all methylation brain tumour classes and the corresponding sub-family.** The first three columns denote the family, its corresponding sub-families and their shortcut. Last column determines the number of samples.

4 Methods

This chapter describes the methods and resources that have been used to solve the problems addressed in this thesis.

4.1 Model Architecture

The following subsections provide a brief explanation of the general basics of deep learning structures and their parameters that have been used for the implementation of the AC-GAN. This includes the different types of layers, the hyperparameters and the different types of activation functions that have been used in this thesis.

4.1.1 Deep Neural Networks Layers

According to O'Shea and Nash [39] Deep Neural Networks (DNN) are computational processing systems that have been inspired by biological nervous systems. Normally, DNN is built from one input layer, several hidden layers and one output layer. In our case, the Generator has two input layers, while the Discriminator has two output layers. Different layers perform different operations. The following paragraphs give an overview of all layers that have been used in this thesis.

Convolutional This layer is responsible for extracting information from a local subset of inputs by multiplying this subset with a vector (in 1D case) or square matrix (in 2D case) called the kernel. The kernel goes through all adjacent subsets of the input and their multiplications result in a new matrix. The weights of the kernel are the same for the entire layer [39]. Convolutional layer accepts input of any dimension [40]. In this thesis, a 1D convolutional layer is employed.

Transposed Convolutional This layer is a transposed convolution, a deconvolution, which up-samples the input while at the same time applying a convolution to it [41].

Dense This layer connects every input to every output by calculating the weighted sum of input and corresponding weights matrix [39].

Flatten This layer converts the high-dimensional input into a low-dimensional output, which is normally 1D output [42].

Reshape This layer transforms the inputs into the given shape, which is the desired dimension of the output array [43]. In our case, this is important for the Generator as it has two different inputs that must have the same shape in order to merge them.

Concatenate This layer merges a list of inputs (that have the same shape except for the concatenation axis). The output array is the concatenation of all inputs and its new shape changes only for the concatenation axis. For instance, the concatenation of inputs with shapes [5, x, 2] and [5, y, 2] is [5, x+y, 2] [44]. In our case, it merges the inputs of the sample and class into one matrix.

Dropout This layer applies a dropout into the inputs, in which it randomly ignores some input units by setting these units to 0. This technique is normally applied during the training phase. By doing this overfitting can be avoided [45].

Batch Normalization This layer normalise the input in such a way that the output will have a mean close to 0 and a standard deviation close to 1 [46] [47].

Activation This layer applies an activation function in order to transform an input signal into an output signal. This output signal will be forwarded to the next layer [48] [49]. Usually, most of the used activation functions are non-linear activation functions [48] that limit the output values to a certain range. These activation functions help in introducing the non-linearity to the neural network as without them the neural network is just a linear regression model [48]. Figure 7 introduces the activation functions that have been used in this thesis [50].

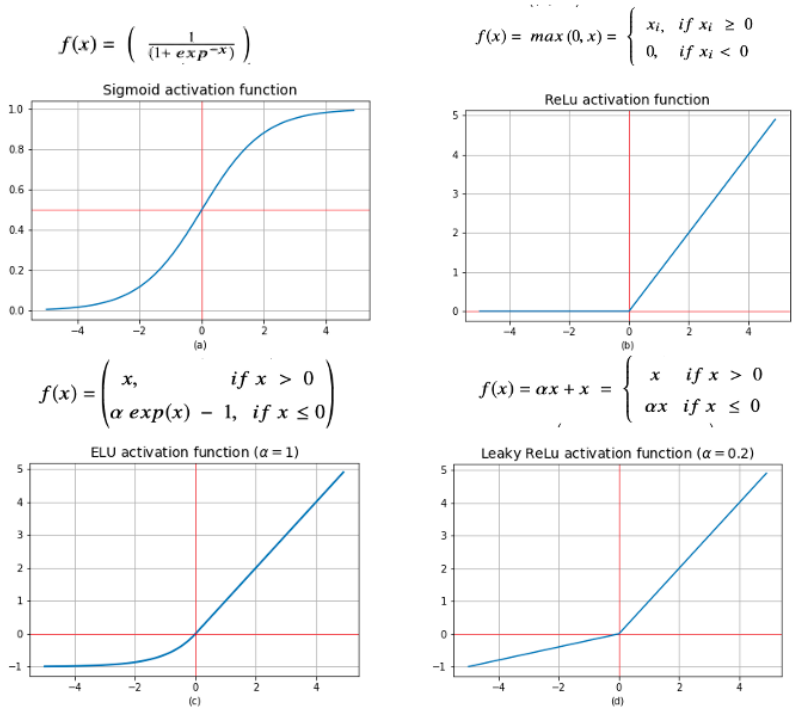


Figure 7: **Activation functions and their corresponding plots. (a)** Sigmoid activation function, **(b)** ReLu activation function, **(c)** ELU activation function and **(d)** Leaky ReLu activation function.

4.1.2 Hyperparameter

The following paragraphs describe the hyperparameters that are necessary for training an AC-GAN.

Sample A dataset consists of many samples, in which each sample is one row. In our case, it is one cancer type with the corresponding beta-values/CpG features (see Chapter 3). The dataset, represented by all its samples, will be used to train the AC-GAN. In this thesis, the number of samples in the training phase correspond to 100 for each iteration [51].

Batch Size A training dataset is normally divided into one or more batches. Each batch contains a pre-defined number of samples. Every batch will be forwarded through the AC-GAN model architecture and at the end of the computation the model will be updated with regards to the error between the expected (true) output and the calculated output [51].

Epoch An epoch is defined as going through the entire dataset once and it is comprised of one or more batches [51]. In our case, the maximum epoch size is 500.

Learning Rate Based on the computed error, the learning rate controls the amount of change the model needs to conduct between each iteration. The learning rate depends on the chosen loss function and the optimization strategy [52]. In our case, the learning rate is 0.0002 with Adam as an optimization strategy.

Kernel-initializer A kernel-initializer is a function responsible for initialising the weights of the network when creating the model [53]. In our case, the kernel-initializer initialises the weights using a random normal generator with a standard deviation equal to 0.02.

Latent Space The latent space refers to a space in which the data is represented differently by an encoder function. This space should also preserve the correlation of the data point [54]. In our case, the latent size is 100 and it is used as input for the Generator.

4.2 GAN Problem Consideration

Section 4.2.1 describes the main problems that can occur while training any GAN model. These problems include mode collapse and overfitting. In order to overcome these problems, two methods have been used in this thesis and are presented in sections 4.2.2 and 4.2.3.

4.2.1 Overfitting and Mode Collapse

Overfitting occurs when a machine learning algorithm does not learn the data efficiently [39]. In other words, it learns the training data very well but fails to fit the test data [55]. Overfitting normally is caused by a limited number of training data samples or the model architecture is too complex given the data [56]. In our case, overfitting happens when the Generator generates data that is nearly identical to individual training data samples [57]. Mode collapse is a problem that happens when the Generator tries to map different latent points to the same or similar output sample [58]. At the same time, the Discriminator can not distinguish or at least identify this failure and thus it gets stuck in a local minimum, which the Generator takes advance of [59]. In short, mode collapse is defined as all the modes collapse into one single mode. The next two subsections introduce two methods for avoiding the mode collapse problem.

4.2.2 Early Stopping

A strategy for overcoming mode collapse is called early stopping. The aim of early stopping is to avoid over (and unnecessary further) training. The model stops when it realises that there are no more additional improvements in the loss. This has the advantage of decreasing the learning time [60]. Another reason for using early stopping is that this is a simple technique for regularizing the training of the AC-GAN [61].

Early stopping in this thesis is only considered after the 1000^{th} iteration in the main loop. The aim of this is to prevent it from stopping too soon. At this iteration, the current values of *dReal* and *dGenerated* (see section 4.3.1) are saved in two variables *best_ dGenerated* and *best_ dReal*. Starting from the 1001^{st} training iteration, two cases are considered: 1) If one of the current calculated values of *dReal* or *dGenerated* is smaller than the *best_ dGenerated* or *best_ dReal* respectively, then *best_ dGenerated* and *best_ dReal* will be updated by the current corresponding values. 2) If the first case does not occur 300 times successively then the training will be automatically stopped. The model at the 300 iterations earlier will be chosen as the best model and its parameters will be saved.

4.2.3 Wasserstein Loss

Another strategy for avoiding overfitting is the usage of Wasserstein loss. Integrating this loss to the GAN model would results in a new model called Wasserstein GAN (WGAN) [62]. Wasserstein loss, also called Earth Mover's distance, aims to minimize the probability distribution between the original data points and the generated data points [62]. In addition to these advantages, this loss intends to make the learning more stable [62]. This loss is usually used for image retrieval problems due to its effectiveness [63]. In comparison to the normal GAN, in which the Discriminator yields 0 for generated and 1 for real, the Discriminator model in WGAN returns real values describing the scores the realness or fakeness (real number) of a given sample in the dataset [64]. The modification for our AC-GAN was made according to the description provided by Jason Brown in [64]. In this thesis, the Wasserstein loss is used as a tool in order to avoid mode collapse/overfitting to the majority class and not used as an evaluation for the Discriminator.

4.3 Evaluation Metrics

For the evaluation of the used AC-GAN models, different evaluation metrics were used. This section introduces these metrics. The goal of these evaluation metrics is to assess the performance of the Discriminator and the Generator [65]. Validation metrics used to evaluate the Discriminator are presented in sections 4.3.1 and 4.3.2, while the remaining subsections deal with evaluating the Generator.

4.3.1 Loss Function

Loss functions are used to calculate the model error. The lower the error value is the better the model will be obtained [36]. For our use case, lower loss indicates having a model that can generate good methylation data. The following Table 2 presents the loss functions that have been used for the AC-GAN models:

Metric	Formula
Binary Cross Entropy (BCE)	$\frac{1}{n}\sum_{i=1}^{n} -(y_i' \log(y_i) + (1 - y_i') \log(1 - y_i))$
Sparse Categorical Cross-entropy (SCC)	$-\sum_{i=1}^{n} y_i' \cdot \log(y_i)$

Table 2: **Loss functions for the AC-GAN.** The first column defines the name of the loss function and the second column defines the corresponding formula. Formula was adapted from Nava et al. [6].

where y_i is the true input, y_i' is the predicted output and n is the total number of samples to be evaluated [6].

The core idea of the loss function is to compare the predicted value y_i' (obtained from the AC-GAN Generator) and the true value y_i (obtained from the raw dataset (real), or randomly generated data (generated)). This action takes place during the training process. In our case *dReal* and *dGenerated* are the losses for the real and generated samples, while *dReal2* and *dGenerated2* are the losses for the class.

4.3.2 F1, Precision, Recall and Accuracy

To evaluate the performance of the Discriminator, *F1-score*, *Precision*, *Recall* and *Accuracy* have been calculated. That happens during training by first computing the confusion matrix *[[TP, FN],[FP, TN]]*, where *TP*, *TN*, *FP*, and *FN* are true positive, true negative, false positive and false negative, respectively. In our case, these values are defined below:

- *TP* = The sample to evaluate is real and the prediction is real. This is then a correct prediction.
- *FN* = The sample to evaluate is real and the prediction is generated. This is then a wrong prediction.
- *FP* = The sample to evaluate is generated and the prediction is real. This is then a wrong prediction.
- *TN* = The sample to evaluate is generated and the prediction is generated. This is then a correct prediction.

With these values, *Precision*, *Recall*, *F1-Score* and the *Accuracy* have been computed [66]. The following Table 3 describes these evaluation measurements [6].

Metric	Formula
Precision	$\frac{TP}{TP+FP}$
Recall	$\frac{TP}{TP+FN}$
F1-score	$\frac{2TP}{2TP+FP+FN}$
Accuracy	$\frac{TP+TN}{TP+FP+TN+FN}$

Table 3: **F1, Precision, Recall and Accuracy equations.** Formulas was adapted from Nava et al. from [6].

4.3.3 K-Nearest Neighbors and Distance

To assess the performance of the Generator, principal component analysis (PCA) was applied. PCA reduces the dimensionality of the data [67]. Here, the data was reduced to two principal components (PC1 and PC2). The raw data and the generated data from the AC-GAN model were projected to these two PCs. To find out to which class the newly generated sample belongs to, the K-Nearest Neighbors (*KNN*) algorithm was used. *KNN* is a supervised algorithm that helps in finding the nearest neighbours of a given point [68].

Additionally, the Euclidean distance, referred to as *Dist*, was employed in order to compute the distance between the generated data and samples from the raw dataset. In this thesis, there are a total of 39 classes in the raw dataset. For each class, the Generator generates 10 samples. This results in a total of 39*10 = 390 new samples that should be evaluated. The following two paragraphs explain how to evaluate the Generator using *KNN* and *Dist* for two cases: (i) the correct assessment of the classification and (ii) whether the model is overfitted or not. This was both visually and computationally evaluated. The next paragraphs describe the evaluation of these two cases.

Evaluating the Correctness of the Classification The centroid point for each class of the raw data in the PCA space is calculated. For each class, the nearest centroid (using *KNN*) and the distance from the centroid (using Euclidean distance) to each of the 10 generated samples are computed. *KNN* value is computed as follows: For each generated point (blue point in Figure 8), the distances for all centroids are computed and sorted. These values are stored in *centroids_position_list*. The centroid of the associated real class (red star in Figure 8) with regards to the generated point is remarked. The *KNN* value is computed by counting the number of the closest centroids until the real class centroid appears. To put it differently, it is the subsequent index of the associated centroid in the sorted *centroids_position_list*. Three *KNN* and *Dist* values are considered in this thesis:

- *KNN all*: the median of the *KNN* value of all 390 samples.
- *KNN low*: the median of the *KNN* value of the lowly represented classes. The classes which have less than 20 samples are determined as lowly represented classes.
- *KNN high*: the median of the *KNN* value of the highly represented classes. The classes which have more than 100 samples are determined as highly represented classes.
- *Dist all*: the mean of the *Dist* value of all 390 samples.
- *Dist low*: the mean of the *Dist* value of the lowly represented classes. The classes which have less than 20 samples are determined as lowly represented classes.
- *Dist high*: the mean of the *Dist* value of highly represented classes. The classes which have more than 100 samples are determined as highly represented classes.

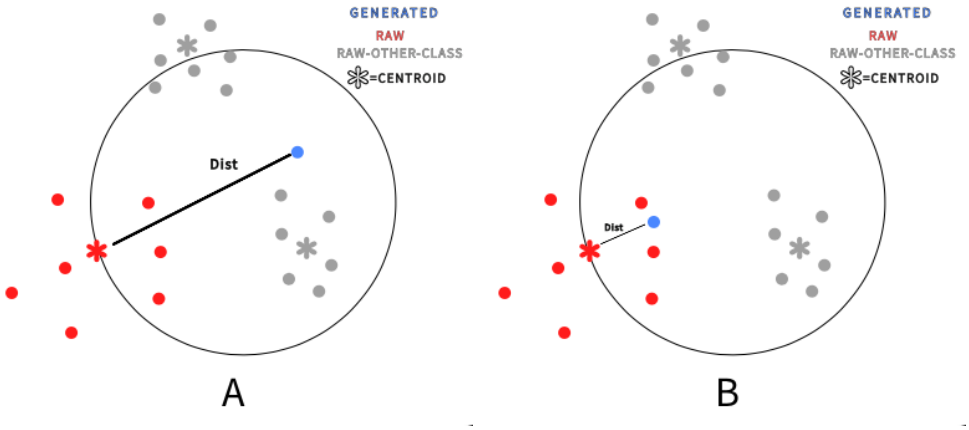


Figure 8: **Calculation of *Dist* and *KNN* illustrated.** The description of the data points are in the figure. (A) and (B) show respectively the worst and best case scenarios. Source: Own construction.

Evaluating of the Overfitting Criteria of the Model The computation in this step is similar to the approach explained in the previous paragraph. First, the nearest samples in the raw dataset to all the generated points for each class are computed using *Dist*. Then, the model is defined as overfitted if more than half of the generated samples have been predicted to the same sample raw class.

4.3.4 Methylation Profile

Another useful validation metric used in this thesis is the line plot to visualise the distribution of the density DNA methylation profile, as shown in Figure 2. For each class, these (profile) line plots have been created containing the distribution of the real and generated as described in the algorithm below.

Algorithm 1 Calculation of DNA Methylation Profile

```
for each class do
    for for each real sample in class do
        calculate histogram
        extract sample histogram values from hist object
        add them to "array of real class histogram values"
    end for
    for  each of the 10 generated sample in class do
        calculate histogram
        extract sample histogram values from hist object
        add them to an "array of generated class histogram values"
    end for
    create line plot with 95 % confidence interval for an "array of real class histogram values"
    add line plot with 95 % confidence interval for an "array of generated class histogram values"
end for
```

4.4 Workflow

Figure 9 illustrates a workflow that has been done to solve the problem of this thesis. This workflow is divided into two phases consisting of 5 steps. The first phase answers the question: How many different numbers of CpG features does a given AC-GAN architecture need, whereas the remaining steps (phase two) deal with answering the question: What architecture does the AC-GAN need to generate lowly represented DNA methylation cancer types. A detailed description and explanation of each step follows in the next two sections.

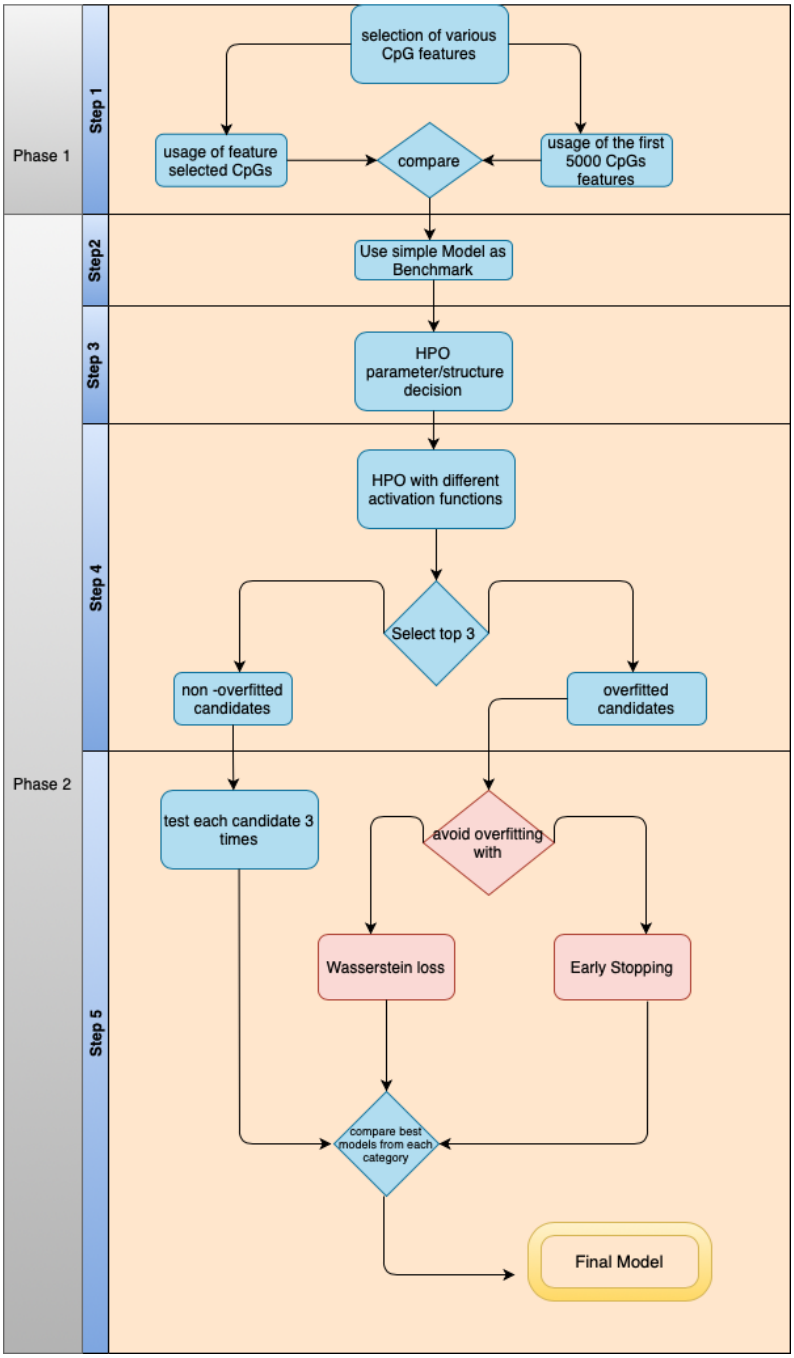


Figure 9: **Flow chart illustrating the 5 step approach** Source: Own construction.

4.5 Feature Selection

Due to the large computing time and the out-of-memory issue on the cluster, the dataset has to be smartly reduced. To resolve this, a feature selection method was applied. Feature selection is a technique in the context of machine learning to remove irrelevant and/or redundant features and select only the relevant ones. In other words, for a dataset with N rows and M columns (or attributes, dimension), feature selection aims to reduce M to M', where $M' \leq M$ [69]. The feature selection for this thesis has been done by Chia-An Lee, a master student from the AG Ishaque. The following points summarise Chia-An Lee's approach for selecting the best features:

1. Standardise the entire methylation matrix using z score.
2. Remove all features with high intra-class variance (Divide the methylation matrix into 13 sub-methylation matrices for this and then calculate the variance).

3. Remove all features with low inter-class variance (Remove the high variance features selected in the previous step before performing this step).

4. Filter the features that exceeded a standard deviation (i.e. $var() > 1$) for intra-class (about 5193 features were removed).

5. Filter the features, where $var() < 0.7$ for inter-class. As a result, only 976 CpG sites remained.

In this thesis, two experiments have been defined, in which a different number of CpG features were studied. These experiments are illustrated in step 1 of the Workflow in Figure 9. In the first experiment, the first 5000 CpG columns (also called features) of the dataset have been selected to define a new dataset. In the second experiment, the best 976 CpGs features were selected by Chiaan Lee. The architectures of the Generator and Discriminator used for both experiments are illustrated in Fig 10 and 11, respectively. The architecture itself originally was implemented for fashion MNIST data, which has been modified [70]. Same for *Batch size*=64, *epoch size*=500 and *latent dimension*=100 were carried out for both feature selection experiments. The results of these both experiments are evaluated (as described in section 4.3) and compared with each other. The dataset with the best results was selected and forwarded to the next phase.

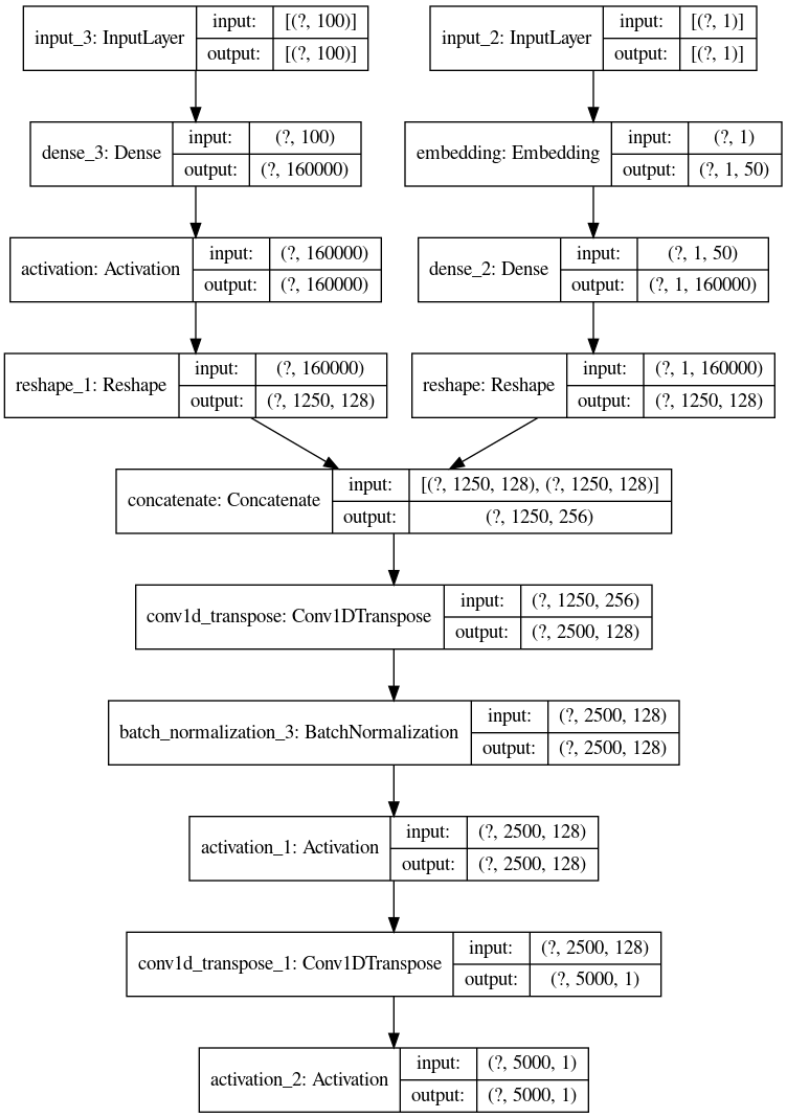


Figure 10: **Network architecture of the initial Generator model for phase 1.**

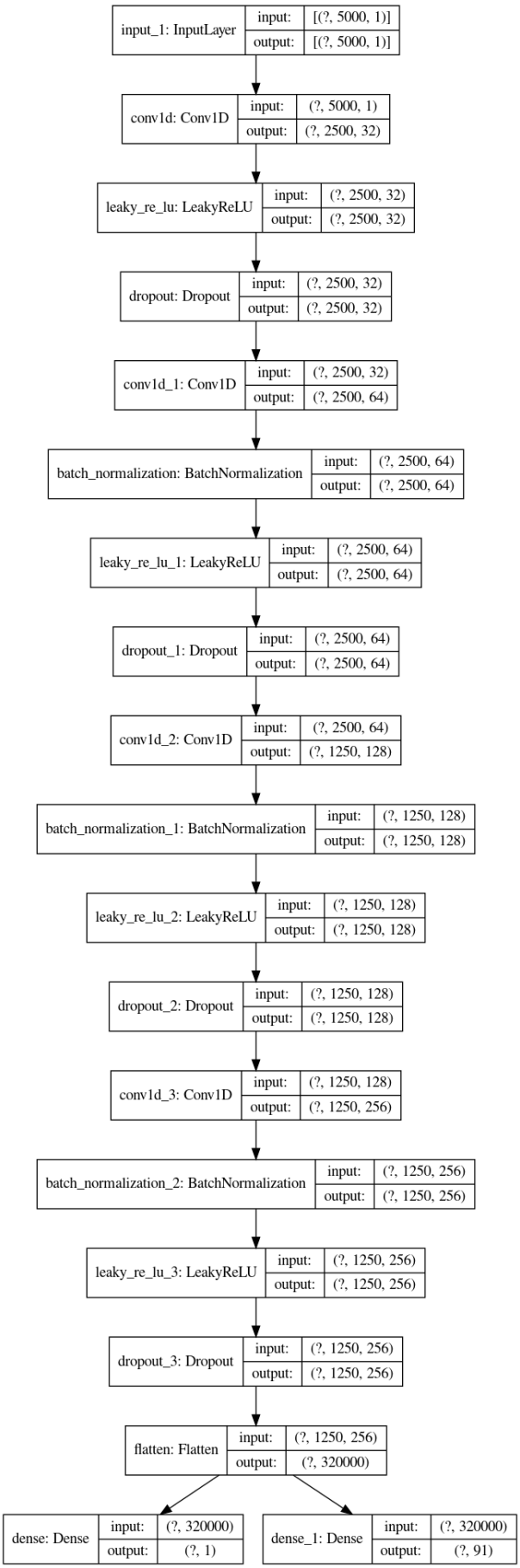


Figure 11: **Network architecture of the initial Discriminator model for phase 1.**

4.6 Hyperparameter Optimization

Every machine learning algorithm has hyperparameters that should be tuned [71]. These hyperparameters can heavily affect the performance of the learning algorithm [72]. According to Hutter et al. [73], there are number of hyperparameters sets in deep learning. Some examples of these sets are (i) parameters for the network architecture itself (number of layers, units per layer), (ii) learning parameters (learning rates, number of samples), (iii) weight initialization parameters, etc. Due to the large number of hyperparameter sets and the time frame for completing this thesis, the focus of this thesis is only on a limited number of these parameters. The hyperparameters used in this thesis are 1) number of layers, 2) number of nodes in each layer, 3) activation functions and 4) *batch size*. To be mentioned, the latent dimension remained unchanged. Seeing that hyperparameter optimisation (HPO) was carried out in this thesis. Its object is to choose the right parameters from all possible hyperparameter sets.

The following subsections describe the second phase of the workflow, which is of the stepwise HPO, in Figure 9. Each step will be explained individually.

4.6.1 Step 2: Usage of the Benchmark Model

This step is considered a prepossessing step for the HPO. The initial AC-GAN architecture described in Figure 10 and Figure 11 was modified to fit our dataset. Since the dataset has no complex architecture, layers that were less relevant were removed, such as Convolutional, Transposed Convolutional, Leaky ReLU, Batch Normalization and Dropout. These layers are more necessary for image generation in comparison to our dataset which is a sequential dataset [39] [15]. Removing the less important layers was completed for both the Generator and the Discriminator. This modified model will be called the "benchmark AC-GAN model". This benchmark model was trained and tested on the selected dataset from step 1. Figures 12 and 13 present the Generator and Discriminator of the benchmark models in black, respectively. The purpose of the red box will be explained in section 4.6.2.

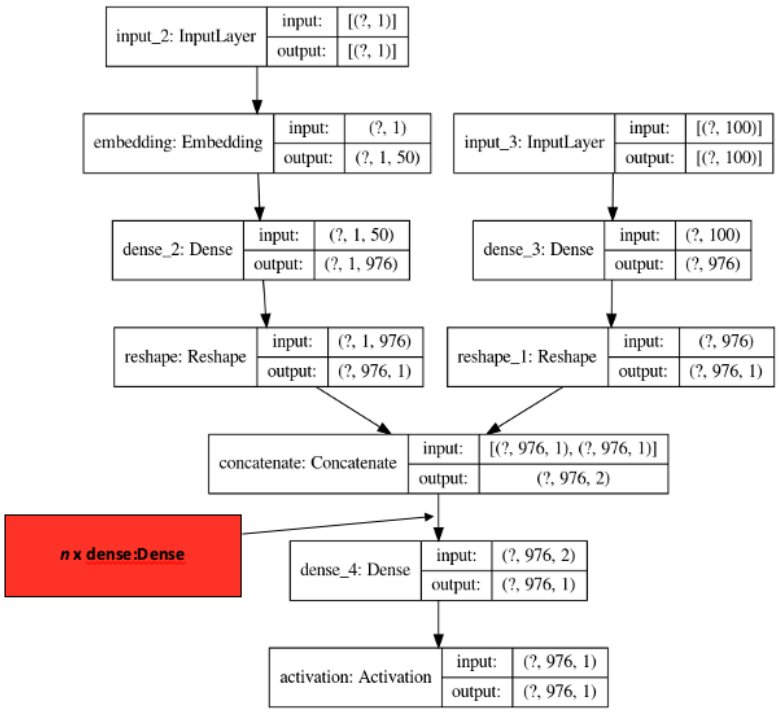


Figure 12: **Network architecture of the benchmark Generator model for phase 2.**

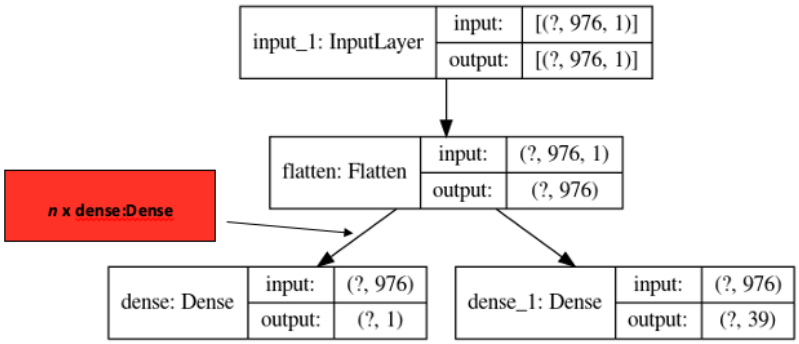


Figure 13: **Network architecture of the benchmark Discriminator model for phase 2.**

4.6.2 Step 3: HPO Parameter/Structure Decision

This step is the main part of the HPO. Different models were obtained by modifying the benchmark model from the previous step. The modification of the Generator and Discriminator was mainly for the number of dense layers ($\in\{0, 1\}$), number of nodes ($\in\{64, 256, 512\}$) and batch size ($\in\{32, 64, 100, 120\}$). Elu activation function was used for all these models. Elu was chosen because it outperforms other activation functions [74]. The combinations of all these parameters are demonstrated in Table 4 for the Discriminator and Generator in columns 2 and 3, respectively. Each row in this Table defines one model. These modifications are also displayed in the red box in Figures 12 and 13. The aim of defining all these models is to detect the ideal models including the ideal parameters for our dataset.

The procedure was as follows: Only the non-overfitted models from analysing the results obtained from the defined 16 models were further studied. First, one more dense layer was added for the non-overfitted Generator models. Second, these new models were further analysed and only the non-overfitted models were selected. Third, from these selected models, a random number of nodes were considered. In addition, the order of the number of nodes was also studied in the case of number of layers bigger than 1. Lastly, the same approach was conducted for the Discriminator. As a consequence of all these created models, the best AC-GAN models (including Generator and Discriminator) were passed on for the next steps for further investigations.

By trying all these model combinations, the following questions can be investigated and answered:

1. Does the Generator, the Discriminator or both models have to be modified?
2. How many dense layers should be added to each model?
3. What is the ideal combination of nodes for each layer and what should be the number of nodes?
4. What is the best *batch size*?

Model	Neurons for Discriminator	Neurons for Generator
model 1	[ ]	[ ]
model 2	[ ]	[64]
model 3	[ ]	[256]
model 4	[ ]	[512]
model 5	[64]	[ ]
model 6	[64]	[64]
model 7	[64]	[256]
model 8	[64]	[512]
model 9	[256]	[ ]
model 10	[256]	[64]
model 11	[256]	[256]
model 12	[256]	[512]
model 13	[512]	[ ]
model 14	[512]	[64]
model 15	[512]	[256]
model 16	[512]	[512]

Table 4: **All HPO combinations for Step 3.** The number of model is define in the first column. In the second and third column, the number of neurons for the Discriminator and Generator are respectively define.

4.6.3 Step 4: HPO with Different Activation Functions

In this step, the effect of the activation function will be studied. For this reason, the best models from the previous step were selected and investigated again, but with Sigmoid and Relu instead of Elu. All other hyperparameters were the same. To be mentioned, the activation function of a particular model is the same for all dense layers in that model. This enables the fair analysis of the role of the activation function. Consequently, the best three overfitted and three non-overfitted model-parameters candidates of all activation functions are selected for further analysing. The selection is based on the evaluation metrics described in section 4.3.

4.6.4 Step 5: Testing of Overfitted and Non-overfitted Candidates

The reliability of the 6 models (see 4.6.3) is tested in two ways: 1) Each of the non-overfitted candidates will be tested three times and 2) The overfitted models will be tested once with the usage of Wasserstein loss and once with early stopping with the hope to prevent overfitting. After that, the best model parameters from each testing category will be selected based on the evaluation metrics. As a result, 3 final model parameters are obtained. Finally, these 3 models will be compared with each other and the best parameter combination will be selected as a final model. This final model should be able to generate lowly represented cancer type data.

5 Implementation Details

The scrips have been implemented in Python 3.9.6[1]. The initial implementation of AC-GAN is based on Jason Brownlee's blog[2]. Here, TensorFlow Keras[3] library has been used for all implementations. Pandas[4] package was used for analysing the visualzing of the data and results. Moreover, NumPy[5], Matplotlib[6], Seaborn[7] and scikit-learn[8] has been used. The code, models and scripts to create the results from this thesis can be found under: https://github.com/anthonyhami/Bachelor_Thesis_AC_GAN_Final.

[1] https://www.python.org
[2] https://machinelearningmastery.com/how-to-develop-an-auxiliary-classifier-gan-ac-gan-from-scratch-with-keras/
[3] https://www.tensorflow.org/api_docs/python/tf/keras?version=nightly
[4] https://pandas.pydata.org
[5] https://numpy.org
[6] https://matplotlib.org
[7] https://seaborn.pydata.org
[8] https://scikit-learn.org/stable/

6 Results

This chapter consists of two different evaluation parts. The first section describes the evaluation of the usage of different numbers of CpG features for a given AC-GAN architecture. The second section describes the evaluation of different AC-GAN architectures to generate lowly represented DNA methylation cancer types.

6.1 Model Based on Various CpG Features

This section describes the evaluation of AC-GAN results using various CpG features. The results are the outcome of two main experiments, where (1) the first 5000 CpG and (2) the best 976 selected CpG positions have been used. The first 2 subsections represent the results of the evaluation of the Discriminator during the training phase and the Generator during the testing phase. The last paragraph summarises the final result for this section. In order to compare and judge the quality of both experiments, the results were collected together in Table 5 and Table 6

6.1.1 Evaluation of the Discriminator during the Training Phase

The first column of Table 5 defines the evaluation metrics used while training the Discriminator model. In the second and third columns, the outcome of these evaluation metrics for the first 5000 CpG features (Exp1) and the best 976 selected CpG features (Exp2) are demonstrated, respectively. It can be observed that the class and sample losses for real and generated (*dReal, dReal2, dGenerated2*) for the selected CpG feature are frequently lower than for the first 5000 CpG features. Moreover, the *F1, Precision and Recall* scores of the 976 positions are significantly higher than for the first 5000 positions. As for the *accuracy*, the results clearly show that the model of Exp2 could learn the real samples better than Exp1. To mention, *accuracy* of 100% for generated samples and 0% for real samples were obtained when applying Wasserstein loss.

Evaluation parameter	first 5000 CpG features (Exp 1)	best 976 selected CpG features (Exp 2)
dReal	0.798	0.670
dReal2	0.432	0.003
F1	0.145	0.812
Precision	0.750	1.0
recall	0.086	0.688
dGenerated	0.608	0.753
dGenerated2	0.824	7.41e-05
Accuracy real/generated	real: 4% generated:96%	real: 67% generated:30%

Table 5: **Training results for Discriminator for different datasets (step 1).** The first column defines the evaluation metrics (see section 4.3) used while training the Discriminator model. In the second and third column, the outcome of these evaluation metrics for the first 5000 CpG features (Exp1) and the best 976 selected CpG features (Exp2) are respectively demonstrated. Note that Wasserstein loss is not included in (Exp2).

6.1.2 Evaluation of the Generator during the Testing Phase

Similar to Table 5, Table 6 displays the outcome of the Generator evaluation for Exp1 and Exp2. The last column describes the results for the best 976 CpGs using Wasserstein loss. It can be seen that all *KNN* measures and distance results were repeatedly better for Exp2 than for Exp1. Although only a few classes out of 39 were correctly predicted. It is important to mention that in this phase more correct classes were predicted for Exp2 than Exp1. Nevertheless, both models are overfitted, even if the model with the 976 has minimally fewer overfitted classes in comparison to the 5000 CpGs. To solve the overfitting problem, Wasserstein loss has been used with the 976 CpG dataset. On the one hand, bad *KNN* and *Dist* results were obtained. On the other hand, the generated data have no longer been overfitted and the methylation distributions look reasonable.

Evaluation parameter	first 5000 CpGs(Exp1)	976 selected CpGs (Exp 2)	976 selected CpGs Wasserstein loss
KNN all	15	5	20
KNN low	23	12	18
KNN high	12	4	18
Dist all	2,963	3.030	4.610
Dist low	4,344	4.091	4.401
Dist high	2,587	2.457	5.156
Overfitted	Yes	Yes	No
Number of overfitted classes	39/39	31/39	0/39
Number of correctly predicted classes	2	6	0
Number of correctly predicted classes low	0	1	0
Number of correctly predicted classes high	0	3	0

Table 6: **Testing results for Generator for different datasets (step 1).** The first column defines the evaluation metrics (see section 4.3) used while testing the Generator model. In the second and third column, the outcome of these evaluation metrics for the first 5000 CpG features (Exp1), the best 976 selected CpG features (Exp2) are respectively demonstrated. Note that Wasserstein loss was used in (Exp2).

6.1.3 Summary

The training with the selected features provides comprehensively better results. Therefore, it was decided to further continue the evaluation of AC-GAN in the second part of this thesis only based on the new filtered dataset represented by those selected features. By using Wasserstein loss, overfitting can be avoided, however, the generated data looks more randomly generated and not class-based. Thus, Wasserstein loss was only used in the final evaluation of the complete thesis for final model comparison (see 6.2.5).

6.2 Models Based on Various AC-GAN Architectures

This section evaluates the results from my workflow for the HPO as described in section 4.6. To decide which AC-GAN architecture generates the best lowly represented tumour classes, a hyperparameter tuning has to be applied. The benchmark models was modified as described in section 4.6.1.

The first subsection represents the result of the benchmark models with and without the usage of Wasserstein loss. A different number of layers and nodes combination for the AC-GAN Generator and/or Discriminator model were explored. The results of these combinations are demonstrated in the second subsection. The third subsection represents the results of the previous subsection while using different activation functions. The following subsection describes the testing results for the best candidates while the fifth paragraph is about the testing results for the best-overfitted candidates including the usage of Wasserstein loss and early stopping. The last paragraph summarises the final result of this section.

6.2.1 Benchmark Model

In order to compare and judge the quality of the benchmark model with and without the usage of Wasserstein loss, the results were collected together in Table 7 and Table 8.

Table 7 represents the results of the evaluation metrics for the benchmark model (similar to Table 5). It can be observed that the class loss for real and generated (*dReal2 and dGenerated2*) are very low in comparison to the sample loss for real and generated (*dReal and dGenerated*). Also, the *F1* and *Recall* scores are very low whereas the *Precision* is 1. The *accuracy* results clearly show that the benchmark model without Wasserstein could learn the generated samples better than the real samples, while the *accuracy* with Wasserstein was the same as in the previous chapter (acc real 100%, acc generated 0%).

Similar to Table 6, Table 8 displays the outcome of the metrics defined in section 4.3 for the benchmark model without and with the usage of Wasserstein loss, respectively. It can be seen that the results

shown for the benchmark model with the usage of Wasserstein loss are significantly better than without using Wasserstein loss. Nevertheless, the *KNN* value for minor cancers types (*KNN low*) is still high. The *KNN* for the high represented classes (*KNN high*) is low for the benchmark model Wasserstein loss in comparison to without the usage of Wasserstein loss. Even with the usage of Wasserstein, both models were overfitted. The *accuracy* for benchmark model with Wasserstein loss was similar to the previous chapter (acc real 100%, acc generated 0%). In summary, the results were better in comparison to the previous models, even though the architecture has been simplified to only a few complex layers.

Evaluation parameter	benchmark model
dReal	0.711
dReal2	0.001
F1	0.420
Precision	1.0
Recall	0.265
dGenerated	0.682
dGenerated2	4.0888e-07
Accuracy real/generated	real:27% generated:80%

Table 7: **Training results of the benchmark model for step 2.** The first column defines the evaluation metrics (see section 4.3) used while training the benchmark model. In the second column, the outcome of these evaluation metrics for the benchmark model is demonstrated.

Evaluation parameter	Benchmark model (without Wasserstein)	Benchmark model (with Wasserstein)
KNN all	10	4
KNN low	13	14
KNN high	8	2
Dist all	3,267	2,16
Dist low	4,1423	3,80
Dist high	2,633	1,90
Overfitted	Yes	Yes
Number of overfitted classes	38/39	34/39
Number of correctly predicted classes	2	5
Number of correctly predicted classes low	0	0
Number of correctly predicted classes high	1	2

Table 8: **Testing results of the benchmark model for step 2.** The first column defines the evaluation metrics (see section 4.3) used while testing the benchmark model. In the second and third column, the outcome of these evaluation metrics for the benchmark model without Wasserstein and the benchmark model with the usage of Wasserstein loss are respectively demonstrated.

6.2.2 Decision of Structure

Throughout the described procedure (see 4.6.2), it was possible to find out which parameters are important for the HPO. It was observed that the Discriminator needs to be modified with a maximum of 2 dense layers and the Generator can be modified by 0-2 additional dense layers. It was also been detected that the Discriminator should have 512 nodes per layer. The Generator can have 64, 256 or 512 nodes, while the order of the nodes does not play a role. It has been noticed, that the *batch size* of 100 or 120 is ideal to avoid overfitting. Thus, the models are trained for less time. Overfitting is observed to be obtained for *batch size* of 64 and 32. The full results for this can be seen in the GitHub (see section 5).

6.2.3 Hyperparameter Optimisation with Different Activation Functions

Based on the results from the previous subsection, different activation functions for each model combination have been used. Sigmoid activation function mostly leads to an overfitted model, even if the *KNN* or *Dist* results were good. The methylation profiles for the generated samples were not good while there were high peaks for the values 0 and 0.9999. Out of all, the used activation functions, Elu delivered good evaluation results. Yet, a few models were overfitted. *KNN* and/or *Dist* values were mostly very

good. The results with Relu were also not often overfitted, but the *KNN* and *Dist* values were too high compared to Elu. Table 9 and Table 10 define the selected top 3 candidates of the over-fitted and non-overfitted models including their hyperparameters, respectively. The evaluation results in the end of training and testing phases for all these selected models are summarised in Table 11 and Table 12.

Hyperparameter	Model 1	Model 2	Model 3
Number of neurons/layers Discriminator	[512]	[512]	[512]
Number of neurons/layers Generator	[512,512]	[64,256]	[ ]
Batch	120	120	100
Activation function	Elu	Elu	Sigmoid
Trainig Step	11500	11500	14000

Table 9: **Top 3 non-overfitted candidates and their corresponding hyperparameters.** The first column defines the hyperparameters while the remaining columns define the corresponding output for each model.

Hyperparameter	Model 1	Model 2	Model 3
Number of neurons/layers Discriminator	[512,512]	[512,512]	[512]
Number of neurons/layers Generator	[256,512]	[64,256]	[512,64]
Batch	120	120	120
Activation function	Simoid	Sigmoid	Elu
Trainig step	11500	11500	11500

Table 10: **Top 3 overfitted candidates and their corresponding hyperparameters.** The first column defines the hyperparameters while the remaining columns define the corresponding output for each model.

Table 11 summarise the results for the top non-overfitted candidates. The sample losses for real and generated were still high, while Model 3 had the lowest loss score. In contrast to the class loss for real and generated, the losses were low, especially the generated class loss for Model 1 and Model 2. Model 3 had the highest score for *F1* and *Recall* score. The *Precision* scores for all models are 1.0. The highest *accuracy* score for Real and Generated had also Model 3, which shows that this model learned the real and generated samples properly. It can be detected, that the testing phase evaluation parameters were all very similar to each other.

Description	Evaluation parameter	Model 1	Model 2	Model 3
Training phase	dReal	0.682	0.692	0.508
	dReal2	0.001	0.001	0.032
	ACC Real	0.467	0.560	0.830
	F1	0.586	0.704	0.927
	Precision	1.0	1.0	1.0
	Recall	0.422	0.547	0.867
	dGenerated	0.651	0.653	0.460
	dGenerated2	0.0006	0.0003	0.020
	ACC Generated	0.880	0.650	0.940
Testing phase	KNN all	4	4	5
	KNN low	7	10	7
	KNN high	2	2	1
	Dist all	3.240	3.271	3.434
	Dist low	4.340	4.850	4.500
	Dist high	1.812	1.682	1.834
	Number of correctly predicted classes	10	15	14
	Number of correctly predicted classes low	2	1	2
	Number of correctly predicted classes high	5	6	5

Table 11: **Top 3 non-overfitted candidates evaluation.** Top: Results of during training phase. Down: Results during testing phase. The second column presents the used metrics (see section 4.3). The last three columns show the results of the three non-overfitted models.

Table 12 represents the results of the 3 overfitted models. The sample losses for these candidates were again very high for all models, whereas for generated loss for Model 2 has the lower loss value. For class loss, Model 3 outputs the lowest class loss values in comparison with Model 1 and 2. The *F1*-score is high for all models. The *Recall* score for Model 3 is the highest for the three models. Similar to Table 11, the *Precision* scores for all overfitted models are also 1. All the real *Accuracy* values are now low, but the *Accuracy* for generated were all above 0.9, which means that all overfitted candidate learned generated samples better than real samples. Regarding the testing phase: The *KNN* and *Dist* values are for all overfitted models almost similar, however, the *Dist high* for Model 1 is especially low. Regarding the class prediction results, Model 1 had the highest result, and all low predicted classes are the same. The results for each activation function can be found in the GitHub (see section 5).

Description	Evaluation parameter	Model 1	Model 2	Model 3
Training phase	dReal	0.644	0.6266	0.6140
	dReal2	0.0841	0.105	0.003
	ACC Real	0.580	0.610	0.730
	F1	0.718	0.736	0.846
	Precision	1.0	1.0	1.0
	Recall	0.5625	0.585	0.734
	dGenerated	0.440	0.337	0.6241
	dGenerated2	0.02339	0.036	0.0005
	ACC Generated	0.97	0.93	0.74
Testing phase	KNN all	5	7	5
	KNN low	6	6	7
	KNN high	2	2	3
	Dist all	3.165	3.928	3.360
	Dist low	3.795	3.798	3.379
	Dist high	1.733	3.477	2.047
	Number of correctly predicted classes	9	4	6
	Number of correctly predicted classes low	2	2	2
	Number of correctly predicted classes high	4	1	3

Table 12: **Top 3 overfitted candidates evaluation.** Top:Results of during training phase. Bottom:Results during testing phase. The second column presents the used metrics (see section 4.3). The last three columns show the results of the three overfitted models.

6.2.4 Testing Top Non-overfitted Candidates

Throughout the described procedure (see section 4.6.4), the best model parameters have been tested for each model three times. The full testing results can be found in the GitHub (see section 5). It could be detected that for Model 1 and 2 with Elu activation function there was no improvement and the results were slightly worse in terms of *KNN* and *Dist*. To be mentioned, two out of nine attempts Model 1 and 2 were once overfitted. In contrast to Model 3 which had Sigmoid as an activation function, none of the models were overfitted. Also, the *KNN* score for low becomes better in comparison to the results in the top 3 none overfitted candidates results. Also, the predicted numbers of classes were higher, especially for high represented classes.

6.2.5 Testing Top Overfitted Candidates

This section provides the results for the top overfitted candidates. The results are shown in the GitHub (see section 5). As expected, the usage of Wasserstein loss and early stopping could avoid overfitting. The *KNN* and *Dist* values were high especially for *KNN low* and *KNN high*. The result for early stopping was better as such the training stopped mostly after 1800 training's iterations. To be noticed, the *KNN* values (*all, low, high*) improved in comparison with all previous model's results. The number of correctly predicted classes when using both Wasserstein loss and early stopping is always between 15-20, however, the generated samples look more randomly generated and are not based on the class distribution. Unfortunately, these models are not reliable, also because the generated samples try to learn only one specific raw sample from the lowly represented classes.

6.3 Summary Result

Table 13 and 14 summarise the best models of (i) testing, (ii) the usage of Wasserstein and (iii) early stopping (see section 4.6.4). Figure 14 shows the PCA result for these 3 models. The red and the blue points represent the true class and all other classes, respectively. The green points represent the generated new points by the usage of Model 1, the black points represent the generated new points by the usage of Model 2 and the yellow points represent the generated new points by the usage of Model 3. The PCA plots for cancer type "GBM" and "IHG" are demonstrated on the left and right sides of these figures, respectively. To be mentioned, the "GBM" is one of the highly represented classes in our dataset, while "IHG" is one of the lowly represented classes. Figure 15 represents the line plots, as described in section 4.3.4, for cancer types "GMB" (left) and "IHG" (right). The remaining lowly represented cancer types and their corresponding PCA and line plots can be found in the Appendix section 9 and the results for the other cancer types can be found in the GitHub (see section 5).

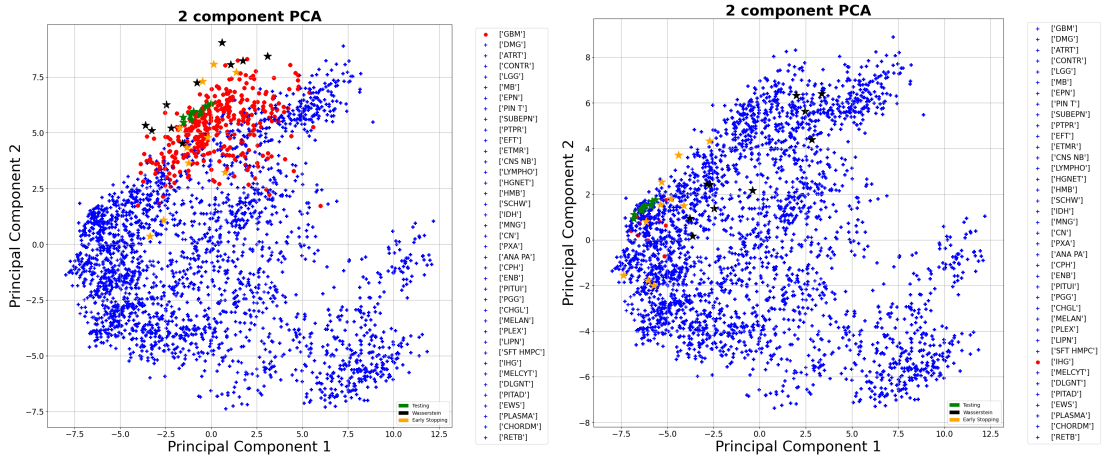
In summary, it is difficult to judge what is the best model. The common feature of all selected models is that they are able to generate correct samples for the over-represented classes, whereas the lowly represented classes are rather poorly or rarely correctly presented. The usage of early stopping demonstrated that the AC-GAN does not need all the training's iteration to generate realistic methylation data. Moreover, there is no clear decision about the number of dense layers to be added for the Generator and Discriminator. Obviously, models with Elu by testing showed that it has a risk for overfitting. On the other hand, the Sigmoid activation function by testing did not provide any overfitted model. Adding Wasserstein and/or early stopping to the implementation showed that the overfitting problem can be solved. However, the generated data points are still randomly scattered. Notably, the newly generated points are class-aware when adding the early stopping criteria to the models. From the line plots, it can be seen that the real and generated lines almost overlap, which indicates the good quality of the AC-GAN models.

Hyperparameter	Model 1 (Testing)	Model 2 (Wasserstein)	Model 3 (Early stopping)
Number of neurons/layers Discriminator	[512]	[512]	[512,512]
Number of neurons/layers Generator	[]	[512,64]	[256,512]
Batch size	100	120	120
Activation function	Sigmoid	Elu	Sigmoid
Trainig step	14000	11500	1762

Table 13: **Hyperparameters for the final models.** The last three columns present the hyperparameters for the best selected models from each category, respectively: Testing, Wasserstein and Early stopping.

Description	Evaluation parameter	Testing	Wasserstein	Early stopping
Training phase	dReal	0.570	/	0.370
	dReal2	0.020	/	0.533
	ACC Real	0.610	0.0	0.840
	F1	0.806	/	0.930
	Precision	1.0	/	1.0
	Recall	0.70	/	0.875
	dGenerated	0.45	/	0.30
	dGenerated2	0.021	/	0.364
	ACC Generated	0.9499	0.0	0.940
Testing phase	KNN all	7	7	7
	KNN low	6.5	14.5	8.5
	KNN high	1	4	9
	Dist all	4.01	4.30	3.40
	Dist low	4.09	5.60	3.42
	Dist high	1.90	3.03	3.34
	Number of correctly predicted classes	12	19	20
	Number of correctly predicted classes low	5	4	5
	Number of correctly predicted classes high	5	6	5

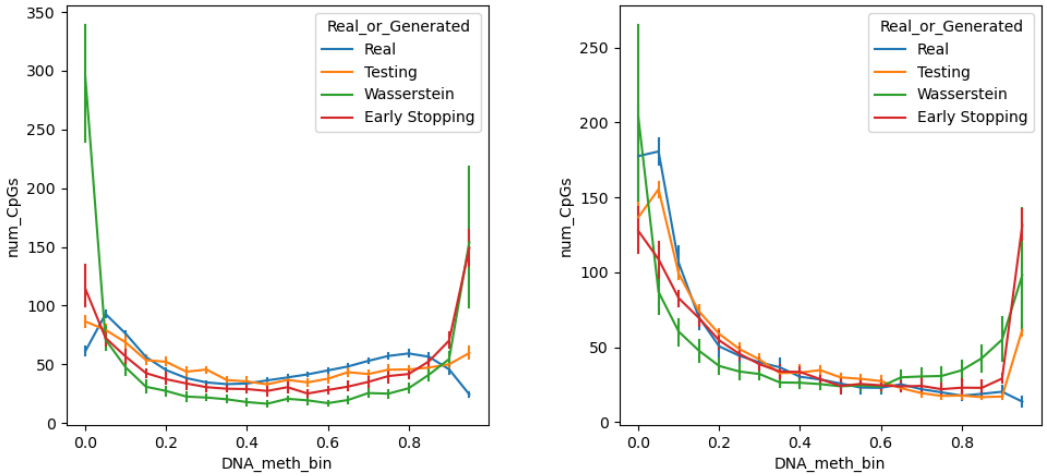
Table 14: **Evaluation results for the final three models.** Top: Results of during training phase. Bottom: Results during testing phase. The last three columns present the results for the best selected models from each category, respectively: Testing, Wasserstein and Early stopping.



(a) PCA plot for cancer type "GBM"

(b) PCA plot for cancer type "IHG"

Figure 14: **PCA plots for Model 1-3** (see Table 13). The red and the blue points represent the true class and remaining classes, respectively. The green points represent the generated new points from Model 1. The black points represent the generated new points for Model 2, and the yellow points represent the generated new points for Model 3.



(a) Line plot for cancer type "GBM"

(b) Line plot for cancer type "IHG"

Figure 15: **Line plots for Model 1-3** (see Table 13 and section 4.3.4). The x- and y-axis describe the DNA methylation bin with respect to the number of CpGs. The blue line represents the true class. The orange, green and red lines presents the bins for the final three models (Testing, Wasserstein and Early stopping), respectively.

7 Discussion and Outlook

This final chapter summarises the findings of the thesis. The first section outlines the main ideas behind the work and recapitulates the working steps. In the second section, some limitations of this thesis are discussed. The third section gives a brief insight into the future work for this project, while the last section concludes the work done.

7.1 Summary

The aim of this thesis is to generate lowly represented tumour types with the usage of an AC-GAN. This has been realised with a five-step approach (Figure 9). Thereby, these five steps concentrate mainly on answering the primary and pre-defined questions of this thesis: (1) How many different numbers of CpG features does a given AC-GAN architecture needs, and (2) What architecture does the AC-GAN need to generate lowly represented DNA methylation cancer types. The five-step approach can be summarised in the following paragraphs.

1. Step In the first step, an initial AC-GAN model receives various CpG features as an input. After training the model, the results will be then evaluated using the pre-defined metrics. It has been observed that the number of features in the training dataset plays a major role in the training phase and on the model outcome. In this thesis, 976 and 5000 features were considered. The AC-GAN with an input of the best 976 selected CpG features produces a better result in comparison to an input of the first 5000 CpG features. This implies that the AC-GAN can learn better with less but selected CpG features than with more and unselected CpG features. In other words, the feature selection contributed to the outcome of the model in such a way that the AC-GAN was able to distinguish between the different classes and to generate more accurate data. Yet, by analysing the results the trained models were still overfitting the training data. This could be avoided by using Wasserstein loss. This however, the generated data looks more randomly generated and not class-based.

2. Step In the second step, a benchmark model has been applied. In spite of the fact that the architecture of this model has been simplified its performance has slightly improved. Based on this it could be concluded that the usage of complex layers, such as convolutional layers, are not necessary for generating methylation data. Nevertheless, all generated data are still overfitted, even with the usage of Wasserstein loss.

3. Step In the third step, different architectures for the Generator and Discriminator have been tested via modifying the benchmark model. To avoid overfitting and get reliable results, it turned out that (1) It is worth using a higher *batch size*, while lower *batch size* leads to mode collapse, (2) The usage of dense layers have significantly improved the performance of the AC-GAN in comparison to the usage of convolutional layers, (3) It is also recommended to use dense layers that consist of more than 64 nodes, (4) It was observed that the Discriminator needs to be modified with a maximum of 2 dense layers, while the Generator can be modified by 0-2 additional dense layers, (5) It has also been detected that the Discriminator should have 512 nodes per layers, (6) The Generator can have 64, 256 or 512 nodes, while the order of the nodes does not play a role and (7) More than two dense layers for the Generator or Discriminator caused to overfitting.

4. Step In the fourth step, different activation functions have been evaluated. It could be stated that the usage of Sigmoid is a double-edged sword. Initially, most of the models having Sigmoid as activation function generated overfitted data. This was not an encouraging reason to further use Sigmoid, but with a lot of luck few models deliver a good performance. Elu, on the other hand, delivered less overfitted results and the quality of the generated data was significantly better. The models having ReLu as activation function were also less overfitted, but the quality of the generated data is lower than the models having Elu as activation function.

5. Step In the last step, the previous best overfitted and non-overfitted models have been tested. The usage of early stopping demonstrated that the AC-GAN does not need all the training's iteration to generate realistic methylation data. Obviously, models with Elu by testing showed that it has a risk for overfitting. On the other hand, the Sigmoid activation function by testing did not provide any overfitted models. Adding Wasserstein and/or early stopping to the implementation showed that the overfitting problem can be solved. However, the generated data points are still randomly scattered. Notably, the newly generated points are class-aware when adding the early stopping criteria to the models. Another finding is that the models can generate highly represented classes nearly perfectly, while for lowly represented classes it was not always good. Nevertheless, for few lowly represented classes it looked like that the models have learned a distribution around a single sample. This indicates that for the lowly minority classes there are not enough samples used to learn.

7.2 Limitation

During the conduction of the research in order to solve the defined problem in this thesis many obstacles and limitations occurred. One of the main downsides of this research was the dataset itself: Due to the large computational time and the out-of-memory issue on the compute cluster, the complete dataset could not be able to be used for the training. Another problem in the context of the dataset was its structure: Samples from different classes are not fully separated. In other words, clusters are hard to determine. This was observed from the PCA plot. Thus, it could be difficult for AC-GAN to learn to distinguish the different classes. Another point to be raise is due to the limited time of the bachelor thesis, it was not possible to consider more hyperparameters that may have a positive influence on the training of the AC-GAN. Another challenge with regard to the training computational time is the possible number of the model that can be investigated: The more complex the model architecture is the more time the model needs to be trained. This again is not possible during the processing time of the bachelor thesis. The last issue was by employing some of the deep learning library-TensorFlow Keras-functions without a success: KerasCallback for early stopping and KerasTuner for tuning the hyperparameters. An assumption behind the failure of success of these functions is their integration limitations to AC-GAN because it has two different deep learning models. To compensate for this, a manual implementation was developed.

7.3 Future Work

This section describes the future work that could be done to extend this project and to improve the current results.

One possible extension of this project could be the expansion of the HPO, which can be used to study and investigate other parameters such as learning rate, Kernel-initializer, sample size, and optimization strategies. Another idea would be to use a complex AC-GAN architecture, in which some layers, such as

convolution layers, can be replaced with other layers, such as attention layers. In addition to the visual evaluation of the generated samples in the PCA-space, it could be interesting to train the AC-GAN with the PCA transformed training's data as an input. Another suggestion for investigating AC-GAN models would be to use different input data, which are created based on other feature selection algorithms. The approaches of Song et al. [75] and Raweh et al. [76] are to be recommended for the selection of the best features of the training's data. Alternative evaluation methods, such as Random Forest, can also be used in order to examine the correct classifications of cancer types of the generated samples. Last but not least, other class-aware GAN models, such as InfoGAN and cGAN, could be tested.

7.4 Conclusion

GAN models were successfully used in different fields in terms of bioinformatics [77] [15] and this thesis demonstrated that AC-GANs are also able to generate synthetic DNA methylation data. The focus of this thesis was on studying AC-GAN models and model architectures for generating new samples for specific (namely lowly represented) classes. Although it was not always possible to perfectly generate samples of the lowly represented tumour classes, AC-GAN demonstrated good performance for generating samples of the highly represented tumour classes. This inconvenient result, however, helps in understanding and studying the various AC-GAN model-architectures that can be further improved in the future for solving the class imbalanced problem.

8 References

[1] M. Fakhr, M. Hagh, D. Shanehbandi, and B. Baradaran, "Dna methylation pattern as important epigenetic criterion in cancer," *Genetics research international*, vol. 2013, p. 317569, 12 2013.

[2] V. Hovestadt, D. T. Jones, S. Picelli, W. Wang, M. Kool, P. A. Northcott, M. Sultan, K. Stachurski, M. Ryzhova, H.-J. Warnatz *et al.*, "Decoding the regulatory landscape of medulloblastoma using dna methylation sequencing," *Nature*, vol. 510, no. 7506, pp. 537–541, 2014.

[3] M. Bibikova, B. Barnes, C. Tsan, V. Ho, B. Klotzle, J. M. Le, D. Delano, L. Zhang, G. P. Schroth, K. L. Gunderson *et al.*, "High density dna methylation array with single cpg site resolution," *Genomics*, vol. 98, no. 4, pp. 288–295, 2011.

[4] K. Wang, C. Gou, Y. Duan, Y. Lin, X. Zheng, and F.-Y. Wang, "Generative adversarial networks: introduction and outlook," *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 4, pp. 588–598, 2017.

[5] D. Capper, D. T. Jones, M. Sill, V. Hovestadt, D. Schrimpf, D. Sturm, C. Koelsche, F. Sahm, L. Chavez, D. E. Reuss *et al.*, "Dna methylation-based classification of central nervous system tumours," *Nature*, vol. 555, no. 7697, pp. 469–474, 2018.

[6] L. Nava, O. Monserrat, and F. Catani, "Improving landslide detection on sar data through deep learning," 2021.

[7] P. Roy, B. Saikia *et al.*, "Cancer and cure: a critical analysis," *Indian journal of cancer*, vol. 53, no. 3, p. 441, 2016.

[8] S. H. Hassanpour and M. Dehghani, "Review of cancer from perspective of molecular," *Journal of Cancer Research and Practice*, vol. 4, no. 4, pp. 127–129, 2017.

[9] H. Mohsen, E.-S. A. El-Dahshan, E.-S. M. El-Horbaty, and A.-B. M. Salem, "Classification using deep learning neural networks for brain tumors," *Future Computing and Informatics Journal*, vol. 3, no. 1, pp. 68–71, 2018.

[10] W. Jiao, G. Atwal, P. Polak, R. Karlic, E. Cuppen, A. Danyi, J. de Ridder, C. van Herpen, M. P. Lolkema, N. Steeghs *et al.*, "A deep learning system accurately classifies primary and metastatic cancers using passenger mutation patterns," *Nature communications*, vol. 11, no. 1, pp. 1–12, 2020.

[11] J. I. Orozco, T. A. Knijnenburg, A. O. Manughian-Peter, M. P. Salomon, G. Barkhoudarian, J. R. Jalas, J. S. Wilmott, P. Hothi, X. Wang, Y. Takasumi *et al.*, "Epigenetic profiling for the molecular classification of metastatic brain tumors," *Nature communications*, vol. 9, no. 1, pp. 1–14, 2018.

[12] I. Dagogo-Jack and A. T. Shaw, "Tumour heterogeneity and resistance to cancer therapies," *Nature reviews Clinical oncology*, vol. 15, no. 2, pp. 81–94, 2018.

[13] Y. Dor and H. Cedar, "Principles of dna methylation and their implications for biology and medicine," *The Lancet*, vol. 392, no. 10149, pp. 777–786, 2018.

[14] F. Bormann, M. Rodríguez-Paredes, F. Lasitschka, D. Edelmann, T. Musch, A. Benner, Y. Bergman, S. M. Dieter, C. R. Ball, H. Glimm *et al.*, "Cell-of-origin dna methylation signatures are maintained during colorectal carcinogenesis," *Cell reports*, vol. 23, no. 11, pp. 3407–3418, 2018.

[15] M. Marouf, P. Machart, V. Bansal, C. Kilian, D. S. Magruder, C. F. Krebs, and S. Bonn, "Realistic in silico generation and augmentation of single-cell rna-seq data using generative adversarial networks," *Nature communications*, vol. 11, no. 1, pp. 1–12, 2020.

[16] D. E. Handy, R. Castro, and J. Loscalzo, "Epigenetic modifications: basic mechanisms and role in cardiovascular disease," *Circulation*, vol. 123, no. 19, pp. 2145–2156, 2011.

[17] J. I. Loizou, R. Murr, M. G. Finkbeiner, C. Sawan, Z.-Q. Wang, and Z. Herceg, "Epigenetic information in chromatin: the code of entry for dna repair," *Cell Cycle*, vol. 5, no. 7, pp. 696–701, 2006.

[18] J. Gayon, "From mendel to epigenetics: History of genetics," *Comptes rendus biologies*, vol. 339, no. 7-8, pp. 225–230, 2016.

[19] G. Navarro, N. Franco, E. Martínez-Pinilla, and R. Franco, "The epigenetic cytocrin pathway to the nucleus. epigenetic factors, epigenetic mediators, and epigenetic traits. a biochemist perspective," *Frontiers in genetics*, vol. 8, p. 179, 2017.

[20] S. B. Baylin, "Dna methylation and gene silencing in cancer," *Nature clinical practice Oncology*, vol. 2, no. 1, pp. S4–S11, 2005.

[21] M. V. Greenberg and D. Bourc'his, "The diverse roles of dna methylation in mammalian development and disease," *Nature reviews Molecular cell biology*, vol. 20, no. 10, pp. 590–607, 2019.

[22] M. Fatemi, A. Hermann, H. Gowher, and A. Jeltsch, "Dnmt3a and dnmt1 functionally cooperate during de novo methylation of dna," *European journal of biochemistry*, vol. 269, no. 20, pp. 4981–4984, 2002.

[23] L. D. Moore, T. Le, and G. Fan, "Dna methylation and its basic function," *Neuropsychopharmacology*, vol. 38, no. 1, pp. 23–38, 2013.

[24] S. Khan and A. K. Hilliker, *DNA Methylation and Cancer*. New York, NY: Springer New York, 2018, pp. 208–209. [Online]. Available: https://doi.org/10.1007/978-1-4614-1531-2_761

[25] E. Daura-Oller, M. Cabre, M. A. Montero, J. L. Paternain, and A. Romeu, "Specific gene hypomethylation and cancer: new insights into coding region feature trends," *Bioinformation*, vol. 3, no. 8, p. 340, 2009.

[26] K. D. Robertson, "Dna methylation and human disease," *Nature Reviews Genetics*, vol. 6, no. 8, pp. 597–610, 2005.

[27] G. I. potenzieller Marker, "Dna-methylierung und krebs," *IPA-Journal 01/2015*.

[28] W.-S. Yong, F.-M. Hsu, and P.-Y. Chen, "Profiling genome-wide dna methylation," *Epigenetics & chromatin*, vol. 9, no. 1, pp. 1–16, 2016.

[29] Z. Wang, X. Wu, and Y. Wang, "A framework for analyzing dna methylation data from illumina infinium humanmethylation450 beadchip," *BMC bioinformatics*, vol. 19, no. 5, pp. 15–22, 2018.

[30] Y. Li, K. Kang, J. M. Krahn, N. Croutwater, K. Lee, D. M. Umbach, and L. Li, "A comprehensive genomic pan-cancer classification using the cancer genome atlas gene expression data," *BMC genomics*, vol. 18, no. 1, pp. 1–13, 2017.

[31] C. Koelsche, D. Schrimpf, D. Stichel, M. Sill, F. Sahm, D. E. Reuss, M. Blattner, B. Worst, C. E. Heilig, K. Beck *et al.*, "Sarcoma classification by dna methylation profiling," *Nature communications*, vol. 12, no. 1, pp. 1–10, 2021.

[32] S. M. Abd Elrahman and A. Abraham, "A review of class imbalance problem," *Journal of Network and Innovative Computing*, vol. 1, no. 2013, pp. 332–340, 2013.

[33] R. Longadge and S. Dongre, "Class imbalance problem in data mining review," *arXiv preprint arXiv:1305.1707*, 2013.

[34] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[35] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[36] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[37] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier gans," in *International conference on machine learning*. PMLR, 2017, pp. 2642–2651.

[38] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2016, pp. 2180–2188.

[39] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *arXiv preprint arXiv:1511.08458*, 2015.

[40] K. Team. Keras documentation: Conv1d layer. Accessed: 2021-08-22. [Online]. Available: https://keras.io/api/layers/convolution_layers/convolution1d/

[41] tf.keras.layers.conv1dtranspose | TensorFlow core v2.6.0. Accessed: 2021-08-21. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv1DTranspose?hl=de

[42] tf.keras.layers.flatten | TensorFlow core v2.6.0. Accessed: 2021-08-21. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Flatten?hl=de

[43] tf.keras.layers.reshape | TensorFlow core v2.6.0. Accessed: 2021-08-20. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Reshape?hl=de

[44] K. Team. Keras documentation: Concatenate layer. Accessed: 2021-08-20. [Online]. Available: https://keras.io/api/layers/merging_layers/concatenate/

[45] ——. Keras documentation: Dropout layer. Accessed: 2021-08-20. [Online]. Available: https://keras.io/api/layers/regularization_layers/dropout/

[46] J. Bjorck, C. Gomes, B. Selman, and K. Q. Weinberger, "Understanding batch normalization," *arXiv preprint arXiv:1806.02375*, 2018.

[47] K. Team. Keras documentation: BatchNormalization layer. Accessed: 2021-08-20. [Online]. Available: https://keras.io/api/layers/normalization_layers/batch_normalization/

[48] S. Sharma and S. Sharma, "Activation functions in neural networks," *Towards Data Science*, vol. 6, no. 12, pp. 310–316, 2017.

[49] K. Team. Keras documentation: Activation layer. Accessed: 2021-08-20. [Online]. Available: https://keras.io/api/layers/core_layers/activation/

[50] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," *arXiv preprint arXiv:1811.03378*, 2018.

[51] J. Brownlee, "What is the difference between a batch and an epoch in a neural network?" *Machine Learning Mastery*, vol. 20, 2018.

[52] G. Chassagnon, M. Vakalopolou, N. Paragios, and M.-P. Revel, "Deep learning: definition and perspectives for thoracic imaging," *European radiology*, vol. 30, no. 4, pp. 2021–2030, 2020.

[53] K. Team. Keras documentation: Layer weight initializers. Accessed: 2021-08-22. [Online]. Available: https://keras.io/api/layers/initializers/

[54] D. Winant, J. Schreurs, and J. A. Suykens, "Latent space exploration using generative kernel pca," in *Artificial Intelligence and Machine Learning*. Springer, 2019, pp. 70–82.

[55] R. Webster, J. Rabin, L. Simon, and F. Jurie, "Detecting overfitting of deep generative networks via latent recovery," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 273–11 282.

[56] B. Adlam, C. Weill, and A. Kapoor, "Investigating under and overfitting in wasserstein generative adversarial networks," *arXiv preprint arXiv:1910.14137*, 2019.

[57] Y. Yazici, C.-S. Foo, S. Winkler, K.-H. Yap, and V. Chandrasekhar, "Empirical analysis of overfitting and mode drop in gan training," in *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2020, pp. 1651–1655.

[58] I. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," 2017.

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

[60] A. Lodwich, Y. Rangoni, and T. Breuel, "Evaluation of robustness and performance of early stopping rules with multi layer perceptrons," in *2009 international joint conference on Neural Networks*. IEEE, 2009, pp. 1877–1884.

[61] L. Prechelt, "Early stopping-but when?" in *Neural Networks: Tricks of the trade*. Springer, 1998, pp. 55–69.

[62] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*. PMLR, 2017, pp. 214–223.

[63] Y. Dukler, W. Li, A. Lin, and G. Montúfar, "Wasserstein of wasserstein loss for learning generative models," in *International Conference on Machine Learning*. PMLR, 2019, pp. 1716–1725.

[64] J. Brownlee. How to implement wasserstein loss for generative adversarial networks. Accessed: 2021-05-20. [Online]. Available: https://machinelearningmastery.com/how-to-implement-wasserstein-loss-for-generative-adversarial-networks/

[65] A. Borji, "Pros and cons of gan evaluation measures," 2018.

[66] D. M. W. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," 2020.

[67] M. Ringnér, "What is principal component analysis?" *Nature biotechnology*, vol. 26, no. 3, pp. 303–304, 2008.

[68] M. Mohammadi, M. Dawodi, W. Tomohisa, and N. Ahmadi, "Comparative study of supervised learning algorithms for student performance prediction," in *2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*. IEEE, 2019, pp. 124–127.

[69] H. Liu, *Feature Selection*. Boston, MA: Springer US, 2010, pp. 402–406. [Online]. Available: https://doi.org/10.1007/978-0-387-30164-8_306

[70] J. Brownlee. How to develop an auxiliary classifier GAN (AC-GAN) from scratch with keras. Accessed: 2021-05-21. [Online]. Available: https://machinelearningmastery.com/how-to-develop-an-auxiliary-classifier-gan-ac-gan-from-scratch-with-keras/

[71] M. Feurer and F. Hutter, "Hyperparameter optimization," in *Automated machine learning*. Springer, Cham, 2019, pp. 3–33.

[72] J. Wu, X.-Y. Chen, H. Zhang, L.-D. Xiong, H. Lei, and S.-H. Deng, "Hyperparameter optimization for machine learning models based on bayesian optimizationb," *Journal of Electronic Science and Technology*, vol. 17, no. 1, pp. 26–40, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1674862X19300047

[73] F. Hutter, J. Lücke, and L. Schmidt-Thieme, "Beyond manual tuning of hyperparameters," *KI-Künstliche Intelligenz*, vol. 29, no. 4, pp. 329–337, 2015.

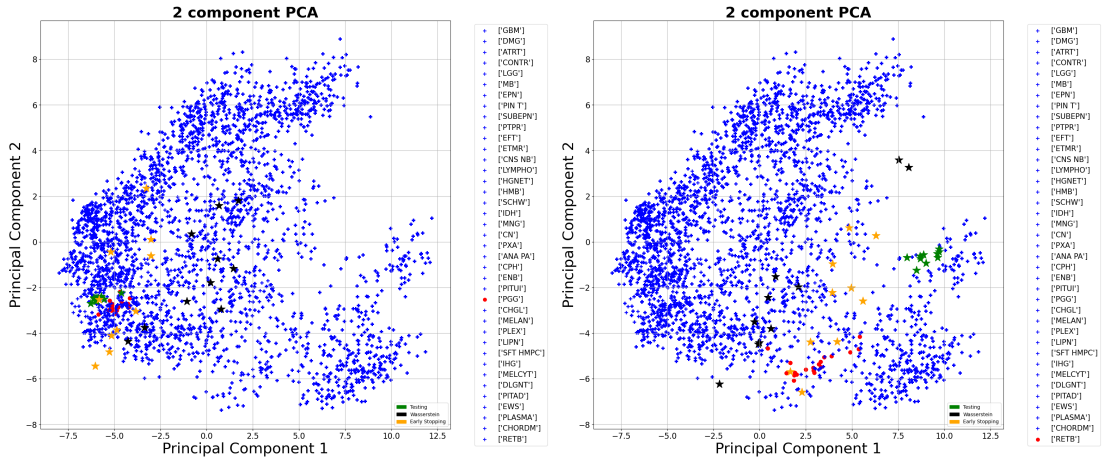
[74] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, 2015.

[75] Z. Song and J. Li, "Variable selection with false discovery rate control in deep neural networks," *Nature Machine Intelligence*, vol. 3, no. 5, pp. 426–433, 2021.

[76] A. A. Raweh, M. Nassef, and A. Badr, "Feature selection and extraction framework for dna methylation in cancer," *Int. J. Adv. Comput. Sci. Appl*, vol. 8, no. 7, pp. 30–36, 2017.

[77] L. Lan, L. You, Z. Zhang, Z. Fan, W. Zhao, N. Zeng, Y. Chen, and X. Zhou, "Generative adversarial networks and its applications in biomedical informatics," *Frontiers in Public Health*, vol. 8, p. 164, 2020.

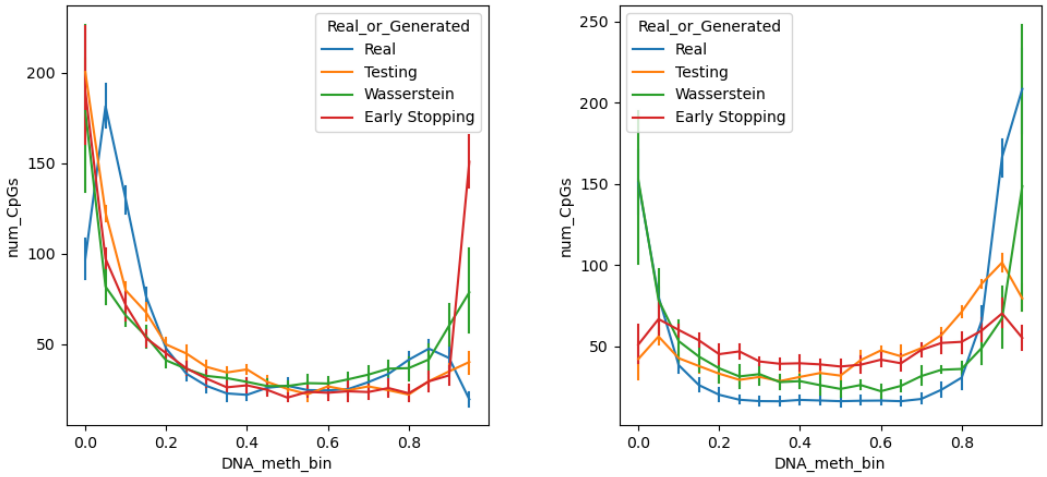
9 Appendix



(a) PCA plot for cancer type "PGG"

(b) PCA plot for cancer type "RET B"

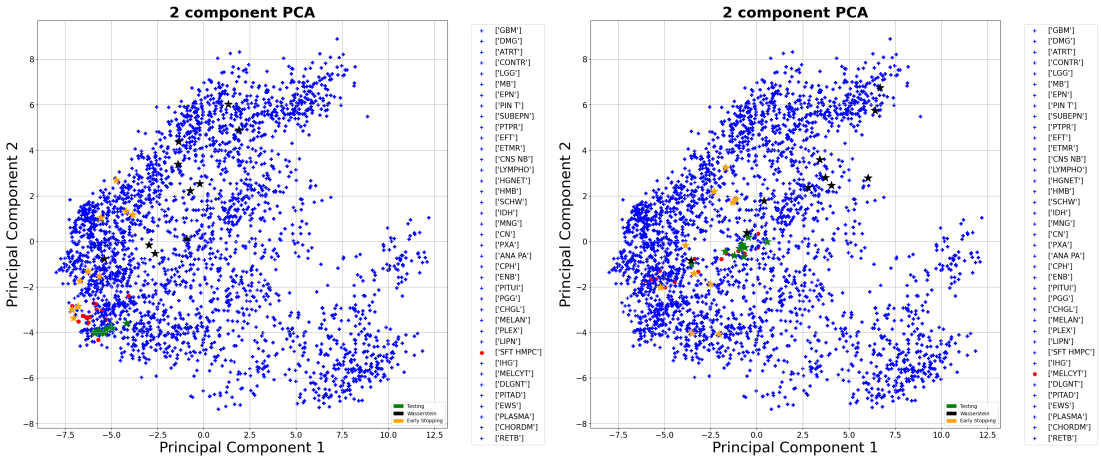
Figure 16: **PCA plots for Model 1-3** (see Table 13). The red and the blue points represent the true class and remaining classes, respectively. The green points represent the generated new points from Model 1. The black points represent the generated new points for Model 2, and the yellow points represent the generated new points for Model 3.



(a) Line plot for cancer type "PGG"

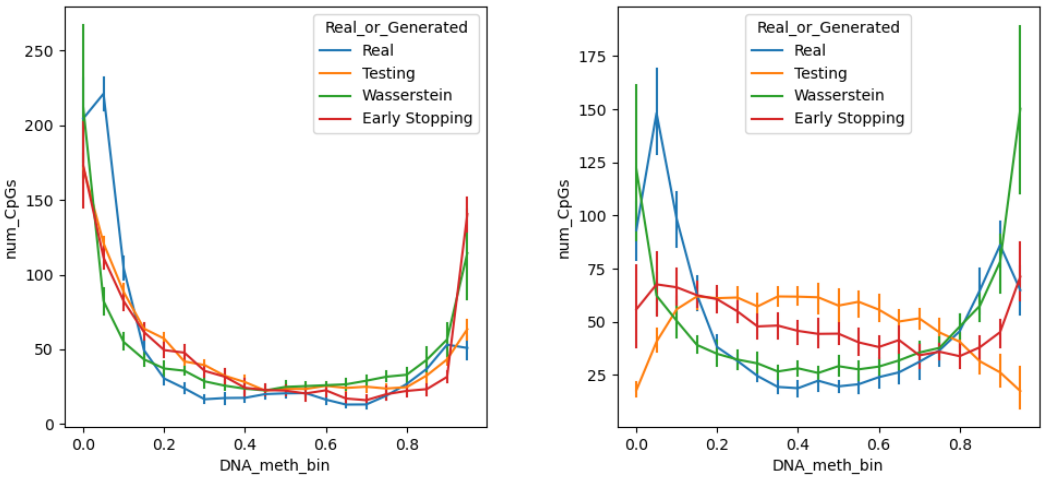
(b) Line plot for cancer type "RET B"

Figure 17: **Line plots for Model 1-3** (see Table 13 and section 4.3.4). The x- and y-axis describe the DNA methylation bin with respect to the number of CpGs. The blue line represents the true class. The orange, green and red lines presents the bins for the final three models (Testing, Wasserstein and Early stopping), respectively.



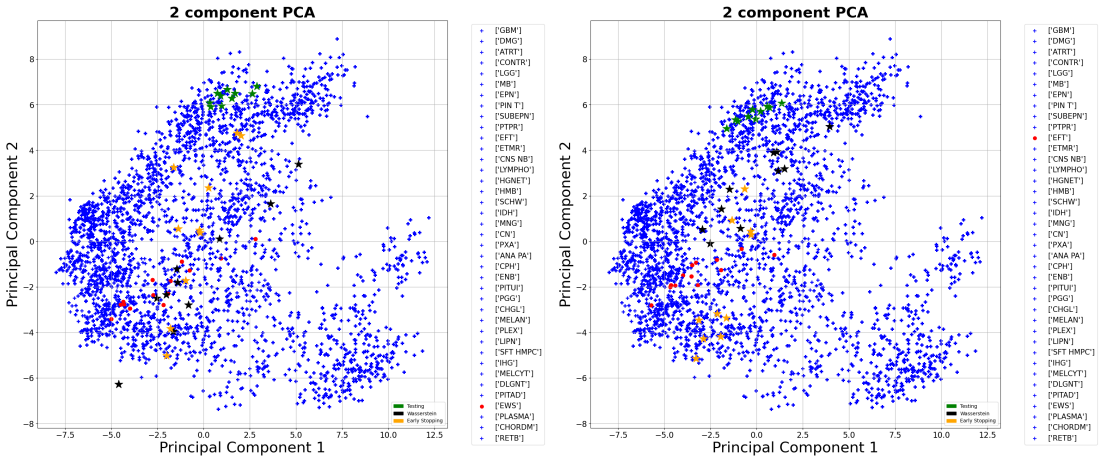
(a) PCA plot for cancer type "SFT HMPC" (b) PCA plot for cancer type "MELCYT"

Figure 18: **PCA plots for Model 1-3** (see Table 13). The red and the blue points represent the true class and remaining classes, respectively. The green points represent the generated new points from Model 1. The black points represent the generated new points for Model 2, and the yellow points represent the generated new points for Model 3.



(a) Line plot for cancer type "SFT HMPC" (b) Line plot for cancer type "MELCYT"

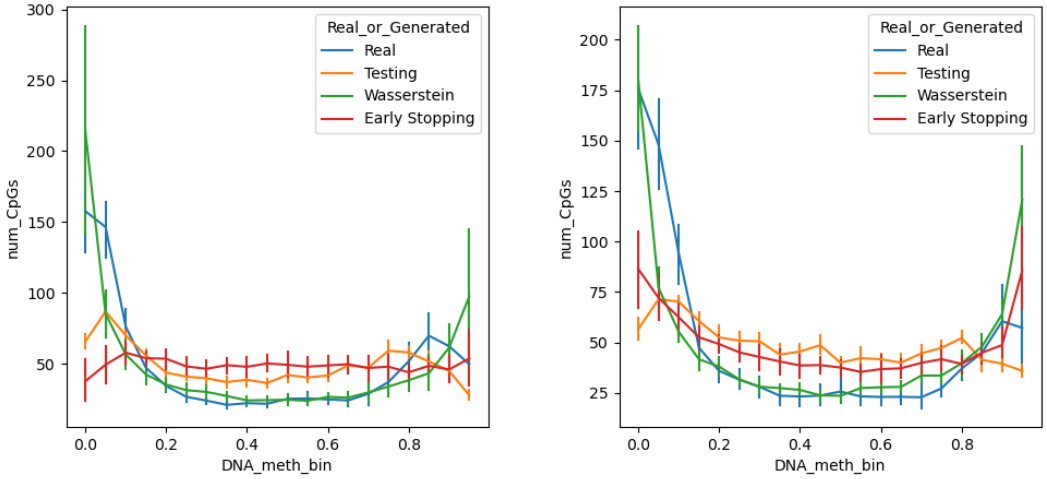
Figure 19: **Line plots for Model 1-3** (see Table 13 and section 4.3.4). The x- and y-axis describe the DNA methylation bin with respect to the number of CpGs. The blue line represents the true class. The orange, green and red lines presents the bins for the final three models (Testing, Wasserstein and Early stopping), respectively.



(a) PCA plot for cancer type "EWS"

(b) PCA plot for cancer type "EFT"

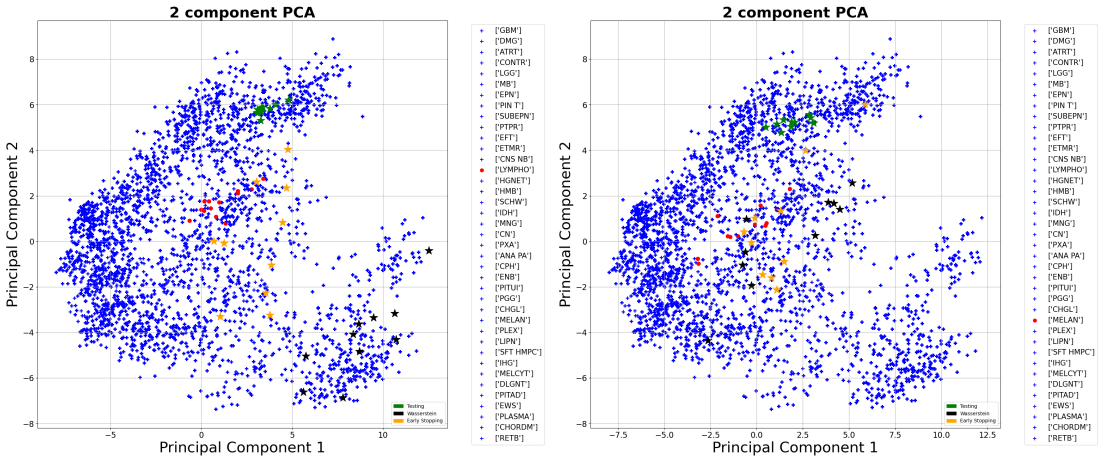
Figure 20: **PCA plots for Model 1-3** (see Table 13). The red and the blue points represent the true class and remaining classes, respectively. The green points represent the generated new points from Model 1. The black points represent the generated new points for Model 2, and the yellow points represent the generated new points for Model 3.



(a) Line plot for cancer type "EWS"

(b) Line plot for cancer type "EFT"

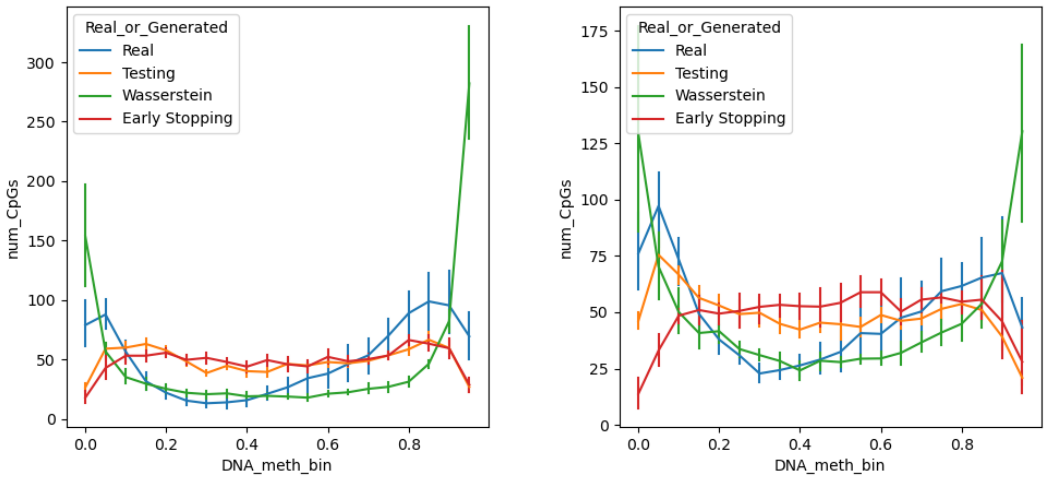
Figure 21: **Line plots for Model 1-3** (see Table 13 and section 4.3.4). The x- and y-axis describe the DNA methylation bin with respect to the number of CpGs. The blue line represents the true class. The orange, green and red lines presents the bins for the final three models (Testing, Wasserstein and Early stopping), respectively.



(a) PCA plot for cancer type "LYMPHO"

(b) PCA plot for cancer type "MELAN"

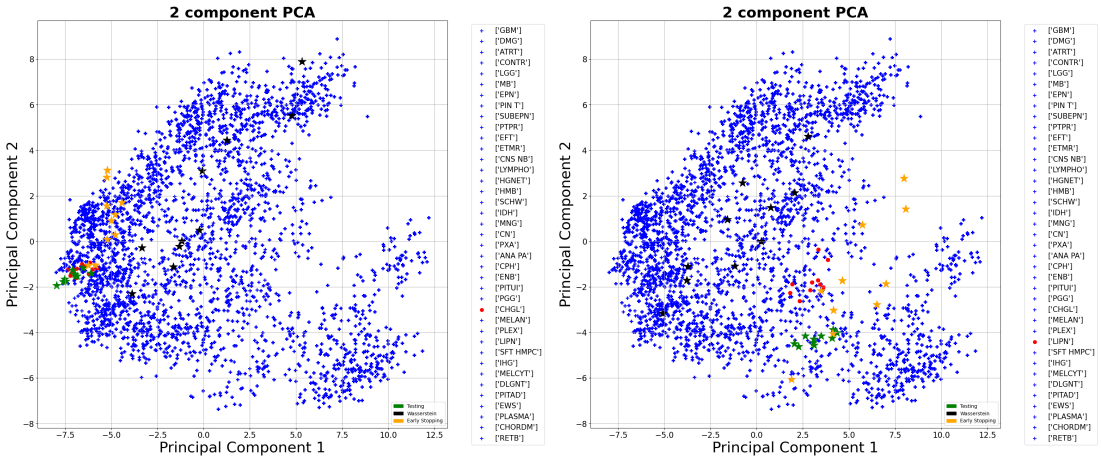
Figure 22: **PCA plots for Model 1-3** (see Table 13). The red and the blue points represent the true class and remaining classes, respectively. The green points represent the generated new points from Model 1. The black points represent the generated new points for Model 2, and the yellow points represent the generated new points for Model 3.



(a) Line plot for cancer type "LYMPHO"

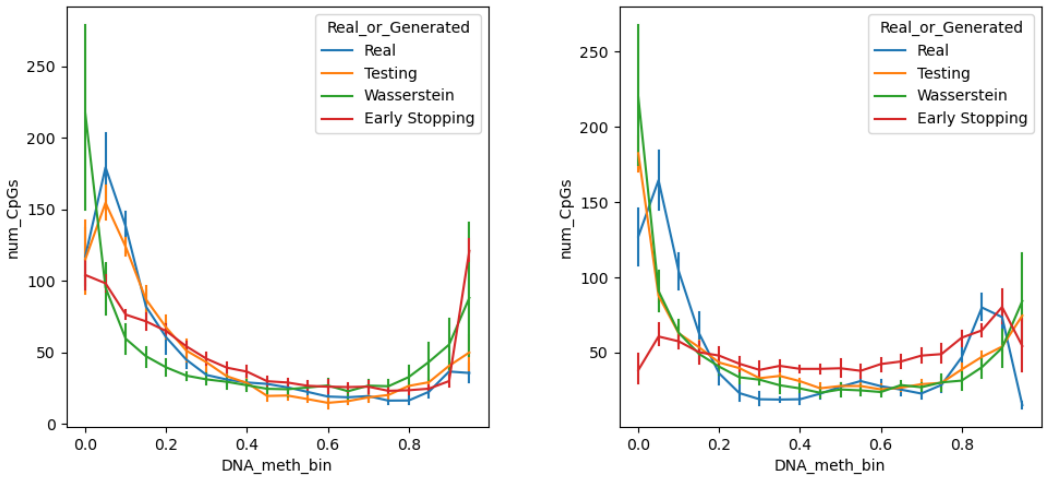
(b) Line plot for cancer type "MELAN"

Figure 23: **Line plots for Model 1-3** (see Table 13 and section 4.3.4). The x- and y-axis describe the DNA methylation bin with respect to the number of CpGs. The blue line represents the true class. The orange, green and red lines presents the bins for the final three models (Testing, Wasserstein and Early stopping), respectively.



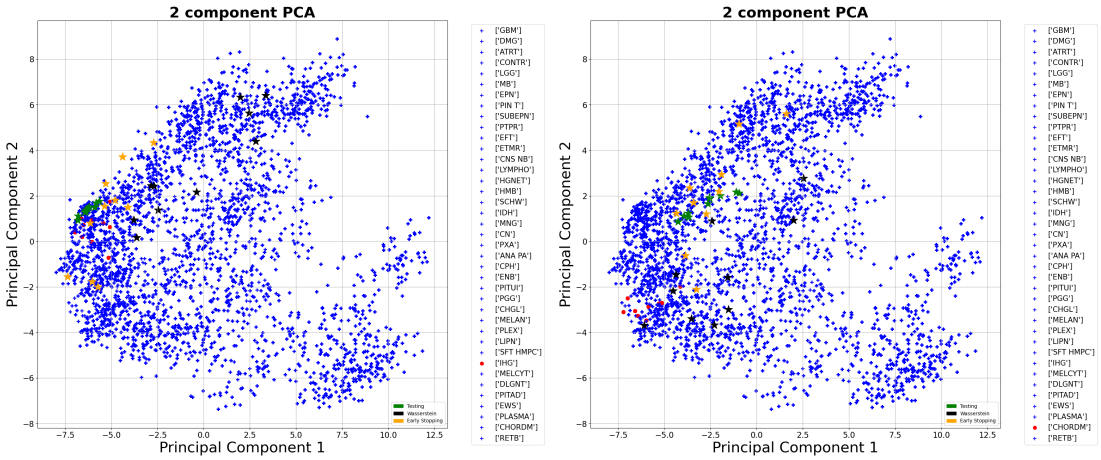
(a) PCA plot for cancer type "CHGL" (b) PCA plot for cancer type " LIPN"

Figure 24: **PCA plots for Model 1-3** (see Table 13). The red and the blue points represent the true class and remaining classes, respectively. The green points represent the generated new points from Model 1. The black points represent the generated new points for Model 2, and the yellow points represent the generated new points for Model 3.



(a) Line plot for cancer type "CHGL" (b) Line plot for cancer type "LIPN"

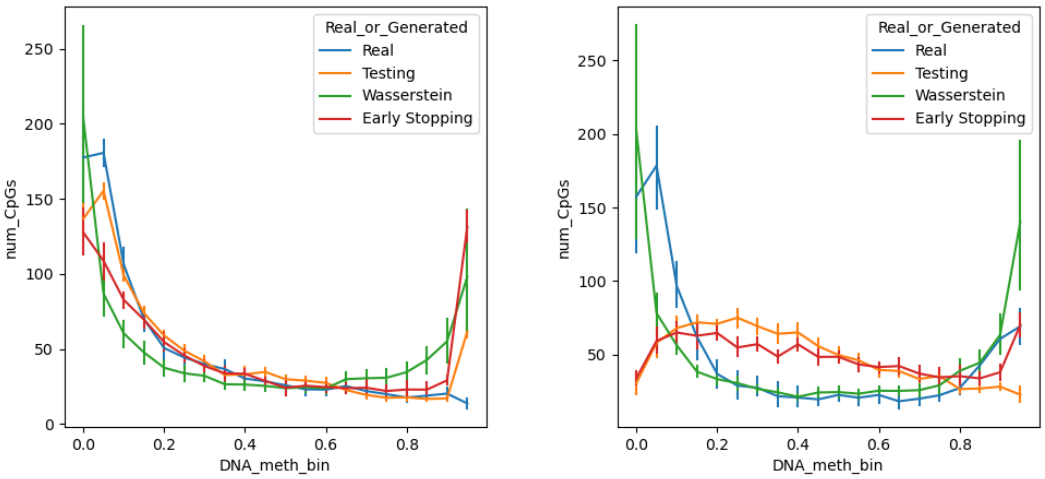
Figure 25: **Line plots for Model 1-3** (see Table 13 and section 4.3.4). The x- and y-axis describe the DNA methylation bin with respect to the number of CpGs. The blue line represents the true class. The orange, green and red lines presents the bins for the final three models (Testing, Wasserstein and Early stopping), respectively.



(a) PCA plot for cancer type "IHG"

(b) PCA plot for cancer type "CHORDM"

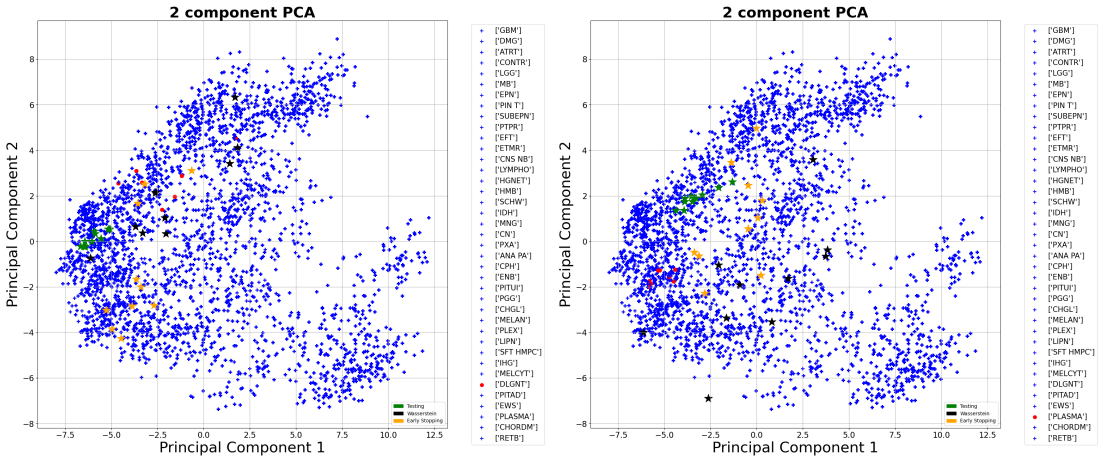
Figure 26: **PCA plots for Model 1-3** (see Table 13). The red and the blue points represent the true class and remaining classes, respectively. The green points represent the generated new points from Model 1. The black points represent the generated new points for Model 2, and the yellow points represent the generated new points for Model 3.



(a) Line plot for cancer type "IHG"

(b) Line plot for cancer type "CHORDM"

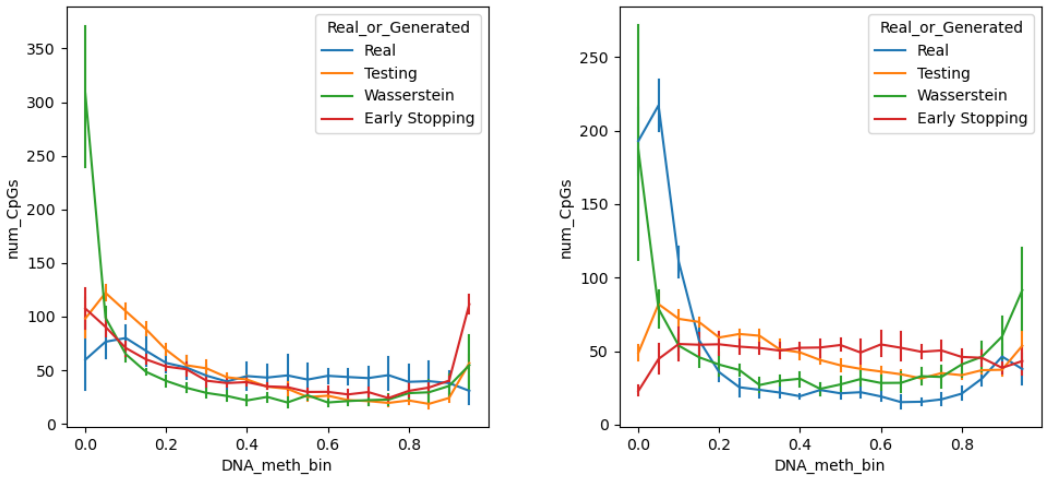
Figure 27: **Line plots for Model 1-3** (see Table 13 and section 4.3.4). The x- and y-axis describe the DNA methylation bin with respect to the number of CpGs. The blue line represents the true class. The orange, green and red lines presents the bins for the final three models (Testing, Wasserstein and Early stopping), respectively.



(a) PCA plot for cancer type "DLGNT"

(b) PCA plot for cancer type "PLASMA"

Figure 28: **PCA plots for Model 1-3** (see Table 13). The red and the blue points represent the true class and remaining classes, respectively. The green points represent the generated new points from Model 1. The black points represent the generated new points for Model 2, and the yellow points represent the generated new points for Model 3.



(a) Line plot for cancer type "DLGNT"

(b) Line plot for cancer type "PLASMA"

Figure 29: **Line plots for Model 1-3** (see Table 13 and section 4.3.4). The x- and y-axis describe the DNA methylation bin with respect to the number of CpGs. The blue line represents the true class. The orange, green and red lines presents the bins for the final three models (Testing, Wasserstein and Early stopping), respectively.