Comenius University, Bratislava
Faculty of Mathematics, Physics and Informatics

Detecting Modified Bases in MinION Data

Master Thesis

2018
Bc. Rastislav Rabatin

Comenius University, Bratislava
Faculty of Mathematics, Physics and Informatics

Detecting Modified Bases in MinION Data

Master Thesis

Study programme:	Informatics
Study field:	Informatics
Department:	Department of Computer Science
Supervisor:	doc. Mgr. Tomáš Vinař,PhD.

Bratislava, 2018
Bc. Rastislav Rabatin

28804922

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Rastislav Rabatin
Študijný program: informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: anglický
Sekundárny jazyk: slovenský

Názov: Detecting Modified Bases in MinION Data
Detekcia modifikovaných báz v dátach platformy MinION

Cieľ: MinION je sekvenovací prístroj, ktorý produkuje postupnosť elektrických signálov na základe analýzy molekúl DNA. Tieto signály možno preložiť do reťazca nad abecedou {A,C,G,T}, ktorá reprezentuje príslušnú molekulu DNA. Metódy na preklad signálu do reťazca sú netriviálnou aplikáciou metód strojového učenia, ako napríklad rekurentných neurónových sietí.

Reťazcová reprezentácia v sebe nezahŕňa množstvo modifikácií báz DNA, ktoré však spôsobujú zmeny elektrických signálov a teda ich možno z týchto signálov identifikovať. Cieľom práce je vyvinúť metódu na identifikáciu takýchto modifikácií za pomoci metód učenia bez učiteľa alebo kombináciou metód s učiteľom a bez učiteľa.

Vedúci: Mgr. Tomáš Vinař, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 14.12.2016

Dátum schválenia: 14.12.2016 prof. RNDr. Rastislav Kráľovič, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	Bc. Rastislav Rabatin
Study programme:	Computer Science (Single degree study, master II. deg., full time form)
Field of Study:	Computer Science, Informatics
Type of Thesis:	Diploma Thesis
Language of Thesis:	English
Secondary language:	Slovak

Title: Detecting Modified Bases in MinION Data

Aim: MinION is a sequencing platform that produces sequences of electrical signals from DNA molecules. These signals can then be translated into a string representation of the DNA (a string over alphabet {A,C,G,T}). Such translation is non-trivial and requires advanced machine learning methods, such as recurrent neural networks.

This string representation does not cover various base modifications, however such modification cause changes in the electrical signals produced by the device. The goal of this thesis is to develop a method to identify these modifications by unsupervised (e.g. outlier detection) or semi-supervised methods.

Supervisor:	Mgr. Tomáš Vinař, PhD.
Department:	FMFI.KAI - Department of Applied Informatics
Head of department:	prof. Ing. Igor Farkaš, Dr.
Assigned:	14.12.2016
Approved:	14.12.2016 prof. RNDr. Rastislav Kráľovič, PhD. Guarantor of Study Programme

.....
Student

.....
Supervisor

Acknowledgement

First, I would like to thank my supervisor doc. Mgr. Tomáš Vinař,PhD. for his guidance and his patience mainly during the last days.

I am also thankful to doc. Mgr. Bronislava Brejová, PhD. and Mgr. Vladimír Boža,PhD. for their helpful advice.

Special thanks to my family and friends for their support.

Abstrakt

Cieľom tejto diplomovej práce je výpočtovými metódami identifikovať modifikované DNA bázy z dát platformy MinION.

MinION je prenosné zariadenie na sekvenovanie DNA, ktoré nepožaduje DNA amplifikáciu počas pripravovania vzoriek na sekvenovanie. DNA modifikácie sú tak stále prítomné na DNA vlákne, počas sekvenovania, ktoré sa realizuje tým, že DNA vlákno prechádza cez nanopór. Modifikované bázy spôsobujú posuny v meranom signále, ktoré môžu byť neskôr identifikované pomocou výpočtových metód.

Súčasné nástroje na identifikáciu modifikovaných báz z dát platformy MinION potrebujú trénovaciu množinu, ktorá je zložená z modifikovaných a nemodifikovaných (kanonických) báz, kde pre kazdú bázu vieme, že či je modifikovaná. Je ťažké a drahé experimentálne pripraviť takéto dáta. V tejto práci na identifikáciu modifikovaných báz používame kombináciu metód s učiteľom a bez učiteľa. Natrénujeme autoenkóder na dátach bez modifikácií, aby sme sa naučili charakteristiky nemodifikovaných báz. Následne analyzujeme rekonštrukčnú chybu autoenkódera na to, aby sme identifikovali bázy, ktoré sa nezhodujú s charakteristikami, ktoré sme sa naučili.

V našej práci sme sa sústredili na DNA metyláciu, ale náš prístup môže byť použitý na identifikáciu ľubovoľných DNA modifikácií. Naše výsledky ukazujú, že z rekonštrukčnej chyby autoenkódera nevieme rozlíšiť medzi metylovanými a nemetylovanými DNA bázami len z jedného čítania. Avšak keď použijeme rekonštrukčnú chybu z viacerých čítaní, tak dostaneme celkom sľubné výsledky: pre väčšinu metylácií je desať čítaní postačujúcich na to, aby sme vedeli rozlíšiť medzi metylovanými a nemetylovanými vzorkami.

Kľúčové slová: MinION, DNA metylácia, detekcia anomálií, hlboké učenie, autoenkódery

Abstract

The goal of this master thesis is to computationally identify modified DNA bases from raw MinION data.

The MinION is a portable DNA sequencing device, which does not require DNA amplification in the sample preparation step. Consequently, DNA modifications are still present in the DNA strand, which is sequenced by passing through the nanopore. Modified bases cause shifts in the measured signal which can later be identified computationally.

Current tools for the identification of modified bases from MinION data require a labeled training set which is composed of modified and non-modified (canonical) bases. It is quite difficult and expensive to experimentally create this kind of dataset. In this thesis, we use a semi-supervised approach to this problem instead. We train an autoencoder on a dataset without modifications to learn characteristics of the non-modified bases. Then we analyze the reconstruction error of the autoencoder to identify bases that do not conform to the learnt characterization.

In our work, we have focused on DNA methylation but our approach can be used for the detection of any DNA modification. Our results show that from the reconstruction error of the autoencoders, we cannot differentiate between methylated and unmethylated DNA bases only by using a single read. However, when we aggregate reconstruction errors from multiple reads, we get a more promising result: for most of the methylations, ten reads are enough to differentiate between methylated and unmethylated samples.

Keywords: MinION, DNA methylation, anomaly detection, deep learning, autoencoders

Contents

List of Figures

List of Tables

Introduction

MinION is currently the smallest DNA sequencing device. Compared to other sequencing technologies it can produce very long reads. The sequencing is performed by transporting the DNA molecule through a nanopore and measuring the changes in the current caused by passing DNA nucleotides. We can determine the sequence of nucleotides that passed through the nanopore from the changes in the electric current, albeit with high error rate.

The sequencing using MinION does not require DNA amplification step in the library preparation process. This step is necessary for many sequencing technologies. DNA amplification removes modifications which play a quite important role in the regulation of gene expression and cell differentiation. Therefore, the MinION is a promising technology for detection of modifications in DNA. These modified bases create a different disruption to the electric current while passing through the nanopore than the canonical bases (A, C, T, G). Based on the signal deviations, we can determined which bases are modified.

One of the most important modification of DNA is methylation. Recently, there were published several approaches to the methylation detection from the MinION sequencing data. There are several ways that we can approach this problem. Some research groups used the classification approach ([MAB+17] and [SWZ+16]). This approach requires a labeled dataset for training which is a quite strong requirement on the data. It is quite hard and expensive to create a labeled dataset which contains methylation in many different contexts.

The goal of this thesis is to design a method which uses only non-modified DNA in the training phase. The algorithm should learn the characteristics of the non-modified DNA and then in the testing phase, it should decide what bases are modified.

In the first chapter, we are going to describe the MinION sequencing technology and how the DNA modifications affect the signal measured by this device. At the end of this chapter, we are going to discuss the previous approaches to the problem of methylation detection and give an overview of our approach.

In the second chapter, we are going discuss the anomaly detection problem and how it relates to our problem. We can actually think of DNA modifications as anomalies. Then we are going to describe several methods which are commonly used to solve the

anomaly detection problem and at the end of the chapter, we are going to describe the method which we are going to use in more details.

In the last chapter, we are going to present our methods, experimentally evaluate them and analyze the results from these experiments.

Chapter 1

Background and Problem Formulation

1.1 DNA

Deoxyribonucleic acid (DNA) molecule stores most of the genetic information of a living organism. This molecule is composed of two strands (**template** and **complement**) and has a helix shape. Each strand is composed of four **nucleotides** – adenine (A), cytosine (C), guanine (G) and thymine (T). Consequently, we can represent DNA as a string over the alphabet $\{A, C, T, G\}$. The process of determining the order of nucleotides from biological samples is called **DNA sequencing**. One of the main problems of DNA sequencing is that we cannot sequence the whole DNA molecule at once. Instead, the sequencer produces short substrings of the original DNA called **reads**. This thesis is mainly focused on a particular nanopore sequencing device called MinION. We are going to introduce the main principles of the nanopore sequencing in the next section.

1.2 DNA Sequencing using MinION

The MinION device is a portable DNA sequencing instrument in the size of a regular USB flash drive. This device is manufactured by Oxford Nanopore Technologies (ONT) and it is the first commercially sold nanopore sequencing device. One of the main advantages of this device is that it produces very long reads compared to other sequencing technologies at a very low price. The main drawback is that the MinION reads suffer from very high error rates ($10 - 15\%$).

The central component of the MinION is a **nanopore** (Figure 1.1) which is a small hole with an internal diameter of the order of one nanometer which is designed so that only a single strand of the DNA sequence can pass trough it. MinION uses an array of 512 nanopores so we can sequence multiple reads at the same time.

In the first step of the sequencing, a double-stranded DNA sequence is split into

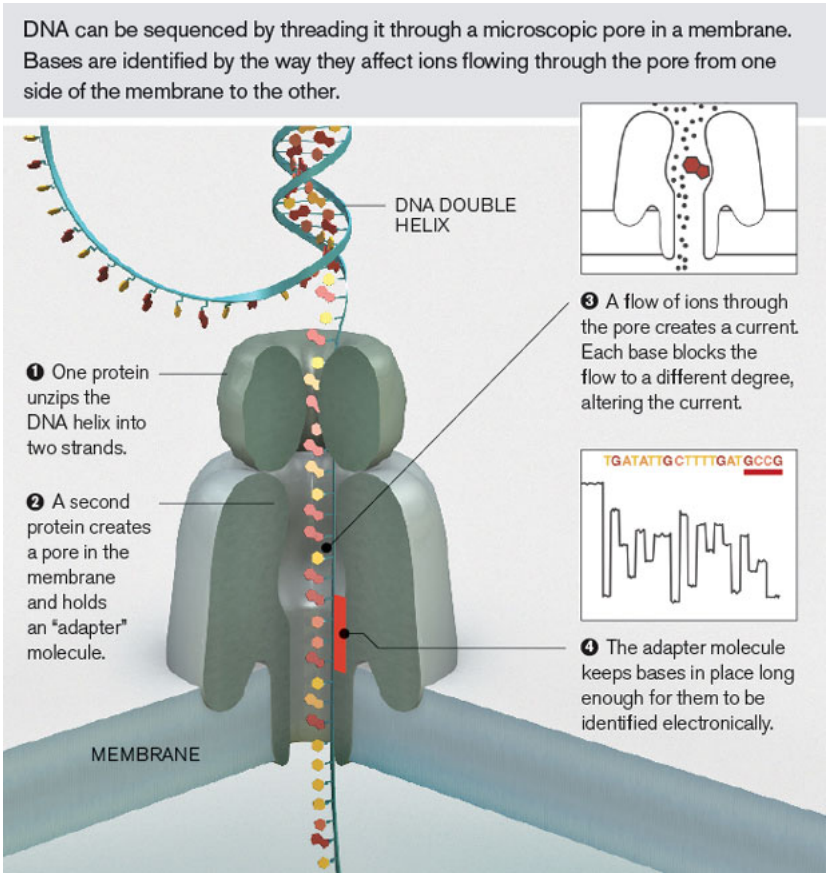


Figure 1.1: Overview of the nanopore sequencing (Source: [Sch16])

two strands which are then connected by a hairpin adapter. Special adapters (proteins) are also attached at the beginning and at the end of the whole DNA sequence in order to lead the sequence through the nanopore. When we turn on the electric field DNA is attracted towards the anode which forces the DNA sequence to travel through the nanopore. The **template** strand is driven through the nanopore first, followed by the **complement** strand. The speed at which DNA is traveling through the nanopore is controlled by the current and the adapters attached to the sequence.

When DNA sequence passes through the nanopore, it alters ionic current flowing through the nanopore. Every nucleotide causes a different disruption in the electric current. From these changes in the current we can determine the sequence which is passing through the nanopore. This process of translating the sequence of the electric current measurements into a sequence of letters A, C, T, G is called **base calling**.

Unfortunately, the situation is not so simple and the current level measured at a particular time is not influenced by only one nucleotide. It is influenced by multiple nucleotides and we do not know exactly which consecutive sequence of nucleotides influences the measurements.

The technology that we described above is called **2D nanopore sequencing** because we use both of the DNA strands. However, the most current versions of the

MinION uses only the template strand and the complement strand is discarded. This technology is called **1D sequencing**.

A Model of The Sequencing Process

We can look at the sequencer as a finite state automaton (FSA) where the nanopore is a head of the FSA that can read k letters at a time. The DNA strand that we want to read can be seen as a word on the input tape of this automaton. As the reading head slides through the input tape, it records signal measurements on the output tape. Unfortunately, the speed at which the head slides through nanopore is not completely constant and we can have errors in the process. This FSA is basically a hidden Markov model and it is quite often used for modeling nanopore sequencing data (e.g. [DDY+16]).

Event Segmentation

Event segmentation is a common pre-processing stage in the pipelines that process the data from MinION. In this stage, we split the raw signal measured by the MinION into segments where every segment should approximately correspond one nucleotide. These segments are called **events** and are usually represented by three numbers – the duration of the segment, the mean of the raw signal in that segment and the standard deviation of the signal. Some approaches even try to align these events to the reference sequence so every event corresponds to exactly one base in the reference sequence [SQE+16].

1.2.1 Squiggles

The raw signal measurements are usually called **squiggles**. See the plot of couple squiggles in Figure 1.2. This plot shows 72 squiggles which should correspond to the same sequence of bases. We would like to show couple important properties of the nanopore squiggles.

The first thing that we can notice is that the nanopore signal is basically a step signal with a quite high amount of noise. We can also notice a quite high variance of the signal per base.

The second thing that we would like to show is that the signal in one segment does not depend only on one base. For example, examine all the adenines in the plot. The mean signal level for all adenines is not the same. Also notice the transition from thymine to thymine approximately in the middle of the plot.

One property of the squiggles that we cannot see in the plot is that the segments do not have the same durations. In order to align multiple squiggles on the same sequence of bases and show it in one plot we have to either downsample the signal in every

segment or warp the squiggle in the time-domain. In this case, the second approach was used. For very squiggle, we plot a different number of points in the same segment.

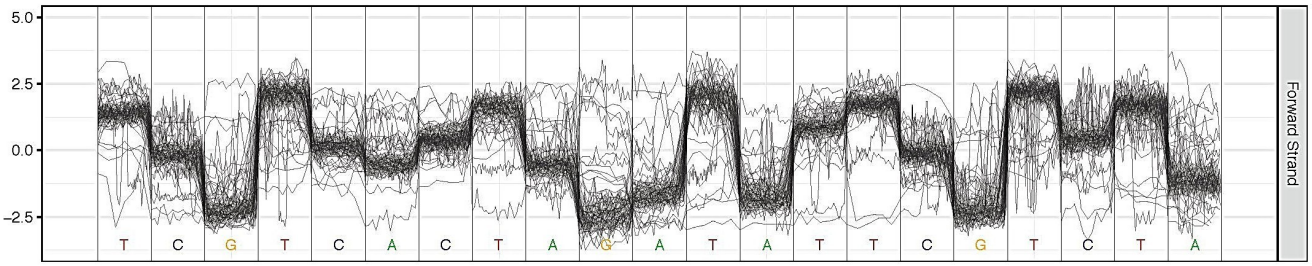


Figure 1.2: Multiple squiggles aligned on a pattern (Source: [SQE+16]). In this plot, we have 72 curves (squiggles) and every squiggle should correspond to the base sequence that is on the x-axis.

1.3 DNA Modifications

In this section, we are going to discuss different DNA modifications and explain the importance of these modifications.

1.3.1 Epigenetics

In this thesis, we are going to focus on epigenetic modifications of DNA. **Genetic** modifications are direct changes in the sequence of nucleotides. For example, cytosine changes to adenine or a full gene translates to a different location in the genome. Ont he other hand, **epigenetic** modifications are heritable changes of DNA which do not directly modify the sequence of nucleotides. They do not influence the genetic code which is stored in the cells but they influence the machinery which reads the genetic code. These changes regulate the gene expression and cell differentiation.

The most studied epigenetic modifications are methylation and histone modifications. In this thesis, we are going to focus on the methylation since the dataset that we have used for the experimental evaluation of our method is composed of several methylation modifications created synthetically in vitro. In order to evaluate our method, we need a dataset for which we know the locations and types of the modifications. In other words, we need a dataset with ground truth. To our knowledge, this dataset contains the greatest amount of different modifications among all the datasets that are publicly available.

In the rest of this thesis, we are only going to focus on methylation but we would like to emphasize that our method can be used for detection of any epigenetic modifications that create different changes of the signal in the nanopore than the **canonical bases** (A, C, T, G).

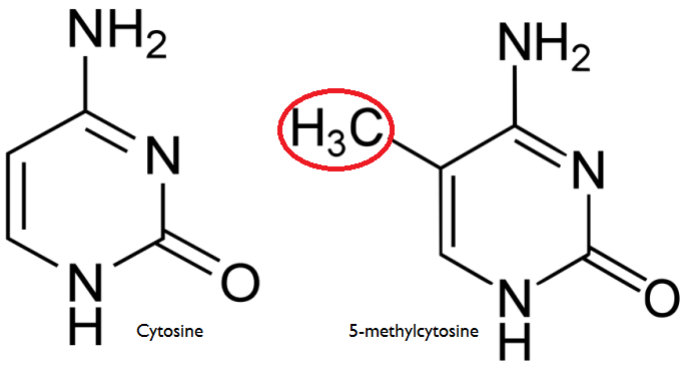


Figure 1.3: 5-methylcytosine (Source: [Wik18]). The added methyl group is in the red circle.

1.3.2 DNA Methylation

Methylation [Wik17b] is a process by which methyl group (CH_3) is added to a nucleotide. Cytosine and adenine can be methylated but thymine and guanine cannot be methylated. When we add CH_3 to cytosine we get 5-methylcytosine (5-mC) which you can see in the Figure 1.3 and when we add CH_3 to adenine we get 6-methyladenine (6-mA). Additionally, there are many other cytosine and adenine variants which can be seen in DNA. For example, by oxidation we can obtain 5-hydroxymethylcytosine (5-hmC).

Methylated bases are created by enzymes which are called methylases. There are various types of methylases which methylate different pattern of DNA. We are going to call this pattern a **methylation patterns**. For examples, the enzyme HhaI methylates GCGC patterns in DNA.

Methylation plays an important role in the regulation of gene expression. For example, it can silence tumor suppressors which protects cells against cancer. Some bacteria (e.g. Escherichia coli) even use methylation marks for a protection against viruses. Methylation plays a different role in various organisms and there is still ongoing research in this field. In this next section, we are going to introduce some of the technologies that are commonly used in these studies.

1.3.3 Technologies for Detection of Methylation

Currently, there is a couple of sequencing technologies which can detect methylation. In this section, we are going to describe in high level two technologies that compete with the nanopore sequencing technology and in the next section we are going to focus on the nanopore sequencing technology by itself.

Many sequencing technologies cannot detect DNA methylation because they create many copies of the DNA strand (DNA amplification) using **polymerase chain reaction (PCR)** which removes the methylation marks from DNA. The DNA am-

plification step is not necessary in the preparation of the nanopore sequencing library therefore we can sequence the extracted DNA directly. One approach that tries to overcome the problem with DNA amplification is bisulfate sequencing [Res17]. This technology in the first step converts unmethylated cytosine to uracil. Then we can amplify the DNA and use a short read sequencing technology (e.g. Illumina) to read the converted DNA. The main limitation of this technique is that we rely on the fact that the conversion process converted all unmethylated cytosines. Another problem is that DNA can be largely degraded during the conversion process into uracil.

The other technology that is commonly used for detection of methylation is Single molecule real time sequencing (SMRT) [Pac17]. SMRT sequencing similarly to the nanopore sequencing identifies the methylated bases directly from DNA without any conversion process. The main idea of this technology is that in the first step fluorescent dyes are attached to every base in the sequence and then the dyes are clipped off the DNA using polymerase. During the clipping, a light is emitted and based on the color of the emitted light we can determine the sequence of bases. The detection of methylation using this technology is based on the observation that the delay between two light impulses is greater in the case of methylated bases. Similarly as MinION, this sequencing technology can produce quite long reads but the main disadvantage of this technology is the cost and the size of the device compare to the MinION.

1.4 Methylation Detection using MinION

The main assumption behind the detection of any DNA modification using MinION is that the modified bases create a different disruption to the electric signal than the canonical bases. As we mentioned before, we are going to focus on the methylation. Methylation is one of the DNA modifications which creates some changes to the signal compared to the canonical bases. However, the changes are not so big as you can see in Figure 1.4. We can notice a small systematic shifts in the signal on the first and second guanine in the GCGC pattern. For TCGA pattern, we can notice a higher variance of the samples for the methylated adenine and a systematic shift on the next adenine.

As we can see in the Figure 1.4, the methylation detection from nanopore sequencing data is not so easy problem. In the rest of the section, we are going to describe different approaches to the problem that were used by different research groups and at the end of this section were are going to give a brief overview of our approach.

1.4.1 Early Approaches

The research of the detection of methylation using nanopore sequencing started a long time before the MinION was released (e.g. [WSH$^+$10]). The goal of the research

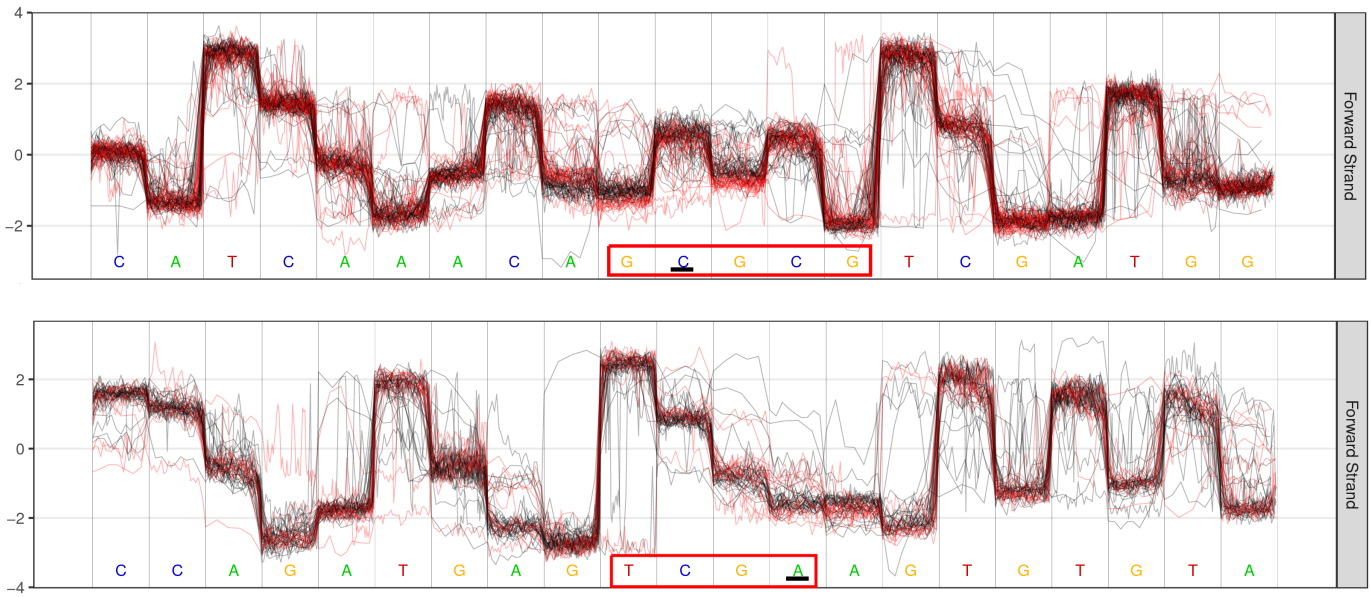


Figure 1.4: Comparison of methylated and unmethylated signal. The methylation motif is highlighted by the red box and the methylated base is underlined. The red squiggles are methylated and the black squiggles are unmethylated. In the first plot, we have 32 coverage for methylated samples and 41 coverage for unmethylated samples. In the second plot, we have 16 methylated samples and 30 unmethylated samples. These plots were created using Tombo [SQE+16] from the dataset which we are going to use in Chapter 3.

groups at that time was to prove that methylation can be detected from nanopore sequencing data. They usually used short sequence of a special form constructed only for this experiment and sequenced them many times. Afterwards, they compared the distributions of the signal for the sequences with methylation and without methylation. Most of the groups only tried to distinguish 5-mC and cytosine. However, later it was also showed that even 5-hmC can be distinguished from cytosine (e.g. [LDB+13]) using the signal from nanopore sequencing. The hypothesis that methylation can be detected from the nanopore sequencing data was proved many times.

As we mentioned in the Section 1.2, the current level measured at the nanopore is influenced by some k-mer inside the nanopore. Therefore the current measured for a cytosine variant is dependent on the context surrounding the nucleotide which means that we need measurements for the same variant in many different contexts in order to develop a general algorithm which can detect DNA modifications in arbitrary sequences. The MinION is a high throughput sequencer therefore it can quickly generate a lot of data. The release of the MinION encouraged couple research groups to try to develop a general tool which can detect methylation in an arbitrary contexts.

1.4.2 Supervised and Semi-supervised Methylation Detection

The problem of methylation detection can be formulated in many different ways. In this section, we are going to discuss several approaches and formulations of the problem that were used by other research groups. We would like to compare them mainly in terms of what kind of requirements they have on the data in the training and testing phase.

Classification

The most simple variant of the problem is a classification problem. This approach to the methylation detection was also used in nanopolish [SWZ+16] and mCaller [MAB+17]. These two groups use different models to solve this problem. Nanopolish project uses hidden Markov models (HMMs) [DEKM98] and mCaller uses multi-layer perceptron (MLP) [GBC16]. Nanopolish learns to differentiate between cytosine and 5-mC in CG contexts methylated by M.SssI and mCaller lerns to differentiate between adenine and 6-mA. Nanopolish group created their training set by synthetically methylating the samples and mCaller used SMRT sequencing to label their dataset. Both of the groups showed that these two methylated bases can be detected from MinION data.

Classification is a typical supervised machine learning problem. The training set is composed of samples methylated by a particular methylase and unmethylated samples. Additionally, we know for every sample if it is methylated or unmethylated. This is actually quite a strong requirement on the dataset. As we mentioned in Section 1.3.2,

a particular methylase methylates only contexts based on some methylation patterns. Most of the methylases only methylate a single motif. If we want to train a classification model which can detect methylation in various contexts, we have to create a dataset by using several methylases which can be quite tedious and expensive or we can use another sequencing technology to heuristically label our dataset.

Semi-supervised Classification

A more interesting formulation of this problem is **anomaly detection** or **semi-supervised classification**. We can think of methylated bases as anomalies in our data. In this formulation of the problem, the training set is composed of only unmethylated samples and the algorithm should learn some characteristics of these samples. Afterwards, the algorithm is given a random sample and the task of the algorithm is to tell which positions are methylated.

The reason why this task is more interesting is that we can use this algorithm to discover new variants of the cytosine and new contexts in which these variants of cytosine can occur. Additionally, it is easier to create an unmethylated dataset. We can do it by using PCR reaction. This approach can be even further generalized into detection of any modifications in DNA.

A very similar approach to the anomaly detection approach, which we have described, was used in SignalAlign project [RJE+16]. At first, they initialize their pair hidden Markov model on the unmethylated samples and then they finish the training on an *unlabeled* dataset which contains a mixture of methylated and unmethylated samples. To model the signal distributions they use hierarchical Dirichlet processes [TJBB05] instead of Gaussian mixture model that was used in nanopolish. The methylation patterns that they can detect are hard wired into the structure of HMM. The basic assumption of this approach of their approach is that they require that the unlabeled training set contains the methylation patterns which they want to detect.

Another approach that is similar to the anomaly detection in terms of the requirements on the dataset is statistical testing approach which was used in nanoraw [SQE+16]. This approach has basically no training phase. In the testing phase, they take an unmethylated dataset and methylated dataset and compare these two datasets using Mann-Whitney test [TM87]. In the first step, they compute the mean level current for every segment (Section 1.2) in the signal and aggregate all the reads that were aligned to a particular site in the genome. Afterwards, they use Mann-Whitney test to compare the distributions of means for a particular segment that was aligned to a particular base in the reference sequence. They use Fisher's method [Whi05] to aggregate the p-values from the statistical test for several bases surrounding the methylated base. Afterwards, they use the aggregated p-value to decide whether the

base is methylated. In the testing phase, this algorithm requires to have unmethylated dataset and a dataset which we want to test if it contains any methylations. The most problematic assumption of this approach is that both of the datasets have to be from the same genome. This approach does not test methylation on a single read but it tests methylation status of a genome site.

1.4.3 Outline of Our Approach

In this work we employ a semi-supervised approach to the problem of methylation detection. We first train a machine learning model on a dataset containing only unmethylated bases in order to learn the characteristics of the corresponding signal and then use this model to detect methylated bases in a dataset which contains a mixture of methylated and unmethylated bases. This approach can be actually used for the detection of any modifications in DNA, not only typical methylation patterns.

In the next chapter, we state the anomaly detection problem in more general terms (Section 2.1), explain the nature of the anomalies that we are interested in (Section 2.2), describe several techniques which are used in the evaluation of this problem (Section 2.3), and introduce several methods that are commonly used to solve the anomaly detection problem (Section 2.4). At the end of the chapter, we describe autoencoders, which is the method that we will later use to solve the methylation detection problem (Section 2.5).

Chapter 2

Anomaly Detection

Anomaly detection is a semi-supervised learning problem which uses a training set that contains only negative samples. As we mentioned in the previous chapter, we want to design a method for detection of modified (non-canonical) bases. There are many non-canonical bases and is it quite expensive to create a training set that contains many different DNA modifications. Additionally, not all of the DNA modifications are known and we are also interested in discovering new DNA modifications. Therefore the anomaly detection approach is more appropriate than the supervised classification approach.

In this thesis, we are going to focus on different methylations because we needed a labeled dataset in order to validate our approach that contains many different DNA modifications and methylations are DNA modifications that can be created in vitro quite easily. In the case of methylation, the normal samples are the unmethylated (PCR) samples and the abnormal samples are the methylated samples. Our learning algorithm is presented with unmethylated samples in the training phase and has to learn some characteristics of the PCR samples. In the testing phase, the algorithm is presented with a single sample and has to say if the sample is methylated or not.

There are different types of anomalies that people are trying to detect and therefore we can see different definitions of the anomaly detection task and the terminology that is used. While reading the literature, we met with very similar problems such as novelty detection, outlier detection or one-class classification. All of these problems are quite similar and there is no clear distinction between them. The terminology that is used usually actually depends on the application domain. We chose to call to our problem anomaly detection task.

Chapter Organization

At first, we are going to define more formally what is an anomaly detection problem and compare this task to the classical supervised an unsupervised learning problems

(Section 2.1). In the Section 2.2, we are going to discuss the nature of the anomalies that we are interested in. Then we are going to describe some of the common methods used for the evaluation of anomaly detection models since it is not so straight forward as in the case of classical supervised learning tasks (Section 2.3). Afterwards in the Section 2.4, we are going to introduce some of the methods that are commonly used to solve anomaly detection problems. The goal of this section is to give a general overview of these methods and not to go into too many details. At the end of this chapter (Section 2.5), we are going to focus more on autoencoders since these are the methods that we have decided to use to approach our problem.

2.1 Problem Statement

The objective of the anomaly detection problem is to find a function $f : \mathbb{R}^n \to \mathbb{R}$ where the input is a **feature vector** for a particular sample and the output is an **anomaly score**. We want to find a function which is going to return small numbers for normal samples and large numbers for abnormal (anomalous) samples and we are only allowed to use normal samples in the training phase of our algorithm. In other words, our training set is composed of only normal samples and our testing set is composed of a mixture of normal and abnormal samples. The anomaly score expresses the confidence of the classifier that that a sample belongs to the normal class. In many cases, the anomaly score is not enough and we want a definitive decision whether the sample is anomaly or not. In that case, we have to set a threshold on the anomaly score. The usual approach to do so is to use cross-validation. The choice of the threshold is directly related to the evaluation of the anomaly detection model which we are going to discuss in Section 2.3.

There are several similar terms that denote basically the same problem. Every term emphasizes different property of the problem that we are trying to solve. We can call our problem a **novelty detection** problem since we are trying to detect something novel (new or unusual [Web18]) – something that we have not seen in the data before. It can also be called **one-class classification** problem since we have only data from one class in our training set. At the testing phase, we have data from both of the classes so it is quite similar to two-class classification problem. Additionally, we can also call it **outlier detection** problem since outlier is defined in the dictionary [Web18] as "a statistical observation that is markedly different in value from the others of the sample". In our case, we want to find samples that are different in value from the samples in the training set. However, we can also talk about outliers in our training set. These are the samples that pollute our training set. More formally, these are the samples that come from the distribution of normal samples but have a quite low probability (density) to

be generated from this distribution (lie in the low density regions). We are going to use the term **outlier** to denote this kind of samples. The samples that do not come from the distribution of normal samples we refer to as **abnormal** or **anomalous**.

The most popular machine learning library – scikit-learn [PVG+11] – differentiates between two tasks – *novelty detection* and *outlier detection*. Outlier detection methods are the methods that are robust to the outliers in the training set and the novelty detection methods are the methods that do not expect outliers in the training set. The methods that are robust against the outlier try to fit the data around the central modes of the distribution of the normal samples and ignore the deviated samples. The samples far from the central modes are considered as outliers.

In some literature [CBK09], this problem is specifically called a semi-supervised anomaly detection because it is somewhere between fully supervised approach and an unsupervised approach. In the supervised anomaly detection, we have labeled data from both classes but we usually suffer from the problem that we have too little data from the abnormal class. That means not all the types of the anomalies are present in the training set. In the fully unsupervised case, we have unlabeled training set and our goal is to split the samples into normal or abnormal group. The usual assumption is that the abnormal samples are far less common than the normal samples.

2.2 Methylation as an Anomaly

In this section, we are going to discuss the nature of the anomalies that we want to detect.

We can observe short jumps that last for one or two observations in the signal from the nanopore sequencing . These jumps are just outliers which we are not interested in. It is obvious that something unusual has happened in the nanopore but when we want to detect methylation we are not really interested in these jumps. For example, the jumps might be caused by the fact that some other molecule other than DNA just passed through the nanopore and it caused some uncharacteristic disruption in the signal.

The anomalies that we are actually interested are **contextual anomalies** (sometimes called conditional anomalies). The context in this case are the bases that generated the signal. We want to be able to detect situations when the signal is consistently different for a given context then we would normally expect. A nice example of contextual anomaly from a real life [CBK09] is temperature for a location. When we have to decide if a given temperature is abnormal we have to know the location where the temperature was measured.

Unfortunately, there are several problems with the context in our case. We are

going to discuss these problems in the rest of this section.

The first problem is that base calling is a quite hard problem. Currently, the techniques based on neural networks show the best results. Even when we use neural networks to translate the signal to bases we still achieve high error rates ($10 - 15\%$). It is also hard to train these neural networks because it is quite hard to create a high quality ground truth.

As we mentioned the base calling problem is quite hard, but in the ideal case we would not only have the sequence of bases but we would also have the alignment of bases to the signal. The current state of the art neural networks approaches [THD+17] do not create a segmentation of the signal into bases. We have to create the segmentation in the postprocessing stage by a different algorithm.

The last problem that we want to mention regarding contexts is that the number of different contexts can actually grows exponentially. Let us say that the length of the context that influences whether the segment of the signal is abnormal is l, then we are actually interested in 4^l of contexts. This means that the amount of data that we have is not so important. The more important thing is that all of the contexts are sufficiently covered by the training set. The length of the context that we are interested is not completely known but the commonly used constant for MinION R9.4 is six. The high number of different contexts makes it hard to train a separate model for every context. Even though we are going to test out approach only on a small number of DNA modifications we want to design a method that can discover new modifications in the data. So we are basically interested in all of the possible contexts.

2.3 Evaluation of The Anomaly Detection Methods

As we mentioned before (Section 2.1), the goal of our anomaly task is to learn a function $f : \mathbb{R}^n \to \mathbb{R}$. Now, the question is how can we evaluate the performance of this learnt function on unseen data.

For classification problems, we usually compute metrics like accuracy, precision and recall on the testing set to evaluate the performance and when we want only a single metric for model comparison we usually use F1 score which is harmonic mean of precision and recall. We can also look at the binary classification models as functions $\mathbb{R}^n \to \mathbb{R}$. However, the difference is that they are usually trained to output numbers close to one for positive class and numbers close to zero for negative class and the threshold for determining whether the sample is positive or negative is usually 0.5. We do not have any information about the positive class in the training process of the anomaly detector so the output of the model is not a label but an anomaly score which expresses how confident is the detector that the sample is positive or negative. When

we want to get the labels from the anomaly score we have to have another dataset which we are going to use for determining the threshold on the anomaly score. The dataset has to have labels and has to contain mixture of normal and abnormal samples. This dataset is usually called development or validation set. The usual strategy is to compute anomaly score for all of the samples in the the dataset and then compute F1 score for several anomaly score thresholds and choose the one that gives the highest F1 score. We can even decide whether higher precision or recall is more important for us. The thresholds that are usually tried are the anomaly scores that were assigned to the samples in the development set. So if the dataset has n samples then we can try all of the n thresholds. This process is called **cross-validation**. In the final evaluation stage of the model, we compute the scores on another labeled dataset called a testing set and use the threshold that was chosen by the cross-validation and report the metrics for this threshold.

The problem with classification metrics is that it does not express how well is the model going to perform if we change the threshold. Therefore when we evaluate anomaly detection techniques, we also usually use other metrics that take various thresholds into account. There are two methods that are commonly used to analyse the effect of the threshold on the performance of an anomaly detector on unseen data.

The first one is called **Receiver Operating Characteristic curve** (ROC curve). In this method, we plot a curve where on the x-axis is *false positive rate* (FPR) and on the y-axis is *true positive rate* (TPR). We compute these two metrics for several anomaly score thresholds and plot these points. When we connect these points we get a curve which is our ROC curve. The curve of a random classifier is a diagonal line $y = x$ and the curve of a perfect classifier is the one that goes directly to the point $(0, 1)$ and then to $(1, 0)$. The perfect classification is $(0, 1)$. From this curve we can better decide on the anomaly score threshold since we can see how much TPR changes when we try to decrease FPR and vice versa.

The second method is called **precision-recall curve** (PR curve). In this method, we plot precision against recall. Recall is on the x-axis and precision is on the y-axis. Similarly as in the case of ROC curve, we compute these two metrics for several thresholds and connect the points to create a curve. The score of a perfect classifier is $(1, 1)$ so the curve goes from $(0, 1)$ to $(1, 1)$ and then to $(1, 0)$. The curve that corresponds to a random classifier is a horizontal line $y = \frac{P}{P+N}$ where P is the number of positive samples and N is the number of negative samples. Similarly as in the case of ROC curves, we can better decide on the trade off between precision and recall using these curves.

The main difference between PR curves and ROC curves is that ROC curves are insensitive to the imbalance in classes which we can notice from the score that is achieved by a random classification. In some literature, the authors consider ROC

curves too optimistic for the evaluation of problems which have high number of negative samples compare to positive samples. For a more detailed comparison of PR curves and ROC curves we would like to refer the reader to the research paper *The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets* [SR15]. As we are going to see in the last chapter, we had to also deal with the imbalanced classes. Most of the research papers in the anomaly detection report results from both of the evaluation methods therefore we are also going to do the same.

When we want to analyse the results for different thresholds we are also going to a look at the curves but sometimes we want to have a single metric for a comparison of several models. In that case, we compute the **Area Under the Curve** (AUC). AUC is a number between zero and one and one is the best score for both of the curves (PR and ROC curve) mentioned before. The score of a random classifier for ROC curve is always 0.5 but for PR curve this number depends on the ratio of positive and negative samples as we mentioned before.

There are also some methods for evaluation of anomaly detection methods such as Excess-Mass or Mass-Volume curves [Goi16] that do not require labeled data but they are not so commonly used and they are computationally inefficient for high dimensional data and non-linear models. They require Monte Carlo estimation of integrals which is quite inefficient for a high dimensional data.

2.4 Anomaly Detection Methods

In this section, we are going to summarize the main categories of methods that are commonly used in the anomaly detection. We use the classification of the methods into categories from *A review of novelty detection* [PCCT14]. They have splitted the methods into five categories: probabilistic methods, distance-based methods, domain-based methods, information theoretic methods and reconstruction-based methods. We mostly use the ideas from the reconstruction based methods and probabilistic methods in our work. The goal of this section is to give a general overview of the methods and ideas that they are based on. More detailed survey of these methods can be found in one of these research papers: A review of novelty detection [PCCT14], Anomaly detection: A survey [CBK09], Novelty detection: a review (part one [MS03a], part two [MS03b]).

2.4.1 Probabilistic Methods

The goal of probabilistic methods in the anomaly detection is to estimate the probability density function (PDF) of the normal samples and the samples in the low density

regions are assumed quite likely to be anomalies. These methods also offer a quite obvious method of evaluation which does not require labeled data. We can basically estimate the volume of the regions for which the anomaly score is below a given threshold. Unfortunately, the estimation of this integral is a quite hard task for most of the models and Monte-Carlo integral estimation methods are unusually inaccurate.

The methods in the category of probabilistic methods are usually split into two subcategories – parametric and non-parametric. In the rest of this subsection, we are going to introduce some of the common methods from both of these categories.

The typical representative of the non-parametric methods is kernel density estimator (Parzen-Rosenblatt method). In this method, we estimate the density around every sample using a kernel function and the final density for a point x is just an average of all densities among all the samples. More formally:

$$p(x) = \frac{1}{n} \sum_{i=1}^{n} K(||x - x_i||^2) \tag{2.1}$$

where K is a kernel function and x_i is a sample from the training set. The most commonly used kernel function is Gaussian function.

The typical representatives of the parametric methods are Gaussian Mixture Models (GMMs), Hidden Markov Models (HMMs) and naive bayes models. In fact, these methods were also used for methylation detection. The group led by Jared T. Simpson [SWZ+16] worked on the methylation classification problem and they used two HMMs – one for modeling unmethylated samples and the other for modeling samples that were methylated by a particular methylase.

Rand et. al. [RJE+16] also used HMM for methylation detection but their approach was semi-supervised and they mixed HMM with hierarchical Dirichlet processes which is a non-parametric distribution.

The most commonly used machine learning library – scikit-learn [PVG+11] implements classical GMM estimation using maximum likelihood (MLE) but additionally it also implements *elliptic envelope* method.

Elliptic envelope [RD99] is basically multivariate Gaussian model but the parameters of this model are estimated using a more robust method to the outliers in training set. The objective of this method is to find k samples from the training set that have the lowest determinant of the sample covariance. The constant k is a hyperparameter which expresses how many outlier we expect in the training set. The reason why we want to achieve the minimum determinant is that determinant directly relates to the volume of the elliptic contours of the normal distribution and we basically want contours which make a tight envelope around our samples.

2.4.2 Distance-based Methods

Techniques in this category are based on k-nearest neighbour (K-NN) method or clustering approaches.

The basic idea behind the k-NN approaches is that the normal samples have very close neighbours in the training set but the abnormal samples have their neighbour far away. For example, the simplest method could use the sum of squared distance to the k-nearest neighbours as an anomaly score. The similarity metric (distance metric) is usually another hyperparameter of these techniques.

The most commonly used method from this category is Local Outlier Factor (LOF) which is also implemented in scikit-learn. This method is based on the fact that the density around an abnormal sample is lower than the local density around a normal sample. This method compares the local density of a point to local densities of the k-nearest neighbours from the training set. Local density of a point is computed using the distances of the k-nearest neighbours. If the neighbours are close then the density is high and if the neighbour are far away then the density is low.

In the clustering based approaches, we usually use the distance to the closest representatives of any of the clusters as an anomaly score. So in the training phase, we cluster the training set and we keep the representatives of the clusters (centroids). Then in the testing phase, we compute the distances from the sample that is being tested to the centroids of the clusters and use the minimum distance as an anomaly score. The basic assumption of these methods is that the normal samples belong to one of the clusters and the abnormal samples do not belong to any of the clusters. LOF and clustering can be combined into one approach [HXD03].

2.4.3 Domain-based Methods

The methods in this category are based on **Support Vector Machines** (SVMs). The goal of these methods is to describe a boundary surrounding the points in the normal class. The points that are outside of this boundary are considered as anomalies. There are basically two main ways to apply SVMs to the anomaly detection problem.

The first one is called *One-class SVM* [SPST$^+$01]. In this method, we want to find a hyperplane with maximum margin which separates our training data from the origin. In other words, the only negative sample that we have in our dataset is the origin. In order to learn non-linear boundaries we can apply the kernel trick similarly as in the case of binary classification SVM. Since the separating hyperplane does not have to exist we can also apply the soft-margin trick from the binary SVM.

The second adaption of SVMs to the anomaly detection is called *Support Vector Data Description* (SVDD) [TD04]. In this method, we want to find the smallest hypersphere which encloses all the points from the training set. Similarly as in the One-class

SVM, we can allow to have some points outside of the hypersphere to deal with the outliers in the training set. We just add a penalty into objective function to penalize for the points outside of the hypersphere (the soft margin trick). We would also like to mention that one-class SVM and SVDD are equivalent for some of the kernels (e.g. Gaussian kernel [SPST+01]).

2.4.4 Information Theoretic Methods

The information theoretic methods are based on the assumption that the anomalies change the information content of the dataset. The common metrics of information content are entropy and Kolmogorov complexity. A basic information theoretic method would try to find a subset of k samples from the dataset that give the highest drop in entropy. This subset is then considered to be the set of the anomalies. Entropy is a measure of disorder or uncertainty. The lower the entropy is the more ordered the dataset is.

2.4.5 Reconstruction-based Methods

The last category of anomaly detection methods that we are going to discuss are reconstruction-based methods. The methods in this category are mostly based on the neural networks.

These methods learn some representation of the normal samples from the training set and in the testing phase they are presented with some sample and their goal is to reconstruct the sample – recall the most similar sample using the representation that the model has learnt. The usual choice of an anomaly score for these methods is a reconstruction error. We can meet with several methods in this category. We can split these methods into two subcategories: subspace methods and biologically motivated neural networks.

The main idea of the subspace methods is that they learn an embedding of the samples into a lower dimensional space in which the normal samples can be better distinguished from the abnormal samples. When the model compresses the data into a lower dimensional vector space then it has to prioritize what information is important that characterizes the samples in the training set. When we use this model for anomaly detection we expect that is has learnt to extract from the data some characteristics which distinguish the normal samples from the abnormal samples. Currently, the most used methods from these category are autoencoders. Our anomaly detection method is based on autoencoders therefore we are going to spend one full section on them (Section 2.5).

The biologically motivated models try to model human brain and mostly associative memory. They are mostly based on Hebbian theory [Heb49]. This theory tries to

explain how neurons interact and how new synaptic connections are created in the brain. These neural networks are not trained by backpropagation as it is usually done for deep learning models. The training algorithms are based on how biological neurons work and interact with each other.

2.5 Autoencoders

Autoencoders (AEs) are neural networks which get on the input a vector x and their task is to output the same vector x. Training a neural network that copies the input to output is not so useful therefore these AEs are usually constrained in some way to learn a more useful representation of the data. A typical way to constrain AEs is to have a layer in the middle of the neural network that is smaller than the input. See the typical structure of AEs that are constrained in this way in Figure 2.1. This type of AE is composed of two parts: **encoder** network and **decoder** network. The encoder compresses the input into lower dimensional space and the decoder decompresses (reconstructs) the input back from the **code** that was produced by the encoder. The smallest layer in the middle is usually called **bottleneck layer** and the features that are learnt by this layer are called **bottleneck features**. AE can be also seen as a dimensionality reduction technique. We learn to reduce the dimensionality to only preserve the most important features from the data. We can also think of AE as a lossy compression algorithm.

Training AEs

AEs are classical neural networks which can be trained by backpropagation [GBC16] to minimize the reconstruction error. As a reconstruction error we can use for example mean squared error or cross entropy. More formally, let us say that our network is composed of encoder $f : \mathbb{R}^n \to \mathbb{R}^k$ and decoder $g : \mathbb{R}^k \to \mathbb{R}^n$ then we can train it to minimize the mean squared reconstruction error:

$$\mathcal{L} = \sum_i ||x_i - g(f(x_i))||^2 \tag{2.2}$$

where x_i are our samples.

Applications

AEs have several use cases. One common use case is when we use the bottleneck features as an input to another classifier. In the process of lossy compression AE removes the noise from the input so the representation is robust against the noise.

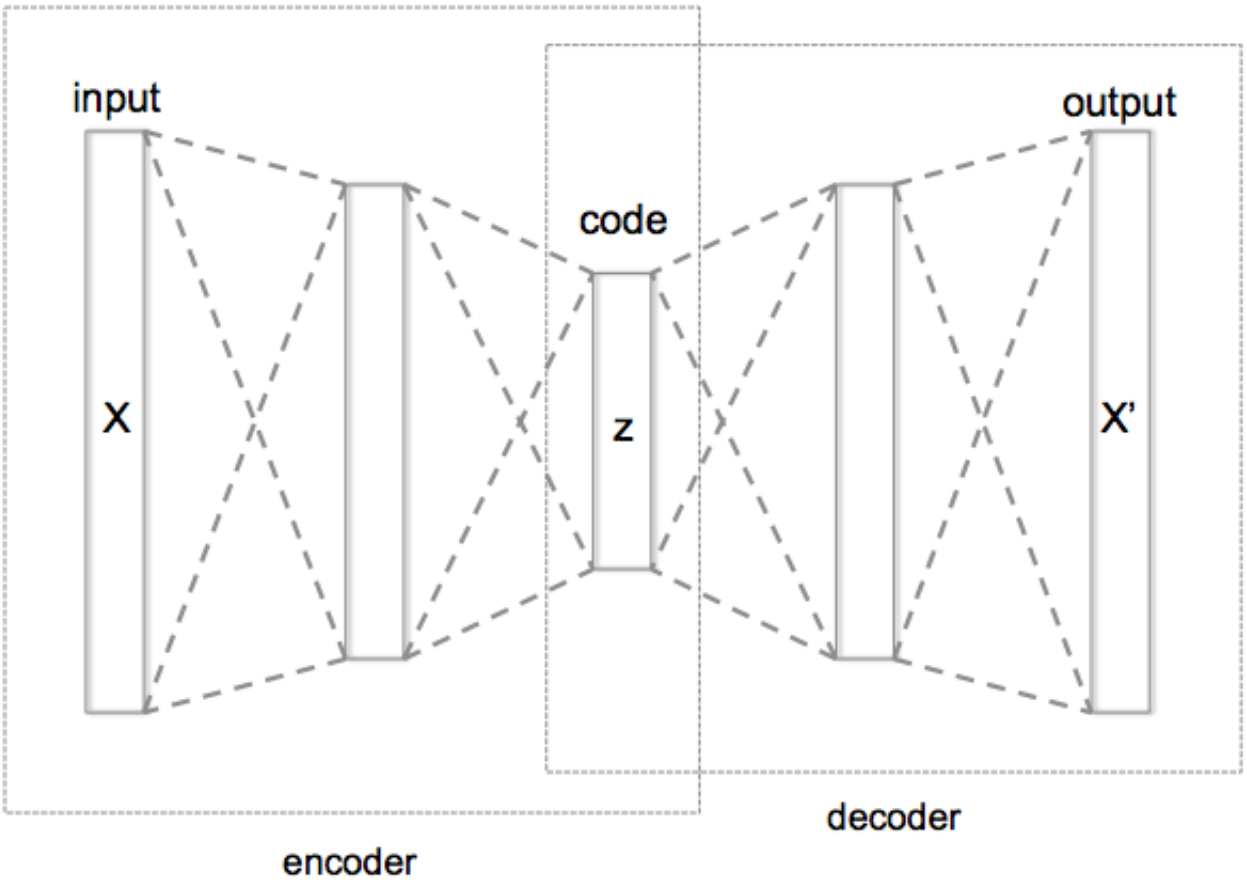


Figure 2.1: Autoencoder structure. [Wik16a]

AEs can be also applied in the field of information retrieval. We can think of the code that was produced by AE as a hash of a sample. We can hash all of our samples using AE and then search in the database of hashes for similar samples.

The most important application for us is in the domain of anomaly detection. In this case, we train AE on the normal samples and then we expect that it is going to learn some features that distinguish the normal samples from the abnormal samples. Consequently, AE will not be able to reconstruct the abnormal samples so well as it can reconstruct the normal samples.

2.5.1 Variants of AEs

In this subsection, we are going to introduce various types of AEs. They mostly differ in the way that they constrain AE to learn a better representation of the input.

Principal Component Analysis

We can think of Principal Component Analysis (PCA [GBC16]) as a special case of AE. In this case, we have a encoder $f(x) = Wx$, where W is a weight matrix and a decoder $g(x) = W^T x$. In other words, we have a linear AE and the weights for the encoder and decoder are tied. The idea of the tied weight is also used in training non-linear autoencoder [MF13]. We can think of it as another technique of constraining AE. When the linear AE is trained to minimize the mean squared error with the constraint that the rows of W are orthogonal to each other then the rows of the matrix W are going to be the subspace of the principal components [GBC16].

Denoising AE

Another interesting technique that is used to force AE to learn some useful representation is to give it a partially corrupted input to make it harder for AE to reconstruct the original sample. This way AE is forces to learn to filter out the random noise in the data and learn a more robust representation. For example, we can add Gaussian noise to the input and then AE has to learn to reconstruct the input without the noise.

Sparse AE

Sparse AE [GBC16] is an AE which is constrained to learn a sparse representation of the input. This can be achieve by L_1 or L_2 regularization on the hidden code so the loss function for a samples x is:

$$\mathcal{L} = ||x - \hat{x}||^2 + \lambda||z||_1 \tag{2.3}$$

where $\hat{x}$ is the output of AE when we give it an input x, z is hidden code for x and λ is the regularization coefficient.

Contractive AE

Contractive AE [GBC16] introduces a constraint on the derivatives of the encoder function. The loss function is then:

$$\mathcal{L} = ||x - \hat{x}||^2 + \lambda \sum_i \sum_j \left(\frac{\partial f(x_j)}{\partial x_i} \right)^2 \tag{2.4}$$

where x_i is the i-th component of the input vector. The constraint on Jacobian of the encoder basically forces the samples that are close in the input space to lie also close to each other in the bottleneck space.

Chapter 3

Methods and Experimental Evaluation

In the first chapter, we have introduced the MinION and explained how methylated bases affect the raw signal produced by the MinION. In the second chapter, we have introduced the anomaly detection problem, explained strategies that are commonly used for the evaluation of methods solving the problem, introduced several methods which are commonly used to solve this problem, and clarified how this problem relates to the methylation detection problem. At the end of the chapter, we have described in more details the autoencoders which are the main method which we are using in our work.

In the first part of this chapter, we present the dataset that we use in our experiments (Section 3.1), describe the pre-processing strategies (Section 3.2), and show how we split the reads into subgroups and into signal windows which we use as an input to the autoencoder (Section 3.3).

In the second part of this chapter, we explain the general training strategies that we have used for hyperparameter tuning and training the autoencoders (Section 3.4) and present two architectures of the neural networks (Section 3.5).

Finally, we analyze the reconstruction error of the trained autoencoders and show the results for the methylation detection problem (Section 3.6).

3.1 Dataset

In our experiments, we have used a dataset which was produced by the authors of the nanoraw paper [SQE+16]. At first, DNA was extracted from E. coli (strain K-12 MG1655) and amplified according to protocols provided by Oxford Nanopore Technologies. Then the DNA samples were split into several groups and every group was barcoded. DNA **barcoding** is a technique which adds a special genetic marker to DNA samples so we can later on differentiate between different groups of samples using this marker. Afterwards, some of the groups were methylated. This experiment was done

by two laboratories – Loman Lab and Pennacchio Lab and three sequencing libraries were created in total. Every library was sequenced by a different MinION device using 2D sequencing kit (SQK-LSK208) and R9.4 250bps flow cells. Notice the structure of the dataset in Table 3.1.

The methylation reactions were not validated by any other methylation detection technology such as bisufate sequencing or SMRT (Section 1.3.3). Authors of the dataset stipulated that not all of the methylases that they have tried worked successfully - the enzymes might have been inactive (Nicholas Loman. Personal communication. 2016).

As you can notice, some of the subsets of the data are not included in the table (e.g. dataset with identifier m03). The authors of the nanoraw paper have sequenced also other samples but we have not used them in our experiments and they have also not reported the results for them in the nanoraw paper [SQE+16]. The links to the dataset are available in Appendix A.

Library	Barcode	Identifier	Methylase	Motif	Methylation class
lib1	NB01	meth01	TaqI	TCG<u>A</u>	6mA
	NB05	control1			
	NB06	meth02	BamHI	GGAT<u>C</u>C	5mC
	NB08	meth04	EcoRI	GA<u>A</u>TTC	6mA
lib2	NB04	meth08	HhaI	G<u>C</u>GC	5mC
	NB05	meth09	MpeI	<u>C</u>G	5mc
lib3	NB02	meth10	SssI	<u>C</u>G	5mC
	NB03	meth11	dam	G<u>A</u>TC	6mA
	NB05	control2			

Table 3.1: Nanoraw dataset [SQE+16]. Every library represents one sequencing run. The column identifier contains identifiers of the subsets of the data that we are going to use in this thesis. Motif is the DNA pattern that is methylated by the given methylase. The nucletide that is underlined is the one that should be methylated. Methylation class is the new non-canonical bases that is created after the methylation reaction. The control groups are the groups which were not methylated (negative samples).

3.2 Pre-processing

After the sequencing, the reads were base called by Metrichor and stored in HDF5 files [The10]. The base called reads have a quite high error rate therefore we have decided to realign the raw signal from reads to the reference sequence. We have downloaded the reference sequence from *European Nucleotide Archive* (accession number U00096.3 version 3). Afterwards, we have used Tombo package ([SQE+16] and [Oxf18]) to realign

the signal to the reference sequence. Tombo in the first stage maps the base called reads onto the reference sequence and then it aligns the signal observations to the reference. Tombo is just a new version of nanoraw with additional features. We have configured tombo to use `BWA-MEM` [Li13] for the mapping of reads to the reference sequence. The process of realigning the signal to the reference sequence is sometimes called **resquiggling**.

Tombo supports two methods of resquiggling the signal – resquiggle with and without events. We have used the method that uses events that were produced by Metrichor. Metrichor does not work on the raw signal. At first, it segments the signal into events and then it runs the base calling algorithm on the events (Section 1.2). The method that uses events directly uses the alignment of events to the base called bases from Metrichor. When we have started with our experiments, Tombo was not released yet. Only nanoraw was available which only supported resquiggling using events. Tombo is still in development and currently also supports eventless resquiggle. The problem with eventless resquiggle is that a model for R9.4 250bps chemistry is not included in Tombo. On the other hand, the requiggle algorithm that uses events does not need any model. Currently, this chemistry is not supported by Oxford Nanopore Technologies. One of many reasons why eventless resquiggle was replaced by resquiggle using events is that the current base callers do not produce the alignment of events to the base called bases. The whole nanopore research has moved from events to a raw signal.

MinION stores the raw data in Fast5 files which is a file format build on the top of the standard HDF5 file format. The raw signal is stored in these files in the form of sequence of 16-bit unsigned integers. The signal processing pipeline that is recommended by Tombo is to use median of absolute deviations (MAD) to normalize the signal through the whole read. The other strategy could be to used is mean normalization. Median is more robust to outliers therefore it is recommended to use median normalization. The pseudocode for the median normalization algorithm is in Figure 3.1. The reason why we need to do the normalization is that the nanopores that are used currently for the sequencing are protein structures which are not so stable and they change their properties throughout the time (nanopore degradation). Therefore also the produced signal changes. Simple shifting and scaling the signal showed to give good results in many nanopore applications [SQE+16].

3.3 Splitting Reads into Subsets and Signal Windows

After this stage, we have signal observations aligned to bases in the reference sequence. The average number of bases aligned to one base is 15. Since the reads are quite long we only work with short windows of 512 signal observations. These windows should be

$shift = median(x)$
$mad = median(abs(x - shift))$
return (x - shift) / mad

Figure 3.1: Normalization of the signal using median of absolute deviations. x is a vector of 16-bit unsigned integers with represent the signal observations from a single read.

large enough because they should contain approximately 34 bases. It is a quite standard approach in the nanopore community to segment the reads into smaller windows (e.g. [SWZ+16], [MAB+17] or [THD+17]). It is not unusual when we get reads of length $10,000$ from MinION which can have around $150,000$ signal observations. Additionally, we know that a single signal observation is influenced by approximately six surrounding bases which means that when we are interested in determining whether couple bases in the middle of the window are methylated we do not need such a large signal context.

3.3.1 Training, Development and Testing Dataset

As we mentioned in Section 3.1, our data is split into groups by the methylase that was applied on the DNA samples. We also split our data into training, development and testing set. The reads that mapped onto first third of the reference sequence are in the training set, the reads that mapped onto second third are in the development set and the reads that mapped onto last third are in the testing set. We use this split in order to make sure that our models do not learn any specific characteristics of the particular part of the genome. The reads that overlapped the borders of the thirds were discarded.

Because of this data split into training, development and testing set, we have decided to merged the reads from both of the unmethylated datasets (control1 and control2) into one dataset. We have observed that control2 dataset had a slightly higher quality (longer reads and higher identity) so we did not want to bias our method by using the better or worse dataset.

3.3.2 Extraction of the Signal Windows from the Reads

As we mentioned in Section 2.2, we want to train our model only on the unmethylated reads and then test on a dataset composed of methylated and unmethylated reads. The windows for training ouf our models were constructed as follows. We took unmethylated reads from the training set and slided a window of length 512 with a stride 512. Therefore the windows that we constructed were not overlapping.

In the testing phase, we want to test how well we can detect particular methylation. Therefore for every methylation we use windows which were centered on a particular methylation motif. Additionally, we require that the windows contain only one occurrence of the pattern. We assume that the methylation reaction worked perfectly – all of the occurrences of the pattern are methylated.

3.3.3 Window Filtering

We have noticed that the resquiggle algorithm does not always work well so we have filtered out the windows which had less than 13 bases fully inside the window. By fully inside the window, we mean that all the signal observations for the base are inside the window. When we have 13 bases inside the window then the average number of observations per base is 40 which is quite a lot. For the windows which were centered on a particular methylation pattern we additionally require that the base that is in the middle of the window has at lest five bases to the right and five bases to the left that are fully inside the window. This way we make sure that the windows are reasonably centered on the methylation pattern.

3.3.4 Notation

As you can notice, our dataset is split into several subsets based on various criteria therefore we need some notation to make it simpler to refer to the various datasets. In this section, we are going to establish a notation for our datasets. A dataset can be characterized by a tuple (a, b, c) where a denotes what methylase was used on the dataset, b denotes if the dataset is training, development or testing set and c denotes the pattern on which the windows in the dataset are centered. If the dataset is methylated the a is the identifier of the methylase from the Table 3.1 and if it is unmethylated then $a = \text{neg}$ (negative). When we want to talk about unmethylated samples in general, we are sometimes going to denote them by only writing neg. To make the notation even shorter, we use the abbreviations train, dev, test to denote training, development and testing set, respectively. If the dataset was not centered on any methylation pattern then we omit the last component of the tuple to simplify the notation. If we want to emphasize that the windows are not centered on any methylation pattern then we use $c = \textit{none}$. These windows are just non-overlapping windows from the read. If the dataset was centered on some methylation motif the c is the identifier of the methylase from the Table 3.1. Instead of using the mathematical notation for the tuples we are going to use string of the form *a*-*b*-*c*. Sometimes, we are also going to use a shorter form of the methylase identifier and write mXX instead of methXX.

We would like to also introduce a notation $x_{l:r}$ to denote a subsequence $x_l, \ldots, x_r$ of the sequence x.

3.3.5 Evaluation Strategy

All of the models were trained on the training set, the analysis that we present was done on the development dataset and only the final evaluation was done on the testing set. The metrics we use are PR-AUC and ROC-AUC. The evaluation strategy for the anomaly detection problem was discussed in the Section 2.3 in more details.

3.4 Training Neural Networks

As we mentioned before, we are going to approach this problem of the anomaly detection using neural networks. In this section we are going to describe the general training and hyperparameter tuning strategies that we have used.

All of the neural networks were trained by using Adam optimizer [KB14]. The values of β_1, β_2 and ϵ were not modified during training. We have only used the values from the original Adam paper mentioned above. The weights were initialized by Glorot method [GB10] from normal distribution.

We have used different batch sizes ranging from 512 to 4096 depending on the hardware that the neural network was trained on and the size of the network. Some researchers report that large batch sizes are prone to overfitting [LXTG17]. We have not observed any problems with overfitting or underfiting when changing the batch size. On the other hand, we have observed that when we use larger batch sizes then the training is actually faster since modern GPUs operate with large number of threads and the computation is well scalable.

We have experimented with different learning rate schedules. Most of the networks were trained by this learning rate schedule:

$$\alpha_t = \frac{\alpha_0}{1 + \lambda \cdot t} \tag{3.1}$$

where α_t is a learning rate at epoch t and λ is a learning rate decay. We have also tried manual learning rate decay but it was tedious and did not yield that big improvement. All of the networks that we report here were trained by using the learning rate schedule from Equation (3.1). The decay parameter and initial learning rate were tuned by a manual grid search. The initial learning rates that we have usually tried were 0.1, 0.01, 0.005, 0.001 and 0.0005. For the learning rate decay we have usually tried these two values 0.01 and 0.005.

All of the neural networks were trained on neg-train. During training, we have monitored the loss on neg-dev dataset. We have not observed any signs of overfitting so we have not used any regularization techniques. The final model for evaluation was chosen based on the validation error. Most of the networks were trained for 2000 epochs but our analysis showed that we could train these networks for only approximately

$400 - 500$ epochs. The validation error was still decreasing even after the 500th epoch but during the offline analysis we have not observed any systematic improvement for all of the methylations. Some methylations improved a little bit but then some got a little bit worse.

3.5 Model Architectures

In this section, we are going to introduce the architectures of neural networks that we have used.

3.5.1 Autoencoder with Fully-connected Layers (DAE)

The first architecture that we have tried was a classical autoencoder with fully-connected layers. Every fully-connected layer (dense layer) compresses the dimensionality of data by one half. For example, when we have an autoencoder with a bottleneck layer of size 32 then the encoder layer sizes are 512, 256, 128, 64 and 32, respectively. See the example of DAE32 in Figure 3.2. Since we are interested only in the signal positioned in the middle of the window, for the input $x_{0:512}$, the decoder will only reconstruct the vector $x_{90:332}$. In other words, we crop 90 observations from both of the ends of the input. Every observation is affected approximately by six surrounding bases and the average number of observations per base is 15. The 90 observations correspond to approximately 6 bases at the window boundaries can help to predict bases which correspond to signal $x_{90:512-90}$, however they are influenced by the signal outside of the input window $x_{0:512}$ and consequently are not likely to be accurately predicted based only on the information inside the window. The decoder layer sizes for bottleneck of size 32 are 64, 128, 256 and 332. We have used hyperbolic tangent as an activation function for all of the layers. In our experiments, we have tried the bottleneck sizes of 16, 32 and 64. We are going to refer to this models as DAE16, DAE32 and DAE64, respectively.

3.5.2 Autoencoder with Convolutional Layers (CAE)

The second architecture that we have tried was an autoencoder with convolutional layers. We have also tried different bottleneck layer sizes for this architectures as we have done in the case of DAE. We have tried bottlenecks of size 16, 32 and 64 and we are going to denote these networks by CAE16, CAE32 and CAE64, respectively.

This architecture is also split into two parts – encoder and decoder. In the following two sections we are going to describe the architecture in more details. Also, notice the diagram of CAE32 in Figure 3.3.

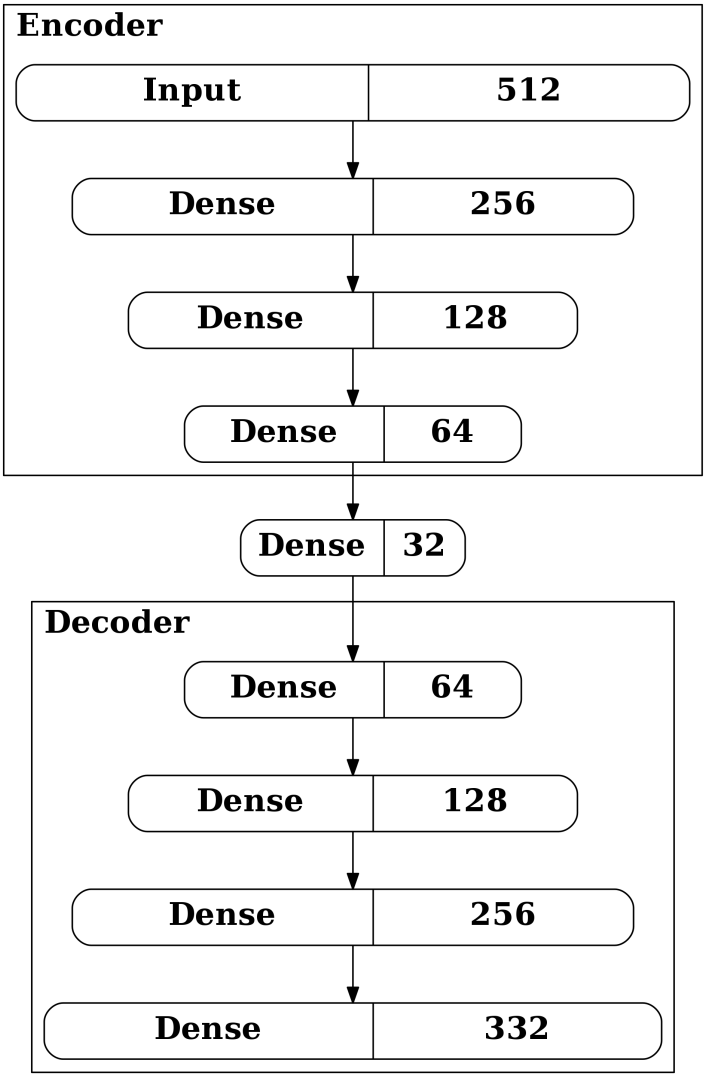


Figure 3.2: DAE32 structure. The individual components are explained in Section 3.5.1. The output dimensions from a layer are in the right part of the boxes.

Encoder

Encoder network is composed of several convolutional modules and one fully-connected layer with a hyperbolic tangent layer. The first convolutional module adds more channels to the data which is usually interpreted as a feature extraction step in the neural network: every channel corresponds to one feature. We basically run multiple convolutional filters through the signal in the time domain. Our input data has only one channel (time). This could be also viewed as a step when we increase the dimensionality of the data. Afterwards, every convolutional module compresses the dimensionality of the data by one half. Every convolutional module is composed of four layers: one dimensional convolution (Conv1D) [DV16], batch normalization layer, ELU and average pooling.

One dimensional convolution is basically the same convolution that is commonly used in image processing. The difference is that we do not slide the filter through the image, instead we slide the filter trough the time steps of the signal. This convolution is commonly used in the audio domain. The number of convolutional filters is kept constant for all the convolutional layers in the network. We have experimented with 32 and 16 filters. The results with 32 filters seemed more promising. Every convolutional module in the network has the same number of filters. Even the last one which means that the input size of the fully-connected layer is $32 \times b$ and output has b units, where b is the number of bottleneck features. The size of the convolution kernel is set to eight, stride is one and the input is padded by zeros so that the size of the output is the same as the input.

Batch normalization [IS15] is a technique which is commonly used to accelerate the training of deep neural networks. The batch normalization addresses the *Internal Covariate Shift* problem. The input of the l-th layer is influenced by all the previous layers. When the output distribution of the first layer is perturbed a little bit due to the change in parameters, the shift is amplified by other layers and at the l-th layer the distribution of the inputs changes in a much more pronounced way. Consequently, small shifts int the input distribution might cause saturation of the units and other problems with stability of the network weights. The technique is designed to prevent this problem by normalizing every feature in the batch to zero mean and variance one and then scaling the data by γ and adding β. In other words, it modifies the distibution so that the features have β mean and γ^2 variance. The parameters γ and β are learnt during the gradient descent, since batch normalization layer is fully differentiable.

In our experiments, we have observed that the neural networks that were trained with the batch normalization allowed us to use higher learning rate compared to networks without batch normalization.

ELUs [CUH15] are non-linear activation units which compared to sigmoid or hy-

perbolic tangent units, do not suffer from the saturation problems. They are similar to ReLUs. The problem with ReLUs is that they suffer from a problem which is called "dying units" problem. Since their derivative is zero for negative values, the gradient cannot be back propagated to other layers, and some parts of the networks might simply become inactive. The other problem is that ReLU does not have negative values, which causes shifts in the mean of activation distributions. ELU has non-zero output for negative values compared to ReLU, therefore it does not suffer from either of the problems mentioned above. The formula for ELU is:

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(e^x - 1) & \text{otherwise} \end{cases} \tag{3.2}$$

where α is a hyperparameter. We have used the default value $\alpha = 1$.

The last layer in the convolutional module is the **average pooling**. This layer basically slides a window trough the input and outputs the mean of every window. The window is slided in the time axis for every channel separately. We have used windows of size two. The purpose of this layer is to downsample the input and reduce the dimensionality. Another possibility to downsample the vector is to use convolution with stride two or max pooling which computes the maximum of a sliding window instead of the mean as in the average pooling. We expect that the input signal is in fact a noisy step signal and average pooling can smooth out the noise in the signal. On the other hand, max pooling can better describe sharp jumps in the signal which would correspond to the base changes in the pore. We have decided to use average pooling.

Decoder

Decoder is composed of as single fully-connected layer with hyperbolic tangent activation units and several deconvolutional modules, where every module increases the dimensionality of the input. We have tried to make the encoder as similar as possible to the encoder so the first fully connected layer has b input units and $32 \cdot b$ output units. Then this vector is reshaped into a $b \times 32$ matrix which serves as an input to the deconvolutional module. The last convolutional layer has only one filter since the output has only one channel. The output of the decoder is actually smaller than the input of the encoder due to the same reasons as in the case of DAE model. So actually the last layer of the network is a cropping layers which crops 90 observations from both of the ends of the output vector.

The structure of the deconvolutional modules is very similar to the convolutional modules. In the convolutional modules, we had convolutional, batch normalization, ELU and pooling layer. Since the purpose of deconvolutional modules is to upsample

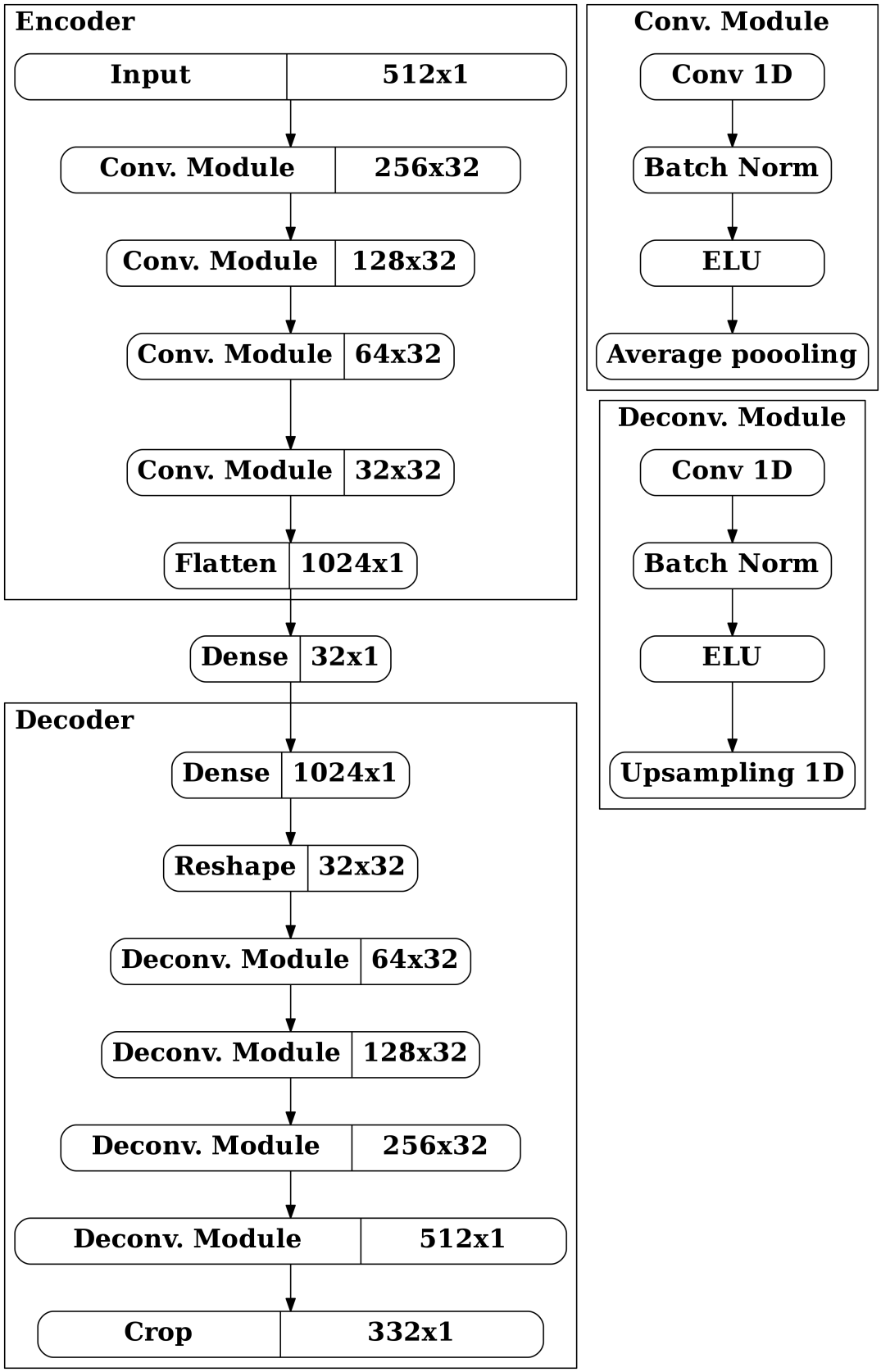


Figure 3.3: CAE32 structure. The individual components are explained in Section 3.5.2. The output dimensions from a layer are in the right part of the boxes.

the input, instead of the pooling layer we use the **upsampling layer**, which duplicates the inputs in the time dimension. All of the other layers are the same as in the convolutional modules. Another option would be to use the transposed convolution [DV16] with stride two. **Transposed convolution** is sometimes called deconvolution. Since convolution is linear operation, it can be described as a matrix multiplication of the input vector by a sparse matrix. By transposing this matrix, we achieve that the input dimensionality of the convolution is swapped with the output dimensionality. Therefore this operation increases the dimensionality. Nice illustrations of this operation can be found in *A guide to convolution arithmetic for deep learning by Dumoulin et al.* [DV16]. The other way to look at the transposed convolution with stride two is that instead of duplicating values as it is done in the case of upsampling layer with convolutional layer, it adds zeros and then convolves the input. For example, let us take the input vector $(1, 2, 3)$ then the transposed convolution would create vector $(1, 0, 2, 0, 3)$ and convolve this vector. In the case of upsampling layer with convolution, we would first upsample the vector to get $(1, 1, 2, 2, 3, 3)$ and then we would convolve it.

3.6 Model Analysis and Evaluation

In this section, we present the analysis of the reconstruction error and evaluation the performance of the models.

3.6.1 Distribution of The Reconstruction Error

Sudden Jumps in The Signal

When inspecting the signal squiggles, we have observed that the signal can unexpectedly jump up for one or two observations. Autoencoder has no way to predict such jumps and we are not really interested in these kind of anomalies in the signal. As we mentioned in Section 2.2, we are interested in contextual anomalies where the signal consistently deviates from the signal expected for a given context. To get rid of sudden jumps in the reconstruction error, we have decided to smooth the error using median filter with kernel size of five.

3.6.2 Mean Squared Error (MSE)

After training the neural networks, we have tried several techniques to compute the anomaly score from the original and reconstructed signal. The obvious choice is MSE:

$$\text{mse}(x, \hat{y}) = \frac{1}{332} \sum_{i=0}^{332} (x_{i+90} - \hat{y}_i)^2 \tag{3.3}$$

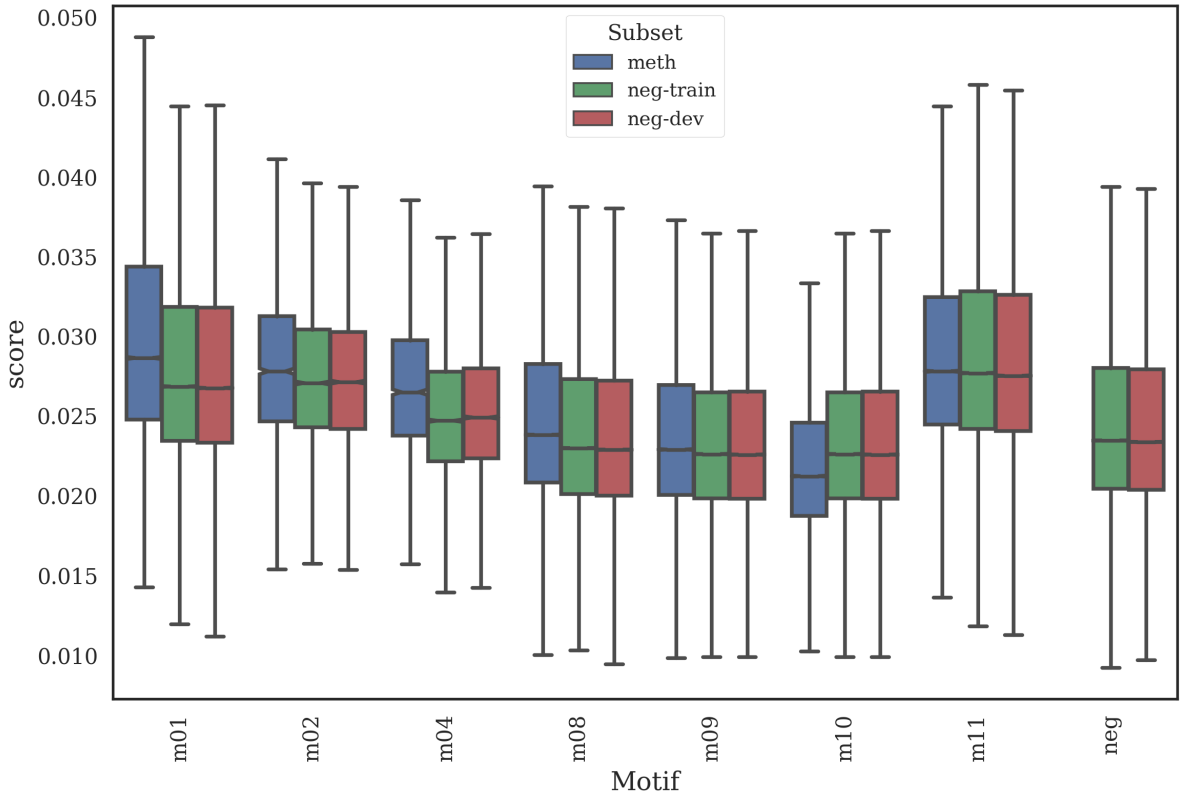


Figure 3.4: The difference in MSE for methylated and unmethylated samples for DAE64. Every boxplot corresponds to the distribution of MSE for a given subset of samples. For a methylase x we show three boxplots which correspond to: x-dev-x (blue), neg-train-x (green) and neg-dev-x (red). The last two boxplots correspond to the distributions of MSE of neg-train-none and neg-dev-none.

where x is the input signal and $\hat{y}$ is the predicted signal.

As you can see in Figure 3.4, the differences in MSE for positive and negative samples are quite small. We can usually see a statistically significant shift in median between the negative and positive samples because the amount of samples is high enough, but the difference is too small for the detection of methylated samples by using MSE on a single read by itself. There is no good threshold for the MSE score which would discriminate methylated from unmethylated samples. The biggest shift can be observed for meth01.

Another interesting property is that some contexts are much easier to reconstruct than others, which is demonstrated by large differences in MSE (e.g. compare m11 and m10). Using MSE, we can actually discriminate between different contexts better than between methylated and unmethylated samples. We can also notice that there are basically no differences between MSE in the training set and the development set, which means that the differences that we see are consistent.

Nevertheless, Figure 3.4 demonstrates that for all of the methylations we can observe higher reconstruction errors in the methylated datasets than in the unmethylated datasets, with a single exception (m10) where the autoencoder can reconstruct the methylated samples better than unmethylated.

3.6.3 Bottleneck Size

In this section we are going to look at the effect of bottleneck size on DAE and CEA separately.

DAE

The comparison of the fully-connected autoencoders for various number of bottleneck features is in Table 3.2. As an anomaly score, we have used MSE. We can notice in this table that DAE64 is the best except for meth10 and meth11. However, all of the autoencoders perform quite poorly on these two methylations. We basically achieve the score of a random classifier. Overall, all of the ROC-AUC scores are quite close to 0.5. The only methylation that has ROC-AUC above 0.6 is meth04.

Methylase	pattern	#pos	#neg	ROC-AUC			PR-AUC		
				16	32	64	16	32	64
meth01	TCGA	52748	254856	0.541	0.558	0.579	0.107	0.114	0.120
meth02	GGATCC	33315	9533	0.480	0.512	0.541	0.141	0.155	0.170
meth04	GAATTC	2951	11342	0.548	0.560	0.610	0.134	0.136	0.155
meth08	GCGC	206150	441005	0.512	0.525	0.547	0.193	0.199	0.213
meth09	CG	145297	377933	0.472	0.484	0.518	0.153	0.159	0.174
meth10	CG	70815	377933	0.431	0.422	0.415	0.073	0.071	0.071
meth11	GATC	89049	293111	0.509	0.509	0.507	0.132	0.131	0.130

Table 3.2: The effect of the bottleneck size on DAE.

CAE

The comparison of the different bottleneck sizes for CAE is in Table 3.3. The anomaly score that was used is MSE. In the case of autoencoder with fully-connected layers, we had a clear winner. In this case it is not so clear. CAE16 is definitely the worst. From the table, it looks like that it works better in `meth10` but we actually could take negative MSE as an anomaly score and then all of the ROC curver would flip around $y = x$ and we would get auc $= 1 -$ old_auc. So actually the best model for this methylation is the one with bottleneck 64.

Methylase	pattern	#pos	#neg	ROC-AUC			PR-AUC		
				16	32	64	16	32	64
meth01	TCGA	52748	254856	0.571	0.590	0.593	0.118	0.125	0.122
meth02	GGATCC	3315	9533	0.515	0.582	0.580	0.153	0.188	0.190
meth04	GAATTC	2951	11342	0.594	0.575	0.632	0.154	0.143	0.163
meth08	GCGC	206150	441005	0.539	0.563	0.556	0.206	0.217	0.216
meth09	CG	145297	377933	0.482	0.500	0.533	0.157	0.165	0.179
meth10	CG	70815	377933	0.449	0.446	0.423	0.076	0.075	0.072
meth11	GATC	89049	293111	0.516	0.525	0.517	0.133	0.136	0.132

Table 3.3: The effect of bottleneck size on convolutional autoencoder.

3.6.4 MSE of CAE and DAE

In this section, we are going to compare DAE with CAE.

At first, we are going to compare MSE on neg-dev. The results are in Table 3.4. You can notice that the reconstruction error of CAEs is always lower. We could say that when the error is around 0.2 or greater then the autoencoder does not reconstruct the signal very well. But on the other hand, when we try to achieve the reconstruction error below 0.05, the autoencoder might converge to the state when it is just passing the signal from the input to the output and does not learn any useful representation of the input.

Bottleneck size	DAE	CAE
16	0.399	0.312
32	0.204	0.130
64	0.086	0.051

Table 3.4: Comparison of MSE for DAE and CAE on neg-dev.

When we look at the reconstruction errors for individual positions in the output window of the autoencoder we can observe only really small shifts in the error distributions. Figure 3.5 shows that the greatest shifts in the distributions between methylated and unmethylated samples are in the segment $150 - 200$ of our signal windows. On the right hand side of the figure, we can see that the differences between training and validation set samples for a particular methylation. The differences are barely noticeable which means that the deviations from unmethylated samples for the given methylation on the left hand side are not random. They consistently appear for the given methylation motif. Also note that the error distributions for positions are shifted towards

larger values since median is below the mean.

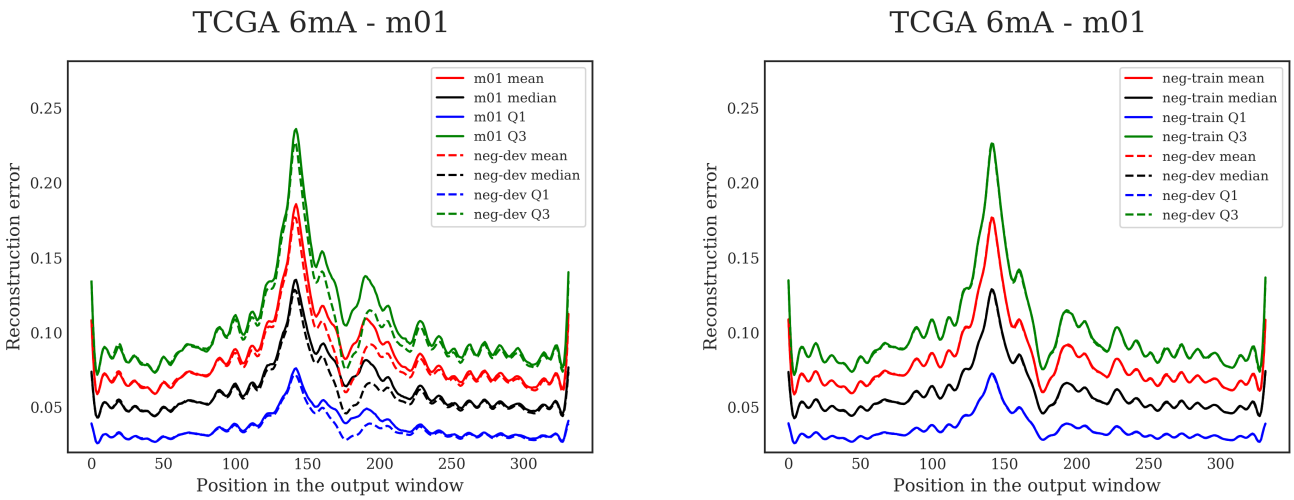
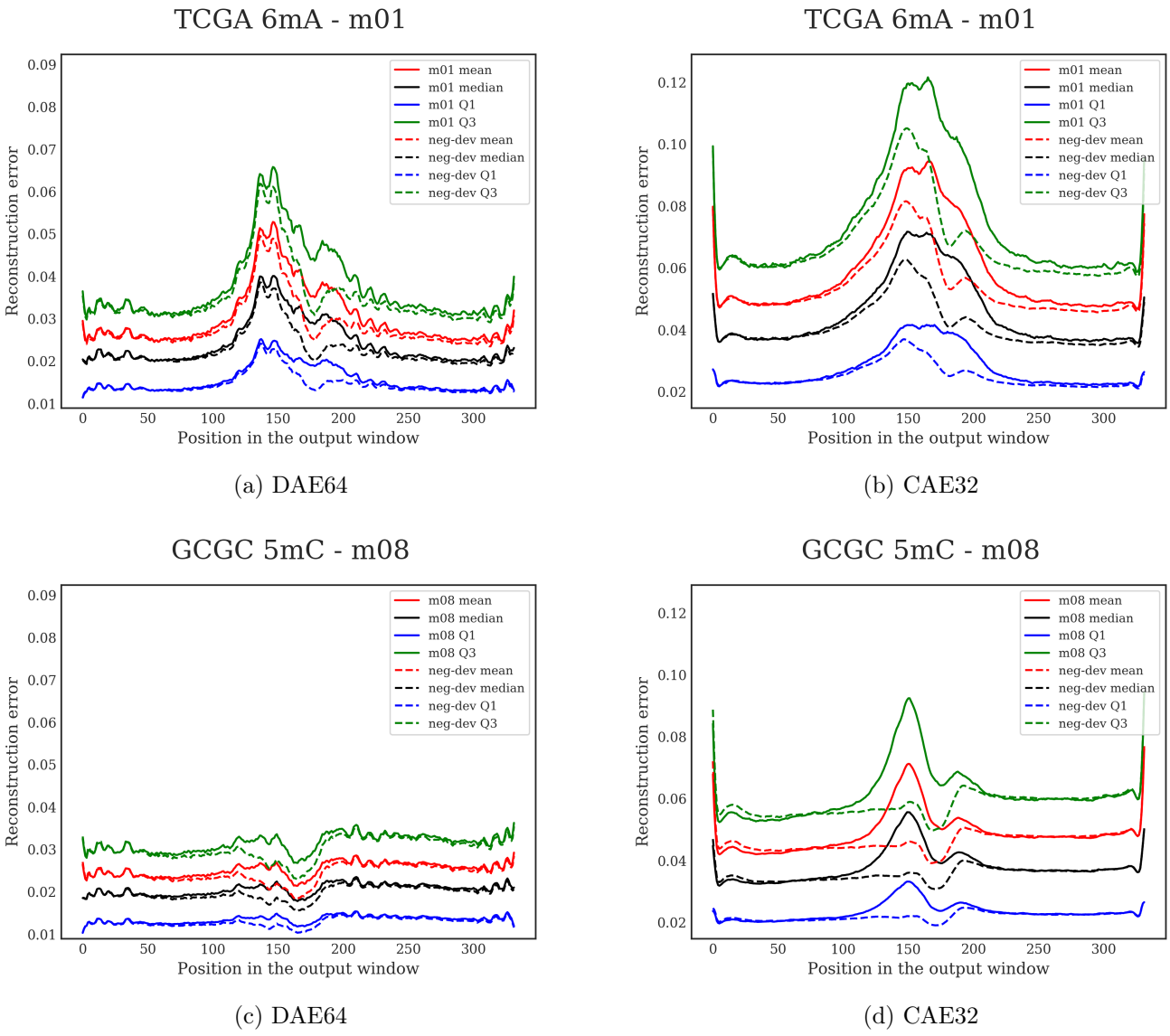


Figure 3.5: Reconstruction error per position. We call these plots also **error profiles** for a particular methylation pattern. X-axis corresponds to individual positions in the output window. Y-axis corresponds to reconstruction error. For every position, we have aggregated the reconstruction errors and computed several statistics. Dashed lines correspond to unmethylated samples and solid lines correspond to methylated samples. Every color corresponds to a different statistic which we have used for aggregation of reconstruction errors (residuals) for a particular position. Notice the legend with description of the statistics. On the left hand side, we have samples that are centered on a methylation pattern `TCGA` corresponding to `meth01` and on the right hand side we have samples from training and development datasets also centered on the same motif. This plot can be viewed as a sequence of boxplots. Every boxplot corresponds to one position in the output window. This error profile corresponds to fully-connected network with bottleneck of size 32.

Figure 3.6 compares the profile of the reconstruction error for CAE32 and DAE64 on the example of two methylases. The difference between methylated and unmethylated reconstruction error for CAE is greater than for DAE in the case of both of the methylations. The profile of the reconstruction error for CAE is smoother, because CAE can model sharp transitions in the signal better than DAE (see Figure 3.7). Increasing the bottleneck size of DAE leads to learning a random noise in the signal.

Even though CAE shows larger differences between methylated and unmethylated samples this is still not sufficient to use it for classification. Notice the first (Q_1) and third (Q_3) quartile of the reconstruction error in the bottom right corner of Figure 3.6. The intervals $[Q_1, Q_3]$ – also called interquartile ranges (IQRs) – for methylated and unmethylated samples have significant overlaps even for a methylation pattern which has the highest difference from the unmethylated samples. From a good anomaly score,



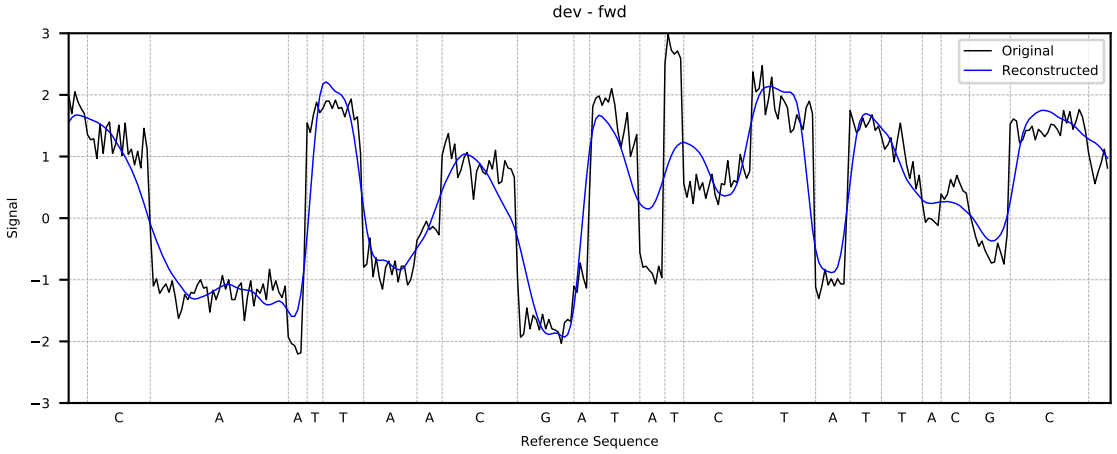
(a) DAE64

(b) CAE32

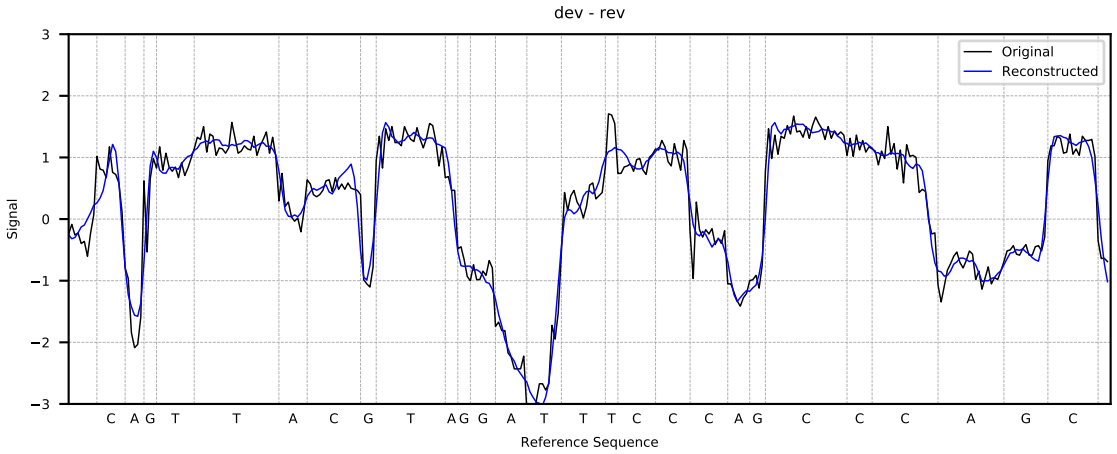
(c) DAE64

(d) CAE32

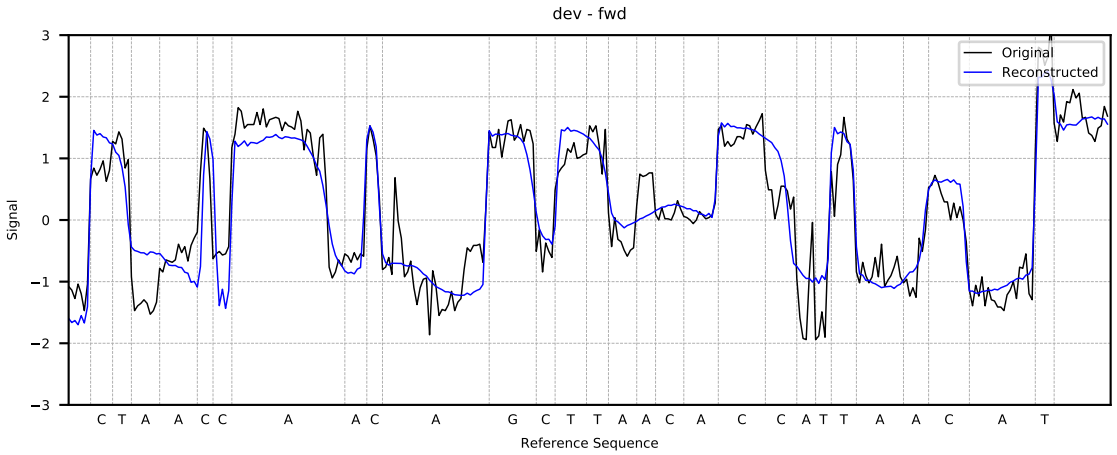
Figure 3.6: Errors profiles between CAE and DAE. Notice that the error scales are not same.



(a) DAE32



(b) DAE64



(c) CAE32

Figure 3.7: Comparison of signals reconstructed by different autoencoders. X-axis corresponds to time and y-axis corresponds to the signal level. Bases of the reference sequence are aligned to the signal (see x-axis and the vertical grid lines). The blue squiggle represents the reconstructed signal and the black squiggle represents the original signal.

we would expect that IQRs of unmethylated and methylated samples do not overlap.

3.6.5 Supervised Classification Based on a Single Read

To ascertain the discrimination power of the reconstruction error of CAE, we have decided to use this score in a simple supervised setup. We have chosen a simple probabilistic model – a multivariate normal distribution with the diagonal covariance matrix – and trained this distribution on methylated and unmethylated samples separately. These distributions were trained on a separate training set and separately for every methylation. The classification score is logarithm of likelihood ratio.

Obviously, we could use more complicated classification models such as SVM, decision tree, or even a neural network. However, the purpose of this experiment was not to find the best classification model, but to evaluate if the comparison of unmethylated and methylated error profiles can give useful results. In the previous experiments, methylation detection using MSE did not yield good results. We can view results in this section as an upper bound on what can be achieved using autoencoder error profiles for the anomaly detection task.

The AUC scores for the classification task are shown in Table 3.5. Notice that CAEs perform better than DAEs in general. Both of the CAEs perform similarly well. CAE32 is the best on `GGATCC` and `GCGC`. CAE64 is the best on `GAATTC`. We do not have clear winners for the other methylations. The differences are quite small and not consistent among ROC-AUC and PR-AUC. For example, for `TCGA`, the best model according to ROC-AUC is CAE64 but according to PR-AUC the best one is actually CAE32.

The interesting observation in the case of `GAATTC` methylation is that none of the models have smooth error profiles, even though CAEs usually have smooth error profiles. However, this might be caused by the fact that for this motif we have the least number of samples, perhaps this indicates overfitting. We can also observe the roughness of the curves on `GGATCC` but it is not as noticeable as in the case of `GAATTC`. We would like to remind that Gaussian models were fitted on different dataset than dataset for which we computed ROC curves and the error profile curves.

AUC score for motif `CG` is quite close to 0.5 for all of the models which means that none of the models can detect it well. The easiest to detect is `GAATTC` motif, but `GCGC` also works quite well.

In general, longer motifs seem be reconstructed better. To examine this in more detail, we have fitted Gaussian model to samples which had some particular k-mer in the middle. We have chosen $k = 6$, since the longest methylation pattern is of length six. Thus, our model of error is an ensemble of Gaussian models, and every Gaussian corresponds to one k-mer. Unfortunately, the results for this ensemble of models were

Methylase		meth01	meth02	meth04	meth08	meth09	meth10	meth11
	pattern	TCGA	GGATCC	GAATTC	GCGC	CG	CG	GATC
#pos		52748	3315	2951	206150	145297	70815	89049
#neg		254856	9533	11342	441005	377933	377933	293111
ROC-AUC	CAE 32	0.65	0.622	0.738	0.718	0.528	0.576	0.605
	CAE 64	0.654	0.575	0.776	0.604	0.53	0.575	0.606
	DAE 32	0.61	0.509	0.65	0.645	0.525	0.577	0.553
	DAE 64	0.629	0.556	0.724	0.622	0.518	0.579	0.562
PR-AUC	CAE 32	0.28	0.35	0.408	0.547	0.297	0.2	0.322
	CAE 64	0.258	0.315	0.457	0.405	0.295	0.198	0.328
	DAE 32	0.24	0.263	0.321	0.466	0.296	0.197	0.269
	DAE 64	0.248	0.3	0.4	0.427	0.289	0.199	0.28

Table 3.5: AUC scores for the supervised classification task based on a single read.

not better. Some contexts could be better reconstructed better and some could not. For comparison, see ROC curves for CAE with bottleneck 32 in Figure 3.8. The AUC score improved just a little bit for `CG` motifs. In order to do a fair comparison of these two approaches, we had to filter out samples from validation set that had k-mer in the middle for which we were not able to estimate a model from the training set. In other words, it did not have at least one unmethylated and one methylated sample.

3.6.6 Supervised Classification Using Multiple Reads

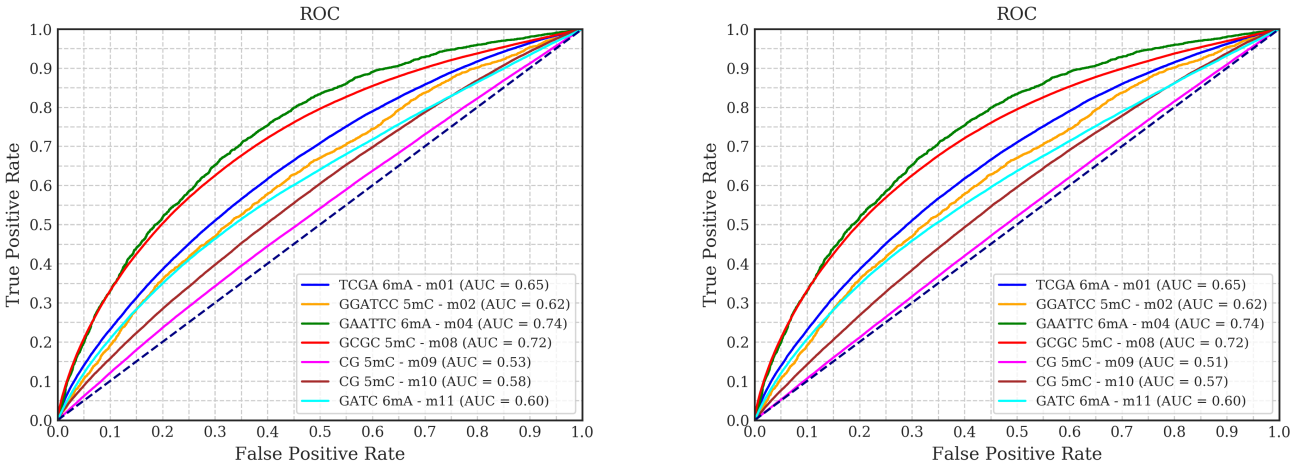
The classification of of methylation from a single read is a very difficult task and it is not clear that a single read contains enough information to perform the task. However, assuming that a single position in a genome is methylated consistently in all read, we can pool the information from multiple reads to achieve better classification results.

To compute the classification score of a single site in the reference sequence, we take a set of samples which were centered on the same motif, have the same methylation status and were aligned to the same position on the genome. For the classification, we simply sum the log-likelihood ratio scores. In other words, we assume that the reads are independent samples from that particular position in the genome.

We expect that when we sum the scores from multiple reads overlapping the same context then the score is going to be more discriminative when the signal is consistent.

Definition 1. *Site has a* ***coverage*** *x in a particular dataset if the dataset contains exactly x reads centered on the site in the reference.*

We have decided to inspect the effect of site coverage on this score. In our dataset,



(a) One Model per Motif

(b) Ensemble of Models

Figure 3.8: Comparison of ensemble of k-mer models and one model per motif. ROC curves on the left hand side correspond to one Gaussian per motif and the plot on the right hand side corresponds to ensemble of Gaussians.

every methylation has a different number of reads and a different number of occurrences in the genome. Therefore the site coverage varies a lot.

The experiment is designed as follows. First we filter out sites with coverage lower than c and then randomly sample c reads overlapping every genome site where a particular methylation pattern occurs. Figure 3.9 shows the results for different coverages. We have chosen ROC-AUC as a metric since PR-AUC is very sensitive to imbalance of positive and negative samples. Increasing the coverage significantly decreases the number of positive samples (Figure 3.9), but the number of negative samples does not decrease that much. Consequently, the PR-AUC plots against coverage mainly show the change in then ratio of positive samples to negative samples.

CAE32 performs the best for most of the methylations. CAE64 does better on meth04, but the estimate of the AUC is quite questionable for this methylation since the number of positive samples drops quickly below 100. All of the models perform poorly on CG methylation in meth10 dataset. The error profile of meth10 (Figure 3.10), shows that the reconstruction error for the methylated samples is actually consistently lower than for the unmethylated samples on the window boundaries but in the center, the reconstruction error of methylated reads surpasses the unmethylated reads, as would be expected. Normally, we would expect that the reconstruction error is greater for methylated samples since the autoencoder was trained only on unmethylated samples. On the other hand, we do not see this strange behaviour of error profile for `meth09` (Figure 3.10). We can notice that the error jumps a little bit in the middle of the window. The differences on the sides of the window are barely noticeable. `Meth09` and

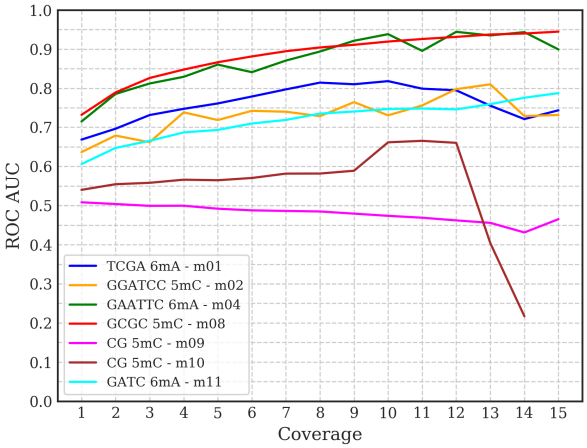
`Meth10` are methylations of the same type. Although, they were created by a different methyltransferase. Both of the methylases should bind additional CH_3 to every cytosine in `CG` context. The authors of the original paper which collected the data mention that these two methyltranferases were obtained from two different companies. Most of the methyltransferases were obtained from *New England Biolabs* but `meth10` and `meth01` were obtained from *Chrometra, Belgium*.

3.6.7 Semi-supervised Classification

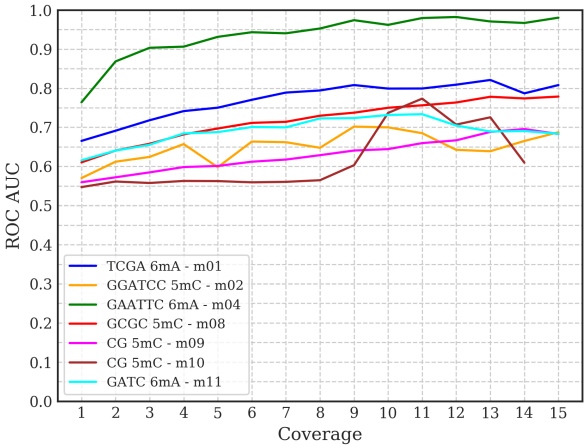
In this section, we are going to describe our approach to detection of methylated samples based on the error profiles. One obvious strategy is to compute mean square error through the whole output window and use this score as an anomaly detection. We have also tried several other strategies to define an anomaly score based on error profile. We are also going to present the results for both of the tasks – detection of methylated samples and detection of methylated genome sites.

Quite common strategy in anomaly detection is to train a probabilistic model on normal data and then use probability density as an anomaly score. In this case, we expect that anomalies are going to get low density compare to normal distributions. We discuss these methods in more details in Section 2.4.1. The goal of the probability distribution in our case is to describe error profile of a particular methylation. Notice that so far we did not given the autoencoder any information about the context of the anomaly. Fitting a distribution to an error profile of a particular methylation is a way to provide some information about the context. The advantage is that we do not need to have a good alignment of signal to bases in order to fit these distributions. We just need some samples that are approximately centered on the methylation motif. The disadvantage is that we still have the problem that some of the bases might be stuck in the nanopore for a longer time and the length of the segment in which the error profile is deviated from the normal error profile might be different for various samples which probably contributes to the noise in error distribution for a particular positions in the output window.

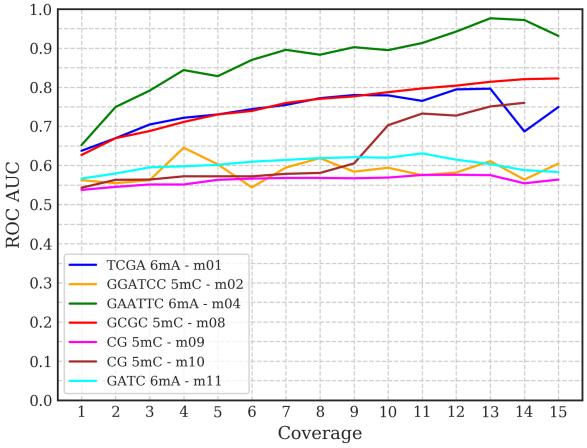
In our experiments, we have tried to fit different probability distributions to reconstruction errors for unmethylated data for a specific methylation and then we have used logarithm of density as an anomaly score. As we could notice from the previous plots of error profiles, the error distributions for different positions in the output window are usually right-skewed. Therefore, we have also tried several skewed distributions. We have tried two versions of multivariate normal distribution – with full covariance matrix and with diagonal covariance matrix. The version with diagonal matrix gave better results probably because it needs to estimate less parameters and is less prone to overfitting.



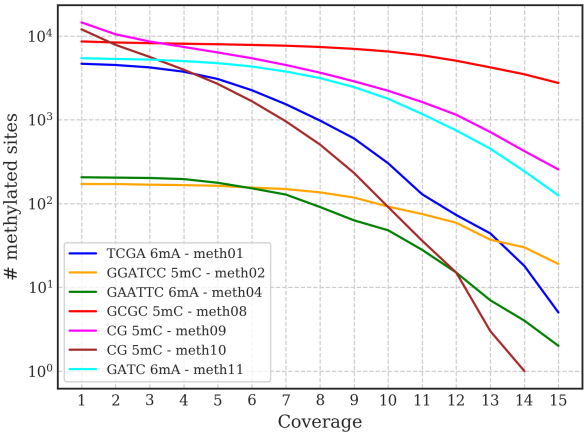
(a) CAE32



(b) CAE64



(c) DAE64



(d) Decrease of the number of methylated sites

Figure 3.9: Supervised classification using multiple reads. The plot in the lower right corner shows how quickly the number of methylated sites decreases as we increase the coverage. Notice that y-axis is in log-scale. All the other plots correspond to AUC scores of the individual models.

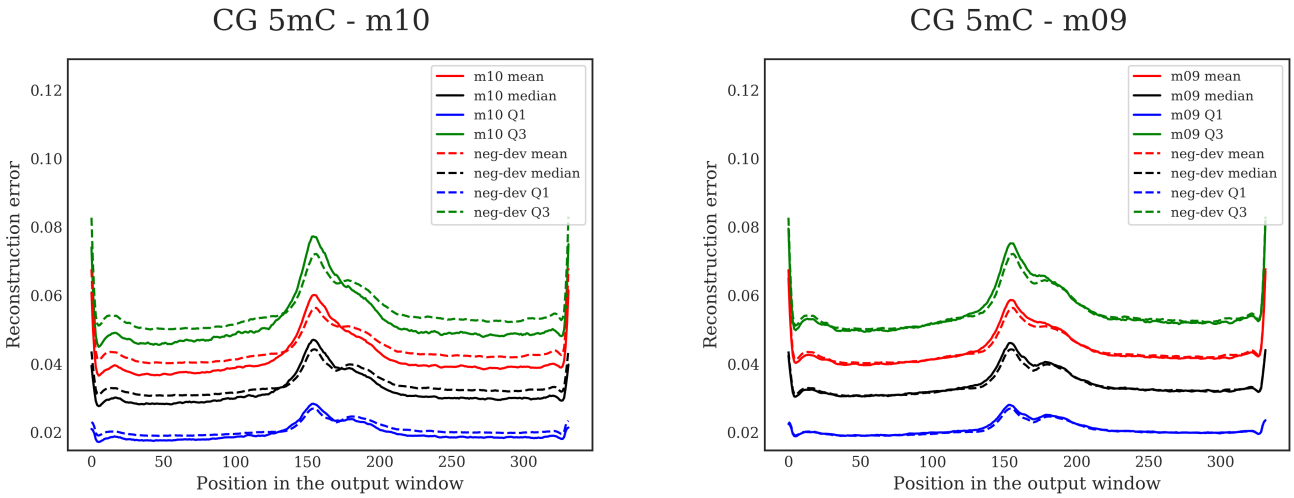
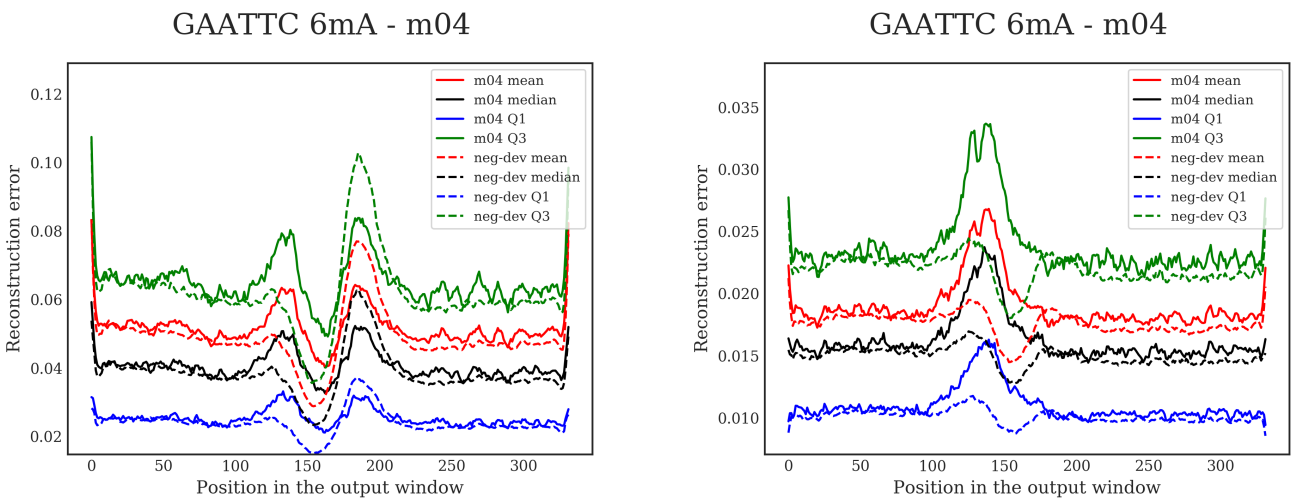


Figure 3.10: Error profile of `meth10` and `meth09` for CAE32.

Multivariate normal distribution with diagonal matrix could be also interpreted as a naive bayes model when we assume that every position in the output window is conditionally independent given the class and we only have data from one class. The features of a sample are reconstruction errors and every feature is distributed according to univariate normal distribution. For every position, we have estimated different parameters. In the same spirit, we have tried to replace the univariate normal distribution by several other distributions. We mostly aimed at skewed distributions. The distributions that we have tried were: log-normal distribution, gamma distribution, inverse normal distribution and skewed normal distribution. The most promising showed up to be the inverse normal distribution. Additionally, we have also tried to slightly improve the MSE method. We call this "MSE middle score" because we compute MSE only using the errors in the middle of the output window. More precisely, we use the segment $y_{100:250}$ from the output window of size 332. This method is based on the observation that all the methylations had the most deviated error profile from the unmethylated error profile in the middle of the window. The comparison of some of the techniques for CAE with bottleneck 32 are in Table 3.6. As we can notice, "MSE middle" is the best technique. It improves a lot the basic MSE which takes all of the errors in the output window and averages them. We can also see that for `meth04` the best model is inverse gaussian distribution. This is actually specific for this autoencoder. For CAE64 and DAE64, we can observe that inverse gaussian is not so significantly better. There is actually disagreement between PR-AUC and ROC-AUC in this case. The reason why targeting the middle of the window does not help for `meth04` is that we can observer deviations from unmethylated profile also on the sides of the window. See the comparision of the error profile for `meth04` in CAE64 and CAE32 in Figure 3.11.



(a) CAE32 (b) CAE64

Figure 3.11: Error profile of meth04 – GAATTC. Notice that the reconstruction error scales are not same.

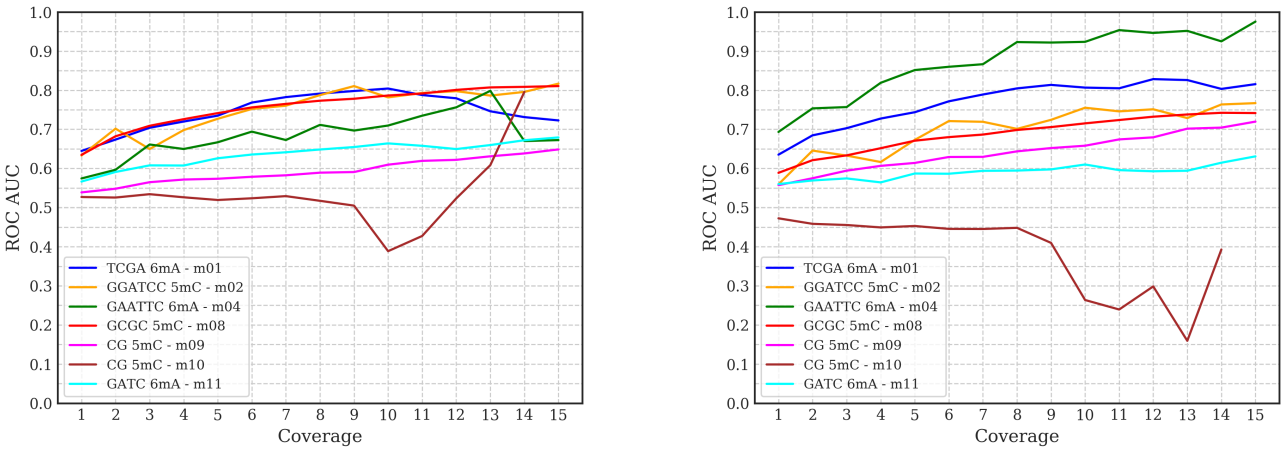
Methylase		meth01	meth02	meth04	meth08	meth09	meth10	meth11
pattern		TCGA	GGATCC	GAATTC	GCGC	CG	CG	GATC
#pos		52748	3315	2951	206150	145297	70815	89049
#neg		254856	9533	11342	441005	377933	377933	293111
ROC-AUC	mse_simple	0.592	0.583	0.573	0.565	0.499	0.446	0.529
	mse_middle	0.634	0.624	0.574	0.63	0.51	0.479	0.564
	inv_gauss	0.586	0.58	0.598	0.569	0.491	0.449	0.524
	diag_norm	0.579	0.56	0.592	0.576	0.486	0.462	0.521
	skew_norm	0.586	0.573	0.598	0.575	0.49	0.452	0.524
PR-AUC	mse_simple	0.222	0.318	0.254	0.36	0.278	0.136	0.244
	mse_middle	0.255	0.346	0.256	0.422	0.284	0.147	0.27
	inv_gauss	0.218	0.316	0.276	0.364	0.275	0.137	0.241
	diag_norm	0.217	0.3	0.278	0.373	0.27	0.14	0.243
	skew_norm	0.219	0.308	0.28	0.369	0.272	0.138	0.243

Table 3.6: Single read semi-supervised classification with CAE32.

We have also tried non-probabilistic anomaly detection techniques such as isolation forest and one-class SVM. The problem with these techniques is that they have several hyperparameteres which needs to be tuned for every methylation pattern separately which is not the path the we wanted to go. Additionally, the training algorithm of one-class SVM is quite slow so we had to downsample the dataset. Isolation forest performed better than one-class SVM and gave slightly worse results than multivariate

normal with full covariance matrix. We have not done any grid search to tune the isolation forest. The estimation of all the distributions that we mentioned above does not require any hyperparameter tuning which is lot better in our case since the number of methylation patterns that we are interested might grow a lot and we do not want to be limited to a specific set of motifs.

We have also tried to detect methylated sites in genome. The comparison of CAE32 and CAE64 for MSE middle is in Figure 3.12. We can notice that bottleneck 32 is better for most of the methylations except `meth04`. The other nice property taht we can notice is that for most of the methylations the AUC score approaches 0.8.



(a) CAE32

(b) CAE64

Figure 3.12: Semi-supervised detection of methylated genome sites. The plot with the number of positive samples for various coverages can be found in Figure 3.9.

3.6.8 Addition of Context to the Autoencoder

As we mentioned in Section 2.2, we are interested in contextual anomalies. So far the autoencoder has not been given any information about the context but we could notice from the error profiles that it can discriminate different contexts (Figure 3.4). The error profiles for different contexts can look significantly different. The autoencoder can reconstruct some of the contexts better then the other contexts but this is actually undesirable. We would actually want the anomaly score to be more stable for different contexts.

In Section 2.2, we have mentioned couple problem regarding incorporation of the context into to the model. We have to basically cope with two main problems – large number of contexts and errors in determining the context for a particular sequence of signal observations. To deal with the problem with errors, we have used bases from the reference sequence which were aligned to our signal. See Section 3.2 for more

details. Additionally, we have decided to use only the bases that are in the middle of the window. We took the bases that were aligned to the observations in the middle of the window and then we took three bases to the left of the base and three bases to the right (7-mer). We have also applied some additional filtering to the training set. We require that we have at least two bases around the 7-mer that are fully inside the input window. These windows made 0.44% of the training dataset. We have also applied this filtering strategy on the development set so the comparison with other models is fair.

The extracted 7-mer were passed to the autoencoder in one-hot encoding. We have experimented with two variants: when the bases were added as an input to the decoder and when they were added as input to the encoder.

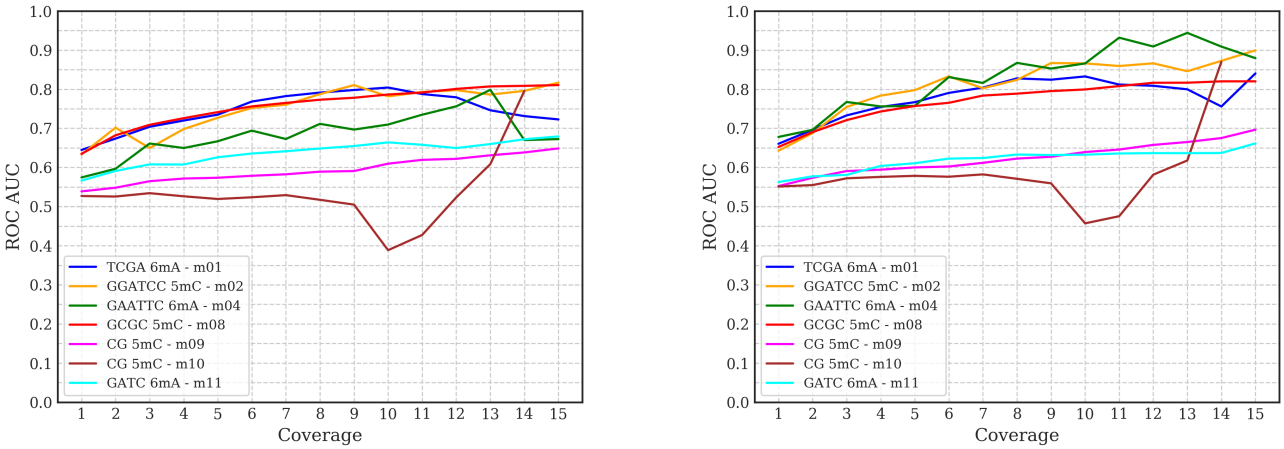
In the encoder case, the bases were merged with signal using fully-connected layer with hyperbolic tangent activations. The output size of this layer was 512 so this layer actually compresses the information since we go from two vectors of size 512 and $7 \cdot 4 = 28$ to one vector of size 512.

In the decoder case, the bases were merged with the bottleneck features using fully-connected layer with hyperbolic tangent units. The output size of this layer was $32 \cdot 32 = 1024$ since we have 32 filters and 32 bottleneck features. This neural network is actually similar to a neural network that receives bases and tries to predict the signal but we additionally give it some features about the signal – the bottleneck features. We have also tried to train a network which would predict the signal from bases but with no success.

The approach with addition of bases on the input of the decoder gave significantly better results. The comparison of CAE32 without bases and CAE32 with bases as an input to decoder for the methylated sites detection task is in Figure 3.13. We can notice that for methylations one, two and four, we are able to reach 0.85 AUC compare to 0.8 AUC. We can observe slight improvements also in the other methylations. The only ones that do not improve are 10 and 11.

We also provide a table (Table 3.7) with scores for a single read methylation detection task. We can notice that addition of bases helps for nearly all of the methylations except `meth11`.

Notice that an alternative approach would be to train a separate neural network for every methylation but this is quite computationally expensive and requires hyperparameter tuning for every methylation separately. The approach when we pass the bases to the autoencoder is quite similar but does not need separate hyperparameter tuning for every motif.



(a) CAE32 without bases

(b) CAE32 with bases

Figure 3.13: Semi-supervised detection of methylated genome sites. The plot on the left-hand side corresponds to classical CAE32 and the plot on the right-hand side corresponds to CAE32 with 7-mer as an additional input to the decoder. The plot with the number of positive samples for various coverages can be found in Figure 3.9. As an anomaly score was used "MSE middle".

Methylase	pattern	#pos	#neg	ROC-AUC		PR-AUC	
				no bases	bases	no bases	bases
meth01	TCGA	52748	254856	0.634	0.655	0.255	0.265
meth02	GGATCC	3315	9533	0.624	0.658	0.346	0.373
meth04	GAATTC	2951	11342	0.574	0.649	0.256	0.301
meth08	GCGC	206150	441005	0.630	0.635	0.422	0.429
meth09	CG	145297	377933	0.510	0.522	0.284	0.292
meth10	CG	70815	377933	0.479	0.504	0.147	0.156
meth11	GATC	89049	293111	0.564	0.561	0.270	0.266

Table 3.7: Semi-supervised classification using a single read. In this table we compare CAE32 without bases and CAE32 with bases as an input to the decoder. As an anomaly score was use "MSE middle".

Conclusion

The goal of this thesis was to design a method which can computationally identify modified DNA bases from raw MinION signal. The conventional approaches to this problem train a classifier on a dataset for which we know what bases are modified. The main limitation of this technique is that it requires an extensive training dataset that contains known DNA modifications in many different contexts.

Instead, we have employed a semi-supervised approach to this problem. We train an autoencoder on the dataset containing only canonical bases to learn characteristics of the signal for the canonical bases and then based on the reconstruction error we classify if the bases are modified or not.

We have experimented with two architectures of the autoencoder – convolutional and fully-connected. Our experiments show that the autoencoders with the convolutional architecture can better learn the characteristics of the signal of canonical bases. The reconstruction error profile for the convolutional autoencoders is smoother than for the fully-connected autoencoders.

The second conclusion drawn from our experiments is that our models cannot detect methylation based on a single read, but when we use multiple overlapping reads we can detect most of the methylations in our dataset. The authors of the dataset which we have used in our experiments have not done any validation if the methylases that they used to prepare the dataset were active. Therefore it is possible that the datasets in which we were not able to detect the methylation contains many unmethylated bases and consequently, we do not have very good ground truth for these datasets.

In our experiments, we have also tried to add the sequence of bases on the input of the decoder. This model actually shows the best results. This approach is similar to training a neural network to predict the signal from bases, but in this case we also include compressed features derived from the signal.

We have also attempted to use recurrent neural networks which seem to be a more natural framework for this type of data, since we are working with sequential data. Unfortunately, we did not get any promising results, and additionally the training of the recurrent neural networks is significantly slower than training convolutional networks.

In future, we would like to experiment more with variational autoencoders, which are probabilistic neural networks. We can use them to model the probability distri-

bution of unmethylated data and then use this probability as an anomaly score. We have done some preliminary experiments with these types of networks but we have encountered a problem that the autoencoder converged after two epochs into a bad local minima. Researchers from the Google Brain [BVV+15] and OpenAI [KSJ+16] teams apparently encountered similar problems in their work. We would like to try the techniques that they used to overcome this problem.

Bibliography

[BB11] Tomáš Vinař Broňa Brejová. *Methods in bioinformatics*, volume 1. Knižničné a edičné centrum, Fakulta matematiky, fyziky a informatiky, Univerzita Komenského, Mlynská dolina, 842 48 Bratislava, 1 edition, 2011.

[BBB+] JM Bernardo, MJ Bayarri, JO Berger, AP Dawid, D Heckerman, AFM Smith, and M West. Generative or discriminative? getting the best of both worlds.

[BBV16] Vladimír Boža, Broňa Brejová, and Tomáš Vinař. Deepnano: Deep recurrent neural networks for base calling in minion nanopore reads. *arXiv preprint arXiv:1603.09195*, 2016.

[BKNS00] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *ACM sigmod record*, volume 29, pages 93–104. ACM, 2000.

[BVV+15] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

[CBK09] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

[CUH15] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[DDY+16] Matei David, Lewis Jonathan Dursi, Delia Yao, Paul C Boutros, and Jared T Simpson. Nanocall: An open source basecaller for oxford nanopore sequencing data. *bioRxiv*, 2016.

[DEKM98] R. Durbin, S.R. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

[DV16] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

[GB10] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[GFGS06] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376. ACM, 2006.

[Goi16] Nicolas Goix. How to evaluate the quality of unsupervised anomaly detection algorithms? *arXiv preprint arXiv:1607.01152*, 2016.

[Heb49] Donald O. Hebb. *The organization of behavior: A neuropsychological theory*. Wiley, New York, June 1949.

[HXD03] Zengyou He, Xiaofei Xu, and Shengchun Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9-10):1641–1650, 2003.

[IS15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[JFM+15] Miten Jain, Ian T Fiddes, Karen H Miga, Hugh E Olsen, Benedict Paten, and Mark Akeson. Improved data analysis for the minion nanopore sequencer. *Nature methods*, 12(4):351–356, 2015.

[KB14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[KSJ+16] Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pages 4743–4751, 2016.

[Lab17] Winnie Tang Lab. Introduction to epigenetics, 2017. [Online; accessed 7-January-2017].

[LDB+13] Andrew H Laszlo, Ian M Derrington, Henry Brinkerhoff, Kyle W Langford, Ian C Nova, Jenny Mae Samson, Joshua J Bartlett, Mikhail Pavlenok, and Jens H Gundlach. Detection and mapping of 5-methylcytosine and 5-hydroxymethylcytosine with nanopore mspa. *Proceedings of the National Academy of Sciences*, 110(47):18904–18909, 2013.

[Li13] Heng Li. Aligning sequence reads, clone sequences and assembly contigs with bwa-mem. *arXiv preprint arXiv:1303.3997*, 2013.

[LQ14] Nicholas J Loman and Aaron R Quinlan. Poretools: a toolkit for analyzing nanopore sequence data. *Bioinformatics*, 30(23):3399–3401, 2014.

[LQS15] Nicholas James Loman, Joshua Quick, and Jared T Simpson. A complete bacterial genome assembled de novo using only nanopore sequencing data. *bioRxiv*, page 015552, 2015.

[LXTG17] Hao Li, Zheng Xu, Gavin Taylor, and Tom Goldstein. Visualizing the loss landscape of neural nets. *arXiv preprint arXiv:1712.09913*, 2017.

[MAB+17] Alexa BR McIntyre, Noah Alexander, Aaron S Burton, Sarah Castro-Wallace, Charles Y Chiu, Kristen K John, Sarah E Stahl, Sheng Li, and Christopher E Mason. Nanopore detection of bacterial dna base modifications. *bioRxiv*, page 127100, 2017.

[MF13] Alireza Makhzani and Brendan Frey. K-sparse autoencoders. *arXiv preprint arXiv:1312.5663*, 2013.

[Mou04] David W. Mount. *Bioinformatics: sequence and genome analysis*. Cold Spring Harbor Laboratory Press, 2004.

[MS03a] Markos Markou and Sameer Singh. Novelty detection: a review—part 1: statistical approaches. *Signal processing*, 83(12):2481–2497, 2003.

[MS03b] Markos Markou and Sameer Singh. Novelty detection: a review—part 2:: neural network based approaches. *Signal processing*, 83(12):2499–2521, 2003.

[Oxf18] Oxford Nanopore Technologies. Tombo, 2018.

[Pac17] PacBio. Advance genomics with single molecule, real-time (smrt) sequencing, 2017. [Online; accessed 21-January-2017].

[PCCT14] Marco AF Pimentel, David A Clifton, Lei Clifton, and Lionel Tarassenko. A review of novelty detection. *Signal Processing*, 99:215–249, 2014.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[RD99] Peter J Rousseeuw and Katrien Van Driessen. A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, 41(3):212–223, 1999.

[Res17] Zymo Research. Learn more about bisulfite conversion, 2017. [Online; accessed 21-January-2017].

[RJE+16] Arthur C Rand, Miten Jain, Jordan Eizenga, Audrey Musselman-Brown, Hugh E Olsen, Mark Akeson, and Benedict Paten. Cytosine variant calling with high-throughput nanopore sequencing. *bioRxiv*, page 047134, 2016.

[SATY17] Dong-Qiao Shi, Iftikhar Ali, Jun Tang, and Wei-Cai Yang. New insights into 5hmc dna modification: generation, distribution and function. *Frontiers in genetics*, 8:100, 2017.

[Sch16] Amanda Schaffer. Nanopore sequencing, 2016. [Online; accessed 30-April-2016] Available from `http://www2.technologyreview.com/news/427677/nanopore-sequencing/`.

[Sim16] Jared T. Simpson. Simpson lab blog, 2016. [Online; accessed 27-January-2016] Available from `https://simpsonlab.github.io/`.

[SPST+01] Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001.

[SQE+16] Marcus H Stoiber, Joshua Quick, Rob Egan, Ji Eun Lee, Susan E Celniker, Robert Neely, Nicholas Loman, Len Pennacchio, and James B Brown. De novo identification of dna modifications enabled by genome-guided nanopore signal processing. *bioRxiv*, 2016.

[SR15] Takaya Saito and Marc Rehmsmeier. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, 10(3):e0118432, 2015.

[SWAS+13] Jacob Schreiber, Zachary L Wescoe, Robin Abu-Shumays, John T Vivian, Baldandorj Baatar, Kevin Karplus, and Mark Akeson. Error rates

for nanopore discrimination among cytosine, methylcytosine, and hydroxymethylcytosine along individual dna strands. *Proceedings of the National Academy of Sciences*, 110(47):18910–18915, 2013.

[SWZ+16] Jared T Simpson, Rachael Workman, Philip C Zuzarte, Matei David, Lewis Jonathan Dursi, and Winston Timp. Detecting dna methylation using the oxford nanopore technologies minion sequencer. *bioRxiv*, page 047142, 2016.

[TD04] David MJ Tax and Robert PW Duin. Support vector data description. *Machine learning*, 54(1):45–66, 2004.

[Tec] Oxford Nanopore Technologies. Oxford nanopore sequencing technologies specifications. [Online; accessed 26-January-2016] Available from `https://nanoporetech.com/community/specifications`.

[THD+17] Haotien Teng, Michael B Hall, Tania Duarte, Minh Duc Cao, and Lachlan Coin. Chiron: Translating nanopore raw signal directly into nucleotide sequence using deep learning. *bioRxiv*, page 179531, 2017.

[The10] The HDF Group. Hierarchical data format version 5, 2000-2010.

[TJBB05] Yee W Teh, Michael I Jordan, Matthew J Beal, and David M Blei. Sharing clusters among related groups: Hierarchical dirichlet processes. In *Advances in neural information processing systems*, pages 1385–1392, 2005.

[TM87] Ronald J Tallarida and Rodney B Murray. Mann-whitney test. In *Manual of Pharmacologic Calculations*, pages 149–153. Springer, 1987.

[Web18] Merriam Webster. Merriam-webster online dictionary. 2018. [Online; accessed 20-April-2018].

[Whi05] Michael C Whitlock. Combining probability from independent tests: the weighted z-method is superior to fisher's approach. *Journal of evolutionary biology*, 18(5):1368–1373, 2005.

[Wik16a] Wikipedia. Autoencoder, 2016. [Online; accessed 30-April-2018].

[Wik16b] Wikipedia. Dna, 2016. [Online; accessed 30-April-2016].

[wik17a] Expectation maximization, 2017. [Online; accessed 7-January-2017].

[Wik17b] Wikipedia. Dna methylation, 2017. [Online; accessed 7-January-2017].

[Wik17c] Wikipedia. Epigenetics, 2017. [Online; accessed 7-January-2017].

[Wik18] Wikipedia. Cpg island hypermethylation, 2018. [Online; accessed 30-April-2018].

[WSH+10] Emma VB Wallace, David Stoddart, Andrew J Heron, Ellina Mikhailova, Giovanni Maglia, Timothy J Donohoe, and Hagan Bayley. Identification of epigenetic dna modifications with a protein nanopore. *Chemical communications*, 46(43):8195–8197, 2010.

Appendix A – Links to Nanoraw Dataset

Identifier	**File size (GB)**	**Link**
control1	54.52	http://s3.climb.ac.uk/nanopore-methylation/Control_lib1.tar
control2	81.44	http://s3.climb.ac.uk/nanopore-methylation/Control_lib3.tar
meth10	22.52	http://s3.climb.ac.uk/nanopore-methylation/meth10_lib3.tar
meth11	41.54	http://s3.climb.ac.uk/nanopore-methylation/meth11_lib3.tar
meth12	75.73	http://s3.climb.ac.uk/nanopore-methylation/meth12_lib3.tar
meth01	32.58	http://s3.climb.ac.uk/nanopore-methylation/meth1_lib1.tar
meth02	49.24	http://s3.climb.ac.uk/nanopore-methylation/meth2_lib1.tar
meth03	43.31	http://s3.climb.ac.uk/nanopore-methylation/meth3_lib1.tar
meth04	36.49	http://s3.climb.ac.uk/nanopore-methylation/meth4_lib1.tar
meth05	25.90	http://s3.climb.ac.uk/nanopore-methylation/meth5_lib2.tar
meth06	41.47	http://s3.climb.ac.uk/nanopore-methylation/meth6_lib2.tar
meth07	46.73	http://s3.climb.ac.uk/nanopore-methylation/meth7_lib2.tar
meth08	64.45	http://s3.climb.ac.uk/nanopore-methylation/meth8_lib2.tar
meth09	51.91	http://s3.climb.ac.uk/nanopore-methylation/meth9_lib2.tar

Appendix B - Source Code

This thesis includes an attached CD, containing the source code of the scripts that were used in the experiments. The source code is also available at `https://gitlab.com/rastislav.rabatin/deepmeth.git`.