William Chakroun Jacobsen

Classifying genes based on promotors with deep neural networks

Master's thesis in TDT4900
Supervisor: Pål Sætrom
January 2022

Master's thesis

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

William Chakroun Jacobsen

Classifying genes based on promotors with deep neural networks

Master's thesis in TDT4900
Supervisor: Pål Sætrom
January 2022

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

NTNU
Norwegian University of
Science and Technology

Classifying genes based on promotors with deep neural networks

William Chakroun Jacobsen

17/01/2022

Acknowledgement

I would like to express my special gratitude to Prof. Pål Sætrom for his insights and expert guidance in the field of bioinformatics. Thank you to my peers for helping me in finalizing my thesis. Lastly, thanks to 10xGenomics for keeping the dataset available.

Abstract

The thesis use single-cell RNA gene expression data from 10xGenomics to study the capabilities of using a deep neural network to classify genes based on the transcription sequences. Genes in the DNA have corresponding transcription factors that attach near the gene's transcription start site. The assignment of the transcription factor is to regulate the gene's transcription to proteins called gene expression. Genes with similar expression profiles are typically controlled by the same transcription factors and often share similar sequence elements. Therefore, can the coexpressed genes can be clustered.

This thesis introduces three methods developed to cluster the gene expression dataset; clique-based, set-cover, and Hierarchical. These methods are compared based on each cluster's robustness (*cohesion*) and the separation between the clusters. The generated clusters are used as classes for the deep learning model based on the paper 'Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning' by Alipanahi *et al.* The deep learning method's job is to extract features from the transcription factor sequences that are similar to the coexpressed genes to classify them. Thus, testing different upstream sequence lengths from the transcription start site and the cohesion of the clustering algorithm. In order to find the ideal *sequence length* and *cohesion* there is used P-value, power analysis is used to find the significance between each cohesion and each sequence length.

Hierarchical clustering reached an internal cohesion of 0.125, clique-based of 0.075, and Set-Cover of ≈ 0.225. Both Set-Cover and Hierarchical retained all genes while clique-based removed genes that were not in a cluster. For the deep learning method, a cohesion of 0.05 had the highest mean AUC score of 0.68, while the other cohesion, 0.075, 0.1, and 0.125, had a score of ≈ 0.5. For sequence length, all sequences, 500, 1000, 1500, and 2000 have a mean AUC score of ≈ 0.53.

In this thesis, the clustering method used was Hierarchical clustering based on superior results in cohesion and retaining genes. In the deep learning aspect, we found that the cohesion of 0.05 indicates a better result. However, there was no significance based on the different sequence lengths.

Sammendrag

Denne oppgaven bruker enkeltcellede RNA-genekspresjonsdata fra 10xGenomics for å studere mulighetene for å bruke et dypt nevralt nettverk for å klassifisere gener basert på transkripsjonssekvenser. Gener i DNA har tilsvarende transkripsjonsfaktorer som fester seg nær genets transkripsjonsstart. Oppgaven til transkripsjonsfaktoren er å regulere genets transkripsjon til proteiner, kalt genuttrykk. Gener med lignende genutrykk styres vanligvis av de samme transkripsjonsfaktorene og deler ofte lignende sekvenselementer. Derfor kan de samuttrykte genene grupperes.

Det blir introdusert tre metoder utviklet for å gruppere genekspresjonsdatasettet; “Clique-based”, “Set-cover” og hierarkisk. Disse metodene sammenlignes basert på hver klynges robusthet (*kohesjon*) og separasjonen mellom klyngene. De genererte klyngene brukes som klasser for dyplæringsmodellen basert på papiret ‘Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning’ av Alipanahi *et al.* Dyplæringmetodens jobb er å trekke ut egenskaper fra transkripsjonsfaktorsekvensene som ligner på de samuttrykte genene for å kunne klassifisere dem. Testing av forskjellige sekvensene fra transkripsjonsstartstedet og samholdet til klyngealgoritmen. Dette er gjort for å finne den ideelle *sekvenslengden* og *kohesjon* så er det brukt P-verdi, og kraftanalyse for å finne signifikansen mellom hver kohesjon og hver sekvenslengde.

Hierarkisk klynging nådde en intern kohesjon på 0,125, “Clique-based” på 0,075, og “Set-cover” på $\approx 0,225$. Både “Set-Cover” og Hierarkisk beholdt alle gener mens “Clique-based” fjernet gener som ikke var i en klynge. For dyplæringsmetoden hadde en kohesjon på 0,05 den høyeste gjennomsnittlige AUC-verdi på 0,68, mens de andre: 0,075, 0,1 og 0,125, hadde en verdi på $\approx$ 0.5. For sekvenslengde har alle sekvenser, 500, 1000, 1500 og 2000 en gjennomsnittlig AUC-verdi på $\approx 0,53$.

I denne oppgaven var clustering-metoden som ble brukt Hierarkisk clustering basert på overlegne resultater i kohesjon og for å beholde gener. I dyplæringsaspektet fant vi at kohesjonen på 0,05 indikerer et bedre resultat. Det var imidlertid ingen betydning basert på de forskjellige sekvenslengdene.

Contents

Figures

Tables

Code Listings

Chapter 1

Introduction

1.1 Review

Each cell in the human body has its functionality and response mechanism [3]. These cells are the building block for every living organism and contain a nucleus. The nucleus is a complex element of the cell, housing the encoding of the organism called the Deoxyribonucleic acid (DNA) [4]. The DNA has two strands and is twisted, containing nucleotides that connect the two strands through a hydrogen bond. The nucleotides have the characteristic letters A, C, T, and G, representing each nucleotide[5]. Together, these letters can form a gene converted into a protein through transcription. On the other hand, not all DNA can transcribe into mRNA and then proteins. These segments are called non-codable DNA [6]. The non-codable DNA is the one that controls the transcription of the gene, and the whole process is called gene expression and contains three steps; initiation, elongation, and termination. The initiation phase enables the generation of mRNA by either enhancing or stopping the RNA-polymerase. The RNA-polymerase is an enzyme that synthesizes RNA from a DNA template. This phase is why every cell has the same DNA. Nevertheless, have different functionality [7]. In the elongation phase, the RNA-polymerase splices the DNA open and uses the gene sequence as a template to produce an mRNA sequence [8], which the mRNA will further translate into a protein. Lastly, the termination phase occurs when the RNA-polymerase reaches the end of the gene sequence and detaches from the DNA [9]. Many methods have evolved to read and understand the structure and sequences of a DNA strain [10]. The gold standard of such a method is the Sanger sequence [11]. The Sanger sequence can read up to 1 million nucleotides daily and was the first to map the human genome. However, the method is slow compared to the human genome size. Therefore, next-generation sequencing (NGS) was developed to speed up this process, which can scan multiple workloads in parallel[12, 13]. The sequencing methods can be split into two classes, bulk, and single-cell (scRNA-seq). Bulk sequencing of RNA will generate a homogenous result based on the cell's input. However, single-cell sequencing takes each cell and uses NGS on its output. As a result, single-cell sequencing generates a sparse data output since not all cells have the same transcription factor. Furthermore, single-cell sequencing contains technical and biological noise that might interfere with the method's output [14, 15].

The sparse matrix the scRNA-seq generates makes computational methods for predicting genes based on the genes' transcriptional signature difficult. As a result, several sophisticated tools have been developed to accomplish this task. Such as DeepBind and GRN [1, 16]. First, to utilize the sparse matrix generated, a correlation matrix is created through the genomic distances on the coexpression of coregulated genes. Secondly, the correlation matrix is clustered using Clique-clustering, Set-Cover, and Hierarchical. Each clustering method has its strength and weaknesses. For example, clique-based creates cohesive clusters but neglects to add all genes to a cluster, resulting in many genes being cluster-less. Set-Cover generates not as cohesive, but all genes are in clusters. Lastly, Hierarchy sets all genes in a cluster and generates somewhat cohesive clusters.

A deep learning model uses the clusters generated by genes' biological patterns for binary classification. A deep learning model utilizes a neural network of activation values and weights. When a model trains, the weights change based on the overall performance. The performance is based on a loss function utilized by an optimizer. The optimizer's job is to generalize the model to learn features from training the model. The features learned will make the model able to classify the dataset given. The model trains on the nucleotide sequence upstream from the transcriptional start site. The model is a convolutional neural network using the convolutional layer's advantage to extract common features from the coregulated genes' promotor sequences. Moreover, utilizing these classifications, a motif pattern should be apparent in the genes in the cluster based on the promotor.

1.2 Claim

Due to the technological advancement in single-cell sequencing and NGS, more biological data have become available for researchers. Because of the increase in data, it has become easier to do computational research on genes and non-codable DNA around. This explosion in computational data have resulted in active research to find the most promising method to uncover coregulated genes in bioinformatics. Therefore, this thesis will use the method of 'Single-cell transcriptomics unveils gene regulatory network plasticity' by Iacono *et al.* to generate the correlation matrix based on coregulated genes and explore the possibilities for this methodology. I will focus on cluster genes based on their coexpression and use novel technological advancements in deep learning to research the possibility of binary classifying these genes based on the classes given and the gene's promoter. Furthermore, the research will focus on the effectiveness of different sequence lengths and the cohesiveness of the cluster in question. In addition, the difference in effectiveness in different traditional and novel clustering techniques.

Research Questions

RQ1. Which clustering method of Clique, Set-Cover, and Hierarchical gives the best clustering of genes from scRNA-seq data?

RQ2. How does the cohesion of a cluster affect the result of the deep learning model?

RQ3. How does the sequence length affect the result of the deep learning model?

1.3 Agenda

This thesis aims to give the reader background information about how the genes affect a cell and how the extraction method occurs. The background section gives an overview of the clustering methods and the correlation algorithms used to accomplish this. Secondly, the methodology uses background information to answer the research questions. Methodology illustrates how the cluster methods were used and evaluated, the architecture and hyperparameters of the deep learning method used, and the evaluation of the model. Thirdly, the thesis will show the results generated from the methodology. The result section illustrates the cohesion and separation from Clique-Based, Set-Cover, and Hierarchy. In addition, the results show four runs of cohesion thresholds and sequence lengths from the transcriptional start site and compare these using P-values. Lastly, a discussion and conclusion on the clustering methods to answer **RQ1**, the results from comparing multiple cohesion runs to answer **RQ2**, and the results from comparing multiple sequence length runs to answer **RQ3**.

Chapter 2

Background

In order to answer the given research questions, the reader needs to understand the basics of the data used in the deep learning model. Therefore, this chapter introduces background information on relevant topics such as basic biology, genetics, computer science, and statistics to understand the method, result, and discussion. Furthermore, related works used to complete this thesis are shown.

2.1 Cell biology

2.1.1 The Cell

Every multicellular living organism contains cells, which is the building block of each living creature. The cell's structure contains a nucleus encapsulated by a nuclear membrane. The nucleus decides the function and structure of the cell, and by the DNA it contains. The cell also contains Cytoplasm. The Cytoplasm is a fluid that provides a platform for organelles in the cell to operate. Organelles are structures that perform various jobs inside cells. One of these organelles is the Ribosome. A Ribosome performs mRNA translation, which is protein synthesis. The mRNA is generated by an RNA-polymerase that transcribes the DNA. Before understanding this process, an introduction to DNA is necessary. [3, 4].

2.1.2 DNA - What is DNA?

As stated in Section 2.1.1, the DNA is contained in the nucleus of all living organisms, which carries the genetic encoding. The DNA structure consists of two strands that are twisted into a double helix visualized in Figure 2.1. Each long strand is built up by a chain of monomer nucleotides generating a phosphate-sugar backbone. The nucleotides consist of deoxyribose attached to a phosphate group and one of four nitrogenous bases. The strands are held with a hydrogen bond between the bases where adenine (A) bonds to thymine (T) and cytosine (C) to guanine (G), as illustrated in figure 2.1. The DNA is structured into larger complexes, known as chromosomes. The DNA structure genetic encoding is synthesized to a gene product that allows the production of proteins. Protein biosynthesis is a core process essential for the cell's functionality. However, not all DNA can be translated into proteins. The DNA contains non-coding and coding DNA. The coding DNA is the encoding for the creation of proteins. However, the non-coding DNA contains regulation of genes, for example, turning on and off a gene [6]. The regulation is called gene expression and is discussed more in detail in Section 2.1.3 [5].

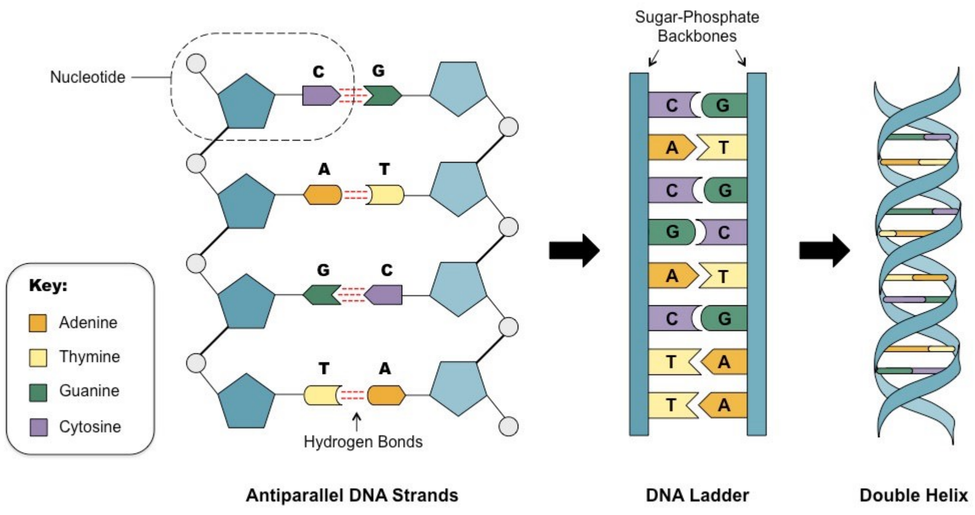


Figure 2.1: Illustration of the polymers of the DNA structure [18].

2.1.3 Gene Expression

Gene expression can be broken into three separate stages: initiation, elongation, and termination.

Initiation

Gene expression is the process used to enable the production of a protein. This process is why all cells in an organism have the same DNA and can differ in functionality. The gene expression is controlled in two stages. Firstly, transcription is controlled by limiting the amount of mRNA produced from a gene. The second stage is a control of post-transcriptional (primary transcripts) events [7]. The first stage is controlled by the cis-regulatory elements, which are the non-coding areas of the DNA. These non-coding areas are such as the promoter, the operator, and the enhancer. The cis-regulatory elements regulate the gene's transcription, which, in turn, generates primary transcripts. The transcription starts with an RNA-polymerase (RNAP) protein binding to the promoter region, as seen in Figure 2.2. The promoter typically lies upstream (3') of the same DNA strand as the transcribed gene and is about 100-1000 nucleotide long [19]. The promoter also contains the activator binding site. The activator is a transcription factor that upregulates the mRNA transcription and increases the interaction between RNAP and the promoter section of the transcribed gene. Furthermore, the activator can bind to the enhancer depending on the DNA binding segment. Further downstream from the promoter is the Operator site of the specific gene. This segment allows the binding of the repressor protein that inhibits or impedes transcription of the gene [8]. According to Cooper Repressors play a key role in regulating transcription in animal cells and serve a critical role in cell growth and differentiation. The repressor serves many roles, including inhibiting the transcription by interacting with general transcription factors and the activator protein [8].

Elongation

Further downstream (5') is the coding DNA (gene), as seen in Figure 2.2. The RNA-polymerase will splice the gene sequence and translate the nucleotide bases to messenger RNA (mRNA); The RNAP uses a single-stranded DNA template to generate the mRNA. The gene sequence is "read" one nucleotide at a time, and the RNAP builds a chain containing complementary base pairs, the mRNA. The generation of mRNA strains is determined by the initiation phase of both the promotor and the operator. The generated mRNA is instructions for synthesizing proteins. The mRNA leaves the nucleus and is exported into the ribosome. The mRNA is translated into proteins in the ribosome by attaching tRNA that contains amino acids. Chaining these amino acids causes the creation of protein. When the completion of gene translation, the next stage is the Termination stage [9].

Termination

At this stage, the termination ends the mRNA transcription, which occurs when the RNAP reaches the termination sequence of the gene. The termination causes the mRNA strand to detach from the DNA [9].

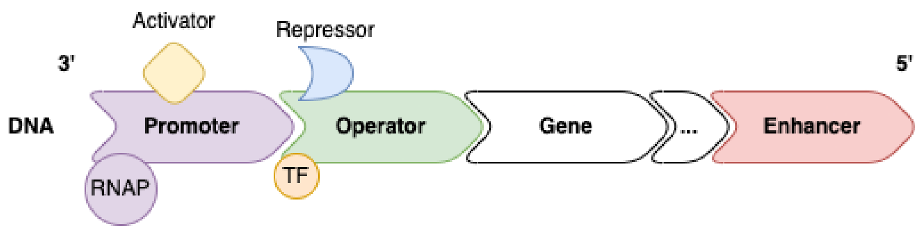


Figure 2.2: Illustration of Transcription area.

2.2 RNA Sequencing

RNA sequencing analyzes mRNA to understand cellular responses in thousands of cells. These responses can be measured using the mRNA generated during the transcription described in Section 2.1. There are multiple ways of understanding cellular responses. First, there is the usage of gene expression, which uses proteins generated in the Transcription phase of the gene to study the cell's functionality. However, it is challenging to research thousands of proteins expressed by the genome that exist in a single cell. Therefore, researchers have turned to mRNA, the recipe for the protein the cell was going to make and correlates to the cells' expression profile. This is done through Sanger and next-generation sequencing (NGS) and is referred to as RNA-sequencing. This method yielded much information about the cell's functionality and was an innovative way of discovering new elements in the cells' applicability. Furthermore, the methods showed individual cells with high resolution on the genomic level [10]. Therefore, by using this methodology on an individual cellular level, the researcher can get a more detailed picture of the cell population that would go unnoticed in analyzing a bulk of cells. This sequencing on an individual cellular level is called single-cell RNA sequencing (scRNA-seq) [20].

2.2.1 Sanger Sequencing

It is essential to understand the history of sequencing to understand the complications of RNA sequencing. In the 1970s, Sanger and his colleagues developed a technique for deciphering genomes called the Sanger sequence. This new sequencing method required fewer toxins and radioisotopes than other older techniques and is now referred to as "the golden standard"[11].

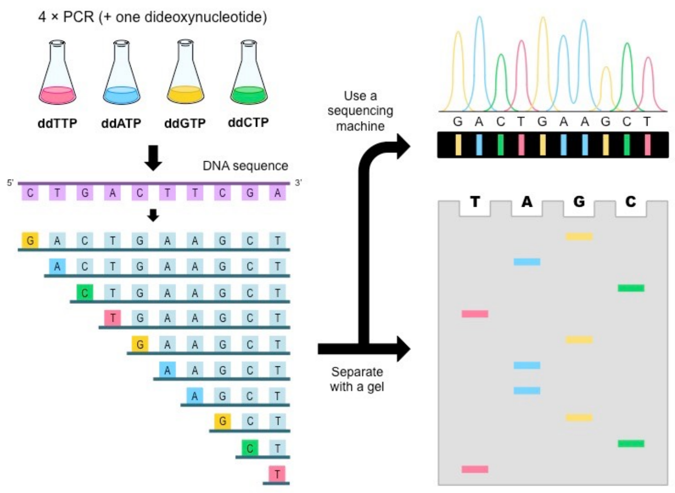


Figure 2.3: Illustration of Sanger sequencing [21].

The Sanger sequencing uses DNA polymerase to transcribe DNA regions and utilizes four test tubes and primers for each of them. When DNA sequences are added to the test tubes, the primer bases are added with the help of the polymerase.

However, the translation stops. This is because each test tube contains its inhibitor of bases A, C, T, and G. These inhibitors are called a dideoxynucleotide (ddNTPs) and lack the OH-molecule attached to the original nucleotide and contain radioactive markers for each base. The lack of OH-molecule in the nucleotide causes the polymerase not to be able to add any more bases to the chain and terminate the polymerase translation. Adding a low concentration of the ddNTPs to each test tube containing one of each base, ddATPs, ddCTPs, ddTTPs, and ddGTPs will cause the generation of multiple random length oligonucleotides chains, also known as short synthetic DNA. When reading the test tubes' results, oligonucleotides are added to an acrylamide gel with each base in each lane.

Electric current is applied to the gel is to sort each chain from shortest to longest in each tube. A technique is known as electrophoresis. A radioactive film in the gel causes each ddNTP to light up and enable localization of the bases[11]. However, alterations have been made to the method. For example, in today's Sanger Sequencing, a fluorescent dye is used rather than radioactive dye, and only one tube is used. As a result, the fluorescent bases will align up and be easier to read using a laser by electrophoresis. This method was utilized to generate the National Human Genome project. However, this project was resourceful. The Sanger sequence is considered a gold standard and is used to fact-check the current methods. However, the method is quite slow. The Sanger sequencing method can read up to 1 million bases per day. For comparison, the human genome is 6.4 billion base pairs long.

The slow sequencing of Sanger stimulated the research and development of next-generation sequencing, which used three major improvements. First, it did not require bacterial cloning, there was an increase in parallel sequencing, and third, there was no use of electrophoresis, separating the DNA, RNA, and proteins based on electrical charge. These three steps enable NGS to process genomes at great speed. These techniques led to widespread use in sequencing since it was cheap, fast, and easy to use. For example, Illumina is a company that launched a bench-top sequencer that reduced the cost per human genome sequence [12].

2.2.2 Next-Generation Sequencing - Illumina

The genome sequencing in today's market is dominated by Next-generation sequencing and has progressed tremendously from the Sanger sequencing presented in section 2.2.1. With the reduction of per-base cost, reading speed has increased, and read-length per gene. This progress has made it easier to develop large NGS applications such as diagnostics and forensics, and in newer times has been used for the genetics of COVID-19 susceptibility [13]. All novel Next-generation sequencing methods have different ways of sequencing DNA in terms of technicalities. However, all NGS technologies share a similar way of sequencing DNA.

The first step in NGS is fragmenting the DNA by either shearing, usually done mechanical, or enzymatic. The fragmented DNA appends adapters on the 3' and

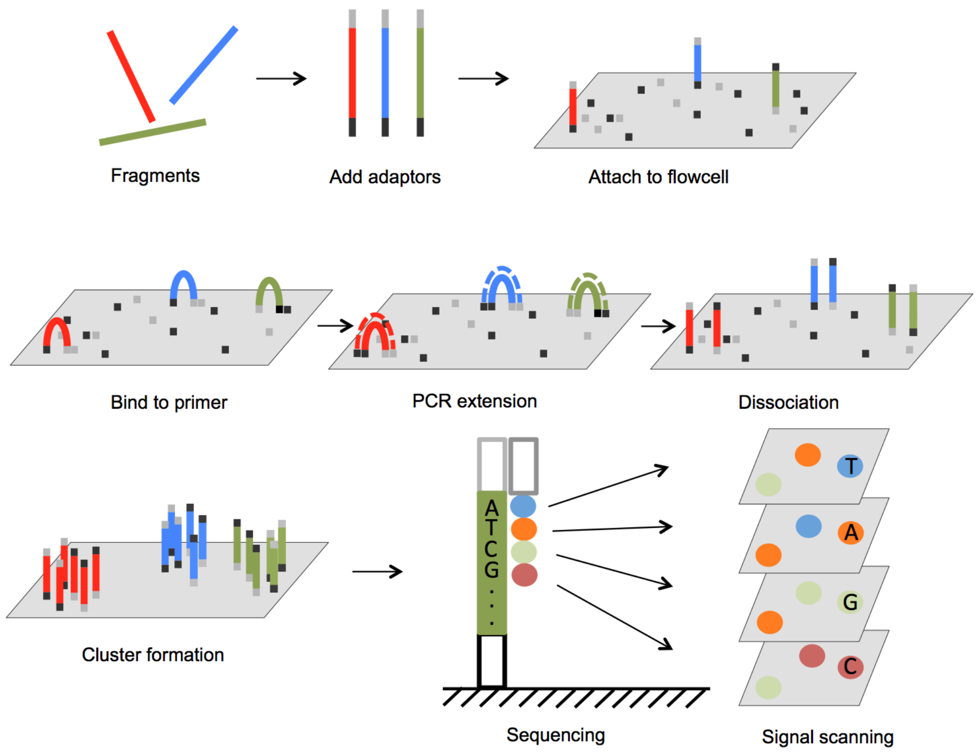


Figure 2.4: Next-generation DNA sequencing technique [22].

5' sides. These adaptors include motifs such as the sequencing binding site for a polymerase protein, indices, and complementary sequences to the oligos. The next step is clustering. The fragments are added to a plane with a lawn filled with complementary binding sites for the oligos on the fragment. Then the fragment is generated by cloning the fragment by using a bridge polymerase chain reaction (PCR). This utilizes the oligos on the planar support plate, where the oligo ends of the fragment attach to the complementary oligo forming a bridge, as visualized in Figure 2.4. After the binding, a polymerase protein generates a complementary strain to the fragment by attaching nucleotides using PCR extension.

Furthermore, the original binding site is broken, and the fragment is complementary cloned. The PCR extension is repeated multiple times, generating cluster formations on the plane. The last step can initialize when the amplification is complete. All reverse strands are removed from the plane since the fragments have been flipped multiple times because of the bridge PCR amplification. The removal causes all the fragments to have the same direction. Next, a polymerase protein attaches to the binding site of the strain and appends fluorescent ddNTPs. When adding a fluorescent nucleotide to the fragment strain, a laser excites the new nucleotide, and a light-sensitive camera catches which base was added to the strain. After the image is taken, the terminator on the ddNTP is removed, which gives the polymerase the ability to append a new ddNTP nucleotide. The appending is done through cycles and is continued until reaching the desired sequence length [22].

This method utilizes the ability to parallel sequencing of the DNA or RNA massively. Each plate in the Illumina processing contains eight lanes that can produce 250 million reads each. Even though the impressive technological advances of sequencing are presented here, the method cannot separate based on individual cells. The next section will present how to sequence individual cells, so-called single-cell sequencing.

2.2.3 Single-cell RNA-Sequencing

Single-cell sequencing can reveal cell population differences or evolutionary relationships in a cell cluster. However, traditional bulk sequencing only gives an average of the cell population. On the other hand, a single cell gives detailed information about a single cell and manages to detect heterogeneity for each cell. However, the method is expensive and returns a sparse data matrix to the researcher. This section describes the biological aspects of single-cell sequencing and its theory. This section will focus on 10xGenomic's single-cell RNA-seq technology. However, a majority of the scRNA-seq technologies follow these steps [14].

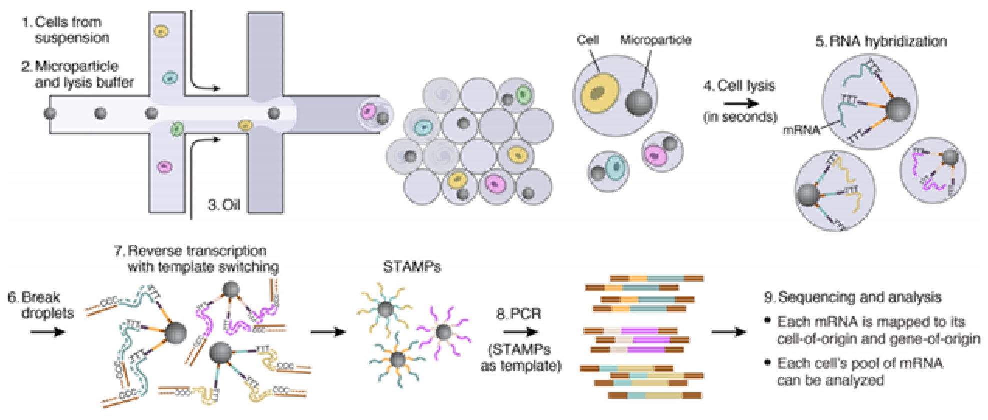

1. Cells from suspension
2. Microparticle and lysis buffer
3. Oil
Cell
Microparticle
4. Cell lysis (in seconds)
5. RNA hybridization
mRNA
6. Break droplets
7. Reverse transcription with template switching
STAMPs
8. PCR (STAMPs as template)
9. Sequencing and analysis
• Each mRNA is mapped to its cell-of-origin and gene-of-origin
• Each cell's pool of mRNA can be analyzed

Figure 2.5: Preparation of single cell analysis with 10xGenetics methodology [23].

Isolate Cells and Barcoding

The heterogeneous cells need to be floating freely in suspension to be extracted individually to start isolating the cells. Common methods for tissue dissociation are either enzymatic digestion or the application of mechanical force to separate the cells. After the mechanical or chemical separation of cells, each cell is injected into an oil emulsion droplet with a single bead, as seen in Figure 2.5 in step 3.

The enzymes in the oil emulsion bursts open the cell and spill the mRNA inside the droplet. There are three primers inside the. Firstly, Each bead has a unique barcode called the cell barcode. That is added to the transcript, so when it is reversed transcripted. This is to give each transcript its cell identity. Secondly is the Unique molecular identifier (UMI) primer, which has a random k-mere sequence. The UMI is added to the reverse transcriptome to identify when the amplification process starts. Lastly is the PCR handle to enable PCR amplification [15].

Amplification

The amplification is done by PCR amplification, which combines the mRNA floating around in the oil. The oligonucleotides generated by the first step are amplified using heat to separate the cDNA. By doing so, a polymerase can synthesize a cDNA for the oligonucleotides. This will continue and clone the original strand exponentially. Then, the primer of the original oligonucleotide binds to the anti-parallel primer, and the polymerase can do its job [14, 15].

NGS preparation and Sequencing

The strands are added to a Next-generation sequencing as shown in Section 2.2.2 to read and convert the biological product from the scRNA-seq. Then, all the cells are added to a singular library, and a primer is added to the library. This is to identify the sample of the cells.

Challenges with Single-cell RNA Sequencing

According to Lindner *et al.*, a gene transcription has temporal fluctuations in response to environmental conditions. The fluctuation can cause identical cells to produce different amounts of a gene. In addition, according to Qi *et al.*, Many features of scRNA-seq are nearly zero, making processing noise of a great significance, and making the noise difficult to distinguish from individual similar cell types. [24, 25] The problems aforementioned are called biological noise and are caused by the nature of cells and genes. Additional to biological noise is technical noise. An example of technical noise is gene dropout events which are caused by an expression level detected in one cell but not in another. The dropout is caused by low capture efficiency, leading to a discrepancy between similar cells [26].

2.3 Computer Science

2.3.1 Clustering Algorithms

This Section introduces ways to cluster the data introduced in the Section mentioned above and further answers the Research Question **RQ1**.

Hierarchical Clustering

Hierarchical clustering is an algorithm that groups similar nodes into groups or clusters. These clusters are generated into a hierarchy of clusters. There are generally two types of hierarchical clustering algorithms. The first one is Agglomerative hierarchical clustering, a "bottom-up" approach. Agglomerative start each node or observation as their own individual cluster, then pairs of clusters are merged based on similarity. This similarity is based on the unit of distance given by a distance matrix [27]. This cluster structure is visualized in the tree-based representation shown in Figure 2.6. This clustering algorithm has a time complexity of $\mathcal{O}(n^2)$ and requires $\mathcal{O}(n^2)$ memory [28]. The second hierarchical clustering algorithm used is the Divisive clustering algorithm. Divisive clustering has a "top-down "approach. All observations start as one cluster and are split downwards recursively as the algorithm moves down the dendrogram.

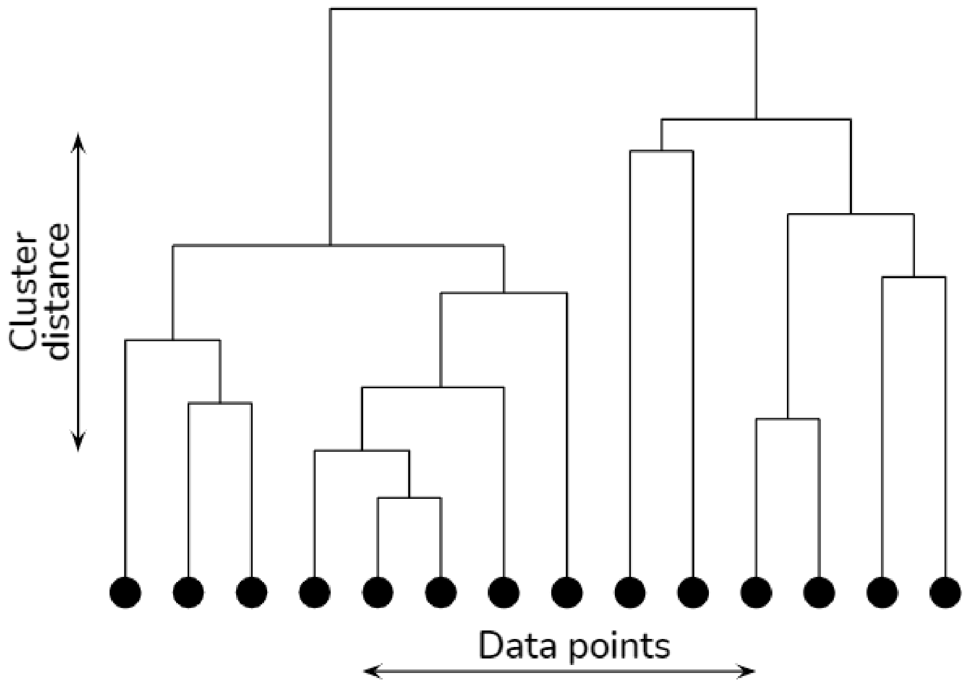


Figure 2.6: Dendogram with the distance from each cluster given in the y-axis and observations in x-axis [29]

Code listing 2.1: Pseudo Code for Hierarchical Clustering

```
clusters = [...]
distance_matrix = Matrix

def calculate_distance_matrix():
    for s in clusters:
        for r in clusters:
            distance = calculate_distance(s, r)
            distance_matrix.store_in_distance_matrix(distance)
```

```
def main():
    calculate_distance_matrix()
    for _ in range(len(clusters)-1):
        # merge_pair method finds the clusters in the distance matrix with the
        # shortest distance to each cluster,
        # and merges r, s to a new cluster k where k not equals r, s.
        distance_matrix.merge_pair();
        # add_pair method adds the new cluster k into the distance matrix in the
        # column and row.
        distance_matrix.add_pair();
        # remove_merged_elements method removes the old cluster elements r and s.
        distance_matrix.remove_merged_elements();
```

2.3.2 Clique Clustering

In graph theory, there is a term clique. The term clique can be referred to as all nodes have something in common. A clique is a subset of graph G nodes where every pair of nodes are adjacent. This algorithm is NP-complete.

> A set, C, is a Clique of the graph G iff $C \subseteq$ vertex set of G and u,v $\epsilon C \wedge u \neq b \implies uv \epsilon E(G)$

In clique, clustering is partitioning the graph G into disjointed clusters where each cluster is formed per the mathematical definition Illustrated above. The clusters generated have a high internal correlation and minimized external agreement. However, maximizing agreements inside a cluster and minimizing the disagreements is an NP-complete problem. This is caused by a brute force search of the entire graph [30].

Set-Cover Clustering

The Set-Cover algorithm is one of Karp's NP-compete problems. By giving a universe of elements, $U = e_1, e_2, ...e_n$, the algorithm tries to find the minimum number of sets to cover all the elements in the given universe, U. There are two main aspects to using this algorithm, the first is pedagogical, and the second is the practical application. First, according to Sherry Liang, many greedy approximation methods have been used for combinatorial problems. Therefore, using the algorithm gives insight into the usage of approximation algorithms in solving NP-hard problems. Secondly, the application of this algorithm can be utilized in many different industries, such as airline industries or the location of cell towers. There are two sets in set-cover clustering, one universe U and one set S, where S is a set of subsets, $S = s_1, s_2, ..., s_n$. Each set in S must cover at least one element of U, and the cost c_i such that $c_1 > 0$. The objective of this algorithm is to find the minimum cost. Sub-collection of sets $X \subset S$ that covers all elements in U. This algorithm is applied in different ways, but the one used to approximate near-optimal solutions for large sets is the Greedy approximation algorithm. This algorithm solves the problem in $lnm - lnlnm + \Theta(1)$ [31] though an iterative greedy heuristic process which selects the largest number of elements in U adds the set s_i to S and removes the elements in s_i in U until $U = \varnothing$ [2].

Code listing 2.2: Pseudo Code for Set covering problem by Sherry Liang [2]

```
T equals {}
while U not equals {}:
    select s_i element of S that covers the highest number of elements
    add s_i to T
    remove s_i from U
End while
return S
```

2.4 Statistics

This Section introduces how to compare the results in Chapter 4 which is used to answer **RQ2**, and **RQ3**.

2.4.1 Correlation Algorithms

Correlation is used to find certain security between two measurements and find the connection between the two measurements. In statistics, correlation normally refers to how much a pair of bivariate or random data are linearly related. Correlation values are useful to indicate a predictive relationship between the two measurements. According to Wikipedia, the most common of the correlation coefficients are pearson's correlation [32].

Pearson's

The Pearson's correlation measures the linear strength between two variables, denoted by r [33]. Pearson's correlation takes each pair of measurements from the y-axis and x-axis and plots them in a graph as illustrated in Figure 2.7. When low values on the x-axis are paired with low values on the y-axis, and high values on the x-axis are paired with high values on the y-axis, a straight line is drawn with a positive slope representing this trend. Using this trend, we can only predict a result on the y-axis by having the x-axis value. However, if the relationship between the two measured variables is weak, this prediction becomes harder to assume. This means that a weak relationship between the variables where the data points are further away from the trend line gives a small correlation value as seen in Figure 2.7 where $r = 0.3$, and data points closer to the trend line gives a large correlation value as illustrated in Figure 2.7 where $r = 0.7$. This correlation value can range between 1 and -1 given the trend line slope, a positive slope gives a positive value, and a negative slope gives a negative value. Furthermore, the correlation value of 0 will indicate no association between the two variables. The correlation value r is calculated using the Equation 2.1 [34].

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{2.1}$$

n is sample size
x_i and y_i are sample points in index i
$\bar{x}, \bar{y}$ are the sample mean

By using the equation 2.1, we can calculate the trend of two given pairs of measurements. A probability value or a P-value is used to validate the correlation with a statistical measurement to find statistical significance between the measurement. The probability value (p-value) tells the probability that randomly drawn data points will result in a similarly strong relationship or stronger. Thus the smaller the p-value, the more confidence we have in predicting that trend line.

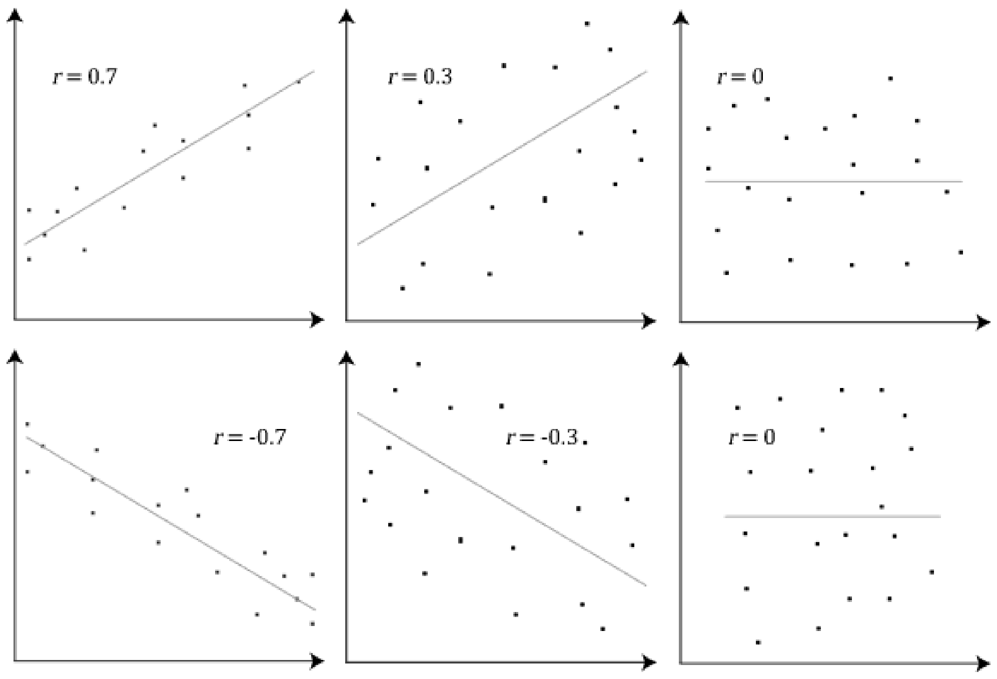


Figure 2.7: Examples of how the linear correlation graph is visualized [33].

Spearman's

While Pearson's correlation uses linear relationships based on data assumptions such as interval or ratio level, linear relativity, and bivariate distribution, Spearman's correlation uses rank to correlate values with the usage of a monotonic function. A monotonic function never increases or decreases as the independent value increases. So if the value on the x-axis increases, the y-axis should constantly increase or decrease. If the value increases and decreases sometimes, it is not a monotonic function. Spearman's correlation uses monotonic relationships between paired data, and like Pearson's, the correlation value is $-1 \leq r_s \leq +1$. Unlike pearson's, Spearman's correlation is a nonparametric statistic, which means it does not require normality of the data. Spearman ranks the samples on the x-axis and y-axis from 1 to n, where n is the number of data points. After the ranking, Spearman uses these rankings to plot the data points. From these datapoints spearman utilizes Pearsons correlation equation shown in section 2.4.1 to give a score r_s [35].

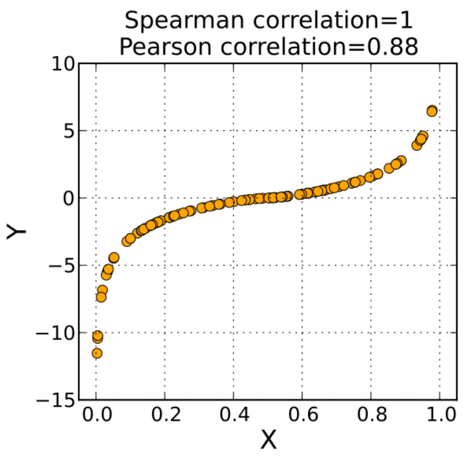


Figure 2.8: Example of monotonic function with pearson and spearman correlation [35].

T-test and Power analysis

The Independent samples t-test is a statistical method for finding statistically significant differences between group means. The test starts by formulating a null hypothesis,H_0, that the means are equal between the observed groups. However, to complete the t-test, there are some assumptions about the dataset.

- **Assumption of independence**: The two groups in question needs to be independent of each other.
- **Assumption of normality**: The data should be approximately normally distributed.
- **Assumption of homogeneity variances**: The data should contain an equal variance.

[36] If these assumptions are met, the t-test can first calculate a t-value. This t-value is calculated with Equation 2.2. However, if the variance in the two independent groups is not equal, then Equation 2.3 is used [37].

$$t = \frac{\bar{x_1} - \bar{x_2}}{s_p\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}} \tag{2.2}$$

$$t = \frac{\bar{x_1} - \bar{x_2}}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}} \tag{2.3}$$

$\bar{x_n}$ is the mean of n'th group
n_n is the number of elements in the n'th group
σ_n is the standard deviation in group n

s_p is the pooled standard deviation

The t-value calculated from the Equation 2.2 or 2.3 can be used to find the p-value which is the probability of when to reject the H_0 hypothesis based on α. The researcher sets α to accept the H_0. If the p-value is below α, the comparison is considered significant.

Furthermore, when conducting multiple comparisons causes the probability of observing a false positive to inflate. This is to the Family-Wise Error rate (FWER), which is the probability of at least one false conclusion in a series of hypotheses,H_n. The FWER is calculated with $FWER = 1(1-\alpha)^n$. If $\alpha = 0.05$, with 6 runs then the $FWER = 1-(1-0.05)^6 \approx 0.26$, which tells that there is a 25% chance of obtaining a false positive. However, by using Bonferroni correction which simply states to divide the α on number of runs; $\frac{\alpha}{n} = \alpha_{Bonferroni}$. Doing so reduces the chance of obtaining a false positive to $\approx$ 5%, or the original value. As well as, multiplying the p-value will have the same effect [38].

Conduction of power analysis can show the confidence of the t-test. Power analysis is the probability of the hypothesis test detecting an effect and if there is a true effect. This can be used to estimate the sample size required for the experiment to detect a true significance between the groups, which leads to the concept of effect size [39]. The effect size measures the sizes of differences between the groups' means, which measures the strength of the relationship between the two groups. The effect size is calculated using Equation 2.4. The effect size can also be used to find if there is a negative effect or a positive effect by not taking the difference but rather m_1-m_2. The effect size refers to the effect a sample has. Therefore, the larger the effect size, the smaller the sample size is needed to achieve the same result [37].

$$E = \frac{|m_1 - m_2|}{\sqrt{\sigma_1^2 - \sigma_2^2}} \tag{2.4}$$

m_n is mean of group n
σ_n is the standard deviation in group n

2.4.2 PCA - Principal component analysis

Principal Component Analysis (PCA) is a dimensionality reduction technique often used to reduce high dimension data structures into lower dimension space while maintaining as much of the data's variation as possible. PCA describes the variance-covariance of a data set through sets of linear combinations of variables called principal components (PC). PCA will also give the percentage of variation for each PC. This variation gives the amount of accuracy contained if a dimensionality reduction is completed. This is done by finding the direction with maximal variability in the graph given by the original data points. The first direction (PC1) is the direction that best fits the data, while the next $i+1$ vectors maximize the variance while being orthogonal to the previous vector. This is done by standardizing each datapoint, computing a covariance matrix, and computing eigenvectors and -values to identify principal components [40].

2.5 Deep learning

Deep learning is inspired by how information is processed in the brain, where the neuron is in focus. Each neuron contains a cell body and an axon, as Illustrated in figure 2.9. The neuron receives signals from a previous neuron, processes it, sends the signal forward to one or more neurons, and is stored as an activation in the cell body of the next neuron. The axon, where the neurons are connected, is called weights. These weights are initially randomized. However, they are tweaked to give the desired output as the individual learns. Finally, these neurons can be stacked to create a layer of neurons called perceptrons [41].

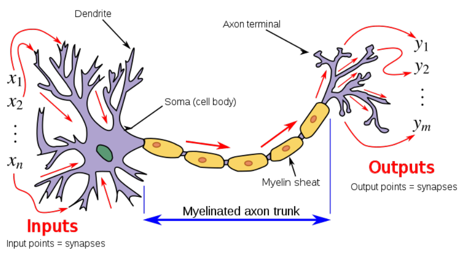


Figure 2.9: Illustration of a neuron based on perceptions[42].

2.5.1 Perceptrons

As in the brain analogy, deep learning uses the same technique. Where the cell body is a node, and the axon is the edge that connects the nodes from one layer to the next layer. As Illustrated in Figure 2.10, The first layer is called an Input layer. This layer is where the data is fed. The next step is the forward propagation of the values through the network, where every node in the $n-1$ layer is connected to the n layer. To retrieve the activation value a_0^n, each node in the $n-1$ layer will multiply with their weight value and take the sum of those values. The next optional step is to use an activation function to modify the values. In Equation 2.5 is how the multiplication and summation is done [43].

$$\sigma(\begin{bmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,n} \\ w_{1,0} & w_{1,1} & \cdots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & \cdots & w_{k,n} \end{bmatrix} \begin{bmatrix} a_0^0 \\ a_1^0 \\ \vdots \\ a_n^0 \end{bmatrix}) = \begin{bmatrix} a_0^1 \\ a_1^1 \\ \vdots \\ a_n^1 \end{bmatrix} \tag{2.5}$$

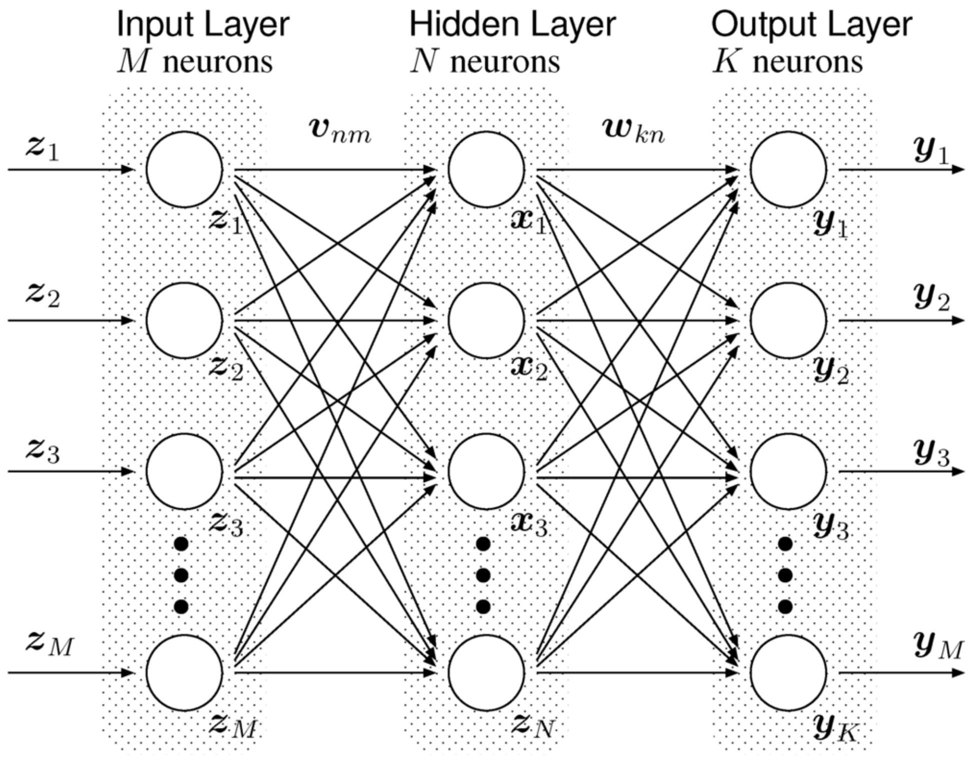


Figure 2.10: Structure of a multilayer perceptron [44]

Where W is the weight matrix between layer a^0 and a^1. a^0 is the activation value stored in the node, while a^1 is the activation value calculated from $\sigma(W \cdot a^0)$. σ is an activation function shown in equation 2.6 where the function contains the input x between 0 and 1; $0 \leq \sigma(x) \leq 1$ [43].

$$\sigma(x) = \frac{1}{1 + e^{-1}} \tag{2.6}$$

As shown in figure 2.10, there can be multiple layers after each other with different amounts of nodes in each layer. A basic deep learning perceptron architecture is based on an input layer, d number of hidden layers, and one output layer, where $d \epsilon \{0, \mathbb{N}\}$. For the network to learn, each weight in the network needs to be rewarded or punished based on its overall performance. The network does back propagation on the network based on a loss value given by the loss function, while an optimizer tries to optimize the network.

2.5.2 Convolutional neural networks (CNN)

Convolutional neural networks (CNNs) are deep learning algorithms used in spatial structures such as images. The algorithm picks up and detects features or common patterns in spatial structures. Rather than connecting all of the nodes between each layer such as described in section 2.5.1, there is rather a filter structure as seen in figure 2.11 that strides over the spatial input structure, and generates a convolved feature. The idea of the convolution as described is to reduce the spatial structure while extracting features from the structure that is critical for getting a good prediction. Multiple filters can be used for learning and extracting different features. With multiple hidden layers of CNNs, the network can learn a hierarchy of features from the input. From lower-level features such as lines, crosses, and dark spots, to more high-level such as facial structures,[45].

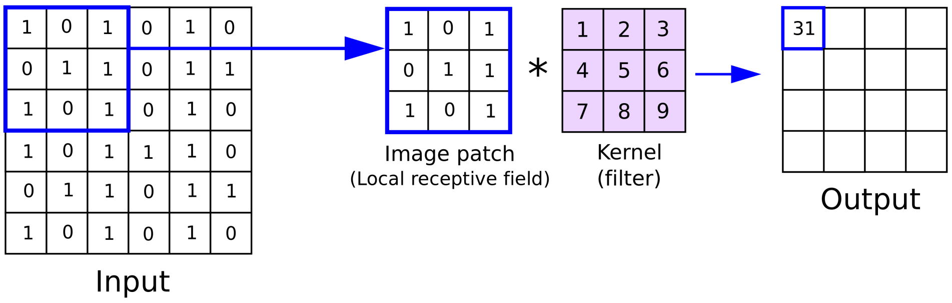


Figure 2.11: Illustration of kernel strides across the input matrix. [46]

$$Z = x * f \tag{2.7}$$

The convolutional step is often represented by an asterisk (*). Where f, which is the filter strides across the input matrix X, which outputs a convoluted matrix Z [47]. This is illustrated in Figure 2.11. Pooling is also a layer usually done in a convolutional neural network. This filter strides in the same way over a spatial structure such as the convolutional filter. However, rather than extracting features, the job of the max pool is to enhance dominant features from the structure [45].

2.5.3 Backpropagation

As stated in Section 2.5.1 deep learning algorithms uses loss functions and optimizers to calculate the accuracy of the network and how to optimize the network. Multiple loss functions and optimizers are utilized in deep learning for actual learning. However, this section will focus on the general application of both the loss and optimizer. After the forward-propagation the network creates a calculated output $\hat{y}$, the loss function utilizes an error function $E(\hat{y}, \theta)$ where θ... This error is propagated back to the previous layer, which is used to alter the weights to minimize the error. This is done to calculate the partial derivative or the gradient concerning the weights. This is done by using the chain rule shown in equation 2.8 [48].

$$\frac{\delta E}{\delta w_{ij}^k} = \frac{\delta E}{\delta a_j^k}\frac{\delta a_j^k}{w_i^k j} \tag{2.8}$$

$$\delta_j^k = \frac{\delta E}{\delta a_j^k} \tag{2.9}$$

$$\frac{\delta a_j^k}{w_i^k j} = \frac{\delta}{w_i^k j}(\sum_{l=0}^{r_{k-1}} w_{1j}^k o_l^{k-1}) = o_i^{k-1} \tag{2.10}$$

The equation shows the change in the error E with respect to w_{ij}^k, which can be equal to the change in E for a_j^k times the change in a_j^k with respect to w_{ij}^k. By using the chain rule, the output layer can be calculated.

$$\frac{E}{\delta w_{i1}^m} = (\hat{y} - y)g_o'(a_1^m)o_i^{m-1} \tag{2.11}$$

Where g_o is the activation function for the output layer, such as the sigmoid function is shown in Equation 2.6, where m is several layers, and o_i^{m-1} is the output from node i in the layer behind. Note that the last layer is only one output perceptron.

For the hidden layers, we utilize the chain rule also the error from the layer in front, which gives:

$$\frac{\delta E}{\delta w_{ij}^k} = \sigma_j^k o_i^{k-1} = g'(a_j^k)o_i^{k-1}\sum_{l=1}^{r^{k+1}} w_{jl}^{k+1}\sigma_l^{k+1} \tag{2.12}$$

By calculating the partial derivative of the function E concerning the weights in the layers $1 \leq k < m$ [48].

2.6 Related Works

Gene Regulatory Networks

In 'Single-cell transcriptomics unveils gene regulatory network plasticity' Iacono *et al.* proposes a different computational framework that can calculate large-scale gene regulatory networks; find the interaction between transcriptions of genes in scRNA-seq. According to Iacono *et al.*, traditional approaches have focused on bulk RNA-seq, while only little effort has been made to derive regulatory networks from single-cell transcriptomics and has been restricted to certain network properties. Further stating that single-cell sequencing is an ideal technology for monitoring interactions between genes in cells [17]. The paper 'Single-cell transcriptomics unveils gene regulatory network plasticity' demonstrates the value of regulatory networks using scRNA-seq data by applying this new computational framework to 11 mouse organs. This is done by three main steps, Data pre-processing, correlation of genes in Z-score space, and building of the regulatory network.

The first step, *Data pre-processing* uses a novel bigSCale framework [17] that handles sparsity and noise from scRNA-seq data. To handle this, bigSCale uses large sample sizes to estimate an accurate numerical noise model. The framework also clusters the cells using recursive clustering to the highest feasible granularity and runs a differential expression between these clusters and the genes they contain. Which indicates the expression changes between two clusters. For the next step, *Measuring correlations in the Z-score space*, Iacono *et al.* utilized *Pearsons*, *Spearman*, and *Cosine* coefficients. these correlations were calculated between each gene using Pearsons and cosine, and spearman was used in a later stage for further control. In the last stage, Iacono *et al.* created an undirected network that utilized Clique based clustering to cluster from the correlation matrix in step two.

By using the technique mentioned above, Iacono *et al.* concluded that the proposed novel computational framework with a regulatory network approach could be used to maximize the biologically relevant information acquired. This technique detects regulatory changes in genes such as the amount of expression and location, which are invisible to traditional techniques such as clustering or differential expression Iacono *et al.* [17].

Gene regulatory network inference in single-cell biology

The paper 'Gene regulatory network inference in single-cell biology' proposed by Akers and Murali is a meta-study researching an ensemble of Gene regulatory network methodologies. Akers and Murali researched the development in GRN, methods to evaluate them, and simulation of scRNA-seq. According to Akers and Murali, single-cell transcriptomics data proposes challenges for GRN inference. These include cell-to-cell stochastic variation in the gene expression, cell cycle, and sparsity due to inadequate sensitivity in scRNA-seq with low expression values. The methods researched were Regression, bayesian network, Boolean network, Differential equations, and TF binding motifs. To evaluate these methods, a ground-truth GRN was generated from synthetic networks to evaluate the aforementioned methods. By doing so, Akers and Murali concluded that even the best methods had an accuracy marginally better than a random predictor. Furthermore, GRNs usually contain tens of nodes which is an underrepresentation of the ground-truth GRN [16].

Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning

'Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning' by Alipanahi *et al.* Proposes predicting and finding motifs in DNA- and

RNA-binding using deep learning rather than traditional techniques. These motifs play a central role in gene regulation, such as transcription, which was introduced in Section 2.1.3. Position Weight Matrices (PWM) are commonly used in finding sequence specificities of these genes are Position Weight Matrices (PWM). PWM is derived from aligned sequences that are believed to have related functionalities and are used to scan over sequences to detect potential binding sites. However, Alipanahi *et al.* utilizes Convolutional neural networks, which can detect the location of binding sites within sequences, even if the location of the binding site is unknown within this sequence. According to Alipanahi *et al.* there are multiple challenges with learning models to identify positions of binding sites and their motifs.

Firstly, the data comes in different qualitative forms. There might be both biological and technical noise. Biological noise is the stochastic gene expression of the protein across a homogeneous population of cells. The noise can be attributed to the pseudo-time of multiple cells, in that the same cells might have different states during the snapshot of the genomics. Technological noise is generated during the Sequencing of the gene, such as described in Section 2.2.1. An example of such noise is during the sequence extraction in the last step, where the machine reads the fluorescent nucleotides. If there are many of one base type in a section and only one of another base, the singular base might be overwritten.

Secondly, the quantity of the dataset is large, with 10 000 to 100 000 sequences, and it is computably difficult to incorporate it all. Lastly, the data gathered has technological noise and biases. Alipanahi *et al.* uses DeepBind to address these challenges by generalizing across multiple different types of datasets, tolerating a degree of noise and mislabeled data, and performing the tasks in a GPU-server for parallelization, which can handle much data throughput.

DeepBind by Alipanahi *et al.* uses sequences with lengths from 14-101 bases. These sequences are labeled using continuous measurements or binary classification. Deepbind uses four stages to compute a binding score $f(s)$ where s is the sequence in question:

$$f(s) = net_w(pool(rect_b(conv_M(s)))) \tag{2.13}$$

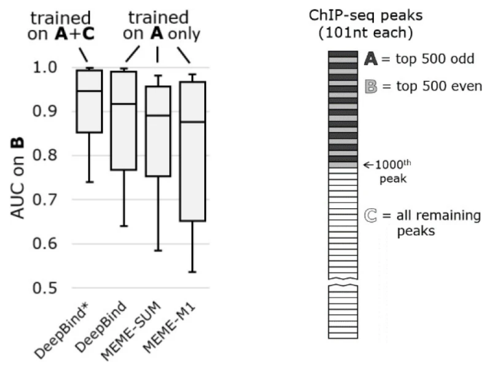


Figure 2.12: "The ChIP-seq performance from Figure 3e are reproduced at left with extra annotations for clarity. At right is the breakdown of ChIPseq peaks used to train a model on each ChIP experiment. We train each method on peaks labeled A ("top 500 odd"), then test each method on peaks labeled B ("top 500 even"). DeepBind* is a special case where we show that including the lower -ranked peaks labeled C ("all remaining peaks") in the training set can significantly improve the accuracy when scoring the top-ranked peaks labeled B" - Alipanahi *et al.*[1].

The sequence *s* is fed into a CNN, rectified, pooled, and then fed into a perceptron network. These stages will be explained in detail in Chapter 3.

DeepBind has been used to identify binary classifications for NRA binding sites. For this Alipanahi *et al.* used 506 preprocessed *in vivo* ENCODE ChIP-seq datasets. To train DeepBind, they used 101-bp sequences centered towards the point source. They compared the model to MEME-ChIP to the same peaks and scored test sequences from either the top PWM (MEME-M1) or the sum of scores for all PWMs (MEME-SUM). Alipanahi *et al.* achieved a higher AUC average score (0.85) than both MEME-SUM (0.82) and MEME-M1 (0.78). This is illustrated in Figure 2.12. Alipanahi *et al.* illustrates that using deep learning for Binary classification and motif exploration has great promises and can triumph over traditional technologies such as PWM as shown [1].

Chapter 3

Methodology and Tools

This chapter will introduce the methods used to achieve the results shown in Chapter 4. First, is the data used in this project and how it was processed. Secondly is how to cluster a gene expression and label to classes and conclude the best clustering method.

3.1 Data | 10xGenomics - PBMC

This section used some of the work of Iacono *et al.*, from the paper 'Single-cell transcriptomics unveils gene regulatory network plasticity' presented in Section 2.6. These include the methods aforementioned such as Data-preprocessing and differential expression.

The analysis presented in the Result section of this paper utilizes PBMC-cells. PBMC, also known as peripheral blood mononuclear cells, contains immune cells that work together to protect from harmful pathogens. PBMC contains three main cell types: lymphocytes, monocytes, and Dendritic Cells. 10xGenomics published the dataset utilized for the paper (2017)[1] which used Illumina's sequencing as described in Section 2.2.2. The dataset contains a total of 8581 cells and represents 33538 genes. The dataset used has been pre-filtred from mitochondria genes and empty reads. More details can be seen in Appendix A.

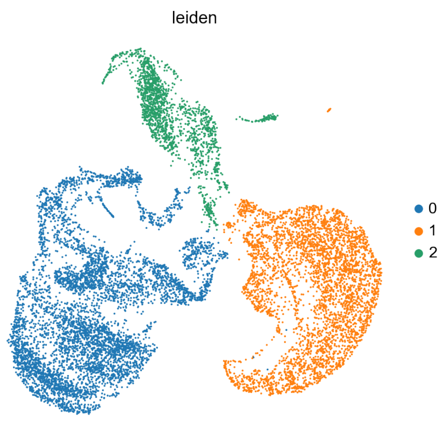


Figure 3.1: PCA dimensionality reduction of gene expressions vs. cells

[1]https://support.10xgenomics.com/single-cell-gene-expression/datasets/2.1.0/pbmc8k

3.2 Data processing and correlation matrix generation - BigSCale2

For data processing of the data from 10xGenomics, the tool BigSCale2 was used shown in the paper 'Single-cell transcriptomics unveils gene regulatory network plasticity' by Iacono *et al.* For a more comprehensive guide for this method, please read 'bigSCale: an analytical framework for big-scale single-cell data.'

For the Data pre-processing, the tool BigSCale was utilized to process the data for generating a correlation matrix. According to Iacono *et al.* the tool first creates a numerical model which allows calculating distances between cells quickly. This is done by grouping the cells that have similar transcriptomes. The cells that are in the same group are treated as replicates to evaluate noise. The grouping is done through normalization of the reads to the library size, taking the log_{10} for each gene, then normalizing those values again. It then used Pearson's correlation for each of the cells and hierarchical clustering. The cells clustered underneath 10% - 15% of the total hierarchical tree height are counted as biologically similar. Then, a P-value is assigned to each gene in the same cluster in the likelihood of a change of expression from one cell to another cell.

Furthermore, the genes are assigned a P-value representing the likelihood of change from one cluster to another. This is done through a change in expression between each of the cells from one cluster to another. All cells are pairwise compared. Genes that repeatedly differ in their expression are given a higher score than those that do not. This allows computing a correlation between all cells using P-values instead of directly expressing values. Which generates a correlation matrix, C_m that will be taken into use in this thesis [17, 49].

3.3 Clustering

In order to answer research question one (**RQ1**), three clustering methods were developed and tested to derive the clustering method that produces the optimal result.

3.3.1 Clique based clustering

For Clique based clustering, the cohesion for each cluster should be high since each cluster can only be generated with nodes that correlate a given threshold with each node in the cluster. This was generated by using the tool bigSCale [2]. According to Iacono *et al.*, the networks were built to retain only the top 0.1% of correlations. This is done to prevent technical factors from being introduced [17]. However, since it did not generate as many clusters as stated by Akers and Murali in the Related work section 2.6 nor large enough clusters caused, this clustering method was retracted from further work.

[2]https://github.com/iaconogi/bigSCale2

3.3.2 Set Cover

For set-cover we utilize the correlation matrix, C_m, and generate set-cover clusters as described in Section 2.3.2. Firstly, a lower threshold is set, t_l. Secondly, the algorithm selects a gene called a hub node in the Universe U, and adds all genes that has $\geq t_l$. The added genes creates a hub/cluster with the hub node and is added to a set S. the genes containing in S is removed from U, and a new gene from U is selected. For Results in Chapter 4 the largest cluster is used.

3.3.3 Hierarchy

The correlation matrix, C_m, introduced in Section 3.2 needed to be converted to a distance matrix for Hierarchy clustering. Since the correlation is using Pearson's and Spearman's, the conversion uses Equation 3.1. Where x is the correlation value for each location in C_m.

$$d(x) = 0.5 * (1 - x) \tag{3.1}$$

This gives a distance matrix, D, that the agglomerative clustering can generate a recursive hierarchical clustering. As stated in Section 2.3.1, the pairs of nodes, in this case, genes, are merged based on the distance between each cluster until there is only one cluster. Using the dendrogram as seen in Figure 3.2 a lower threshold t_l is set, and clusters are generated at a distance lower than the given threshold. As in Section 3.3.1 the largest cluster is used to generate the results.

3.3.4 Cluster comparisons

To test the different clustering methods and compare their cohesion, separation, and coverage results. Each method took the 20 most significant clusters and used them for comparisons, except for the clique-based. Cliquebased clustering only utilized 10 clusters since that is what its maximum produced.

Cohesion

The cohesion is to find the robustness of each cluster in each method. The robustness illustrates how tight each gene compares in terms of transcriptomes for the cluster it houses. The better cohesion might be better for a positive dataset in a binary classification since it contains less biological- and technological-noise. Next, the cohesion for each cluster is calculated by taking the distance for each pairwise gene, summing them up, and averaging them. Finally, the distance calculation is done for each cluster and gives an overview of the cohesion of each method. An example of this is illustrated in Figure 3.2, where the cluster has three genes with

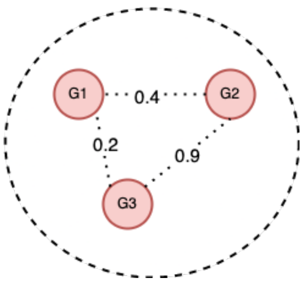


Figure 3.2: Example of a singular cluster with three genes.

edges between them all. The edges e are summed up and averaged $\frac{\sum_{i=0}^{n} e_i}{n}$ where n is the number of edges.

Separation

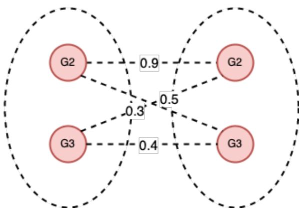


Figure 3.3: Example of two clusters with edges between each gene.

The separation for the clusters is how much breakage is between each cluster. The separation illustrates how easy it is to distinguish the different clusters in a given method. This method uses the distance matrix to calculate the span between each cluster regarding the genes it contains. Firstly, each gene in cluster c_i is compared to each gene in cluster c_{i+j} where j is iterative incremented. Secondly, the separation value is calculated by taking the weight of each edge in between the respected gene and then averaging the sum. This is illustrated in Figure 3.3 where the left cluster's genes' edges between the right's are summed and averaged.

Graph coverage

The graph coverage gives an insight into the coverage by different cluster methods. This illustrates how much of the graph covered. For example, if the graph coverage value is high will show that the separation of the selected clusters and non-marked clusters is high, and therefore the selected method covers a lot of the graph. The ground pillar for this method uses hierarchical clustering. Firstly, Each cluster in the hierarchical method is compared to groups given by the cluster methods we want to take the coverage of, including clique-based, set cover, and hierarchical. These comparisons are to check if one of the genes in the method exists in the ground pillar clustering. If it does, then the cluster will be marked. Secondly, all the marked clusters are removed from the ground pillar clustering, and the remaining clusters are sorted based on size, from largest to smallest. Thirdly, the separation from each marked cluster is taken from the sorted list, where the number of clusters is checked.

3.4 Deep learning

In order to answer research questions, two and three (**RQ2**, **RQ3**), the deep learning model from the paper 'Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning' was used to experiment with the cohesion and sequence length.

3.4.1 Data processing

Unbalanced data set

In binary classification there is a positive set $P_s = p_0, p_1, ..., p_n$ and a negative dataset, $N_s = n_0, n_1, ..., n_m$. The positive dataset is selected by applying a lower threshold, t_l. Then a cluster is chosen from those generated from the clustering algorithm. However, this leads to a small positive dataset and a large negative set, $P_s \ll N_s$. To tackle this problem is in two steps. Firstly, the biological information for clusters close to the positive set will have many similarities, which will cause the deep learning model to have difficulties distinguishing. Another threshold is applied, upper threshold t_u, to combat the specific issue. The job of t_u is to create distance between P_s, and N_s. This is done differently for the different clustering algorithms. Set-Cover removes clusters from the negative set where the correlation of the negative clusters' hub node is higher or equal to t_u.

For Hierarchy, removing close biologically related negative clusters is done by creating clusters for t_u, C_u^j, where j is the number of clusters generated by the t_u. If $P_s \subset C_u^i$, $0 \le i \le j$ then the cluster found, C_u^i is removed from the negative set. This method is illustrated in figure 3.4.

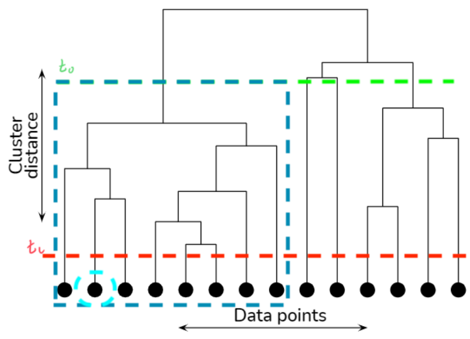


Figure 3.4: Since we want to create a Positive set and negative set for binary clustering, we have used two thresholds, upper-threshold t_u, and lower-threshold, t_l. The lower threshold should be chosen, so the cluster is cohesive enough while having enough positive values for the Deep learning model. The upper threshold should be chosen high enough to remove genes that resemble the positive data set. To illustrate this, the green dotted line is t_u and the red is t_l. First, a cluster is chosen where the distance $< t_l$. In the figure, this cluster is marked in a dotted circle. The other clusters are seen as negative sets. To remove genes that resemble the positive set in the negative, an upper threshold is taken. Where $t_u = 0.43$, all clusters inside the blue square except the positive set are removed from the training set. This leaves the clusters that are outside the blue square as the negative data set. [29]

Secondly, a generator has been created for the model to train on a majority of negatives. The model stochastically selects elements from P_s and N_s, so the sampling selected are equal in size. However, this causes the model to train on various negatives and hopefully see the negatives as noise while the positives are more consistent in the sequence.

Sequencing

The gene's transcriptome and their identification are from the dataset 10xGenomics. However, the 10xGenomics data set does not include sequence and metadata for each gene. Firstly, The metadata can be downloaded from Ensembl genome browser[3] as a .gtf file, which includes the human genome (GRCh38), which is utilized in this paper. This data includes the start and end position of the transcriptome, the transcript direction, and which gene a chromosome contains. An example of this can be seen in Table 3.1.

Table 3.1: Table shows output from the .gtf file after parsing with pyranges.

Chromosome	Feature	Start	End	Strand	gene id
1	gene	1471764	1497848	+	ENSG00000160072
1	gene	2581559	2584533	+	ENSG00000228037
Y	gene	18772705	19077416	-	ENSG00000176728

The .gtf file given from Ensembl can be parsed using the tool Pyranges[4] and was used to generate the Table. Secondly, to fetch each sequence for the genes since we do not need the entire human genome, just a subset of it. Then the GRCh38 dataset fetched from Ensembl was filtered based on the PBMC genes to reduce its size. Furthermore, each chromosome was downloaded from Ensembl to map the gene to the sequence's start and stop position in that chromosome. Now we have all the sequences we need. However, as stated in Section 2, the promotor is before the gene. To read the sequence for promotor we read the s_b before the gene, where $s_b \epsilon 500, 1000, 1500, 2500$. The transcriptome can be reversed transcribed as fetched from the metadata mentioned earlier, which causes the promotor to change direction from being on the 3' to the 5'. The sequence also needs to be reversed, and complementary transcribed.

Table 3.2: Sequence length and how many neucleotides read from start location for gene start shown in table 3.1

Sequence length	Upstream (3')	Downstream (5')
500	400	100
1000	800	200
1500	1100	300
2500	2000	500

After separating the sequences obtained as the method described above, the next step is to convert the sequence into a spatial matrix. Each sequence contains the base pair alphabet A, C, G, and T, as stated in section Theory. In addition, however, the sequence might contain N, which illustrates an unidentified nucleotide. This new letter increases the alphabet to A, C, G, T, and N. Furthermore, a motif length, m, needs to be defined. The motif is the prominent sequence that the deep learning model tries to extract from the sequence given in this thesis $m = 24$,

[3]https://www.ensembl.org/index.html
[4]https://github.com/biocore-ntnu/pyranges

which is placed about the Related work Section , as well as $k = 16$, where k is the number of motifs to be found.

The Equation used to convert a sequence into a spatial matrix for the deep learning model is presented in the next section. It iterates through the sequence where $0 < i < n$, where n is the Sequence length, and $0 < j < 4$. This Equation 3.3 and 3.2 is from the paper 'Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning' referenced in Related works.

$$S_{ij} = \begin{cases} .25, S_{i-m+1} = N \vee i < m \vee i > n-m \\ 1, S_{i-m+1} = j^{th} basein[A, C, G, T] \\ 0, otherwise \end{cases} \tag{3.2}$$

An example of this is illustrated below in Equation 3.3. In this example, $m = 3$, sequence s=AATG, which gives $n = 4$

$$S = \begin{bmatrix} .25 & .25 & .25 & .25 \\ .25 & .25 & .25 & .25 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ .25 & .25 & .25 & .25 \\ .25 & .25 & .25 & .25 \end{bmatrix} \tag{3.3}$$

3.4.2 Deep learning - architecture

While finding the best models related to sequencing analysis and classifications, one model was referred to by many papers, DeepBind by Akers and Murali in the paper 'Gene regulatory network inference in single-cell biology.' So this was the model this paper is based on.

As introduced in Section 3.4.1

$$f(s) = net_w(pool(rect_b(conv_M(s))))$$

The architecture of DeepBind combines a Convolutional neural network as described in Section 3.4.1 with Max pooling and a hidden Perceptron layer. The Convolutional neural network filter's job is to extract the features from the most prominent sequence, the motifs that are the binding sites, $conv_M(s)$. The method takes a matrix from the pre-processing stage, S, which is a $4x(n + m - 1)$ matrix. When the convolution is done, a rectification layer, $rect_b$, has been added to remove negative values $Y_{i,k} = max(0, X_{i,k} - b_k)$. After the rectification stage, the max pool, $pool(Y_{i,k})$ from each filter is taken $z_k = max(Y_{1,k}, ..., Y_{n,k}$, which gives a value for each of the bases. These values are fed into the hidden layer that contains both a bias and another rectification stage. After this stage, a dropout

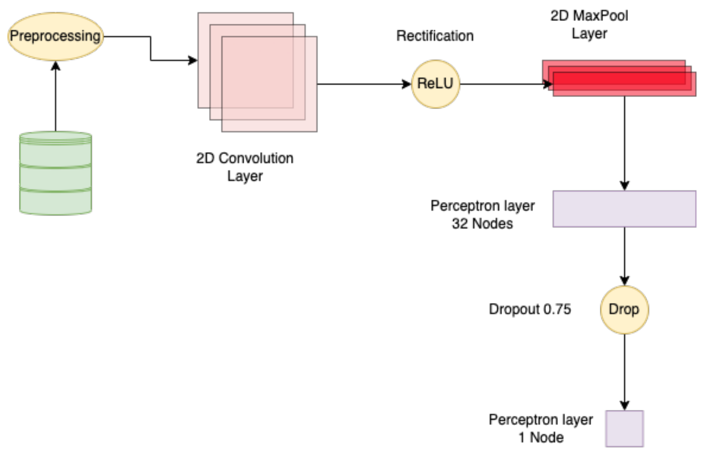


Figure 3.5: Illustration of the architecture of Deep learning model.

is utilized to combat overfitting. Lastly, a loss is calculated from the output, and the model uses backpropagation introduced in Section 2.5.3 to train the model's weights and biases.

3.4.3 Deep learning - Model Training

Training a Deep Learning model is essential for a good result. Training is done by choosing correct hyperparameters and using related research to choose batch size, epoch, optimizers, and loss functions. Stochastic gradient descent(SGD) was utilized for this thesis because of its promising results in the paper DeepBind. SGD used a learning rate of 0.0005, a decay of $1e^{-}1$, a momentum rate equal to Equation 3.4, and Nesterov enabled.

$$f(a,b) = (b-a) * \sqrt{x} + a \tag{3.4}$$

where x is a random uniform value between 0 and 1.

Table 3.3: Hyperparameters used in the deep learning model.

Hyperparameter	Value
learning rate	0.0005
Train-test split	0.1
Activation function	SGD
Loss function	binary cross-entropy
Epochs	200
Batch size	1

The loss function used is binary cross-entropy. This is because the classification for this thesis is based on binary classification. The Batch size was one and ran 200 epochs. A generator was used during training to combat overfitting as described in Section 3.4.1. The generator fetches stochastic values from both the positive and negative in a balanced fashion. Doing so gives an abundance of a diverse negative dataset. However, the model will train on much of the same positive values, hopefully learning its features.

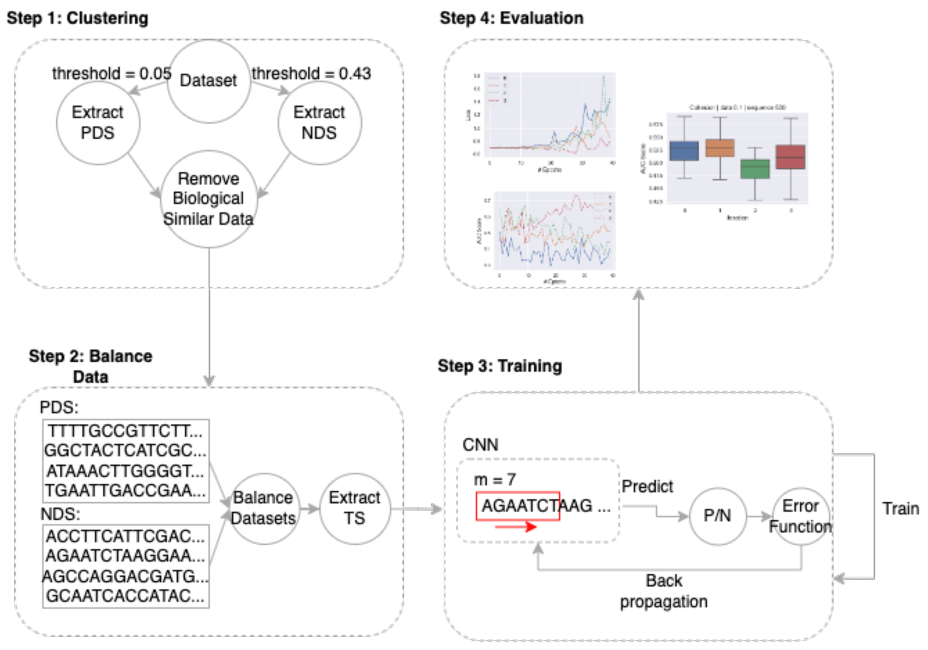


Figure 3.6: This Figure illustrates the pipeline of how the deep learning model manages to train on the given dataset. **Step 1: Clustering**, the dataset is separated into a Positive dataset (PDS) and a negative dataset (NDS) by using the clusters generated by a clustering method. In this example the lower threshold is $t_l = 0.05$ (*cohesion*), and negative dataset is $t_u = 0.43$. The cluster of the PDS does not exist in the NDS. The similar biological data contained in PDS is removed from the NDS as described in Section 3.4.1. **Step 2: Balance Data**, The data is balanced using a generator as described in Section 3.4.3. Then it acquires the Transcription sequence (TS) upstream (5') from the transcription start site from each gene based on the *sequence length*. **Step 3: Training**, the CNN moves a convolutional matrix with length $m = 7$ called the motif across the sequence acquired from the previous step. It predicts if the gene is in the negative or positive dataset. The error function calculates the degree of mistake, and the model learns using back propagation. This step is run multiple times. Lastly, **Step 4: Evaluation**, here is the result of how well the model managed to predict the given dataset based on the *cohesion* and *sequence length*.

3.4.4 Deep learning - Model Evaluation

Evaluation of the deep learning model is essential to understand the model's performance. The evaluation is done in 100 steps, and a new set of negative datasets is picked for each step. The original dataset has been train-test split. A new positive dataset has been selected that the model has never seen, and a new selected stochastic dataset for each step.

The performance metric of the evaluation is Area under the ROC Curve (AUC). AUC is utilized since the loss function is a binary classifier. Binary classification means that the outcome can either be positive or negative. Because of this, the classification can be true positive (TP), false positive (FP), true negative (TN), or false negative(FN). Where one is positive, and zero is negative. These classifications can calculate a Sensitivity and Specificity using Equation 3.5. Furthermore, the Sensitivity and Specificity are used in AUC in a ROC curve. The ROC Curve is plotted on a two-dimensional graph, where the y-axis is Sensitivity, and the x-axis is Specificity. As the name implies, AUC is calculated by calculating the area underneath the curve. A score of one shows 100% correct predictions.

$$\begin{aligned} Sensitivity = Recall &= \frac{TP}{TP+FN} \\ Specificity &= \frac{TN}{FP+TN} \end{aligned} \tag{3.5}$$

3.4.5 T-test and the power analysis

A power analysis was used to find the effect value and calculate the power for a specific sample size to compare sequence length and cohesion runs. The power analysis is completed to find the confidence of the results when comparing each of the runs. By knowing the statistical power, the t-test value is more supported. Some steps were to calculate the necessary sample size for a specified power. The first step was to calculate the pooled mean for each sample. For cohesion, that would be the total mean of 0.05, 0.75, 0.1, 0.125 individually, $\bar{x}_i$, where i is which cohesion. A pooled standard deviation was calculated for each pairwise run, σ_{ij}. With these values, the effect size can be calculated. This is done by taking $e = \frac{x_i - x_j}{\sigma_{ij}}$. When the effect size has been calculated the python library TTestInd-Power's[5] method solve_power utilized the effect size with an $\alpha = 0.05$ and power $= 0.8$ to find the sample size of the effect.

3.4.6 Biological validation of results

By using Biological confirmation can visualize that the model learned features that exist in either the negative or the positive dataset trained on. This is done by taking the convolutional layer's weights, a $(24, 4, 16)$ matrix, where there are 16 represented motifs of length 24. The 4 represents each base, A, C, G, and T. These weights is visualized in Appendix B. Then, by iterating through each of the motifs

[5]https://www.statsmodels.org/dev/generated/statsmodels.stats.power.TTestIndPower.html

detected, we can find the base that is most represented for each column in the filter. This generates a sequence that the model found identifiable in the runs.

```
A [0 0 1 1 1 0 0 0 0 0 1 1 0 0 1 0 1 1 0 1 0 0 0 0]
C [0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0]
G [0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1]
T [1 0 0 0 0 1 1 1 1 0 0 0 1 0 0 0 0 0 1 0 1 1 0 0]

TGAAATTTTCAATGACAATATTGG
```

Code listing 3.1: Example of motif found by the deep learning model

This sequence can be added to the JASPAR DB to check if the motif is found to exist in the gene we have. These motifs are visualized in Appendix C.

Chapter 4

Results

This chapter will present the results of applying the methods introduced in Chapter 3, Methodology and Tools. Firstly, the results from 10xGenomics are presented. Secondly, the comparisons of the different clustering methods. Thirdly, The results from the DeepBind architecture with different clusters and sequence lengths are introduced.

4.1 Clustering

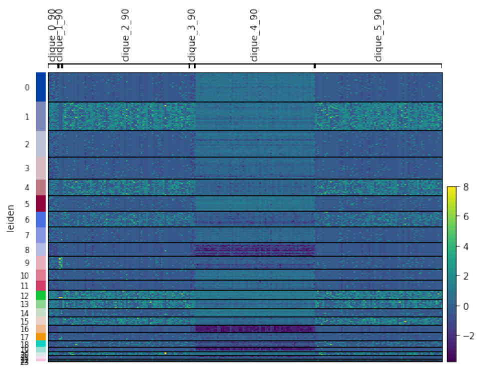
To answer **RQ1**, a visualization of each clustering method, Clique, Set-cover, and Hierarchy, is illustrated through a heatmap in Figure 4.1. Both Set-cover and Hierarchy illustrate only the 20 largest clusters. On the other hand, the Clique method only shows 6 clusters because GRN produced 10. However, only 6 of them contained more than one gene. Therefore, each cluster visualized contains a different amount of genes based on the top 20 biggest clusters for Clique based contains 3466 genes, set-cover with 13591, and Hierarchy with 13591.

Table 4.1: Table illustrates different positive dataset sizes based on lower threshold distance. The upper threshold t_u is 0.43, and a lower threshold, $t_l = 0.05$. The table is generated using Hierarchical clustering

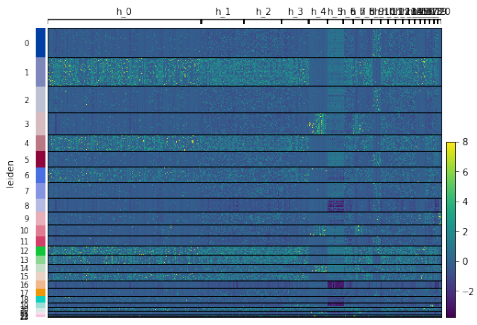
Lower threshold	Positive	Negative
0.05	77	7093
0.075	279	6618
0.1	686	6618
0.125	974	6618

4.1.1 Visualization

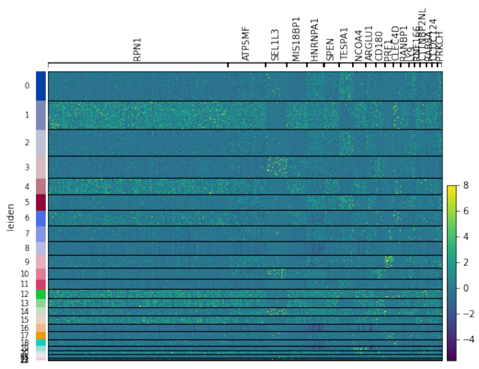
The Heatmaps in Figures 4.1a, 4.1c, and 4.1b are based on the gene expression on the x-axis and are placed in the clustering of Figure 4.1d. Each gene expression is mapped to a cell cluster in Figure 4.1d. An example of this is cluster 1 in Figure 4.1d has a high gene expression in Figure 4.1c and is represented by the lighter color. The dark locations represent low gene expression in the cell cluster, and yellow represents a high expression.



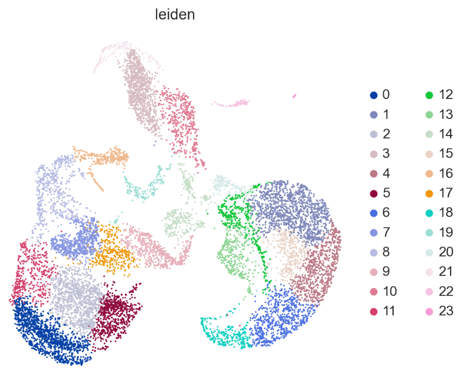
(a) Clique clustering using the tool BigSCale on correlation 0.90 (0.05 distance).



(b) Heatmap of Hierarchy clustering with a threshold of 0.05 (0.90 correlation). This Figure also illustrates the top 20 largest clusters found by the recursive clustering algorithm.



(c) Heatmap of Set-cover clustering mapped to figure 4.1d, correlation is set to be 0.9 which is 0.05 in distance. The figure illustrates the top 20 largest clusters for set-cover clustering.



(d) PCA dimentionality reduction of 10xGenomics PBMC cells aforementioned in section 3.1 and clustered using leiden clustering method.

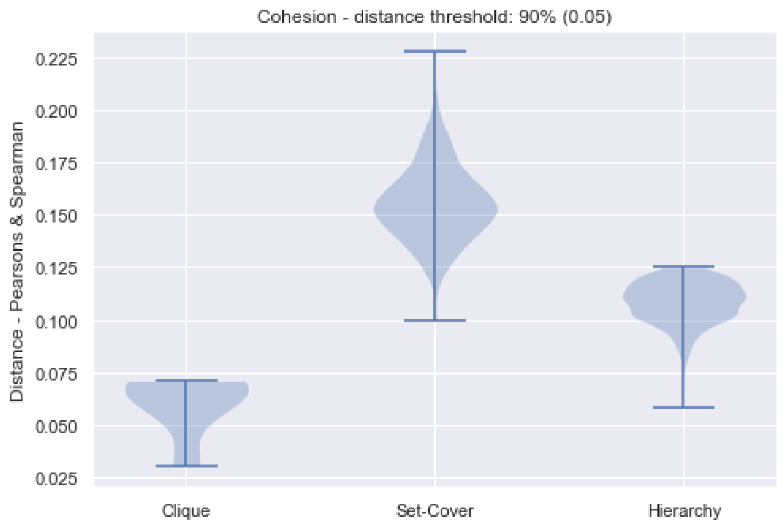
Figure 4.1: Illustration of clustering methods grouped by cliques visualized in Figure 4.1d.

4.1.2 Cohesion, Separation, and Coverage

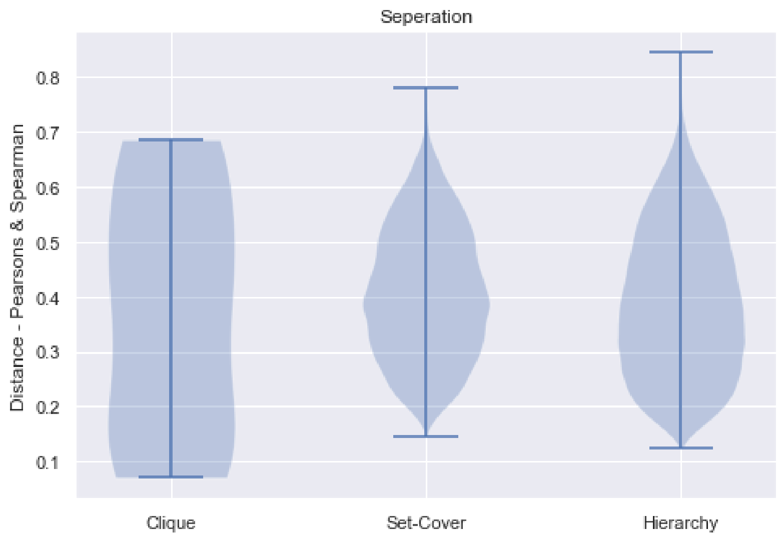
This result visualizes the robustness and separation of each clustering method. For example, Clique and Hierarchy's cohesion suddenly stops at a distance, with Clique on 0.075 and Hierarchy on 0.125. On the other hand, Set-Cover has a mean of around 0.150 and has no sudden stop like the two counterparts, as illustrated in Figure 4.2a.

The separation between the three methods is somewhat similar, with all methods having a lower distance of 0.1 and an upper of 0.7-0.8. However, both Set-Cover and Hierarchy have almost the same distribution, but Hierarchy has a higher span from 0.1 - 0.82 meanwhile Set-Cover has a span of 0.15 - 0.79. The distance between each cluster in Clique-based is nearly equally distributed from 0.05 to 0.7.

As illustrated in Figure 4.3, the distance for the next clusters is quite uniform in Hierarchy for all the illustrations. However, with Set-Cover, the distance seems to increase the more clusters are being appended.



(a) Cohesion on gene clustering methods using a distance threshold of 0.05 (90% correlation).



(b) Illustration of the separation from each cluster and the distance between them. The higher the value, the more separation is the average distance between each cluster.

Figure 4.2: This figure illustrates both the cohesion and separation of genes using the different clustering methods as explained in Section 3.3.4.

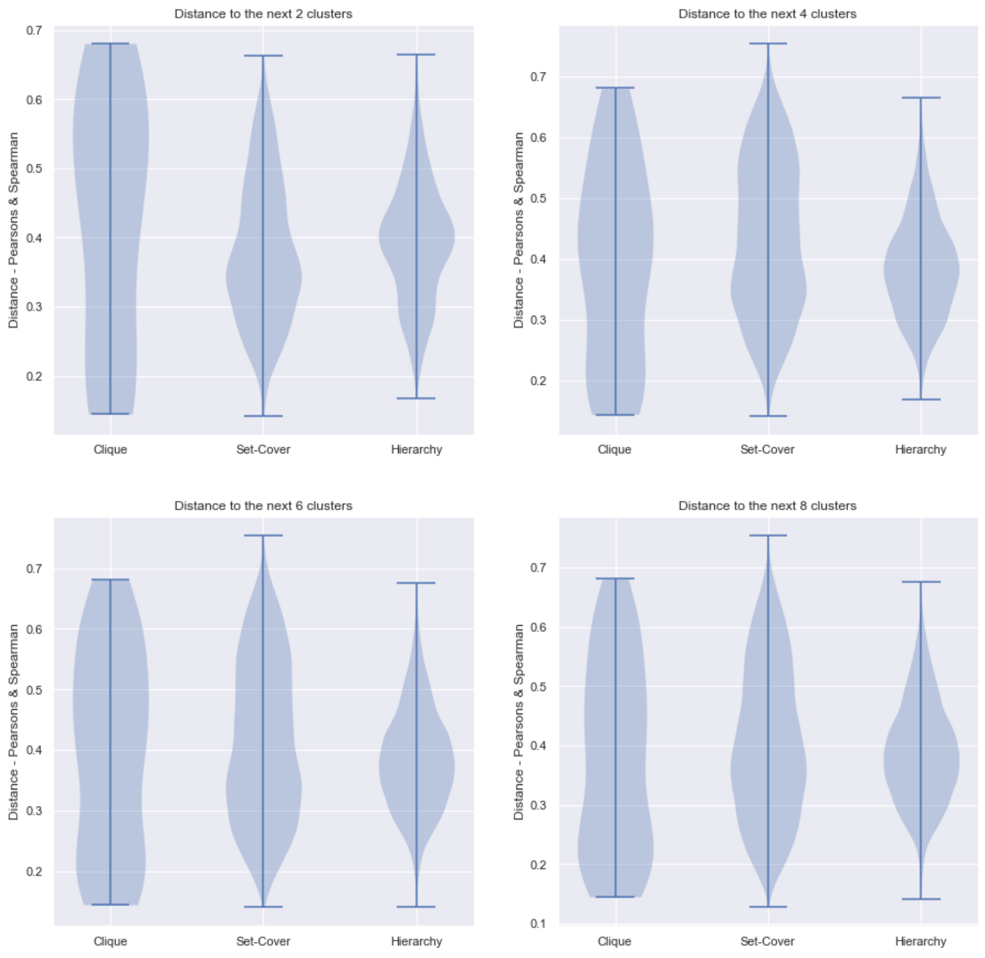


Figure 4.3: Graph coverage To illustrate how much of the graph is covered by the largest cluster in each clustering method.

4.2 Deep learning

4.2.1 Clustering comparisons

A simple comparison has been made to illustrate the difference between Set Cover and Hierarchy. Set Cover received a mean AUC score below 0.5, while Hierarchy scored above 0.7.

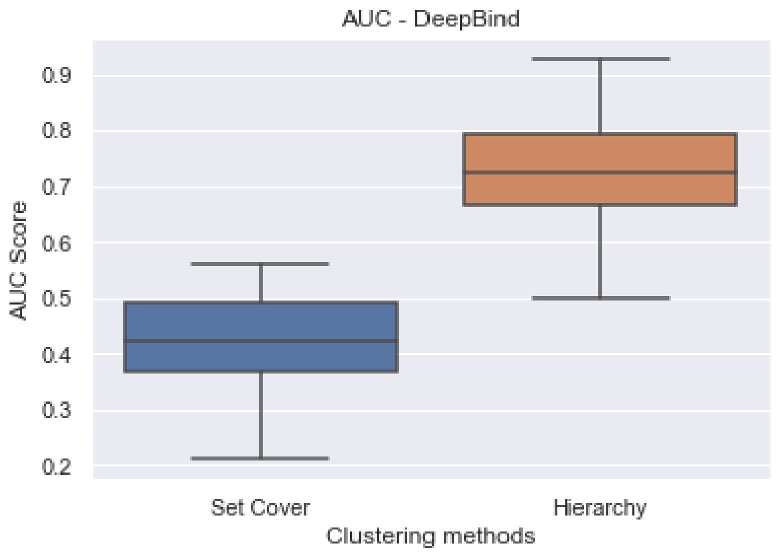


Figure 4.4: Comparisons of Hierarchy and Set-cover. Cohesion: 0.05, Sequence length: 2000.

4.2.2 AUC - Different lower-threshold comparisons

In this Section, cohesion is discussed in order to answer **RQ2**. For the result to be trustworthy, it needs to be reproducible. Therefore, as shown in Figure 4.5, many different cohesions are tested on the dataset using a generator described in the previous chapter. As a result, each graph shows four iterations of identical runs. The more deviation, the less statistical reliability is the cohesion's final result. Furthermore, the power analysis has been taken between each run to support them further.

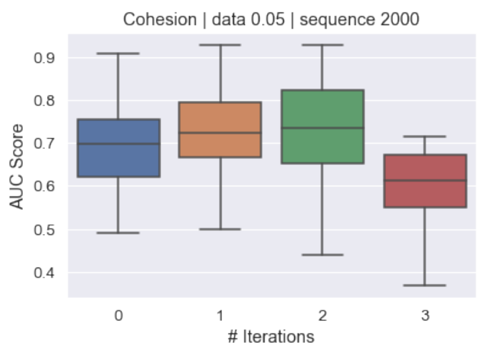
Firstly, a distance of 0.05 illustrated in Figure 4.5a shows a somewhat consistency in each individual run. 0.05 has the AUC mean around 0.7 in the first run, while in the last run, it dips to 0.6.

Distance 0.075 has less consistency between the iterations, where the AUC mean is from underneath 0.5 and up to 0.6. The values under 0.5 are as good as random guesses.

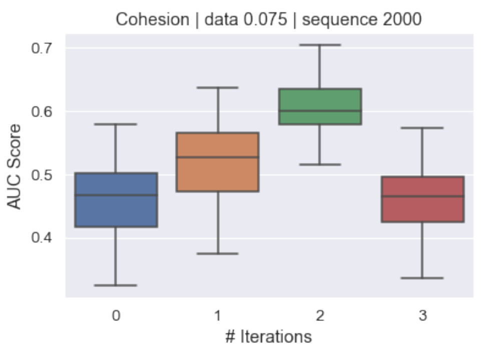
Distance 0.1's AUC score hovers around 0.5, where the first run, 0, has a mean AUC of ≈ 0.45, which shows a worse predictor than a random guesser. On the other hand, run 1 has a mean AUC score over 0.5.

Lastly, distance 0.125 shows the same behavior as 0.75 and 0.1, where the mean AUC score hovers around 0.5.

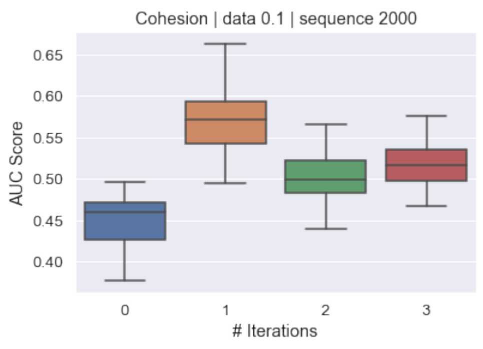
Additionally, to test the statistical significance of each of the runs. These values are illustrated in Table . For example, as visualized, the cohesion of 0.05 in the first row of Table needs to iteraterate five times for each of the other cohesions. However, there is much more uncertainty between 0.075 and 0.1. Therefore, where to say with certainty if there is a statistical significance, there have to be additional 1746 iterations for 0.075 - 0.1 and 1351 runs for 0.075 - 0.125. Furthermore, it takes 166 iterations for group 0.1 - 0.125.



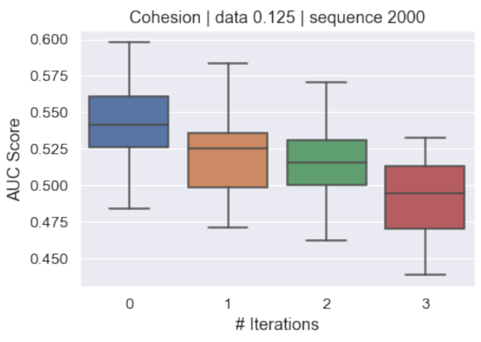
(a) Iteration test of cohesion of 0.05



(b) Iteration test of cohesion of 0.75.



(c) Iteration test of cohesion of 0.1.



(d) Iteration test of cohesion of 0.125.

Figure 4.5: Iteration test to show the reproducability of the result based on four different distances in the lower threshold, 0.05 4.5a, 0.075 4.5b, 0.1 4.5c, and 0.125 4.5d.

Table 4.2: The Power Analysis is for two-group independent t-test samples from the cohesion testing done in figure 4.5. The table illustrates the iterations it takes to confirm the statistical differences. Furthermore, the table illustrates the P-value for the cohesion and effect sizes. The cohesions tested are: 0.05, 0.075, 0.1, and 0.125.

P-value w/ Bonferroni correction	0.05	0.075	0.1	0.125
0.05	-	0.076	0.038	0.023
0.075	-	-	1.000	1.000
0.1	-	-	-	1.00
0.125	-	-	-	-
Effect size	0.05	0.075	0.1	0.125
0.05	-	0.170	0.175	0.165
0.075	-	-	0.004	-0.005
0.1	-	-	-	-0.009
0.125	-	-	-	-
Iterations	0.05	0.075	0.1	0.125
0.05	-	6	5	5
0.075	-	-	1746	1351
0.1	-	-	-	166
0.125	-	-	-	-

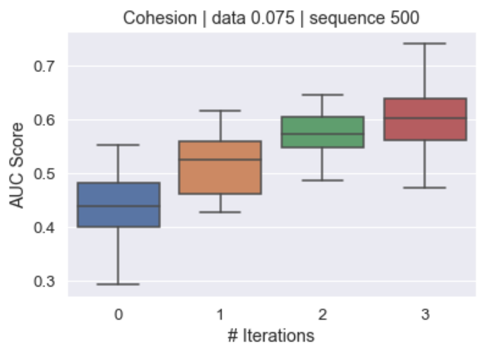
Table 4.3: Table represents the cohesions 0.05, 0.075, 0.1, and 0.125 and their average, standard deviation, and variance.

Cohesion	0.05	0.075	0.1	0.125
Mean	0.683	0.513	0.508	0.518
Standard deviation (σ)	0.108	0.056	0.032	0.027
Variance (σ^2)	0.011	0.003	0.001	0.001

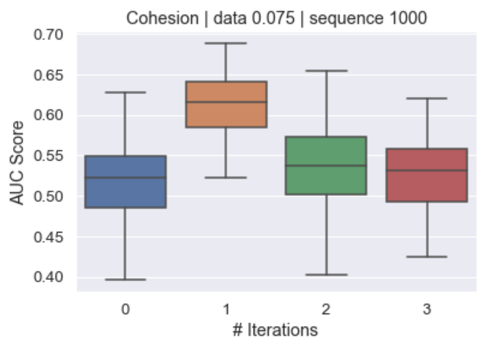
4.2.3 AUC - Different sequence lengths

As in cohesion, the sequence length needs to be tested. In this section, a discussion over how to answer **RQ3** is done. The sequence lengths tested were 500, 1000, 1500, and 2000. This sequence was split 20% upstream and 80% downstream from the gene start. Each of these runs utilized 0.075 as cohesion distance. The sequence length of 500 shows an average AUC median of around 0.5, with the lowest median score of 0.43 to 0.6. Furthermore, the sequence length of 1000 has all the median scores above random choice, 0.5 AUC. However, the inner quartile touches 0.5 in three of the four iterations. The same can be said with a sequence length of 1500. On the other hand, run one shown in Figure C.1e has a lower mean AUC with 0.1 in relation to run one illustrated in Figure 4.6b. Lastly, sequence 2000 has a mean hovering around 0.57 except for run one below 0.5.

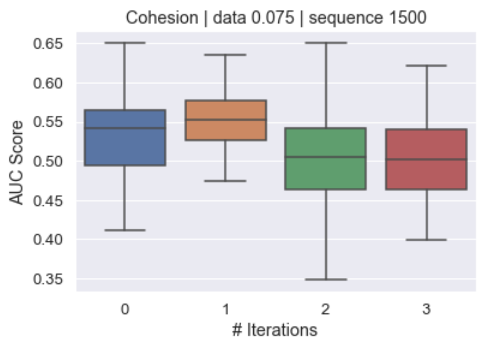
The power analysis illustrated in Table 4.4 shows that to conclude the t-test, all the different runs need to be run at least 41 times and a max of 710 times.



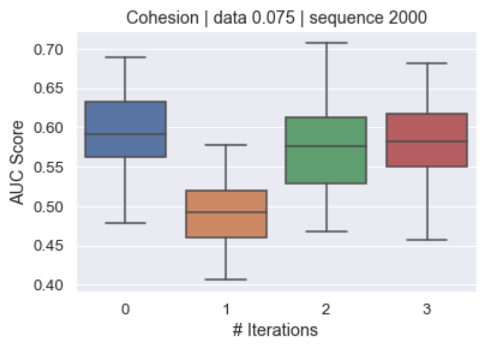
(a) Iteration test for sequence length 500



(b) Iteration test for sequence length 1000



(c) Iteration test for sequence length 1500



(d) Iteration test for sequence length 2000

Figure 4.6: The result shown in this figure is to test for sequence lengths based on nucleotides in the promoter region. The test is to verify the reproducibility of the result.

Table 4.4: The Power Analysis is for two-group independent t-test samples from the sequence length testing done in figure 4.6. The table illustrates the iterations it takes to confirm the statistical differences. Furthermore, the table illustrates the P-value for the cohesion and effect sizes. The sequence length tested are: 500, 1000, 1500, and 2000.

P-value w/ Bonferroni correction	500	1000	1500	2000
500	-	1.000	1.000	1.000
1000	-	-	1.000	1.000
1500	-	-	-	1.000
2000	-	-	-	-
Effect size	500	1000	1500	2000
500	-	-0.017	0.008	-0.026
1000	-	-	0.025	-0.009
1500	-	-	-	-0.034
2000	-	-	-	-
Iterations	500	1000	1500	2000
500	-	166	710	73
1000	-	-	71	456
1500	-	-	-	41
2000	-	-	-	-

Table 4.5: Table represents the sequence lengths 500, 1000, 1500, and 2000 and their average, standard deviation, and variance.

Sequence length	500	1000	1500	2000
Mean	0.531224	0.547730	0.522726	0.557160
Standard deviation (σ)	0.057722	0.048708	0.056486	0.052686
Variance (σ^2)	0.003	0.002	0.003	0.003

4.2.4 Biological validation - Motif detection

As seen in Table 4.6 is the motifs found by the deep learning model from a cohesion of 0.05 and sequence length of 2000. Each motif has a score similar to the JASPER DB motif. Furthermore, is the gene name the motif belongs to, and every gene was found in either the positive or negative dataset.

Table 4.6: Motifs found from deep learning model of sequence 2000 and cohesion 0.05. The score and ID are from the JASPAR DB[50]. The score represents how close the displayed motif is to the original motif. As well as the data set to the gene belongs. If the dataset column has *N/A*, then the gene does not exist in any data sets.

index	motif	score	id	Gene name	Data set
1	TTTTGCCGTTCTTTCTCACCCGGT	29.4789	MA1929.1	CTCF	N/A
2	GGCTACTCATCGCGACTGTCTAGC	33.3121	MA1930.1	CTCF	N/A
3	ATAAACTTGGGGTCTAATTGGTCT	26.9971	MA1714.1	ZNF675	N/A
4	TGAATTGACCGAAGTCTTGTGATC	X	X	X	N/A
5	CAGAGTCCCTGATTCGGACAGACG	33.5324	MA1929.1	CTCF	N/A
6	ACCTTCATTCGACGCTCATGCGGA	34.4783	MA1930.1	CTCF	N/A
7	AGAATCTAAGGAATTAAAGACGTC	31.0683	MA1930.1	CTCF	N/A
8	AGCCAGGACGATGGGCCCAACCCA	32.667	MA1930.1	CTCF	N/A
9	GCAATCACCATACTTTATGCCACT	28.5168	MA1930.1	CTCF	N/A
10	CTCAGACGGCAAGTTACAGAAACG	34.155	MA1930.1	CTCF	N/A
11	TCGATTGATTGATCAAACATGTGT	26.1302	MA1978.1	ZNF354A	Negative
12	CCCCTCGGGTCGGGCCAGATAAAG	35.148	MA1929.1	CTCF	N/A
13	GGCATGCTCGATGCCCCTATCCCC	34.3255	MA1976.1	ZNF354A	Negative
14	AGATCTAGGATCACCGGTGATGAT	28.7608	MA1594.1	ZNF382	N/A
15	CGTACGAAGACTGTAGCGGTTCAG	33.3708	MA1929.1	CTCF	N/A
16	TGAAATTTTCAATGACAATATTGG	32.2484	MA1929.1	CTCF	N/A

Chapter 5

Discussion

5.1 Clustering

The clustering method for the correlation matrix generated by coregulated gene expression is crucial for the deep learning algorithm. Moreover, this method is essential for a great data foundation for the deep learning model to train and evaluate. Therefore, three viable methods were experimented with founded on cohesion, separation, and graph coverage. The visualization of the clustering methods illustrated differences in the illustrative aspect of each method according to gene expression in the PCA cell clustering displayed in Figure 4.1d. However, the heatmap also yielded similarities between each cluster for each method used. That is why using cohesion and separation testing for validation.

Clique-based clustering, as illustrated in Figure 4.2 generated robust clusters and was the most promising to do so with an internal distance of 0.075. However, a drawback of this method is the small number of clusters yielded. For example, this method produced ten cliques, where three of them were only one gene. The same phenomena happened with the paper 'Gene regulatory network inference in single-cell biology' by Akers and Murali presented in the Related Works. Another disadvantage of clique-based clustering was removing all other genes, leaving only the genes inside those clusters, 3466 genes.

Set-Cover exhibited a more favorable outcome as it contained all genes that Clique-based removed and generated clusters for all, although some clusters contained a singular gene. Although these advantages, Set-Cover was lacking in the cohesion validity test. Since this method uses hubs, and all genes with a distance of 0.05 get added by this greedy algorithm does not mean the newly added genes correlate to 0.05 in the distance. The newly added genes can have higher distances between each other. Hence, it indicates Set-Cover unsatisfactory results in the cohesion test displayed in Figure 4.2a.

Agglomerative Hierarchy clustering is the last method assessed. As Set-Cover, Hierarchy retains all genes and positions them in clusters. Furthermore, Hierarchy utilizes the dendrogram to be able to make a hard cut based on the given distance, causing a low cohesion. However, the robustness in the average cluster is inferior to that of Clique-based. The hierarchy method is favored because of its advantages and does not contain the drawbacks of Clique and Set-Cover. Hierarchy and Set-Cover are applied to the deep learning model as seen in Figure 4.4 to further analyze the clustering method's capabilities. The comparison tells us that Hierarchy might have a bigger advantage. Further iterations of both clusters must be done to draw a statistical conclusion. This method is used in the deep learning results.

5.2 Deep learning

5.2.1 Cohesion

To conclude internal cohesion, I have run four different iterative runs for four distances, 0.05, 0.075, 0.1, and 0.125. I have used Hierarchical Agglomerative clustering with an upper threshold, t_u of 0.43 to create two major clusters of $\approx$ 7000 genes each while containing as much separation as possible.

At first glance, a cohesion of 0.05 shows successful results as the mean AUC mean is 0.68, as illustrated in Table 4.3, which shows some discrimination between the positive and negative datasets. Three iterations have an average AUC score of $\approx$ 0.7, while only one with $\approx$ 0.6 as shown in Figure 4.5a. However, causing this might be the strong cohesion between the genes, making it easier for the deep learning model to acquire some prominent features in the positive set. On the other hand, the strong cohesion between the genes in the positive set generates only 77 out of $\approx$ 7000, while 69 are for training, and eight are for evaluation. Therefore, it can cause overfitting of the data.

0.075 illustrates a worse result than 0.05. Two of the AUC mean are beneath a score of 0.5, and two are above, giving a mean score of 0.513; however, iteration one touches the score of 0.5, which illustrates that cohesion of 0.075 does not distinguish positives from negatives. Meanwhile, the lower threshold of 0.075 increases the positive dataset to 279, contrary to the 77 that 0.05 had, giving 28 evaluation genes. This increase in genes might have removed the binary guessing, as seen in 0.05, but it could also have introduced biological noise to the positive dataset making it harder for the deep learning model to pick out features from the dataset.

The cohesion of 0.1 has the biggest span of mean AUC scores of all the tested cohesions. This span is from 0.45 to 0.575, seemingly giving the cohesion of 0.1 the most irregular result out of all the runs. Furthermore, three out of four iterations are either on or below an AUC score of 05, giving the mean AUC score of 0.501. The mean score of 0.501 illustrates that the model guesses randomly for the given data. However, as shown in Table 4.1, the increase in the positive dataset went from 279 to 686, which gave more training data, which in turn introduced more biological noise to the dataset. The introduction noise could justify the unsatisfactory results.

Lastly is 0.125, Which has three out of four iterations above 0.5 AUC contrary to 0.1, giving a mean AUC score of 0.518. This run also shows irregularities in the results; however, the irregularities are not as aggressive as 0.075 and 0.1. This span is only from 0.55 to 0.50, which is not an ideal region for a classification. This poor performance might result from too low robustness adding excessive noise into the positive dataset, making it harder for the model to interpret common features. However, increasing the cohesion gives the model more to train. The positive dataset increased to 974, making it the largest positive dataset.

P-value using Bonferroni correction was applied to each pair of runs. Using $\alpha = 0.05$, we can see that we can only reject two H_0, which are the runs between 0.05 and 0.01, and 0.05 and 0.125. The rejection tells us there is a significance between these runs and no significance between the rest. The Effect size in table 4.2 shows that the 0.05 positively affects all of the compared values. This effect size shows that the cohesion of 0.05 is superior to the other tested cohesions. However, by taking the power analysis of the pairwise runs, we calculated the number of iterations that needed to be done in order to confirm the statistical differences. Here we can see that between 0.075 and 0.1, and 0.075 and 0.125 tells us that there are needed 1746 iterations. Hence, a statistical conclusion between these is not as viable as the difference of 0.05. Further iterations for 0.075 needs to be done in order to conclude. The same can be said between 0.1 and 0.125 since the power analysis gave the number of iterations 166. On the other hand, the large number of iterations tells us that there are no practical differences between these runs.

5.2.2 Sequence length

As for testing cohesion, the same procedure was completed with the sequence length. The sequence length might be too short to contain motifs in the promotor that are identifiable for the classifier or too long that it might contain too much biological noise. All tests were run on 0.075 cohesion but with iterative runs for four sequence lengths, 500, 1000, 1500, and 2000.

The sequence 500, where 400 are upstream, and 100 are downstream, seems to be split, where two iterations are under and on 0.5 AUC score, and two are above. Thus, giving a mean AUC score of 0.531. The sequence in question indicates that the deep learning model does not find apparent features. As discussed in the gene theory, the promotor can contain up to 1000 nucleotides, and taking 400 bp upstream from transcription start will not include the entire promoter region. Hence, causation of inconsistent results; AUC score spans from ≈ 0.4 to ≈ 0.6.

1000 Sequence length shows more favorable results than 500, with all mean AUC scores above 0.5, providing a total mean of 0.547 shown in table 4.5. That indicates the model has found some features. Nonetheless, three out of four inner quartiles in Figure 4.6b touch the 0.5 AUC. A further indication that the model does not fetch crucial features. On the other hand, the three runs have similar results, hinting that the result is not as inconsistent as the 500. It implies that there are features in the 1000 sequence that does not exist in the 500.

The Sequence length of 1500 has a balanced run for the evaluation where all iterations are in the 0.50 to 0.55 domain. This result is not viable for binary classification as it guesses most of the time. These results indicate that the model has not learned features unique to the dataset. As well as, it might introduce biological and technical noise by using 1500 base pairs.

Lastly, the sequence length of 2000, seemingly from the results in Figure C.1d might be better out of the four sequence lengths. Three out of four iterations give a mean AUC score of ≈ 0.6, implying that a length of 2000 is more reliable and gives decent scoring. However, one of the iterations has an AUC score lower than 0.5 of≈ 0.48, making it not as reliable at first glance. Thus, giving a total mean AUC score of 0.557. That might be the same as for 1500, which introduces biological noise. In addition, the large sequence of 2000 bp can cause a loss of accuracy because the model is created for smaller sequences; DeepBind used the model for a bp of 101, and adding 2000 bp might cause it to be not as accurate as the result in paper 'Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning.'

As mentioned in the previous section 5.2.1, the results alone are not enough to compare the runs. Furthermore, a P-value is needed to accept or reject the H_0. As seen in Table 4.4. None of the P-values are above the $\alpha = 0.05$, which tells us that there is no significance between the different runs, and H_0 is accepted. However, by inspecting the effect size, we can see that it is quite small, which, in turn, causes the number of iterations needed to confirm the differences to increase. So the comparisons of the sequence iterations need to be taken with a grain of salt.

5.2.3 Biological validation

To verify that the deep learning model managed to find motifs in the dataset, given the weights in the convolutional layer were converted to motif sequences as shown in Table 4.6. Out of the 16 filters in the model, only six were unique, with MA1929.1, MA1930.1, MA1714.1, MA1978.1, MA1976.1, and MA1594.1. These motifs belong to four genes, CTCF, ZNF675, ZNF354A, and ZNF382. However, only one gene was found in the dataset through motifs. The found gene was ZNF354A and was in the negative dataset. By the nature of the method used for training, an assumption was that the model would learn features from the positive dataset. However, the only found gene was in the negative dataset. Therefore, I speculate that this find was by chance. This result does not support the model finding biological patterns upstream from the transcriptional start site.

5.2.4 Strengths and Weaknesses with methodology

Furthermore, I have used the biggest cluster when comparing sequence length and cohesion. This cluster was chosen to get as many genes for the positive dataset, so the deep learning model had enough training data. However, a weakness with this is that each cluster might have its biological noise profile. Another weakness is the utilization of only the PBCM dataset for the genes. This can cause the model not to be as generic as one hoped for. Hence, the model and results in this thesis can be used for the PBMC dataset and the biggest cluster.

Chapter 6

Conclusion

An analysis of clustering methods, cohesion, and sequence length shows an ensemble of results. In order to answer **RQ1**, Hierarchical clustering is arguably a better clustering method. There are two major reasons for this. Firstly, the Hierarchical method provides all genes and does not discard any genes in its clustering. On the other hand, the clique method discards genes, not assigned clusters. Therefore, it makes Hierarchical to be ideal for the Clique. Secondly, Hierarchical is superior to Set-cover by utilizing the cohesion of the clusters as a validator of the clustering technique. Utilizing a Hierarchical clustering strategy in the deep learning model will cause it to be able to produce cohesive clusters that contain all inserted gene expressions.

Two features from the dataset were tested for the deep learning method: cohesion and sequence length. By investigating the cohesion results, this thesis concluded that cohesion of 0.05 on the dataset provided is superior to the others, which answers **RQ2**. For the results, only using the largest cluster was used for generating the results. In this conclusion, An assumption is that the chosen cluster does not matter. Additionally, the effect size from 0.05 to the others was positive, illustrating that the strength of 0.05 and a few iterations were given from the power analysis. Further, given $\alpha = 0.05$ illustrates a significant difference between 0.05 and 0.1, 0.125. Illustrating the possibility of difference between these runs not based on chance. Furthermore, by analyzing Table 4.2, the number of iterations by the power analysis reaches up to 1746, which illustrates the small differences between 0.075 with 0.1 and 0.125 that it has no practical differences between these runs.

The experiment on sequence length revealed a less favorable outcome. All runs had visibly different results for each iteration in the run. Furthermore, no significance between these runs by examining the P-value was identified. The mean for each run is similar, at around ≈ 0.53, characterizing the model's random nature. This further implies that the sequence length does not affect the result. Since the iterations given by the power analysis are large also demonstrates that there is no functional difference between the various lengths. Therefore, the conclusion for

RQ3 is that the sequence length does not affect the results for the deep learning model. This experiment was based on the assumption that the selected dataset, cohesion, and cluster do not affect the result.

Lastly, extracting motifs from the deep learning model illustrates the practical capability of the deep learning model. However, as displayed in Table 4.6 only one of the genes was found. Furthermore, the gene found was in the negative set. This result weakens the practical application of this methodology for finding motifs upstream in the promoter.

In this experiment, It was derived that Hierarchical clustering is the superior clustering technique for **RQ1**. This superiority is due to its ability to conserve all genes and its high cohesion. For **RQ2**, a cohesion of 0.05 is currently the best cohesion value. This conclusion is based on the higher mean value, P-value with Bonferroni correction, and significance between the other runs. The conclusion of cohesion assumes that the sequence length does not affect the result. However, when utilizing the biological validation, The model did not extract features dominant in the positive dataset. This extraction weakens the practical validity of the model.

To answer the **RQ3**, There was no significance between the runs of 500, 1000, 1500, and 2000. The iterations given by the Power analysis in table 4.4 reveal no practical difference between these runs. Thus, the conclusion for **RQ3** is that the sequence length has no practical effect on the deep learning method. Nevertheless, this conclusion assumes that cohesion does not affect the result. The conclusion for **RQ2** and **RQ2** are under the assumption that there is no difference when testing other datasets and clusters other than the largest one.

6.1 Further Work

6.1.1 T-test - Independent data

The t-test has assumptions about the tested data, one of which is the **assumption of independence**. When evaluating the data, the same dataset was used in all runs causing this assumption to be broken. For further work, ten-fold cross-validation could be used to calculate a total error rate and fortify the methodology and results.

6.1.2 Testing other clusters - Noise profiles

This thesis scoped large clusters to generate as much positive data for the dataset as possible, which caused a methodology weakness. As illustrated in the Heatmap Figure 4.1 the clusters have different signatures, but some are closely related to others. For example, h_0 and h_1 in Figure 4.1b are closely related, but h_5 are visibly distinct. The signatures can cause a distinct noise ratio in the different clusters, affecting the results. A distinct ensemble of runs needs to be concluded to test a different cluster signature to validate the model's capability of picking up other noise profiles.

6.1.3 More evaluation datasets - Generalize model

The deep learning model should be evaluated on other datasets in further work. The evaluation is to validate the model's general versatility. Hence, training and evaluating other scRNA-req datasets such as Mouse PBMC[1]. If the model presents equivalent results on other datasets, it will further establish the model's adaptability to a generic mRNA dataset.

6.1.4 DeeperBind - Improve the model's capabilities

The model used in this thesis is from the paper 'Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning' by Alipanahi *et al.* and According to Alipanahi *et al.* shows great promises in using deep learning in classifying genes based on gene expression. However, the model's design cannot capture positional clues in the classification. This is why there has been a development in the DeepBind called the DeeperBind [51]. The paper 'DeeperBind: Enhancing Prediction of Sequence Specificities of DNA Binding Proteins' by Hassanzadeh and Wang adds Recurrent Neural Network to the DeepBind model, which is to capture the positional dynamics of the sequence, which according to Hassanzadeh and Wang can handle dynamic length motifs rather than setting a static motif length as in DeepBind. Therefore, further work should test whether DeeperBind can achieve higher classification scores than the DeepBind used in this thesis.

[1]https://www.10xgenomics.com/resources/datasets/integrated-gex-and-vdj-analysis-of-connect-generated-library-from-mouse-pbm-cs-2-standard-6-0-1

Bibliography

[1] B. Alipanahi, A. Delong, M. T. Weirauch and B. J. Frey, ‘Predicting the sequence specificities of dna- and rna-binding proteins by deep learning,’ *Nature Biotechnology*, vol. 33, no. 8, pp. 831–838, Aug. 2015, ISSN: 1546-1696. DOI: 10.1038/nbt.3300. [Online]. Available: https://doi.org/10.1038/nbt.3300.

[2] K. A. H. Sherry Liang Khalid Alanazi. ‘Set covering problem.’ (), [Online]. Available: https://optimization.cbe.cornell.edu/index.php?title=Set_covering_problem. (accessed: 23.04.2022).

[3] N. C. Institute. ‘Cell function.’ (), [Online]. Available: https://training.seer.cancer.gov/anatomy/cells_tissues_membranes/cells/function.html. (accessed: 11.04.2022).

[4] MedlinePlus. ‘Cell structure.’ (), [Online]. Available: https://medlineplus.gov/genetics/understanding/basics/cell/. (accessed: 11.04.2022).

[5] medlineplus. ‘What is dna?’ (), [Online]. Available: https://medlineplus.gov/genetics/understanding/basics/dna/. (accessed: 06.06.2022).

[6] N. H. G. R. Institute. ‘Non-coding dna.’ (), [Online]. Available: https://www.genome.gov/genetics-glossary/Non-Coding-DNA#:~:text=Non%2Dcoding%20DNA%20corresponds%20to,DNA%20have%20no%20known%20function.. (accessed: 06.06.2022).

[7] P. Theresa Phillips. ‘Regulation of transcription and gene expression in eukaryotes.’ (), [Online]. Available: https://www.nature.com/scitable/topicpage/regulation-of-transcription-and-gene-expression-in-1086/. (accessed: 12.04.2022).

[8] G. M. Cooper, *The Cell: A Molecular Approach. 2nd edition.* Sinauer Associates, 2000.

[9] K. Zhao YB., ‘Mrna translation and protein synthesis: An analysis of different modelling methodologies and a new pbn based approach.,’ *BMC Systems Biology*, 2014.

[10] H. A. E. J. T. S. et al., ‘A practical guide to single-cell rna-sequencing for biomedical research and clinical applications,’ *Genome Med*, 2017.

[11] . A. R. C. F. Sanger S. Nicklen, ‘Dna sequencing with chain-terminating inhibitors,’ *Proc Natl Acad Sci U S A*, 1977.

[12] E. L. van Dijk, H. Auger, Y. Jaszczyszyn and C. Thermes, 'Ten years of next-generation sequencing technology,' *Trends in Genetics*, vol. 30, no. 9, pp. 418–426, 2014, ISSN: 0168-9525. DOI: https://doi.org/10.1016/j.tig.2014.07.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168952514001127.

[13] Illumina. 'Next-generation sequencing.' (), [Online]. Available: https://www.illumina.com/science/technology/next-generation-sequencing.html. (accessed: 21.04.2022).

[14] A. Haque, J. Engel, S. A. Teichmann and T. Lönnberg, 'A practical guide to single-cell rna-sequencing for biomedical research and clinical applications,' *Genome Medicine*, vol. 9, no. 1, p. 75, Aug. 2017, ISSN: 1756-994X. DOI: 10.1186/s13073-017-0467-4. [Online]. Available: https://doi.org/10.1186/s13073-017-0467-4.

[15] E. Z. Macosko, A. Basu, R. Satija, J. Nemesh, K. Shekhar, M. Goldman, I. Tirosh, A. R. Bialas, N. Kamitaki, E. M. Martersteck, J. J. Trombetta, D. A. Weitz, J. R. Sanes, A. K. Shalek, A. Regev and S. A. McCarroll, 'Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets,' *Cell*, vol. 161, no. 5, pp. 1202–1214, 2015, ISSN: 0092-8674. DOI: https://doi.org/10.1016/j.cell.2015.05.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0092867415005498.

[16] K. Akers and T. Murali, 'Gene regulatory network inference in single-cell biology,' *Current Opinion in Systems Biology*, vol. 26, pp. 87–97, 2021, ISSN: 2452-3100. DOI: https://doi.org/10.1016/j.coisb.2021.04.007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2452310021000184.

[17] G. Iacono, R. Massoni-Badosa and H. Heyn, 'Single-cell transcriptomics unveils gene regulatory network plasticity,' *Genome Biology*, vol. 20, no. 1, p. 110, Jun. 2019, ISSN: 1474-760X. DOI: 10.1186/s13059-019-1713-4. [Online]. Available: https://doi.org/10.1186/s13059-019-1713-4.

[18] R. Swofford. 'How we get information from your dogs dna.' (), [Online]. Available: https://darwinsark.org/information-from-dna/. (accessed: 11.04.2022).

[19] addgene. 'Promoters.' (), [Online]. Available: https://www.addgene.org/mol-bio-reference/promoters/. (accessed: 07.06.2022).

[20] H. B. L. J. B. D., 'Single-cell rna sequencing technologies and bioinformatics pipelines,' *Exp Mol Med*, 2018.

[21] S. Aryal. 'Dna sequencing- maxam–gilbert and sanger dideoxy method.' (), [Online]. Available: https://microbenotes.com/dna-sequencing-maxam-gilbert-and-sanger-dideoxy-method/. (accessed: 07.06.2022).

[22] Y. Lu, Y. Shen, W. Warren and R. Walter, 'Next generation sequencing in aquatic models,' in *Next Generation Sequencing*, J. K. Kulski, Ed., Rijeka: IntechOpen, 2016, ch. 2. DOI: 10.5772/61657. [Online]. Available: https://doi.org/10.5772/61657.

[23] L. Sciences. '10x genomics single cell rna sequencing report.' (), [Online]. Available: https://www.lcsciences.com/documents/sample_data/single_cell_rna_seq/scRNA_seq_html_report_DEMO.html. (accessed: 20.04.2022).

[24] M. Lindner, I. Verhagen, H. M. Viitaniemi, V. N. Laine, M. E. Visser, A. Husby and K. van Oers, 'Temporal changes in dna methylation and rna expression in a small song bird: Within- and between-tissue comparisons,' *BMC Genomics*, vol. 22, no. 1, p. 36, Jan. 2021, ISSN: 1471-2164. DOI: 10.1186/s12864-020-07329-9. [Online]. Available: https://doi.org/10.1186/s12864-020-07329-9.

[25] R. Qi, A. Ma, Q. Ma and Q. Zou, 'Clustering and classification methods for single-cell rna-sequencing data,' eng, *Briefings in bioinformatics*, vol. 21, no. 4, pp. 1196–1208, Jul. 2020, 5528236[PII], ISSN: 1477-4054. DOI: 10.1093/bib/bbz062. [Online]. Available: https://doi.org/10.1093/bib/bbz062.

[26] Y.-L. Bai, M. Baddoo, E. K. Flemington, H. N. Nakhoul and Y.-Z. Liu, 'Screen technical noise in single cell rna sequencing data,' *Genomics*, vol. 112, no. 1, pp. 346–355, 2020, ISSN: 0888-7543. DOI: https://doi.org/10.1016/j.ygeno.2019.02.014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0888754318305809.

[27] D. Blog. 'What is hierarchical clustering?' (), [Online]. Available: https://www.displayr.com/what-is-hierarchical-clustering/#:~:text=Hierarchical%20clustering%2C%20also%20known%20as,broadly%20similar%20to%20each%20other.. (accessed: 06.06.2022).

[28] T.-D. Nguyen, B. Schmidt and C.-K. Kwoh, 'Sparsehc: A memory-efficient online hierarchical clustering algorithm,' *Procedia Computer Science*, vol. 29, pp. 8–19, 2014, 2014 International Conference on Computational Science, ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2014.05.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050914001781.

[29] P. Pai. 'Hierarchical clustering explained.' (), [Online]. Available: https://towardsdatascience.com/hierarchical-clustering-explained-e59b13846da8. (accessed: 23.04.2022).

[30] M. Chrobak, C. Dürr, A. Fabijan and B. J. Nilsson, 'Online clique clustering,' *Algorithmica*, vol. 82, no. 4, pp. 938–965, Apr. 2020, ISSN: 1432-0541. DOI: 10.1007/s00453-019-00625-1. [Online]. Available: https://doi.org/10.1007/s00453-019-00625-1.

[31] P. Slavık, 'A tight analysis of the greedy algorithm for set cover,' *Journal of Algorithms*, vol. 25, no. 2, pp. 237–254, 1997, ISSN: 0196-6774. DOI: https://doi.org/10.1006/jagm.1997.0887. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0196677497908877.

[32] Wikipedia. 'Correlation.' (), [Online]. Available: https://en.wikipedia.org/wiki/Correlation. (accessed: 26.04.2022).

[33] L. Statistics. 'Pearson product-moment correlation.' (), [Online]. Available: https://statistics.laerd.com/statistical-guides/pearson-correlation-coefficient-statistical-guide.php. (accessed: 26.04.2022).

[34] Wikipedia. 'Pearson correlation coefficient.' (), [Online]. Available: https://en.wikipedia.org/wiki/Pearson_correlation_coefficient. (accessed: 26.04.2022).

[35] Wikipedia. 'Spearman's rank correlation coefficient.' (), [Online]. Available: https://en.wikipedia.org/wiki/Spearman%27s_rank_correlation_coefficient. (accessed: 26.04.2022).

[36] statisticshowto. 'Statistics how to.' (), [Online]. Available: https://www.statisticshowto.com/probability-and-statistics/t-distribution/independent-samples-t-test/l. (accessed: 05.05.2022).

[37] statisticshowto. 'Statistics how to.' (), [Online]. Available: https://www.statisticshowto.com/probability-and-statistics/t-distribution/independent-samples-t-test/l. (accessed: 05.05.2022).

[38] M. Jafari and N. Ansari-Pour, 'Why, when and how to adjust your p values?' eng, *Cell journal*, vol. 20, no. 4, pp. 604–607, Jan. 2019, PMC6099145[pmcid], ISSN: 2228-5806. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/30124010.

[39] J. Brownlee. 'A gentle introduction to statistical power and power analysis in python.' (), [Online]. Available: https://machinelearningmastery.com/statistical-power-and-power-analysis-in-python/#:~:text=Statistical%20power%20is%20the%20probability,effect%20size%2C%20and%20statistical%20power.. (accessed: 05.05.2022).

[40] C. Cheng. 'Set covering problem.' (), [Online]. Available: https://towardsdatascience.com/principal-component-analysis-pca-explained-visually-with-zero-math-1cbf392b9e7d. (accessed: 07.06.2022).

[41] R. C. Fong, W. J. Scheirer and D. D. Cox, 'Using human brain activity to guide machine learning,' *Scientific Reports*, vol. 8, no. 1, p. 5397, Mar. 2018, ISSN: 2045-2322. DOI: 10.1038/s41598-018-23618-6. [Online]. Available: https://doi.org/10.1038/s41598-018-23618-6.

[42] T. f. E. Wikipedia. 'Artificial neuron.' (2020), [Online]. Available: https://en.wikipedia.org/wiki/Artificial_neuron.

[43] C. Nicholson. 'A beginner's guide to neural networks and deep learning.' (2019), [Online]. Available: https://pathmind.com/wiki/neural-network (visited on 06/06/2022).

[44] T. Isokawa, H. Nishimura and N. Matsui, 'Quaternionic multilayer perceptron with local analyticity,' *Information*, vol. 3, no. 4, pp. 756–770, 2012, ISSN: 2078-2489. DOI: 10.3390/info3040756. [Online]. Available: https://www.mdpi.com/2078-2489/3/4/756.

[45] W. Zhao and S. Du, 'Spectral–spatial feature extraction for hyperspectral image classification: A dimension reduction and deep learning approach,' *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, pp. 4544–4554, Apr. 2016. DOI: 10.1109/TGRS.2016.2543748.

[46] A. H. Reynolds. 'Convolutional neural networks (cnns).' (), [Online]. Available: https://anhreynolds.com/blogs/cnn.html. (accessed: 08.05.2022).

[47] A. SINGH. 'Demystifying the mathematics behind convolutional neural networks (cnns).' (2020), [Online]. Available: https://www.analyticsvidhya.com/blog/2020/02/mathematics-behind-convolutional-neural-network/ (visited on 20/04/2020).

[48] C. W. John McGonagle George Shaikouski. 'Backpropagation.' (2022), [Online]. Available: https://brilliant.org/wiki/backpropagation.

[49] G. Iacono, E. Mereu, A. Guillaumet-Adkins, R. Corominas, I. Cuscó, G. Rodríguez-Esteban, M. Gut, L. A. Pérez-Jurado, I. Gut and H. Heyn, 'Bigscale: An analytical framework for big-scale single-cell data,' eng, *Genome research*, vol. 28, no. 6, pp. 878–890, Jun. 2018, gr.230771.117[PII], ISSN: 1549-5469. DOI: 10.1101/gr.230771.117. [Online]. Available: https://doi.org/10.1101/gr.230771.117.

[50] JASPAR. 'Jaspar 2022.' (2022), [Online]. Available: https://jaspar.genereg.net/ (visited on 06/06/2022).

[51] H. R. Hassanzadeh and M. Wang, 'Deeperbind: Enhancing prediction of sequence specificities of dna binding proteins,' vol. 2016, Dec. 2016, pp. 178–183. DOI: 10.1109/BIBM.2016.7822515.

Appendix A

10xGenomics - Filtered Data

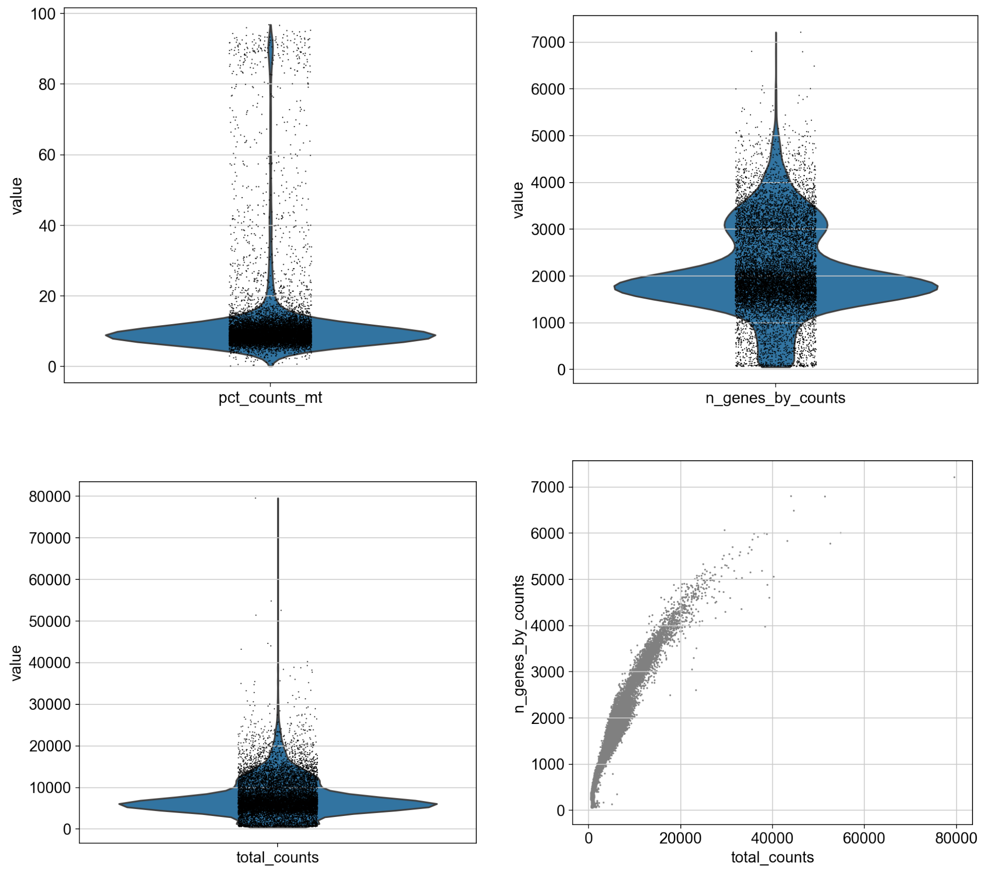


Figure A.1: Detailed information about 10xGenomics data.

Appendix B

Weights by deep learning model

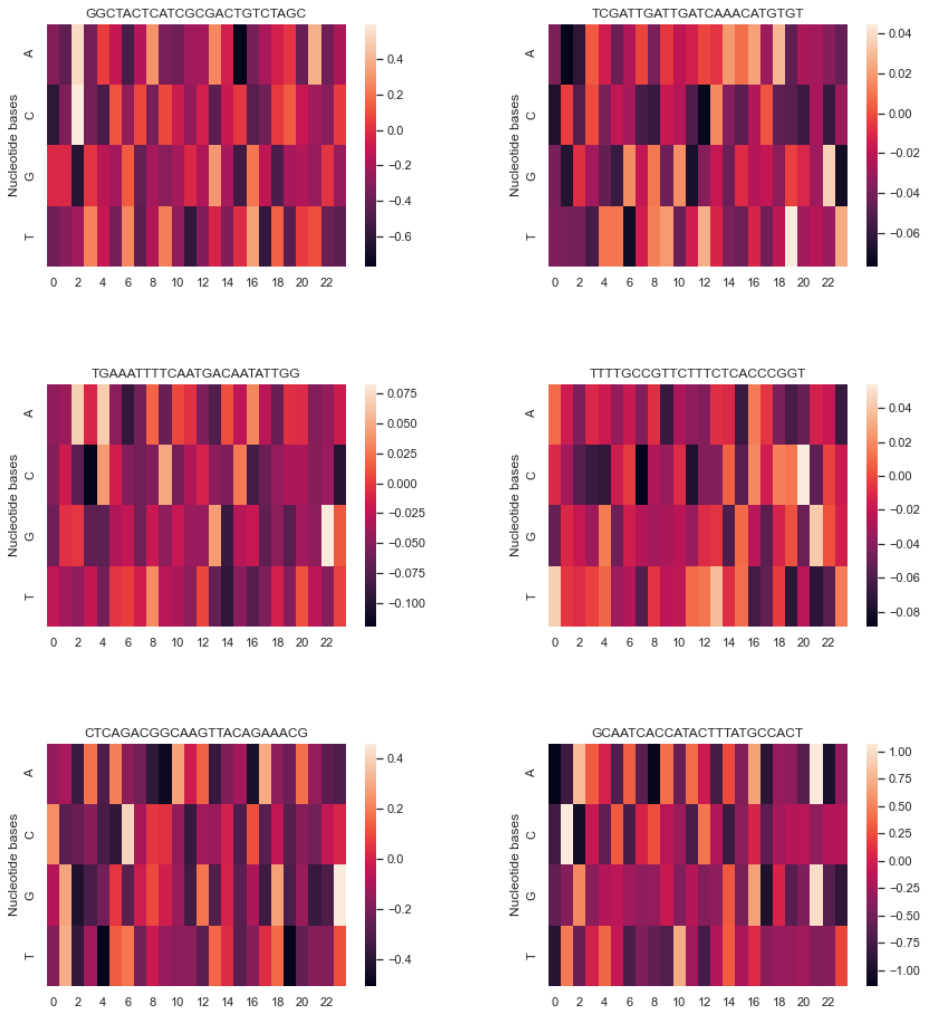


Figure B.1: Filters from the DeepBind model to generate motifs

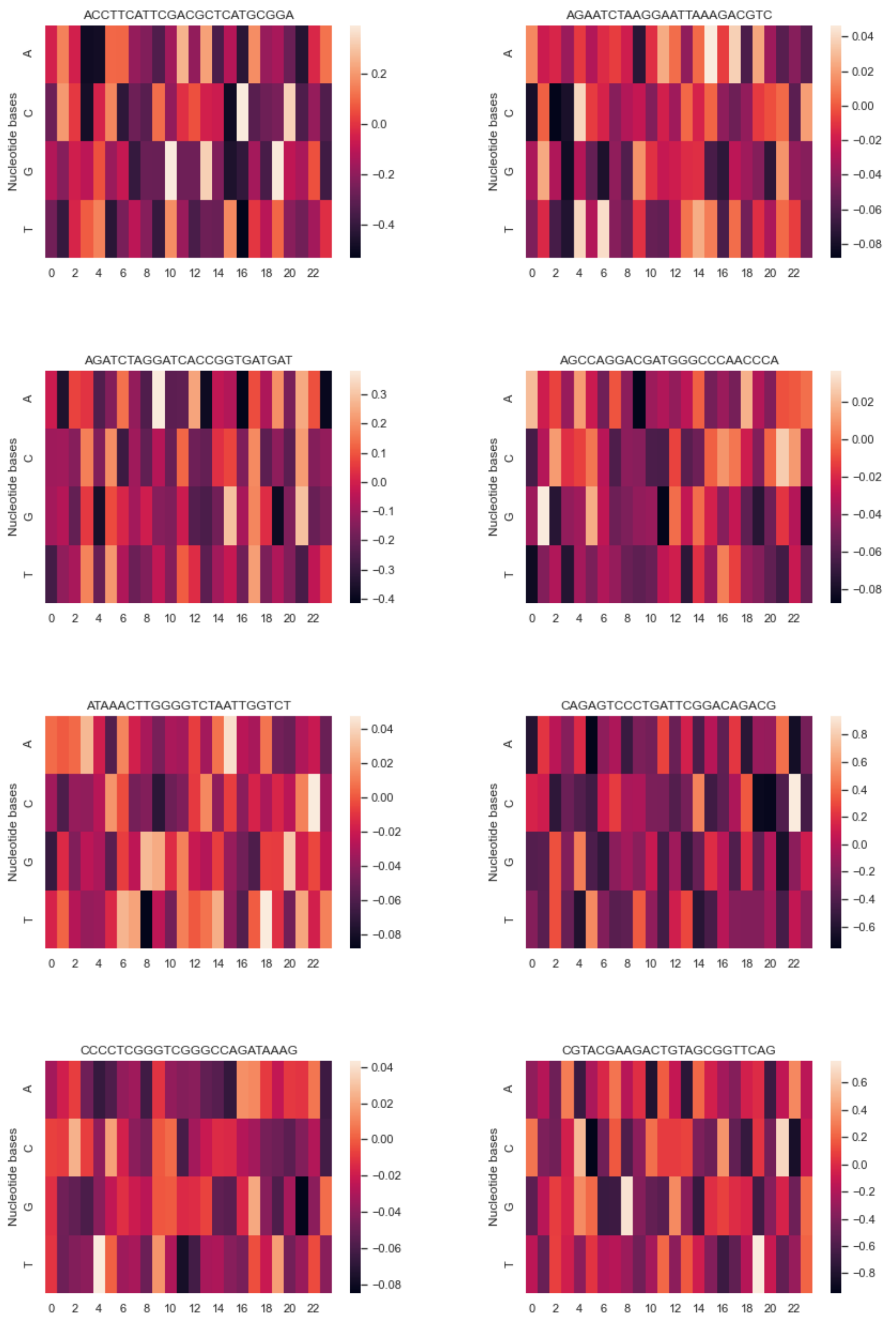
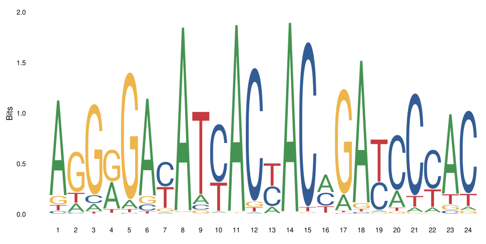


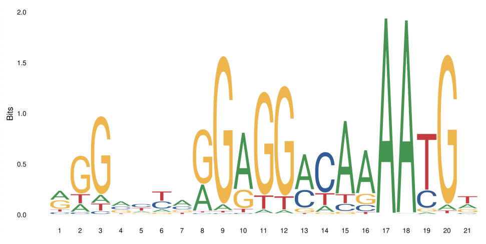
Figure B.2: Filters from the DeepBind model to generate motifs

Appendix C

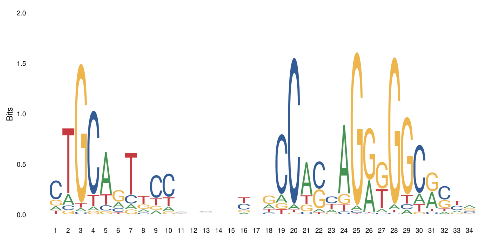
Jaspar DB motifs



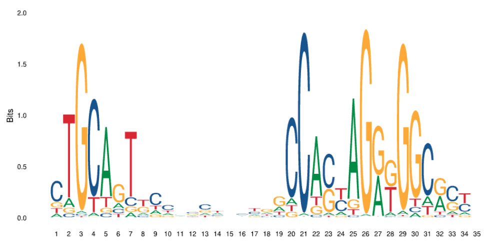
(a) ID: MA1594.1



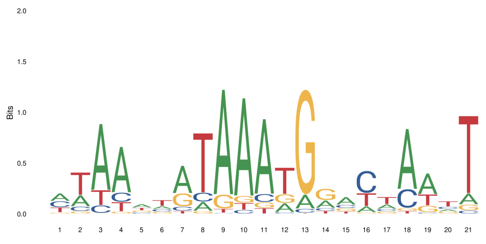
(b) ID: MA1714.1



(c) ID: MA1929



(d) ID: MA1930.1



(e) ID: MA1978.1

Figure C.1: Detailed information of matrix profiles from JASPAR DB [50].