Fakultät für Mathematik
Lehrstuhl für Mathematische Modelle biologischer Systeme

Inference of Gene Regulatory Networks

Master's Thesis by Lorena Mendez

Examiner:	Prof. Dr. Fabian J. Theis
Supervisor:	Volker Bergen
Submission Date:	September 7, 2021

I hereby confirm that this is my own work, and that I used only the cited sources and materials.

München, September 7, 2021

Lorena Mendez

Abstract

Cell identity is defined by an underlying complex system of interactions between transcription factors (TFs) and their target genes also known as Gene Regulatory Network (GRN). The inference of gene regulatory networks is of great interest in the scientific community and has remain as an active area of research for more than 20 years. There are many factors involved that make the inference task extremely challenging. Some of the limitations we have to overcome when inferring GRNs are: the destructive nature of single-cell data, which only allows us to glimpse static snapshots of cellular states and the assumption that correlation between expression patterns correspond to regulatory interactions, which may lead to the detection of spurious network links. We believe that integrating different types of information such as time, RNA velocity, scRNA-seq and scATAC-seq data, we can overcome these challenges and will have a clearer comprehension of the underlying regulatory processes of a cell.

This work seeks to improve the inference of GRN's by incorporating time, scRNA-seq and scATAC-seq data, as a first step before linking regulation with RNA velocity. Having a temporal-dependant GRN inference method allows us to tackle one key aspect of causality: temporal precedence, but also enables us to make a prediction about the future state of a cell using its gene expression profiles. With an inferred future state, we could compute *cell displacements* that directly relate to RNA velocities, which later could help improve GRN inference and viceversa.

Contents

1 Introduction 1

2 Preprocessing and analysis of single-cell RNA sequencing data 3
- 2.1 RNA-seq protocol . . . 3
- 2.2 Preprocessing . . . 4
- 2.3 Advanced Analysis . . . 6

3 Preprocessing and analysis of single-cell ATAC sequencing data 9
- 3.1 ATAC-seq protocol . . . 9
- 3.2 Pre-analysis . . . 10
- 3.3 Core Analysis . . . 13
- 3.4 Advanced Analysis . . . 17

4 Modeling with neural networks 25
- 4.1 Fundamentals . . . 26
  - 4.1.1 Parameter optimization . . . 27
  - 4.1.2 Parameter initialization . . . 28
  - 4.1.3 Weight Normalization . . . 28
  - 4.1.4 Residual Blocks . . . 29
  - 4.1.5 Early stopping . . . 29
- 4.2 Convolutional neural networks . . . 30
  - 4.2.1 Convolutional layer . . . 30
  - 4.2.2 Temporal convolutional neural networks . . . 31
  - 4.2.3 Dilated convolutions . . . 32
  - 4.2.4 Depthwise separable convolutions . . . 32
  - 4.2.5 Attention mechanisms . . . 33

5 The gene thicket: a new method to infer gene regulatory networks 35
- 5.1 ATAC-seq data preprocessing . . . 35
- 5.2 RNA-seq data preprocessing . . . 37
- 5.3 The gene thicket . . . 37
  - 5.3.1 Architecture . . . 37
  - 5.3.2 Interpretability . . . 39
  - 5.3.3 Sign assignment: Activation or Repression . . . 39
  - 5.3.4 Causal validation . . . 40

6 Results 43
6.1 Curated Datasets . . . 43
6.2 Synthetic Datasets . . . 46
6.3 Scalability . . . 48
6.4 Pancreatic Endocrinogenesis . . . 50

7 Conclusion and outlook 53

A Metrics for network evaluation 55
A.1 Area Under the Precision Recall Curve . . . 55
A.2 Area Under the Receiver Operating Characteristic Curve . . . 56

B Datasets 57
B.1 Curated Datasets . . . 57
B.2 Synthetic Datasets . . . 59

C The gene thicket: number of blocks 61

List of Figures 65

List of Tables 69

Index 69

Bibliography 71

Chapter 1

Introduction

The discovery of new cell types and cellular lineage tracing has been only possible with single-cell expression data from singe-cell RNA-sequencing technology. Now, we wonder if we can also infer the underlying gene regulatory networks (GRNs), which control cellular differentiation and are the drivers of cell type transitions. GRNs are represented as graphs, which nodes symbolize genes and edges indicate the existing direct regulatory relationships between two genes. Ideally, the edges have a direction, a weight and a sign. The direction indicates which gene is regulating another gene, the weight reflects the strength of the relationship, and the sign corresponds to either activation (+) or inhibition (-).

Inferring a GRN has many challenges, being causality the main one. For that reason, it has been an active field of research for many years, and there already exist several GRN inference approaches based on single-cell expression data. Some authors developed GRN inference methods based on tree ensemble methods, for example, GENIE3 [IWG+10] (top performer in the DREAM4 *In Silico* network challenge) and GRNBOOST2 [Moe+19]. Other approaches are based on mutual information, like PIDC [CSB17], or on correlation, like PPCOR [Kim15]. Since GRNs are related to the development of cells through time, there have been an increasing number of approaches that incorporate pseudotemporal ordering. For example, regression-based methods like SINCERITIES [PG+18] and GRNVBEM [SC+18], algorithms using ordinary differential equations like SCODE [Mat+17] and GRISLI [AFV20], SINGE [Des+21] based on Granger causality, LEAP [SL17] based on correlation, or SCRIBE [Qiu+18] based on mutual information.

Deep learning was an unpopular approach to infer GRNs because they are difficult to interpret. But now, with the development of interpretability methods and the fast paced progress in deep learning, an increasing number of GRN inference algorithms based on neural networks have been developed. For example, CNNC [YBJ19] and DeepDRIM [Che+21] use convolutional neural networks to infer relationships from an image of a histogram from each pair of genes in a supervised way; Neural Granger Causality [Tan+18] uses either a recurrent neural network or a multilayer perceptron with sparsity-inducing penalties to infer causal gene interactions; and GRGRNN [Wan+20] infers GRNs expressing them as a graph classification problem using supervised and semi-supervised graph neural networks.

Although single-cell expression data is promising for computing GRNs, there are additional difficulties that we must consider when working with this data. Some of these challenges are: high sparsity due to dropouts [KSS14], effects related to the cell-cycle [Bue+15], cellular heterogeneity [WRY16] and variation in sequencing depth among the cells. For this reason, recently developed algorithms integrate different complementary data sources to improve the discovery of causal relationships. This is known as multi-view learning and has already proven to be more informative in GRN inference (e.g. [Zhu+08], [Hec+09] and [AAG14]). From this idea, SCENIC [Aib+17] was created. SCENIC integrates iRegulon [Ver+15] ranking databases and makes use of different algorithms: either GENIE3 [IWG+10] or GRNBoost2 [Moe+19] to infer putative regulatory links, RcisTarget [Imr+15] and iRegulon databases to construct regulons and AUCell to estimate regulon (subgraph of a transcription factor and its target genes) activities

More recently, ATAC-seq data has become popular among scientists, as it provides key insights for gene regulation and requires less cells, time and money than previous technologies for epigenomic readouts. So far, not many methods integrate scATAC-seq data to infer GRNs, as far as we know CellOracle [KHM20] is a pioneer doing that. The main idea of CellOracle is to infer the expression of every gene as a linear function of the expression of its respective transcription factors (TFs) to be able to simulate cell identity after perturbation. This algorithm can be summarized in the following steps: (1) detection of putative TF-gene bonds using scATAC-seq data and scRNA-seq preprocessing, (2) GRN inference using scRNA-seq data and presumptive TF-gene bonds, (3) perturbation simulation.

In this work, we explore the idea of inferring GRNs using a combination of cellular pseudotime, complementary information as in SCENIC or CellOracle, and neural networks. We take particular inspiration in TCDF [NBS19], a model based on convolutional neural networks and attention, created to identify causal links in time series data. This architecture has been applied to simulated financial market data and simulated functional magnetic resonance imaging, but it has never been used in the context of GRN inference.

Chapters 2 and 3, will give an overview of the preprocessing of scRNA-seq and scATAC-seq data, respectively. Then, Chapter 4 focuses on the fundamentals of deep learning modeling. Next, in Chapter 5, we will introduce our model: **the gene thicket** and describe how it works. Following, Chapter 6 describes the results in curated, simulated and biological data. Finally, in Chapter 7, we will discuss the findings, conclude the work and give ideas for future work.

Chapter 2

Preprocessing and analysis of single-cell RNA sequencing data

Gene expression is the process in which the cells use the information encoded in a gene to produce *gene products* (usually proteins) that regulate all the mechanisms in our body. Two of the main components of gene expression are **transcription** and **translation**, represented in Figure 2.1. In the *transcription* part, the information encoded in a gene is copied from DNA into a single strand molecule called messenger RNA (mRNA). While, in *translation*, a gene product is produced according to the instructions of the mRNA molecule.

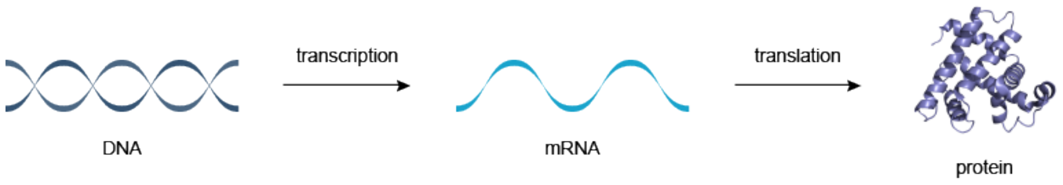


Figure 2.1: Transcription and translation [Bro]

In the transcription process, a group of proteins called transcription factors regulate gene expression. Being proteins themselves, they are also product of gene expression. That means, that the transcription of some genes regulates the transcription of other genes, these are the so-called gene regulatory networks (GRNs) that we want to model (see Figure 2.3). Therefore, the fundamental information for our model is the gene expression profiles at single-cell resolution that the scRNA-seq protocol provides.

In this chapter, we will describe the RNA sequencing workflow and the preprocessing of the data following closely the work of [LT19].

2.1 RNA-seq protocol

Multiple steps conform the RNA-seq protocol: single-cell dissociation, single-cell isolation, library construction, and sequencing. *single-cell dissociation* refers to the process in which the biological tissue samples are digested to generate a single-cell suspension. Then, as its name suggests, in *single-cell isolation* cells are isolated for an individual mRNA profiling. Scientists usually capture each cell, either in a microfluid droplet or wells on a plate, depending on the experimental setup. The mRNA of each cell is labeled with a barcode that is either droplet- or well-specific. This particular step is prone to many errors, which can end up in capturing multiple cells (*doublets or multiplets*), capturing non-viable cells, or simply not capturing a cell (*empty droplets/wells*). Later, the After cell isolation, we capture the cell's mRNA, transcribe it to complementary DNA (cDNA) and amplify it, in the step known as *library construction*. Later, the cDNA libraries are annotated with cellular barcodes and captured molecules are often labeled with a unique molecular identifier (UMI). Finally, the cDNA libraries are pooled together (multiplexed) for *sequencing* to produce read data. Before the analysis of read data, we need to perform quality control, group the data based on their assigned barcodes (demultiplexing) and align in read-processing pipelines. In case of UMIs protocols, read data can be demultiplexed to produce count data of captured mRNA molecules.

A comparison of detailed protocols and a more extensive description can be found in [Zie+17], [Mac+15] and [Sve+17].

2.2 Preprocessing

The preprocessing of RNA-seq data can be summarized in the following steps:

1. **Quality control.** In this step, we examine the distribution of the number of codes per barcode (count depth), the number of genes per barcode, and the fraction of counts from mitochondrial genes per barcode. Outliers of these distributions usually correspond to doublets, dying cells, or cells whose membranes are broken. To minimize the false positive rate, it is recommended to analyze the distributions together.

2. **Normalization.** We must normalize count data to make the counts comparable between cells and to be sure that the differences we observe among cells are not a consequence of technical measurements, but due to biological reasons. There exist many normalization methods that can be applied to RNA-seq data, and indeed they perform optimally in different datasets [Col+19]. The most popular normalization method is *counts per million normalization*, also known as *CPM normalization*. However, there are many others like a *CPM normalization extension* proposed by [WWK18] and *Scran's pooling-based size factor estimation method* [LB+]. Some authors also normalize gene counts to have a better gene-to-gene comparison. This normalization is a scaling of gene counts to have a mean of zero and a standard deviation of one (z scores). Nevertheless, others prefer to omit scaling because the importance of a gene is reflected in the magnitude of its expression.

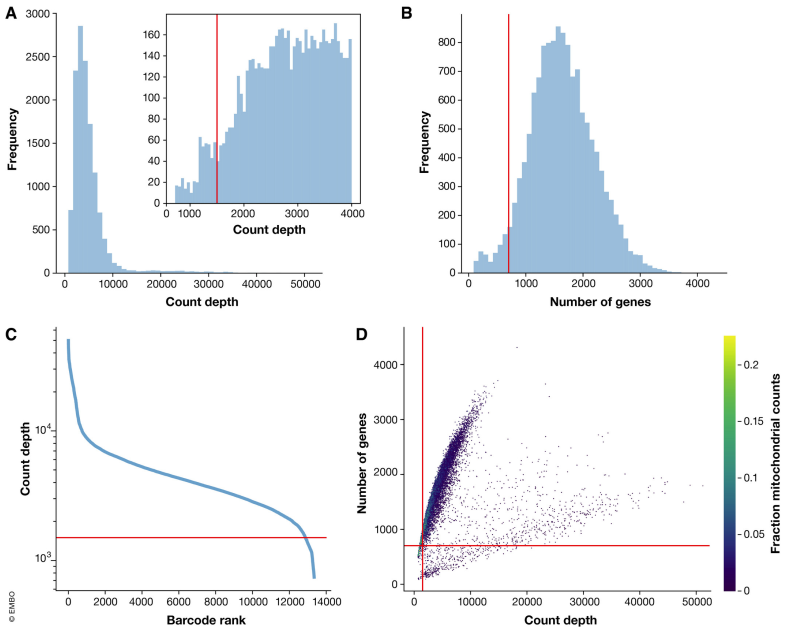


Figure 2.2: RNA-seq quality control distributions with filtering decisions for a mouse epithelium dataset from [Hab+17]. Image source: [LT19]

3. **Log-transformation.** Now, we will transform the count data matrix using the transformation *log(x+1)*. There are three main reasons for applying this transformation:

   - the standard way to measure changes in gene expression is through log fold changes, which are represented by the log-transformed expression values,
   - the mean-variance relationship of single-cell data [Bre+13] is mitigated,
   - and the skewness of the data is reduced, approximating now more to a normal distribution.

4. **Correction for biological and technical effects.** Sometimes, it is useful to remove cell cycle and count depth effects from the data, for example, when inferring developmental trajectories [Bue+15]. We can remove these effects by performing a simple linear regression as in Scanpy [WAT18] or Seurat [But+18], or we can also apply more complex models as scLVM [Bue+15] or f-scLVM [Bue+17].

5. **Correction for batch effects.** When cells come from different experiments, we may observe differences due to the variety of conditions they were handled. This is known as batch effect. There are multiple methods that we can apply to correct

for batch effects. For example, ComBat [JLR07], or data integration methods like scGen [LWT18] and Canonical Correlation Analysis [But+18].

6. **Feature selection.** The idea of this step is to maintain only the highly variable genes (HVGs) [Bre+13], which are the most informative genes in the data. On practice, between 1,000 and 5,000 genes are selected.

7. **Dimensionality reduction and visualization.** The cells can be described by fewer dimensions than the number of their measured genes. We aim to find these dimensions to get a better visualization of the data in two or three dimensions. There are many algorithms for dimensional reduction, for example, principal component analysis (PCA), diffusion maps [Coi+05], uniform approximation, and projection method (UMAP) [MHM18] and t-distributed stochastic neighbourhood embedding (t-SNE) [VDM09].

8. **Imputation.** Single-cell data contain many sources of noise, being drop-outs one of the main ones. Imputation seeks to reduce the level of noise by filling out the zeros, caused by drop outs, with an appropriate value. The method that we use is based on the k-nearest neighbor (KNN) graph G from the previously filtered and normalized count matrix X. Each node of G represents a cell, and each edge between two nodes v, w, indicates that they are in each others' k-neighborhood. There are more sofisticated methods based in k-nearest neighbors graphs, that also aim to reduce batch effects, such as batch balanced KNN (BBKNN) [Pol+20] or Harmony [Kor+19].

2.3 Advanced Analysis

In this section, we will list some of the different methods available, designed to extract crucial biological insights and describe the underlying biological system.

Clustering

There exist different types of cells that are involved in different biological processes. For example, in the pancreas, the *alpha cells* produce the hormone *glucagon*, which increases the levels of glucose in the blood. But there are also *beta cells*, that are crucial for the production of insulin, which decreases the levels of glucose in the blood. These two types of cells are present in the pancreas, and have different biological functions. By clustering, we aim to identify the different types of cells in a tissue, grouping them by the similarity of their gene expression profiles. There are many algorithms that we can use for this step, for example, k-means, hierarchical clustering, and Louvain community detection [Blo+08]. Nowadays, Louvain is the default method used by single-cell analysis platforms Scanpy [WAT18] and Seurat [Hao+21].

Cluster annotation
After we identified clusters of cells, we would like to validate and annotate them with a meaningful biological label. Since sometimes cells that belong to the same cell type are in different clusters due to their cell states, the clusters are referred to as cell identities instead of cell types. Clusters are usually annotated according to the *marker genes*, i.e. the gene signatures of each cluster. To improve the annotation process, reference databases like the mouse brain atlas [Zei+18] or human cell atlas [Reg+17] are becoming more available.

Trajectory inference
Gene expression is a dynamic process that occurs in the cell. RNA-seq data capture snapshots that can reveal the development of a cell, for instance, the transitioning between cell identities, cell differentiation or simply changes in the biological function. Trajectory inference methods aim to reconstruct this continuous process by finding paths in the cellular space that minimize the transcriptional changes between neighboring cells. Here, the time is a latent variable described by a proxy variable called *pseudotime* that ranges from zero to one. There are many available methods such as Monocle [Tra+14], Wanderlust [Ben+14] or Slingshot [Str+18].

Gene regulatory networks
Complex regulatory interactions between genes and molecules determine gene expression, as seen in Figure 2.3. Gene regulatory networks (GRNs) aim to describe these underlying relationships. Many algorithms infer GRNs based on gene co-expression measurements and rely on different methods such as regression, correlation, mutual information, differential equations, three ensemble methods, or even neural networks. Some of them are related to trajectory inference, and others include additional information such as ATAC-seq data, covered in Chapter 3. A couple of existing methods is listed on Chapter 1.

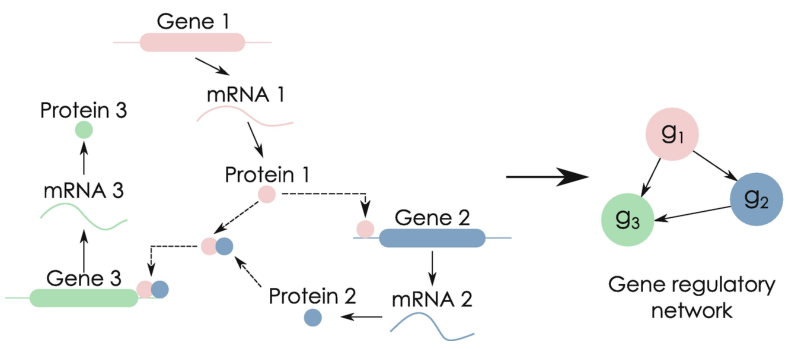


Figure 2.3: Regulatory interactions and GRNs [San+19]

Chapter 3

Preprocessing and analysis of single-cell ATAC sequencing data

The compaction of chromatin (how DNA is stored) determines the gene regulation. The DNA wraps itself around histone proteins, that later form nucleosomes, make a high order structure and loop themselves into the chromosome shape. This packaging is not just structural to reduce the size of the DNA, it is also functional because it determines the accessible DNA regions for RNA polymerase II to start transcription, i.e., the open chromatin regions are the regions that contain the genes that will be transcribed.

Several technologies can be used to identify open chromatin, including DNase-seq, FAIRE-seq, and ATAC-seq. Nowadays, ATAC-seq is the most popular method because it requires less cells, less time and money than the others. In this chapter, we will describe how ATAC-seq works, how to analyze it and the existing tools to perform this analysis.

3.1 ATAC-seq protocol

The ATAC-seq protocol consists of two steps: Tn5 insertion and PCR amplification.

1. **Tn5 insertion:** ATAC-seq uses a genetically engineered hyperactive Tn5 transposase to simultaneously fragment and tag the accessible regions of the genome with sequencing adapters.

2. **PCR amplification:** The tagged DNA fragments are purified, PCR-amplified and sequenced using high throughput sequencing [Pic+14].

ATAC-seq is a short name of assay for Transposase-Accessible Chromatin using sequencing. With ATAC-seq we can get accurate and sensitive measurements of chromatin accessibility in less than one day and it can be performed even using around 500 cells. In contrast, DNase-seq and FAIRE-seq published protocols have considerable more steps, require more than two days of preparation and at least one million cells (FAIRE-seq) or 50 million cells (DNase-seq). A better comparison can be seen in Figure 3.2.

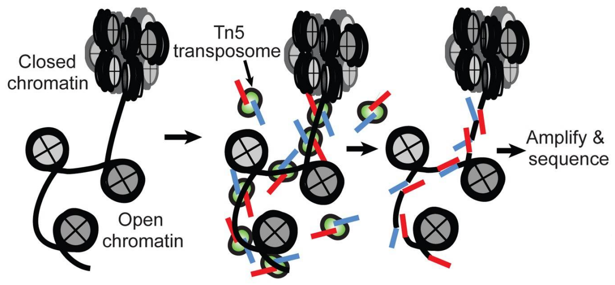


Figure 3.1: Tn5 (green) with sequencing adapters (red and blue) binds to the open chromatin region and cuts the DNA into fragments [Bue+13].

The number of ATAC-seq publications and data has been increasing exponentially since first described in 2013. It shows great potential, specially when trying to disentangle the regulatory interactions that occur in a cell. Recently, different tools and methods have been created for ATAC-seq data analysis. We will quickly describe them in the following sections, following closely the information from the hitchhiker's guide to ATAC-seq data analysis [Yan+20], Harvard ATAC-seq guidelines [Gasa], the assessment of computational methods for the analysis of single-cell ATAC-seq data [Che+19] and Signac [Stu+20].

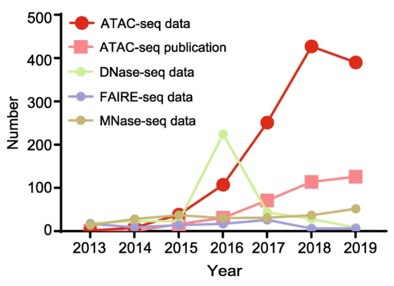
3.2 Pre-analysis

The pre-analysis of ATAC-seq data can be summarized into the following steps:

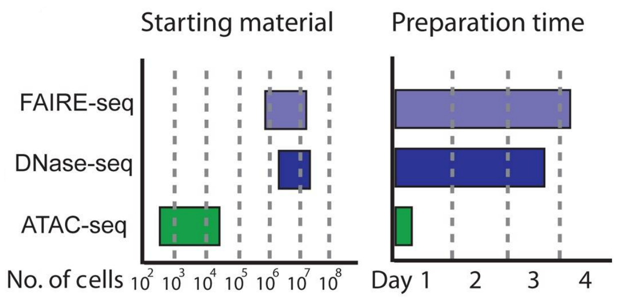
1. **pre-alignment quality control.** This step refers to all base quality scores, GC content, sequence length distribution, sequence duplication levels, k-mer overrepresentation and contamination of primers and adapters in the sequencing data. FastQC [And10] is a popular tool for this step, because it is efficient. It can process a file of around 20 million reads in approximately 5 minutes, using less than 250MB of memory.

2. **adapter removal.** We should remove adapters, in case that the reads appear to be contaminated with them, because this could negatively impact the aligning of the sequences to the reference genome. We can suspect that the reads are contaminated with adapters either from the FastQC report (sections: "Overrepresented sequences" and "Adapter content") or from the size distribution of the sequencing library. There are many tools that can be used for adapter removal, here we will mention two:

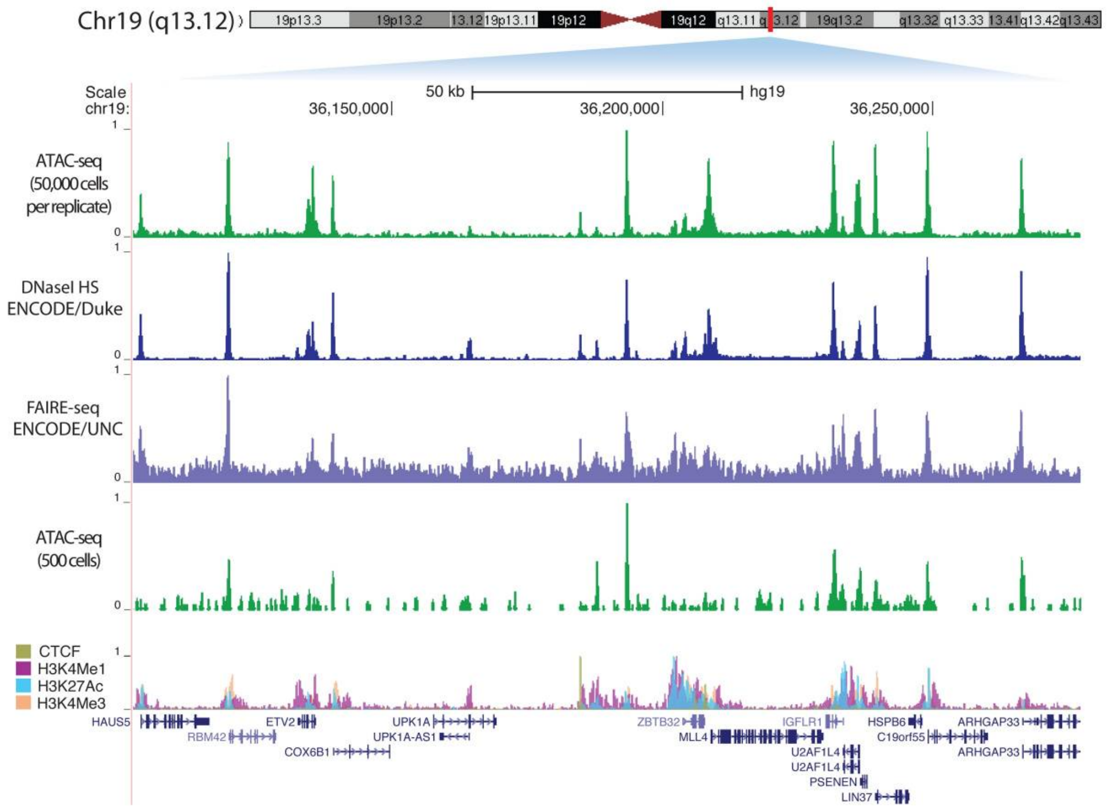
   - *Cutadapt [Mar11].* It looks for a specific given adapter sequence through all the reads, removes the adapter and everything that follows.



(a)



(b)



(c)

Figure 3.2: a ATAC-seq datasets and publications are increasing exponentially b approximate input material and preparation time of open chromatin assays c comparison of open chromatin assays at a locus GM12878 lymphoblastoid. ([Yan+20] [Bue+13])

- *NGmerge [Gas18b].* It aligns each pair of reads, and if they align with 3' overhangs, it clips the overhangs of both reads.

Cutadapt works only if we know the adapter that was used and requires a parameter that specifies the minimal length of match for the adapter sequence (default 3bp). Alternatively, NGmerge works only with paired-end sequencing. With these tools, we can also remove low-quality bases.

3. **read alignment to a reference genome.** BWA-MEM [LD09] and Bowtie [LS12] are two of the most popular aligners, because of their efficiency. According to [Yan+20], a successful ATAC-seq experiment usually has a unique mapping rate over 80%.

4. **post-alignment processing and quality control.**
   - **Fragment size distribution:** we should observe a clear periodicity every 200bp corresponding to nucleosome-free fragments, single nucleosomes, dimers, trimers, etc. as it is shown in Figure 3.3.

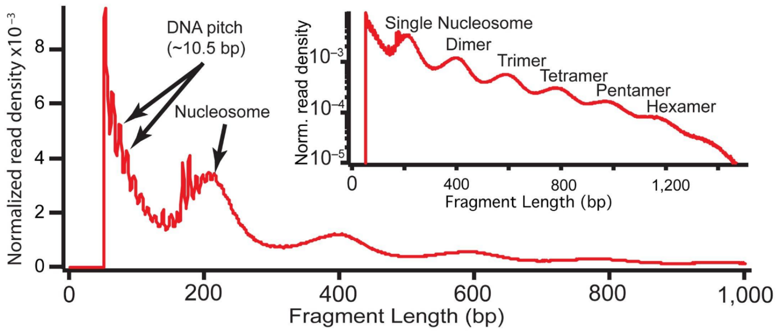


Figure 3.3: Successfull ATAC-seq length distribution from [Bue+13].

   - **Transcriptional start site (TSS) enrichment score:** it is the ratio between the aggregated distributions of reads centered on the TSSs and reads flanking the corresponding TSSs as shown in Figure 3.4. The ENCODE project [Con+12] defined this score to evaluate ATAC-seq data. We should remove cells with low TSS enrichment.
   - **Black list regions:** the ENCODE project has created a list of regions representing reads, which are often associated with artefactual signal. Cells that show a high proportion of reads mapping to the black list should be removed, since they frequently represent technical artifacts [AKB19].
   - **Duplicated reads:** are likely coming from PCR artifacts [PMC18], so they should be removed.

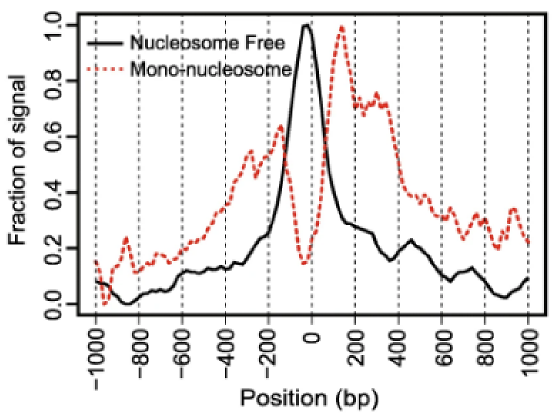


Figure 3.4: TSS enrichment plot from [Yan+20]. Nucleosome fragments are enriched at TSS, while mono-nucleosome fragments are enriched at flanking regions.

- **Mitochondrial reads:** are more accessible due to the lack of chromatin-packaging [Bog12], for which they also must be removed.

Yan et.al [Yan+20] suggest the following pipeline for the pre-analysis of ATAC-seq data: FASTQC [And10] → trimmomatic [BLU14] → BWA-MEM [LD09] → ATACseqQC [Ou+18].

3.3 Core Analysis

In the core analysis, we will identify the accessible chromatin regions (peaks) to later construct a feature matrix and be able to obtain crucial information from the data.

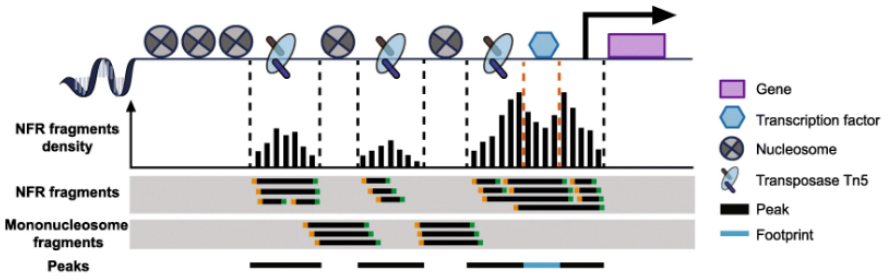


Figure 3.5: Tn5 transposase fragments and tags the accessible regions with sequencing adapters to identify open chromatin regions (black) and transcription factor footprints (blue). The nucleosome-free region (NFR) fragments are represented by peaks [Yan+20].

The core analysis we described closely follows the information described in [Che+19] and can be summarized in the following steps:

1. **Define regions:** this is perhaps the most important part of ATAC-seq data analysis, because it is the basis for constructing of the feature matrix. Most of the methods define their regions based on peak calling, and only a few of them, segment the genome into bins of the same size and count the fragments within each bin (eg. SnapATAC [Fan+21] and Cusanovich2018 [Cus+18]).
   The peaks are the most popular choice, because they represent fragments of open chromatin regions. In Figure 3.6, we can observe a comparison of different peak calling algorithms from [Che+19]. HMMRATAC [TL19] is a method based on Hidden Markov Models, that learns the structure around the chromatin accessible regions. It splits the genome into high signal (open chromatin), moderate signal (nucleosome regions) and low signal (background). MACS2 [Zha+08], Homer [Dut+19], epic2 [SS19] , F-seq [Boy+08] and JAMM [ILO15], compare the shape of the read distribution in a specific region to a random background. And PICS [Zha+11] estimates enrichment scores for each putative peak region.

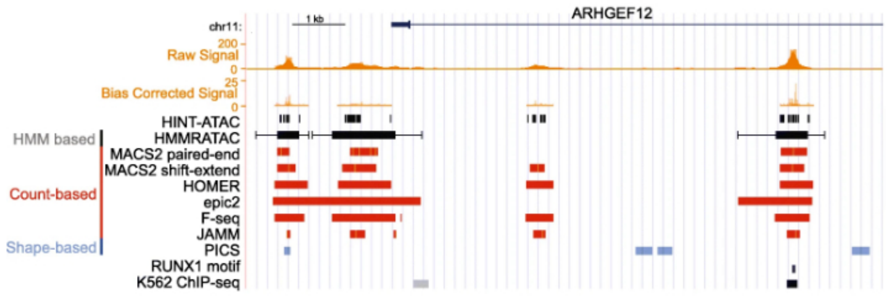


Figure 3.6: Comparison of distinct peak calling algorithms from [Che+19].

   The only aforementioned method that was designed specifically for ATAC data is HMMRATAC. However, more methods have become available recently, like an improved version of MACS2 [Gas18a]. [Gasa] recommend to use Genrich [Gasb], a tool developed by the Harvard's Informatics group that combines all the post-alignment steps and peak calling.

2. **Filter cells:** we need to filter low-quality cells and cells that may represent technical artifacts, such as doublets. Therefore, we must exclude:
   - cells with very few or very high fragments overlapping peaks.
   - cells with a low proportion of fragments overlapping peaks (number of fragments overlapping peaks divided by the total number of fragments $<$ 15-20%).

3. **Filter peaks:** after we filtered cells, we need to make sure to keep only the peaks that are present in one or more of the remaining cells, i.e. non-zero peaks.

4. **Binarization:** once filtering is complete, we can build a peak count matrix, where the rows are the cells, the columns are the peaks and each entry represents the number of fragments corresponding to the specific cell and peak. Since single-cell data are sparse, we do not expect to have more than one read per cell per peak. Therefore, it is convenient to binarize the count matrix for better results. Then, each entry would tell us which regions are open or not in the respective cells.

5. **Normalization:** this is a crucial step when dealing with count data, so we can make sure that the differences among cells have a biological meaning and are not observed due to technical reasons. In this work, we will focus on a two-step normalization procedure, that is a modified version of term frequency-inverse document frequency (TF-IDF) by [Stu+20]. The idea is to normalize cells to account for differences in sequencing depth and peaks to make the differences among them more pronounced. TF-IDF is a popular algorithm in natural language processing used to identify the importance of a word in a document. As its name suggests, it is composed of two terms: the term frequency (TF) and the inverse document frequency (IDF). The TF indicates the relative frequency of a word w in a document, and the inverse document frequency represents the importance of a word based on its rareness. For example, words like "the", "is", "that", appear frequently in most of the documents and they may not be relevant to infer its content. At the end, we can generate a word-document matrix, where each entry $TF(w, d) \times IDF(w)$ will indicate the importance of word w in document d.
   TF-IDF was first used in the context of chromatin data by [Cus+15]. Here, the term frequency indicates the relative frequency of a peak in a cell and is computed as:
   $$TF(i, j) = \frac{\text{total number of counts for peak } i \text{ in cell } j}{\text{total number of counts for cell } j}$$
   and the inverse document frequency represents the importance of a peak based on its uniqueness. It is estimated as:
   $$IDF(i) = log(1 + \frac{\text{total number of cells in the dataset}}{\text{number of counts of peak } i \text{ across all cells}}).$$
   However, when computing the TF-IDF matrix, there are some non-zero entries of the matrix, which mean is close to zero and variance is low. This could result in a poor discrimination between cell types. To overcome this situation, [Stu+20] introduced a modified version of TF-IDF, in which the inverse document frequency is estimated as:
   $$IDF(i) = \frac{\text{total number of cells in the dataset}}{\text{number of counts of peak } i \text{ across all cells}}$$
   and the TF-IDF entries of the final matrix are transformed as:
   $$log(1 + (TF \times IDF) \times 10^4).$$

This TF-IDF modified version showed an improvement in preserving local neighborhoods and cell type separation, when using singular value decomposition (SVD) as dimensional reduction method and uniform manifold approximation and projection (UMAP) for visualization. The combination of TF-IDF and SVD is known as latent semantic indexing (LSI) and it is scalable when using the modified TF-IDF method [Stu+20], because preserves sparsity, and efficient, because SVD is highly optimized to deal with sparse matrices ([BR05], [BRL19]).

6. **Dimensionality Reduction and Visualization:** Often, the data have many features (dimensions) that describe the cells. However, not all features carry the same amount of information. If we keep them all, we could observe several issues like expensive computations, highly correlated dimensions, the data could be hard to visualize and the need of exponential amounts of data to characterize the density (curse of dimensionality). The relevant features are the ones with highest variance, which means, the ones who have the most different behavior across cells. One of the most popular techniques is singular value decomposition (SVD).

   Proposition 3.1 (Singular Value Decomposition)
   Let $X \in \mathbb{R}^{m \times n}$ be a matrix, then X can be written as: $X = U\Sigma V^T$, where U is an orthogonal $m \times m$ matrix, which columns are the eigenvectors of XX^T; V is an orthogonal $n \times n$ matrix, which columns are the eigenvectors of X^TX; and Σ is a $n \times n$ diagonal matrix with the square roots of the eigenvalues of X^TX in the diagonal (also known as singular values), which are written in decreasing order.

   We can use SVD to find the best low-rank approximation of any given matrix $X \in \mathbb{R}^{m \times n}$ with the Eckart-Young-Mirsky theorem 3.2 [EY36].

   Theorem 3.2 (Eckart-Young-Mirsky)
   Let $X = U\Sigma V^T \in \mathbb{R}^{m \times n}$ be the SVD of matrix X. Then, $\forall r \in \mathbb{N}$, such that $1 \leq r \leq k = rank(X)$; U, Σ and V can be partitioned as follows:

   $$U =: \begin{bmatrix} U_1 & U_2 \end{bmatrix}, \quad \Sigma =: \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix}, \quad \textit{and} \quad V =: \begin{bmatrix} V_1 & V_2 \end{bmatrix},$$

   where $U_1 \in \mathbb{R}^{m \times r}$, $\Sigma_1 \in \mathbb{R}^{r \times r}$, $V_1 \in \mathbb{R}^{n \times r}$, and the matrix $\widehat{X}^* = U_1\Sigma_1V_1^T$ is the best rank-r approximation matrix of X, i.e.

   $$\|X - \widehat{X}^*\|_F = \min_{\mathrm{rank}(\widehat{X}) \leq r} \|X - \widehat{X}\|_F = \sqrt{\sigma_{r+1}^2 + \cdots + \sigma_m^2}$$

   Usually, the first component of LSI is highly correlated with the total number of counts of each cell, which means, that it captures technical variation (sequencing depth). Therefore, is a common practice to remove it from the analysis. We can also use other techniques such as Uniform Manifold Approximation and Projection (UMAP) [MHM18], t-distributed Stochastic Neighbor Embedding (t-SNE) [VDM09] or DDRTree [Mao+16].

7. **Clustering:** The idea of clustering is to be able to identify the distinct type of cells present in our data according to their open chromatin differences. There are many algorithms that we can use for this step, for instance, k-means, hierarchical clustering, Louvain community detection [Blo+08] or the smart local moving algorithm [WVE13].

8. **Annotation:** we can perform a more targeted analysis by counting the peaks overlapping specific regions of the genome, such as promoters, enhancers and gene bodies. This is to get more information about the underlying dynamics of the cell, for example, which motifs are available for the transcription factors to bind, or which genes are in the open chromatin to be transcribed. There are many databases that we can use, for example, JASPAR [Kha+18], Homer [Hei+10], SwissRegulon [Pac+07], Factorbook [Wan+12], HOCOMOCO [Kul+18], GimmeMotifs [BVH18] and others. In particular, GimmeMotifs created a non-redundant database including the information about all the aforementioned ones.

3.4 Advanced Analysis

In this section, we will describe different analysis that we could perform to get even more information from the ATAC data.

Co-accessibility

Peaks represent open chromatin regions, in which we can find different cis-regulatory elements, such as promoters and enhancers, that influence the transcription of a gene. Promoters are close to the gene body, while enhancers are usually distant. To predict the interaction between the cis-regulatory elements, Cicero [Pli+18] estimates co-accessibility scores and connects the regulatory elements to their putative target genes. First, Cicero groups similar cells using K-nearest neighbors in a low-dimensional space (UMAP, tSNE, etc.) to make the data more dense. Then, it builds a matrix of raw correlation between peaks, which we can think of as a weighted non-directional graph, where the nodes are the peaks, and the edges are the correlations among them. Having this in mind, Cicero uses the Graphical Lasso algorithm [FHT08] to prune the connections between peaks, penalizing the correlations according to distance (the farther the peaks are, the greater the penalty is, and vice-versa). The output of Cicero are the co-accessibility scores for all pairs of peaks within 500kb of one another. An example can be seen in Figure 3.8. Additionally, Cicero uses the Louvain community detection algorithm [Blo+08] to find clusters of co-accessible peaks and construct modules, also known as cis-co-accessibility networks. An overview of Cicero can be seen in Figure 3.7.

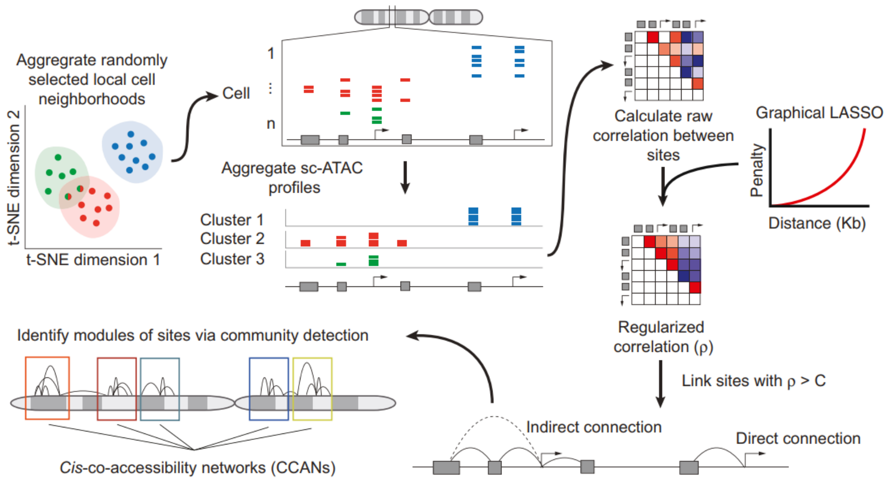

Aggregate randomly selected local cell neighborhoods
t-SNE dimension 2
t-SNE dimension 1
Cell
1
n
Aggregate sc-ATAC profiles
Cluster 1
Cluster 2
Cluster 3
Calculate raw correlation between sites
Graphical LASSO
Penalty
Distance (Kb)
Regularized correlation (ρ)
Link sites with ρ > C
Indirect connection
Direct connection
Identify modules of sites via community detection
Cis-co-accessibility networks (CCANs)

Figure 3.7: Overview of Cicero by [Pli+18]. Cicero finds putative cis-regulatory interactions through co-accessibility.

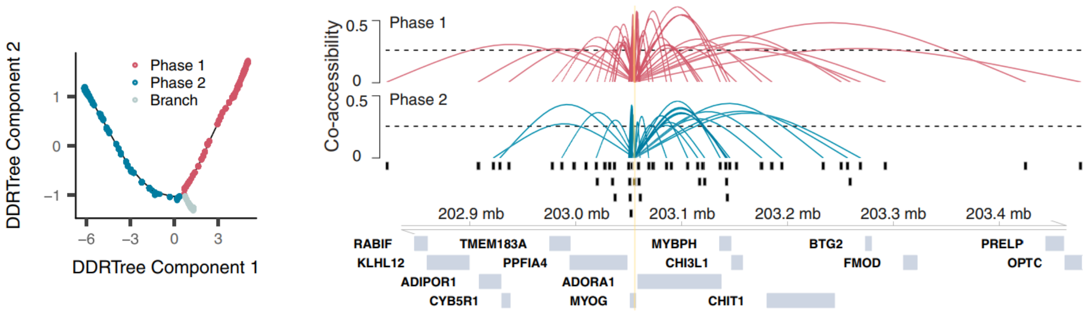


Figure 3.8: Example of Cicero co-accessibility links between MYOG promoter and surrounding distal sites. Red links correspond to phase 1, while blue links to phase 2. The height of connections indicates the magnitude of the Cicero co-accessibility score between the connected peaks ([Pli+18]).

Trajectories

The dynamics of chromatin openness are captured as a snapshot by ATAC-seq data. These snapshots are part of the continuous development of a cell. The idea of building trajectories is to reconstruct this developmental path minimizing the observed changes among peaks between neighboring cells. The ordering of cells along these paths serves as a proxy for developmental time and is known as pseudotime. Trajectory inference was first done using gene expression and served as an inspiration for the open chromatin version. The most known methods for RNA-seq trajectories ([Tra+14], [Ben+14]) are the ones often used for ATAC-seq data.

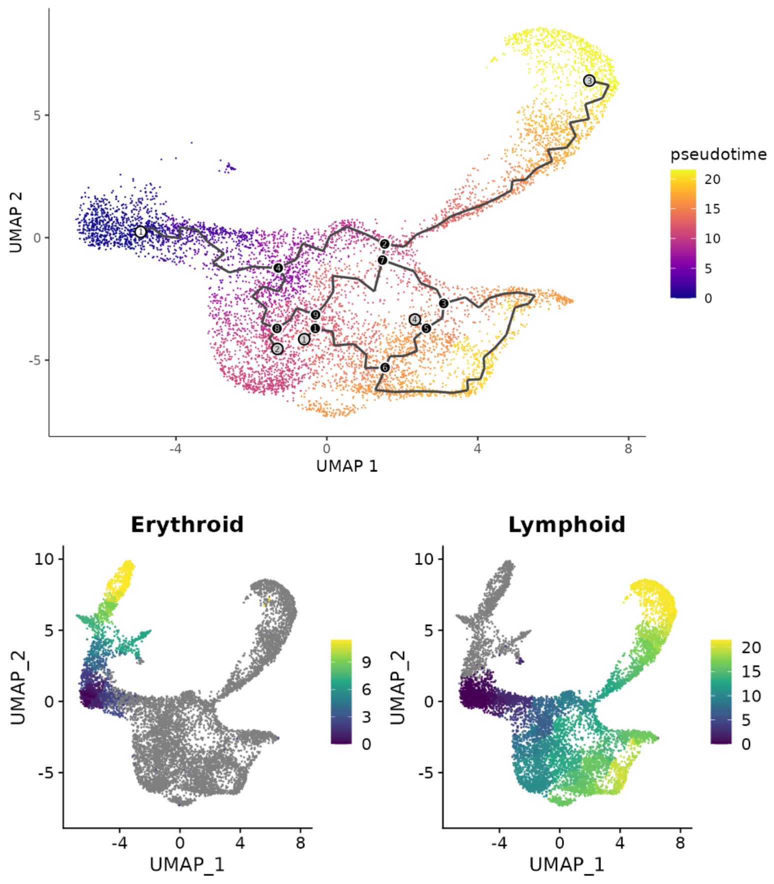


Figure 3.9: Trajectories on CD34+ hematopoietic stem and progenitor cells using Monocle3 [Tra+14]. Erythroid cells are earlier in time than Lymphoid ([SSb]).

Differential Analysis

We would like to identify the most important differences with respect to chromatin accessibility in the cells. To do so, [Stu+20] suggests using the logistic regression method from [Ntr+19]. For this, we must select the two groups of cells we would like to compare. Then, we will adjust a logistic regression model for the cells, using the peak counts as features and group membership as the binary target variable. We will later perform a likelihood ratio test to assess the goodness of fit and adjust the p-values using the Bonferroni correction on the total number of peak counts in the data. Another idea would be to take a look at the fold change accessibility (in peaks). Additionally, we could find the closest gene to each peak and have a better interpretation of the results.

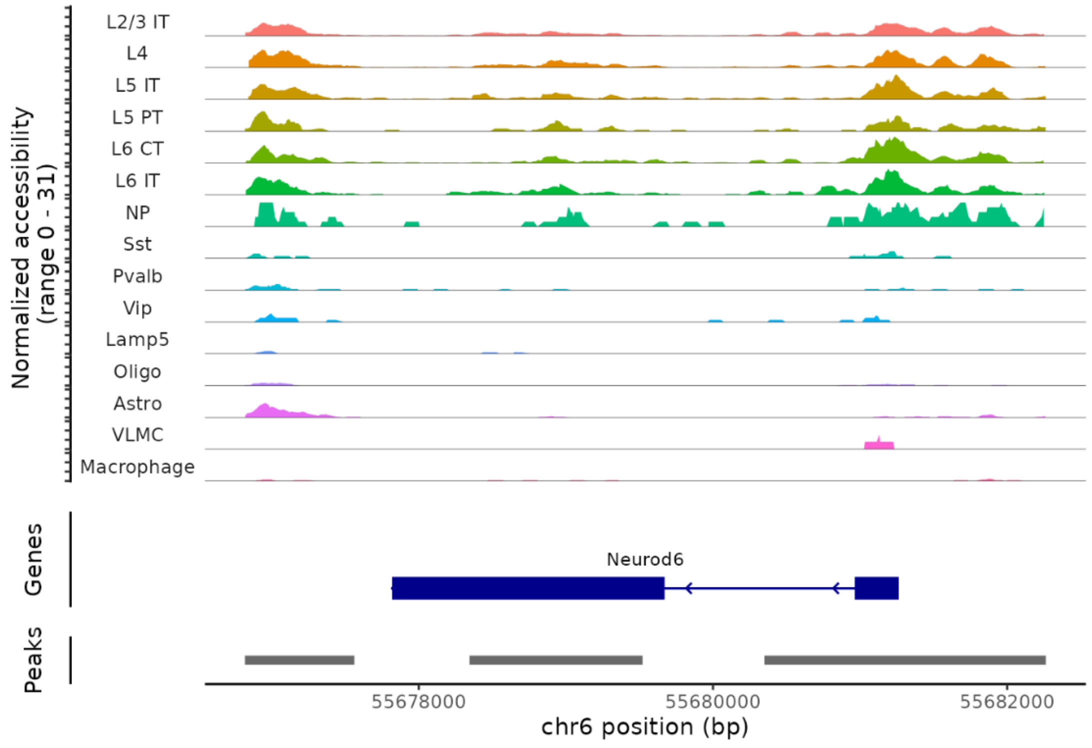


Figure 3.10: Differential analysis of gene Neurod6 peaks across different cell types of the mouse brain ([SSa]).

Another interesting question to answer is how the chromatin accessibility is changing through time. To answer this question, we can combine the pseudotime information from a trajectory inference algorithm and differential analysis. Cicero [Pli+18] helps us to visualize these chromatin accessibility changes (Figure 3.11) and integrates a statistic to indicate whether a peak is changing in pseudotime using a likelihood ratio test from Monocle3 [Tra+14].

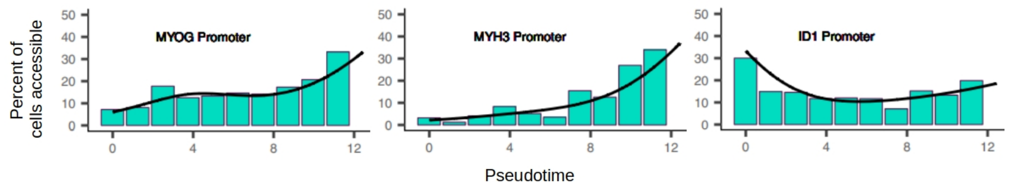
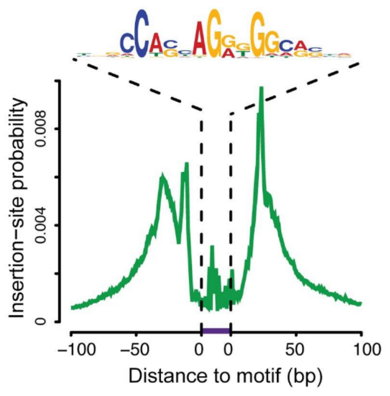


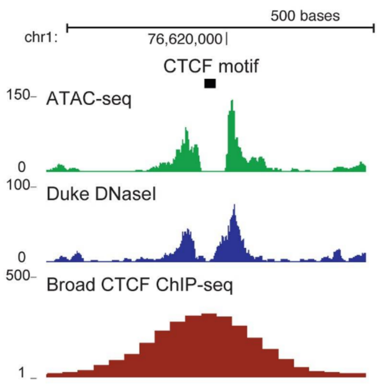
Figure 3.11: Change of promoter accessibility through time in human skeletal muscle myoblasts (HSMM). Black line indicates the pseudotime-dependent average from a smoothed binomial regression ([Pli+18]).

Transcription Factor Footprints

We can study the binding patterns of the transcription factors in the cells to get more information about the regulatory dynamics in a cell. A more comprehensive approach can be made using TOBIAS [Ben+20]. TOBIAS estimates footprint scores for each transcription factor, and this scores can be interpreted as a measure of protein binding. The higher the score, the higher the evidence that a protein was bound at a given position. This helps us to compare and quantify the transcription factor binding activity across cells.



(a)



(b)

Figure 3.12: a Aggregate ATAC-seq footprint of CTCF transcription factor. b CTCF footprints observed through different assays. ([Bue+13])

Motif analysis

- *Motif enrichment*
  We would like to find overrepresented motifs in the set of differentially accessible peaks of each type of cell to identify their potential cell-type-specific regulatory sequences. To do so, we must identify the position and frequency of motifs in each peak region and compare them with a set of background peaks. We will mention a few of the current approaches. [Stu+20] and [Dut+19] perform a hypergeometric test to assess the probability of observing the motif at the given frequency by chance, compared with the background set of peaks. In contrast, [STB20] and [MB10] use a logistic regression approach, where the target variable is the logit of the probability of a peak containing a given motif, and the covariates are the values of the low-dimensional coordinates of the cells when applying PCA to recover 99% of the variance (remember, the original dimension is the number of peaks). [TAD18] generates a motif displacement score [Azo+18] by estimating the ratio of motif occurrence in a small bin (150bp) to a large window (1500bp) from each peak center, and then this score is compared across different conditions with a Z-test.

- *Motif activities*
  A motif activity is the deviation in chromatin accessibility across the set of peaks that contain the motif, i.e., how different the accessibility is for peaks containing the motif compared to the average peak accessibility. The number of fragments that map to a peak is interpreted as its accessibility. Motif activities allow us to identify differentially active motifs across cells, is like gene expression for motifs, and give us a hint to discover potentially transcription factor binding sites. The most popular tool to compute motif activities is ChromVAR [Sch+17], specially designed for scATAC-seq data.

Gene Activities

Gene activities are a measure of chromatin accessibility associated with each gene, an intuitive way to link peaks with genes. There is no specific gene activity score, so there are different ways to estimate it. For example, Signac [Stu+20] sums fragments overlapping the respective gene bodies and promoters, Cicero [Pli+18] sums fragments proximal to a gene transcription starting site (TSS) and its related cis-regulatory elements, and Gene Scoring [Lar+19] computes a distance-weighted sum of fragments overlapping the 50kb upstream and 50kb downstream regions of the gene TSS.
In Figure 3.13, we can observe the activity of three genes in the mouse brain using Signac [Stu+20].

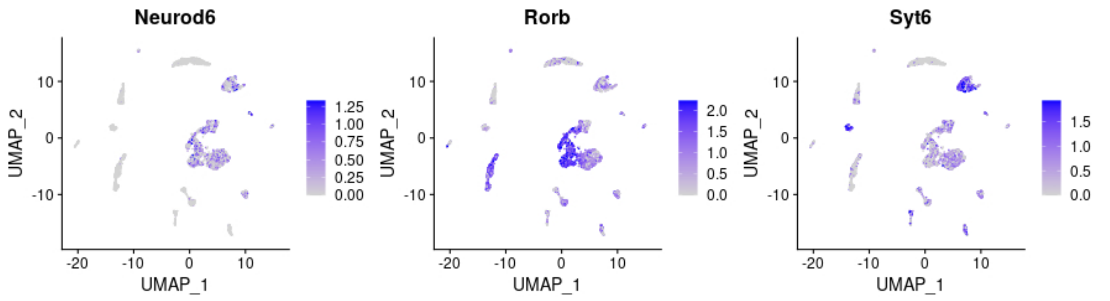


Figure 3.13: Gene activities estimated using scATAC-seq data of cells from the mouse brain ([Stu+20]).

Data Integration
scATAC-seq data give us information about the accessible chromatin regions in a cell, however, these data can be complemented to have a better understanding of the complex regulatory dynamics that happen within it. In this last part, we will talk about scATAC-seq and scRNA-seq integration. The methods reviewed so far, first compute the gene activity scores and build a gene activity matrix, to later integrate it along with the gene expression values. scMVAE [ZC20] fits a multimodal variational autoencoder to learn jointly the features of multilayer profiles, scJoint [Lin+21] transfer labels from scRNA-seq to scATAC-seq gene matrix using a neural network. But, the most popular method is [Stu+19] (used in Signac). This method reduces the dimension of both datasets with Canonical Correlation Analysis (CCA) [But+18], applies L2-normalization to the canonical correlation vectors, identifies mutual nearest neighbors in the shared low-dimensional representation (also known as transfer anchors), estimates scores based on the consistency of anchors, and computes "correction" vectors for each query cell to be jointly integrated.

In this chapter, we mentioned some of the analysis and tools for ATAC-seq data. However, all these are exponentially increasing given to its extremely high potential.

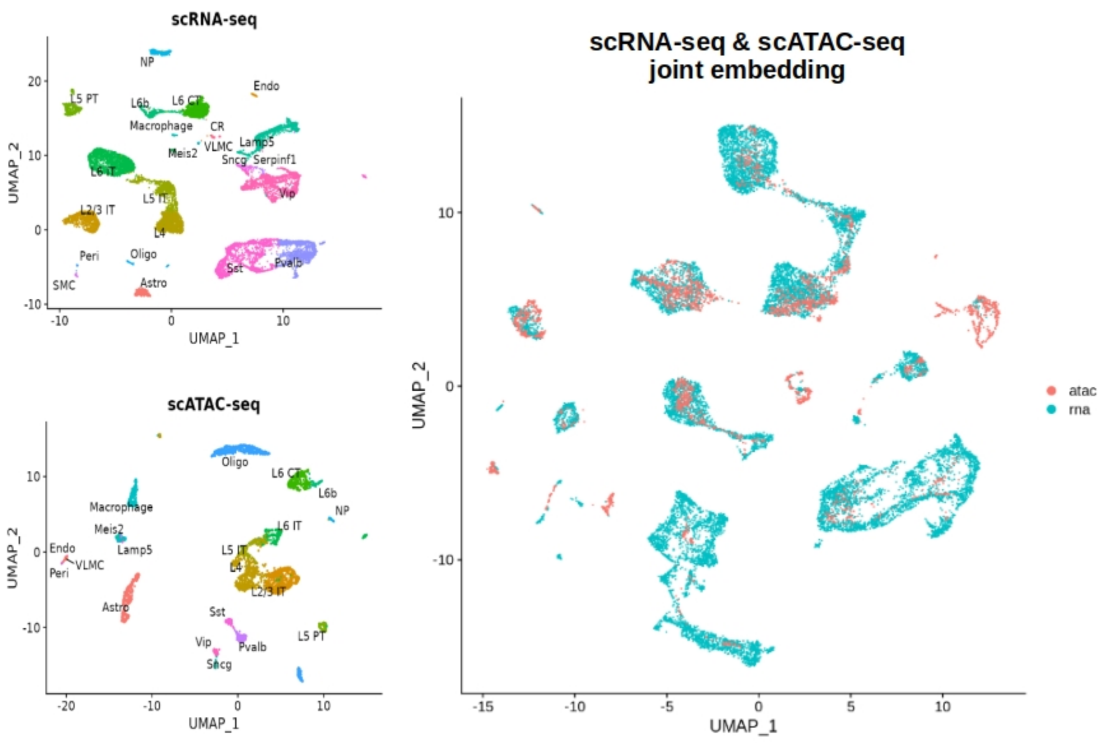


Figure 3.14: scRNA-seq and scATAC-seq joint embedding of cells from the mouse brain using Signac [Stu+20].

Chapter 4

Modeling with neural networks

There are several algorithms that can be used for GRN inference. One approach is to address the problem using neural networks. Neural networks are universal approximators used in different fields. A neural network can belong to one of the following types of algorithms, depending on its learning process:

- **Supervised learning.** In supervised learning, the algorithm compares its prediction with the correct answer to improve itself on each iteration. That means, that for this type of models, we must provide the right answer (*ground truth*). These models are used, for example, in classification or regression tasks.

- **Unsupervised learning.** Finding patterns and structure in the data is the goal of unsupervised learning. The usual problems tackled using this approach are clustering, anomaly detection, association, autoencoder noise removal, and dimensionality reduction. The lack of *ground truth* makes these methods hard to assess, however, there are useful in practice.

- **Semi-supervised learning.** This refers to when we have labeled and unlabeled data. It is useful when both, labeling and extracting relevant features from the data are difficult. An example would be generative adversarial networks (GANs).

- **Reinforcement learning.** Here, an *agent* must learn how to interact with a *dynamic environment* to accomplish an objective. No data are given, the *agent* learns from it choices and tries to maximize its reward.

In this work, we will use a convolutional neural network on time series data to infer GRNs. The problem is set as a regression task, where the time series from putative transcription factors are used to predict the time series of their target genes. Therefore, we are dealing with a *supervised learning task*.

Through this chapter, we will explain the fundamental parts of neural networks and in convolutional neural networks (CNNs), which were originally designed to tackle problems in the field of computer vision. A deeper explanation of the foundations of neural networks can be found in [GBC16] and [Bis14].

4.1 Fundamentals

To understand how a neural network works, it is useful to have the following function $f : \mathbb{R}^D \to \mathbb{R}^L$ in mind:

$$f(\mathbf{x}, \mathbf{w}) := h\left(w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x})\right) = h(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})), \tag{4.1}$$

where $\boldsymbol{\phi} : \mathbb{R}^{D+1} \to \mathbb{R}^M$ is an element-wise function, with $\phi_0 := 1$. $\boldsymbol{\phi}$ and $h : \mathbb{R} \to \mathbb{R}$, are called *basis function* and *activation function*, respectively. The parameters, w_j, with $j \in \{1, ..., M-1\}$ are known as *weights*, and w_0 is known as *bias*.

Definition 4.1 (Neural network)
A neural network is a composition of functions of type 4.1, which basis functions are also functions of type 4.1 [Bis14]. We can write the equation explicitly as:

$$F(\mathbf{x}, \mathbf{W}) := h_K({\mathbf{w}_K}^T h_{K-1}({\mathbf{w}_{K-1}}^T \cdots h_0({\mathbf{w}_0}^T \mathbf{x}))), \tag{4.2}$$

where the subscript k in $\mathbf{w}_k$ and h_k, with $k \in \{0, ..., K\}$, indicates the number of layers.

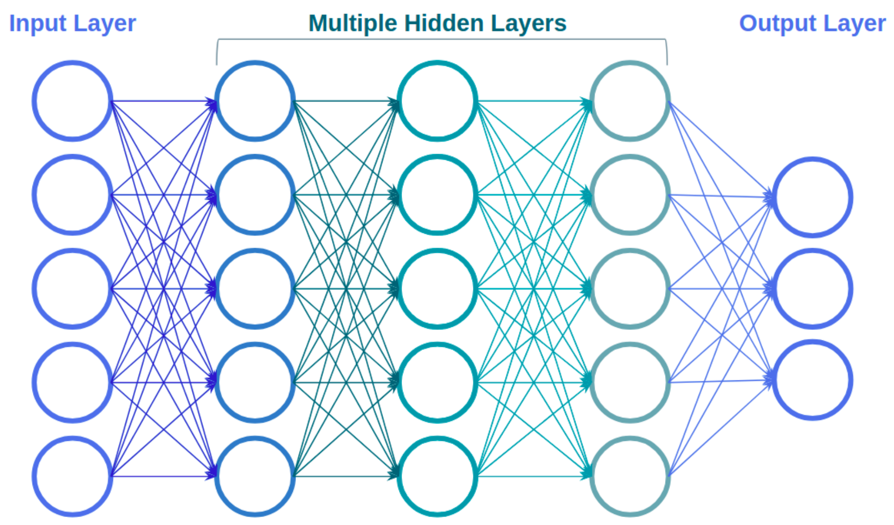


Figure 4.1: Graphical representation of a neural network.

Neural networks are often represented graphically as shown in Figure 4.1. They usually have non-linear activation functions and several layers. The last activation function h_K is chosen according to the function we want to approximate, while the rest of the activation functions are frequently taken from popular non-linear activation functions in the literature. In this work, we use the parametric rectified linear unit (PReLU) [He+15] defined as follows:

Definition 4.2 (PReLU)
The parametric rectified linear unit (PReLU) activation function is defined as:

$$PReLU(y_i) = \max(0, y_i) + a_i \min(0, y_i) = \begin{cases} y_i, \text{ if } y_i \geq 0 \\ a_i y_i, \text{ if } yi \leq 0 \end{cases} \tag{4.3}$$

where a_i is a parameter that the neural network learns during training, often initialized as 0.25.

A graphic representation of PReLU is shown in Figure 4.2.

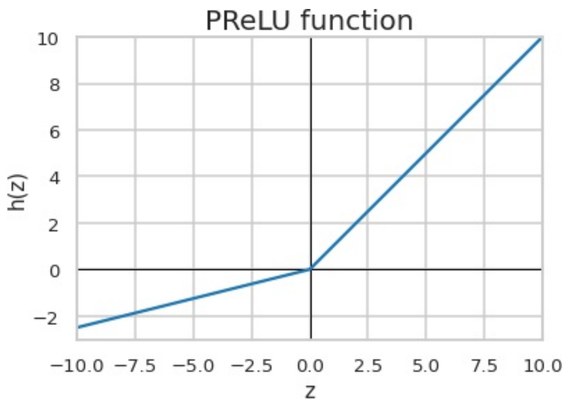


Figure 4.2: PReLU activation function.

4.1.1 Parameter optimization

The process in which a neural network finds the optimal parameters **W** is called *learning*. In this work, we are dealing with a *supervised learning* task, that means that the neural network will approximate **W** by comparing its output $\hat{\mathbf{Y}}$ against the true values **Y**. To do this comparison, we use a *loss function* L, which intuitively provides a measure of how far the given output is from the truth. $L = 0$ would mean that the output of our model is the ground truth, while large values of L would mean the opposite. Therefore, the objective is to find the parameters **W** that minimize the loss function L.

Every time the neural network is fed with an input value **x**, it adjusts the values of **W** using the following *update rule:*

$$\mathbf{W}_{new} = \mathbf{W}_{old} - \alpha \nabla_{\mathbf{W}} L(\mathbf{W}_{old}) \tag{4.4}$$

where the *learning rate* $\alpha \in \mathbb{R}^+$ is a hyperparameter that controls how much we move in the direction of the gradient $-\nabla_{\mathbf{W}} L(\mathbf{W}_{old})$, that indicates the direction in which the loss function decreases the most in every step. The gradient $\nabla_{\mathbf{W}} L(\mathbf{W}_{old})$ is found numerically during training, through an algorithm called *backpropagation*. More detailed explanations can be found in [Bis14].

The update rule 4.4 is applied iterative to the inputs $\mathbf{X}$ to optimize the parameters $\mathbf{W}$. This method is known as *gradient descent* (GD) [Bis14]. However, the choice of the learning rate α in GD is challenging, therefore, different algorithms have been proposed to overcome this limitation. In this work, we use the Adaptative Moment Estimation (Adam) optimizer [KB14], a method based on GD that adapts the learning rate dynamically during training, depending on the mean and variance of the loss function.

4.1.2 Parameter initialization

When we have several layers with activation functions, we can observe the problem of *gradient vanishing* or *gradient exploding*. These problems are encountered when we compute the gradient to optimize the weights as in equation 4.4 using backpropagation. During backpropagation the gradients are computed using the chain rule, which means that we compute the product of several numbers. If these numbers are small, the gradients can be vanishing small preventing the weights to be updated (*vanishing gradients*); and if these numbers are large, the gradients can take on larger values resulting in huge updates to the weights, instability and could prevent the model to learn (*exploding gradients*).

To prevent the problems of *gradient vanishing* and/or *gradient exploding*, we can initialize the weights so they have zero mean and a constant variance through the layers. In this work, we use the *Xavier initialization* [GB10] mechanism, this means that for every layer k:

$$\mathbf{w}_k \sim \mathcal{N}\left(\mu = 0, \sigma^2 = \frac{1}{n_{k-1}}\right) \tag{4.5}$$

where n_{k-1} is the number of neurons in layer $k-1$, and the *bias* should be initialized as zero.

4.1.3 Weight Normalization

To accelerate the training of Deep Neural Networks, Salimans et al. introduced *weight normalization* [SK16]. The idea is to reparametrize each weight vector $\mathbf{w}$ as follows:

$$\mathbf{w} = \frac{g}{||\mathbf{v}||}\mathbf{v} \tag{4.6}$$

where $\mathbf{v}$ is a k-dimensional vector, g is a scalar, and $||\mathbf{v}||$ is the Euclidean norm of $\mathbf{v}$. In such setting, the norms of all the weights are constant $||\mathbf{w}|| = g$. The authors state that weight normalization also improves the conditioning of the gradient and leads to improved convergence of the optimization procedure.

4.1.4 Residual Blocks

Just like parameter initialization, *residual blocks* proposed in [He+16] are another technique that can be performed to avoid *gradient vanishing* and *gradient exploding*. *Residual blocks* make the training of a neural network more stable by adding an identity connection. Then, if a layer is no longer beneficial, the neural network will skip it. Figure 4.3 shows a diagram of a residual block. For us, is important to have the idea of the residual block, however, we would use a modified version of it. Therefore, details of *Batch normalization* are out of our scope, but can be found in [IS15].

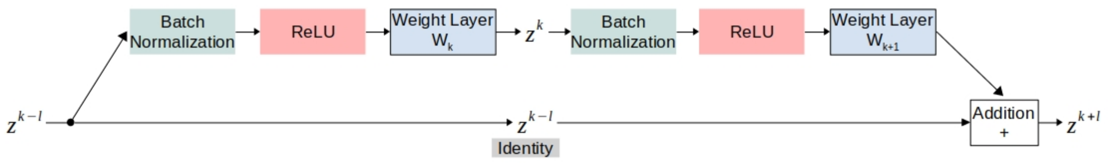


Figure 4.3: Structure of a residual block. [Bec21]

4.1.5 Early stopping

Neural networks are prompt to overfitting while updating the weights using an iterative method (see Equation 4.4). The model improves its fit to the training data with each iteration, and improves its performance on outside data up to a certain point. Past that point, the model loses the ability to generalize. *Early stopping* is a regularization method we can use as a guidance to know how many iterations should be performed in the training, before overfitting. Figure 4.4 provides a graphic explanation. More theory details can be found in [Bis14].

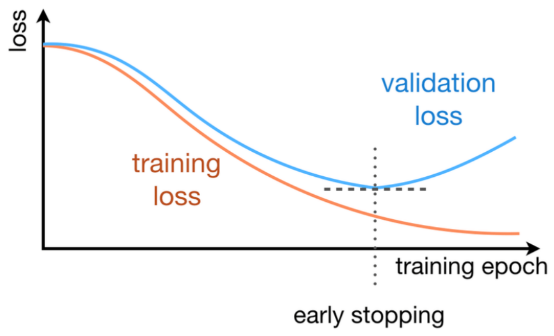


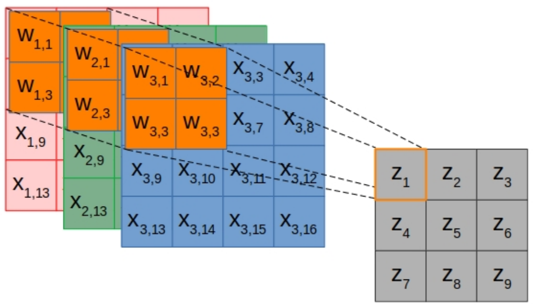
Figure 4.4: Training and validation loss through epochs. Early stopping (dotted line), indicates the point where we should stop the training before overfitting the data. [Vog18]

4.2 Convolutional neural networks

A *convolutional neural network* (CNN) is a type of neural network, typically used to analyze images. CNNs take advantage of the spatial information that images provide, by using *convolutional layers* or *pooling layers* to learn image features, such as vertical lines, horizontal lines, or more complicated patterns.

4.2.1 Convolutional layer

A *convolutional layer* is a special type of layer used by CNNs. In contrast of a linear layer defined in Section 4.1, a *convolutional layer* makes use of *filters* or *kernels* that slide along the input and extract a *feature map* [Zha+88]. These *kernels* have a shared-weight architecture, meaning that their weights are used by multiple elements of the input. The process of how a convolutional layer works is shown in Figure 4.5.



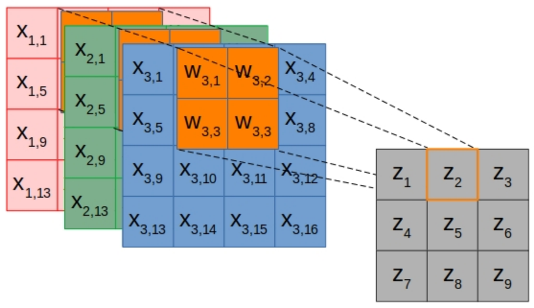


Figure 4.5: Convolutional layer. The input image is represented with its three channels: red, green and blue; the convolution kernel of size 2×2 is shown in orange; the feature map is represented in gray. [Bec21]

The *feature map* is the output of sliding the *kernel* through an image, and each of its entrances can be computed as follows:

$$z_i = \mathbf{w}^T \mathbf{x}_i + b \tag{4.7}$$

where $\mathbf{w}$ are the kernel weights, $\mathbf{x}_i$ is the i-th chunk of the image, and b represents the bias. The number of pixels that the kernel shifts in each step is called the *stride*. Both, the kernel size and the stride are hyperparameters. In Figure 4.5 the kernel size is $2 \times 2 \times 3$ and the stride is 1. We can also observe that the size of the feature map is smaller than the input image. If we want to preserve the size between the input and the feature map (*same convolution*), we need to add zeros in the edge of the input (*zero padding*) as shown in Figure 4.6.

It is important to note that each kernel maps a multi-channel input into a single-channel feature map. That means that each channel in the feature map corresponds to one kernel.

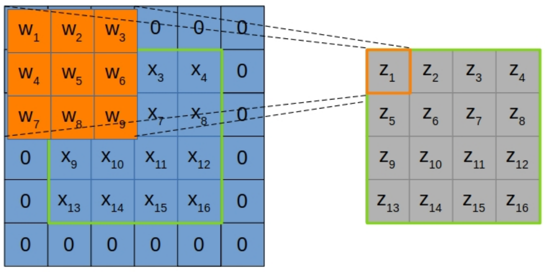

w_1 w_2 w_3 0 0 0
w_4 w_5 w_6 x_3 x_4 0
w_7 w_8 w_9 x_7 x_8 0
0 x_9 x_{10} x_{11} x_{12} 0
0 x_{13} x_{14} x_{15} x_{16} 0
0 0 0 0 0 0
z_1 z_2 z_3 z_4
z_5 z_6 z_7 z_8
z_9 z_{10} z_{11} z_{12}
z_{13} z_{14} z_{15} z_{16}

Figure 4.6: Padding. A channel of the input image with zero padding is shown in blue. The feature map, represented in gray, has the same size of the original image as indicated by the green outline. [Bec21]

Also, it is worth mentioning that CNNs usually have a non-linear activation function after the convolution layers that helps them learn non-linear relations.

4.2.2 Temporal convolutional neural networks

Temporal convolutional neural networks (TCNNs) are CNNs for time series prediction. Although, most of CNN applications are in the field of computer vision, they have also been used to deal with time series data. [BKK18a] proposed TCNNs and showed that they overcome several limitations of recurrent neural networks (RNNs), like gradient exploding or vanishing, and lack of memory retention. Additionally, CNNs allow for parallel computation of outputs. TCNN has a one-dimensional vector as input and as output and uses a modified convolution layer called *causal convolution*, that ensures that every element of the output sequence depends only on observed elements of the input sequence (past and present). Figure 4.7 shows the difference between causal and standard convolutions.

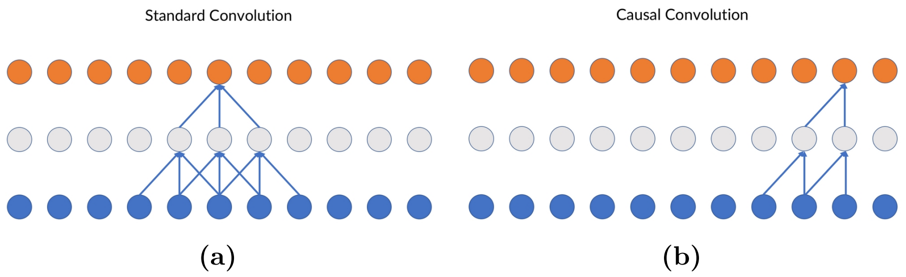


Figure 4.7: Comparison between a standard convolution (a) and a causal convolution (b) [Kla19]

4.2.3 Dilated convolutions

In convolutional layers, the region of the input that produces an output feature (on any layer) is known as *receptive field*. The idea of a *dilated convolution* [YK15] is to capture a large context of the input without increasing the number of parameters by expanding the kernel with gaps. This means, that the network will increase the receptive field of its features, without increasing the number of parameters as shown in Figure 4.8. We can also combine the concept of dilations with causal convolutions into a *causal dilated convolution*, as the one illustrated in Figure 4.9.

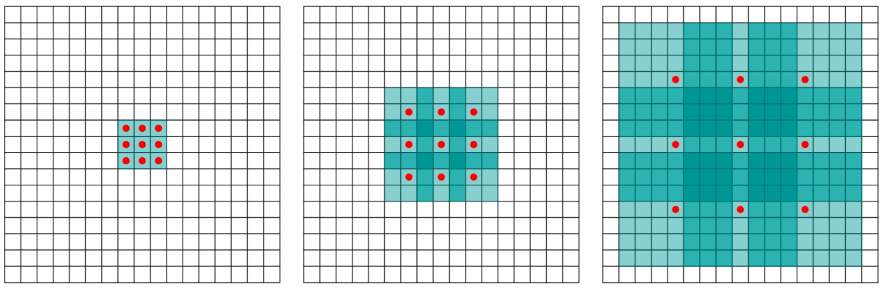


Figure 4.8: Dilated convolution. The receptive field expands without loss of resolution (without parameter increasing).[YK15]

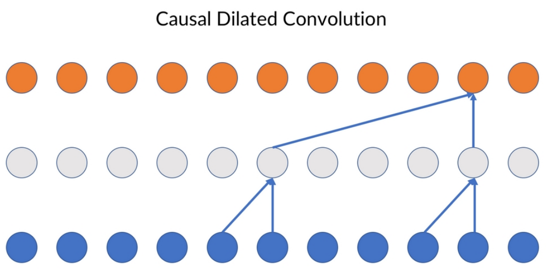


Figure 4.9: Causal dilated convolution. The combination of a causal convolution and dilations [Kla19].

4.2.4 Depthwise separable convolutions

A convolution layer learns spatial correlations within every channel and cross-channel correlations simultaneously. The idea of a *depthwise separable convolution* (introduced by [Cho16]) is to learn both types of correlations independently. First, the spatial information is captured by a regular convolution in each channel individually, and then the cross-channel correlations are captured by a *pointwise convolution*, as it can be seen in Figure 4.10.

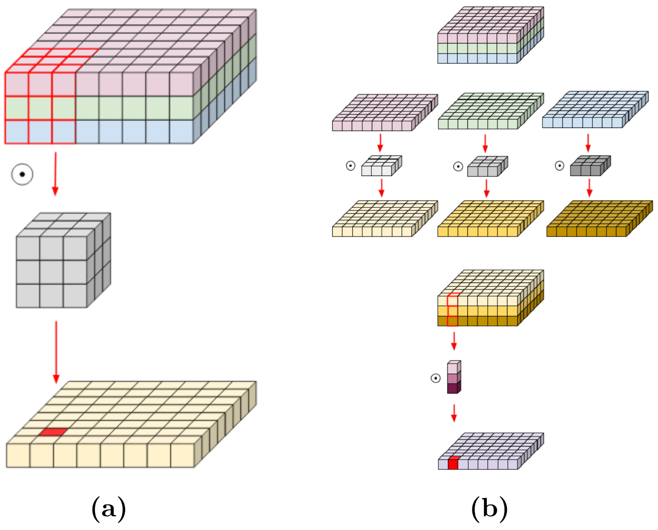


Figure 4.10: Comparison between a standard convolution (a) and a depthwise separable convolution (b) [Ben].

4.2.5 Attention mechanisms

The idea behind an *attention mechanism* is to enable the neural network to focus on the most relevant parts of the input to compute the output. Attention provides scores that serve as an interpretability method. By interpreting attention, we can figure out exactly which parts of the input are relevant for the prediction. It has been proven to be effective for different tasks, for example in [Vas+17], or [Yin+16].

In the literature, we can distinguish between *soft attention*, where the final scores are in the interval $[0, 1]$; and *hard attention*, where these are either 0 or 1. *Hard attention* may be easier to interpret, but has the strong limitation of being non-differentiable, and thus, not optimizable during backpropagation [She+18]. For this reason, many authors prefer to use *soft attention*. To estimate *soft attention*, the *softmax* function σ is applied to the trainable attention vector.

Chapter 5

The gene thicket: a new method to infer gene regulatory networks

The goal of this thesis is to explore the idea of having a new GRN inference method based on neural networks, that takes into account time (or pseudotime ordering), RNA-seq data and ATAC-seq data. From this idea, **the gene thicket** was created. In this chapter, we will describe step by step how our model works. We will start with scATAC-seq and scRNA-seq data preprocessing, then we will detail gene thicket's architecture, and finally we will explain the metrics used to evaluate the performance of the model. An overview of the entire workflow can be seen in Figure 5.1.

5.1 ATAC-seq data preprocessing

The causality inference from gene expression profiles can be problematic due to the level of noise in the data, and also spurious links can be inferred based only on correlations. We can overcome these limitations by adding different data modalities, such as scATAC-seq data. Therefore, as part of the gene thicket's pipeline, we integrated the CellOracle's ATAC pipeline [KHM20]. The idea is to analyse scATAC-seq data to have a list of transcription factors and putative target genes, that serve as a prior information. We can think of it as an initial GRN structure that we will prune, later on, using scRNA-seq data and the gene thicket.

For most of the recent scATAC-seq databases, the pre-analysis mentioned in Section 3.2 and the definition of regions from Section 3.3 are already performed. Since there is no need of the doing the first steps, we will continue as follows:

1. filter cells, filter peaks, binarization, normalization and dimensional reduction steps described in Section 3.3.

2. run Cicero [Pli+18] to estimate co-accessibility scores between peaks, as it is described in Section 3.4. The output will be a list of peak pairs within 500kb apart and their coaccessibility scores.

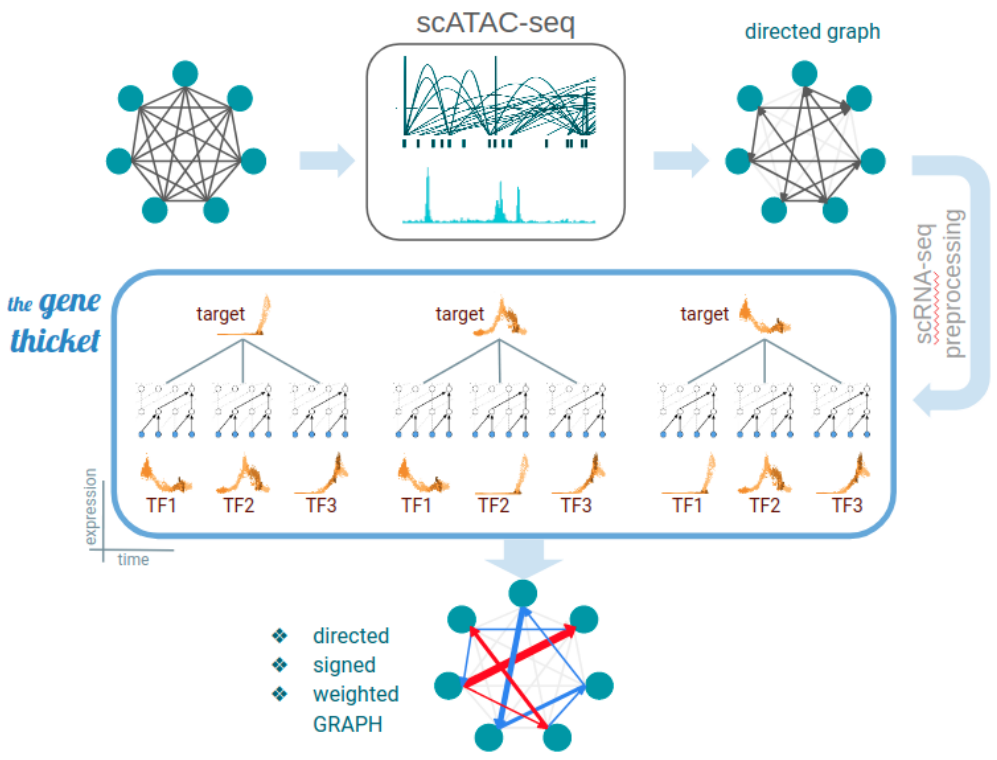


Figure 5.1: The gene thicket's workflow overview. scATAC-seq data is used as a prior to obtain directed regulatory gene connections. Then, pseudotime ordered expression profiles of transcription factors are used to predict the pseudotime ordered expression profile of its putative target genes. The output is a refined network with signed and weighted connections.

3. annotate peaks from Cicero's output using Bedtools [QH10] intersection and filter all regions that have a coaccessibility score < 0.8 with a gene's transcription starting site (TSS), to avoid false positives.

4. scan the remaining peaks to find motifs using GimmeMotifs [BVH18]. For each peak, we will select all the corresponding motifs and create a list of their associated transcription factors (also from GimmeMotifs).

5. create a list of unique genes with corresponding transcription factors from the previous output.

This final list of target genes and their corresponding putative transcription factors will be the input of the gene thicket.

In case that no scATAC-seq data is available, we can obtain the aforementioned list from the ranking cisTarget [Imr+15] databases used in SCENIC [Aib+17]. Empirically, we selected the first 1500 target genes associated with each transcription factor, and then filtered the database to keep only the genes of our final expression data from the RNA-seq data preprocessing step described in the following Section 5.2.

5.2 RNA-seq data preprocessing

The inferred causal relationships depend on the gene expression patterns through time. For that reason, the preprocessing of gene expression profiles from scRNA-seq data is crucial for our model. We will follow the preprocessing steps described in Section 2.2 using scVelo [Ber+20]. First, we will select genes filtering them by a minimum number of counts and high variability (dispersion), then each cell will be normalized by its total size, next the count data matrix will be log transformed, later we will compute principal component analysis (PCA) as dimensional reduction method, and finally, we will impute the data using the k-nearest neighbor (KNN) graph.

5.3 The gene thicket

The motivation to build the gene thicket is to have a method that considers non-linear relationships between target genes and transcription factors, such as GENIE3 [IWG+10] or GRNBoost2 [Moe+19], but that also considers time series data. This is because the change in expression of transcription factors has an impact in the change of expression of target genes, which is observed when the expression profiles are sorted through time. However, algorithms that do not require pseudotime-ordered cells have shown a better performance according to [Pra+20]. Another advantage of having a time series approach is that we can make forecasts about the expression values of target genes using the previous expression values of transcription factors. Then, we could compute displacements and simulate trajectories, in a similar way that CellOracle [KHM20] does.

The idea of using deep learning comes from the fact that neural networks are universal non-linear approximators. We selected temporal convolutional neural networks with attention, because they are scalable and can be interpreted through an attention mechanism. In this work, we created a model to infer GRNs based in this idea, including epigenetic information (ATAC-seq data) as prior.

5.3.1 Architecture

The presented architecture was inspired by the Temporal Causal Discovery Framework (TCDF) [NBS19], used to infer are causal associations in market stocks, and by the Temporal Convolutional Network (TCN) [BKK18b], used to model univariate time series.

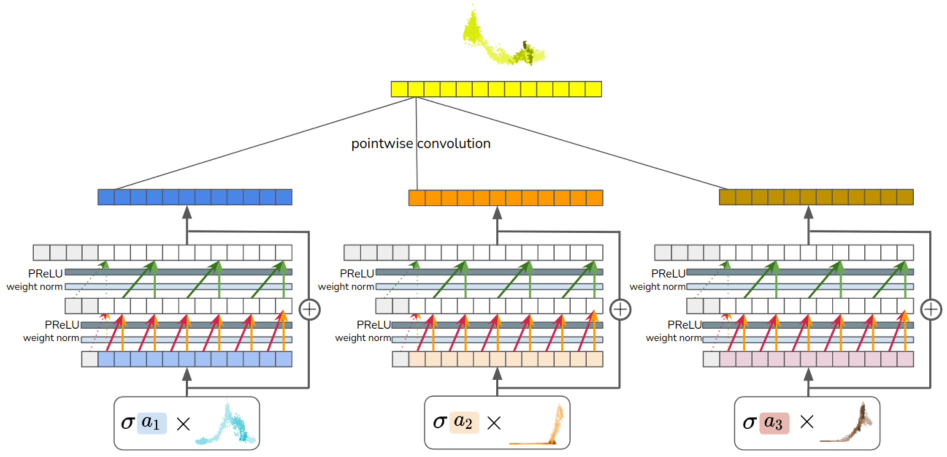


Figure 5.2: Overview of a network that predicts a target gene expression through time. Each transcription factor time series is multiplied by an attention constant σa_i, and then used as an input for a residual block denoted by $\bigoplus$. The residual block consists of convolutional layers with dilations, which have PReLU activation functions and weight normalizations in between. At the end a pointwise convolution is used to combine separable outputs into a single future prediction for the target gene.

The gene thicket consists of N identical and independent neural networks. Each network is built to predict the time step t from a target gene time series, using present (time step t) and past (some time steps $< t$) values of its putative transcription factor time series. We obtained this putative transcription factor list for every gene, from the scATAC-seq data preprocessing 5.1, as it is shown in Figure 5.2. For this, we use one-dimensional temporal separable convolutions with causal padding (4.2.2, 4.2.4). The separable convolutions are brought together by a pointwise convolution, so at the end we can have a single prediction. In this way, we can treat each of the input time series individually, which helps the model's interpretability. We used PReLU (4.2) activation functions, weight normalization (4.1.3) and initialized the weights using *Xavier initialization* (4.1.2).

To increase the network's receptive field, we included dilations (4.2.3) with an exponentially increasing dilation factor. In this way, the receptive field augments exponentially, while the parameters grow linearly, which is particularly useful to deal with delays between cause and effect. The gene thicket also makes use of residual connections, in the form of residual blocks .

To reduce the computation time and make the gene thicket more scalable, we added an early stopping mechanism 4.1.5. We train the network with the first 80% of the time steps and use the remaining 20% as a validation set. If the performance of the model in the validation set does not improve for a determined number of iterations (in our case, 30), then it stops training.

5.3.2 Interpretability

An attention mechanism (4.2.5) was implemented to find out which transcription factor time series are the most relevant for the prediction of their presumptive target gene. For each network $\mathcal{N}_j$, the attention mechanism is a trainable vector $\mathbf{a}_j$ of size $1 \times M$, that is multiplied element-wise with the M time series from the input. That means, that the attention score $a_{i,j}$ is multiplied with the input time series $\mathbf{X}_i$ in the network $\mathcal{N}_j$. Then, we can interpret the attention score $a_{i,j}$ as the importance of time series $\mathbf{X}_i$ when predicting target time series X_j.

At the beginning of the training, the attention scores have an initial value of 1. During backpropagation, the scores are updated, and take values in $\mathbb{R}$. With these scores we compute *soft attention*, as mentioned in 4.2.5. To select relevant attention scores, we apply a *HardSoftmax* function that will set as zero all the scores below a certain threshold τ_j:

$$\text{HardSoftmax}(a) = \begin{cases} \sigma(a) & \text{if } a \geq \tau_j \\ 0 & \text{if } a < \tau_j \end{cases} \tag{5.1}$$

Let $h_{i,j}$ be the value that we obtain when applying the *HardSoftmax* function to the attention score $a_{i,j}$. Then, if $h_{i,j} > 0$, we will consider the time series $\mathbf{X}_i$ as a potential cause of time series $\mathbf{X}_j$.

To determine the threshold value τ_j, we order the attention scores in $\mathbf{a}_j$, from high to low, and search for the largest gap between two adjacent of them. τ_j will be equal to the attention score on the left side of the gap. However, we make a few additional considerations. Let G be set of gaps $\{g_0, g_1, ..., g_{N-1}\}$, then:

1. $\tau_j \geq 1$. Since the attention scores are initialized as 1 and we are using a $softmax$ function, an increasing of this initial value will mean that the network is focusing on the corresponding time series.

2. τ_j should not be the highest attention score. This is to ensure that potential causes of target $\mathbf{X}_j$ are not being truncated, just because their attention scores are weaker than the top score. Consequently, we will always consider at least two time series as potential causes.

Figure 5.3 illustrates the selection of τ_j.

5.3.3 Sign assignment: Activation or Repression

One essential characteristic of a GRN is that their edges have a sign representing an activation or repression relationship between two genes. Since the attention scores are inferred through a pipeline, where they are the input of a sigmoid function, their sign does not reflect the type of the presumptive relationship between the genes. Therefore,

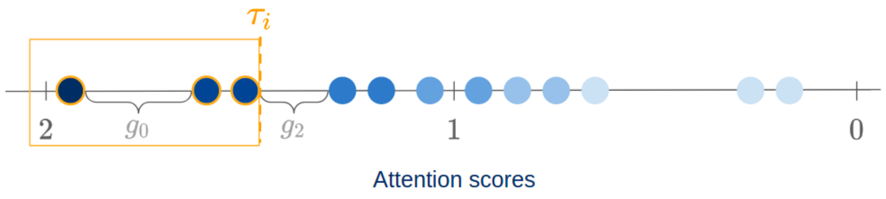


Figure 5.3: The threshold value τ_j is the attention score on the left side of the largest gap g_k, with $k \neq 0$.

as a first simple approach, we decided to estimate the signs using Pearson correlation coefficient (PCC) as it is done by SCENIC [Aib+17].

The Pearson correlation coefficient is a normalised measure of covariance between two variables and it is defined as:

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \tag{5.2}$$

where X and Y are two random variables, cov is the covariance, σ_X is the standard deviation of X, and σ_Y is the standard deviation of Y.

One of the biggest pitfalls of PCC is the fact that only captures linear correlation. We are aware of this limitation and intend to develop a new method to estimate signs in the future.

5.3.4 Causal validation

According to [Eic12], a causal relationship should have the following characteristics:

- *temporal precedence:* the cause precedes the effect.
- *physical influence:* changes in the cause impact the effect.

Now that we have a list of potential causes for each target, we would like to validate that these two causality requirements are fulfilled. Since we are using a *temporal convolutional neural network*, the *temporal precedence* is satisfied. However, we still need to check for *physical influence*.

To make sure that *physical influence* between a potential cause and the effect exist, we will use *Permutation Importance* (PI) [Bre01]. The idea of PI is to permute the values of the time series corresponding to the potential cause, and measure if the loss function of the network increased significantly. If this is the case, that means that the potential cause is likely a true cause of the predicted variable. This comes from the fact that by permuting the values of the potential cause, we are breaking the chronological relationship between cause and effect. In our implementation, we will take this idea to validate the cause $\mathbf{X}_i$ for target $\mathbf{X}_j$ by doing the following:

1. First, train network $\mathcal{N}_j$ and measure the loss at the first and last iterations. We will call them $\mathcal{L}_{init}$ and $\mathcal{L}_{last}$, respectively.
2. Then, permute the values of $\mathbf{X}_i$ in the input dataset.
3. Later, evaluate the trained network $\mathcal{N}_j$ using the permuted dataset and measure the loss $\mathcal{L}_i$.

Let $\Delta_G := \mathcal{L}_{last} - \mathcal{L}_{init}$ and $\Delta_i := \mathcal{L}_{last} - \mathcal{L}_i$ be the loss improvements using the original inputs and the input with the permuted variable $\mathbf{X}_i$, respectively. Then, for a given value $s \in (0, 1]$, if $\Delta_i \leq (\Delta_G \times s)$, $\mathbf{X}_i$ is considered a true cause of $\mathbf{X}_j$. Because that would mean that the improvement on the loss using permuted values of $\mathbf{X}_j$ is relatively small compared to the improvement we obtained by using the original chronologically ordered input. The parameter s is a hyperparameter that we set as 0.8, as in [NBS19].

Chapter 6

Results

In this chapter, we will evaluate the performance of our model (the *gene thicket*) and compare it with the performance of different existing GRN approaches. Although it may seem that this is a straightforward task, to evaluate the quality of any reconstructed gene regulatory network is extremely challenging, because a ground truth does not exist. Moreover, there are diverse factors that have an impact on experimental data and could affect the interactions in a system. These factors can be primary cell types, environmental conditions, technology platforms and cell lines [CM18]. To overcome these limitations, in the first two sections, we will evaluate the model's performance using synthetic and curated data from [Pra+20], respectively. In the last section, we will investigate the model's behavior in real biological data that describe the Pancreatic endocrinogenesis [BP+19].

We compared our model against the tree-based GRN inference algorithms Genie3 [IWG+10] and GRNBoost2 [Moe+19]; an existing algorithm to infer GRNs from time series data called SINCERITIES [PG+18] and the TCDF algorithm [NBS19]. The metrics we use to compare our method are the *Area Under the Precision Recall Curve* (AUPRC) and the *Area Under the Receiver Operating Characteristic Curve* (AUROC), described on Appendix A. To compute AUPRC and AUROC, we think of the GRN inference problem as a binary classification task: if *geneA* regulates *geneB*, then (*geneA*, *geneB*) should be of class 1 (positive class); and if such relationship does not exist, (*geneA*, *geneB*) should be of class 0 (negative class).

6.1 Curated Datasets

We evaluated the model in the four existing types of curated datasets: Mammalian Cortical Area Development (mCAD) [GG10], Ventral Spinal Cord Development (VSC)[Lov+14], Hematopoietic Stem Cell Differentiation (HSC) [Kru+11], and Gonadal Sex Determination (GSD) [Río+15]. More information about the datasets and a summary of the ground truth networks for each one of the models can be found on Appendix B.1.

Figure 6.1 shows the performance of our model against other GRN inference algorithms on the curated data. For mCAD, the *gene thicket* has the best performance, while for the rest of the curated dataset types, its performance can be ranked in the bottom two.

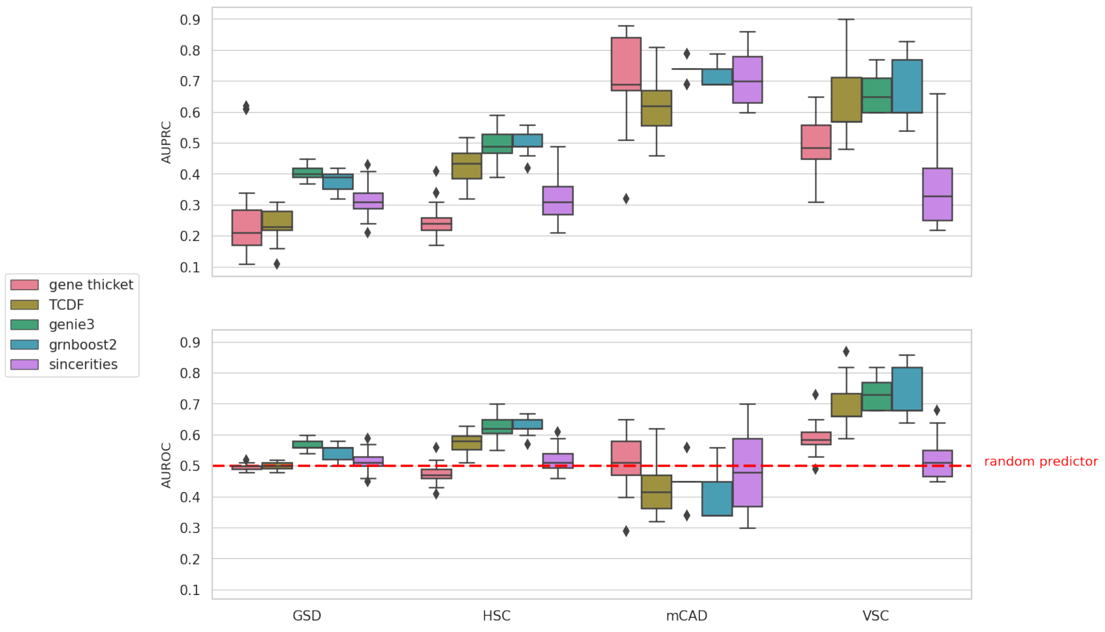


Figure 6.1: AUPRC and AUROC of GRN Inference methods evaluated on the curated datasets. The *gene thicket* has the best performance on the mCAD dataset, while for the rest, its performance lies on the bottom two.

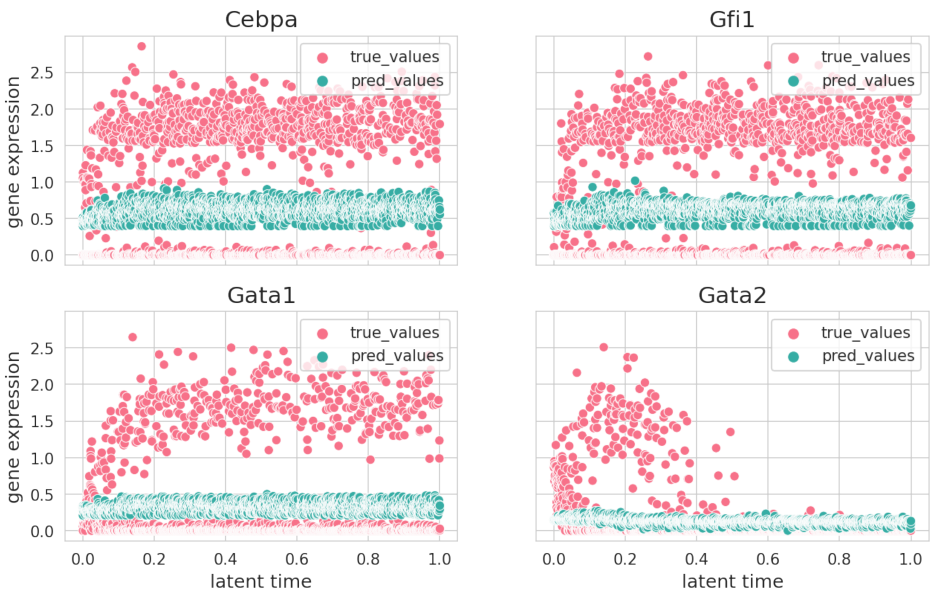


Figure 6.2: The *gene thicket* struggles to predict the expression of Cebpa, Gfi1, Gata1 and Gata2 in version eight from HSC dataset with 2000 cells and drop out rate of 70.

We investigated more about this fact, and discovered that this is due to the different trajectories the cells can follow. Depending on the faith a cell has, the expression of its genes can be either low or high at the same moment of time. This ambiguity is not considered by the model so far and impacts its performance. In Figure 6.2, we can observe an example of how the *gene thicket* struggles to make a good gene expression prediction, because the loss function of the *gene thicket* is the mean square error (MSE), which is not suitable for ambiguities.

We can also note that the performance of the models has the tendency to get worse when the number of genes grows. In general, the AUPRC values of the models have the following order, when sorting them from the largest to lowest: mCAD (5 genes), VSC (8 genes), HSC (11 genes) and GSD (19 genes). This comes from the fact that when the number of genes increases, the complexity of the relationships grows and also the possibility of the models to detect indirect regulatory links.

The curated datasets also allow us to investigate the behavior of the different GRN inference algorithms when having distinct drop out rates, as it can be seen in Figure 6.3. For each type of data, there exist versions with drop out rates of 0, 50 and 70. Unsurprisingly, the performances of the algorithms tend to decrease as the number of drop outs increases.

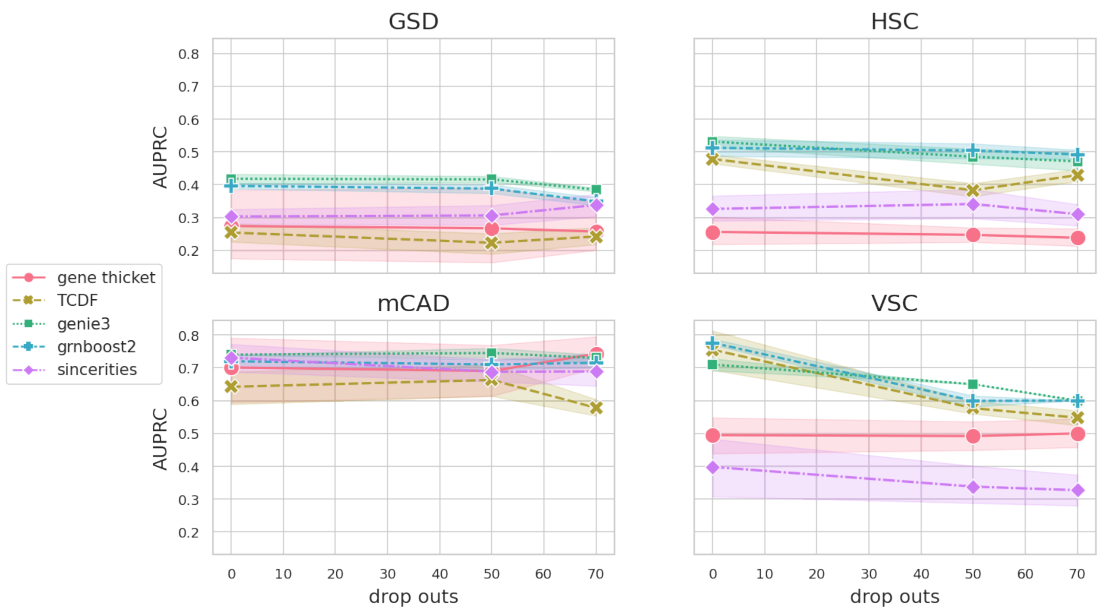


Figure 6.3: Effect of dropouts on the performance of GRN inference methods.

6.2 Synthetic Datasets

We evaluated the performance of the *gene thicket* on the six existing types of synthetic datasets: bifurcating (dyn-BF), bifurcating converging (dyn-BFC), cycle (dyn-CY), linear (dyn-LI), linear long (dyn-LL), and trifurcating (dyn-TF). The datasets description and a summary of the ground truth networks for each one of the types can be found on Appendix B.2.

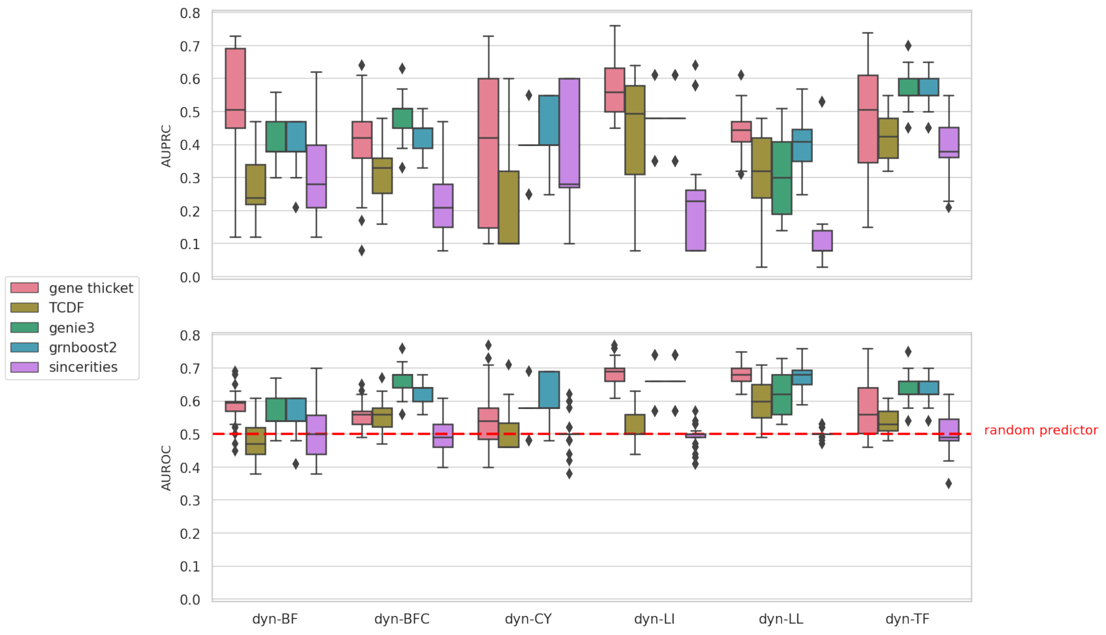


Figure 6.4: AUPRC and AUROC of GRN Inference methods evaluted on the synthetic datasets.

Figure 6.4 shows the AUPRC and AUROC scores of our model against other GRN inference methods on the distinct types of synthetic data. AUROC allows us to compare all the methods against a random predictor. However, in the context of GRN inference, AUPRC is more suitable than AUROC because we have unbalanced classes (see Appendix A). The *gene thicket* shows a good performance with respect to the AUPRC scores on the synthetic data, which is the opposite behavior that we observed in the curated data.

When we analyse the gene expression predictions that the *gene thicket* makes for genes in the synthetic data, we observe that they tend to follow the gene expression patterns as in Figure 6.5. This happens when the patterns do not have ambiguity as in Figure 6.2.

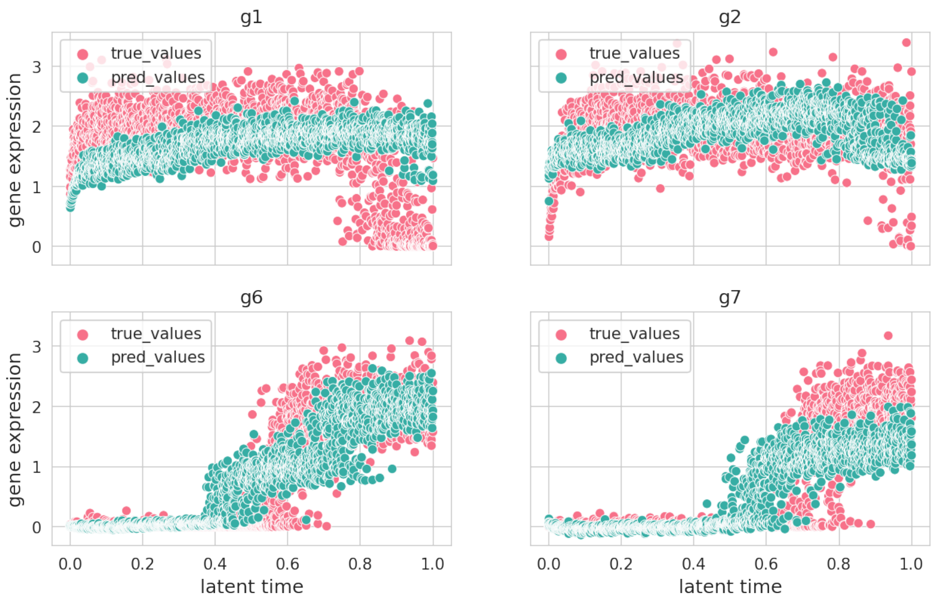


Figure 6.5: The *gene thicket* predicts the expression of genes of version two from the dyn-LI dataset with 2000 cells.

For each type of synthetic data, there exist distinct versions with different number of cells. This gives us the chance to determine how the performance of the GRN inference methods changes with respect to the number of cells, as it is shown in Figure 6.6. The tree ensemble methods are the most consistent. However, there is a noticeable impact on the behavior for the temporal-dependent approaches when the number of cells increases. We would expect that the more information the model has, the more accurate its predictions would be, nevertheless, this is not always the case.

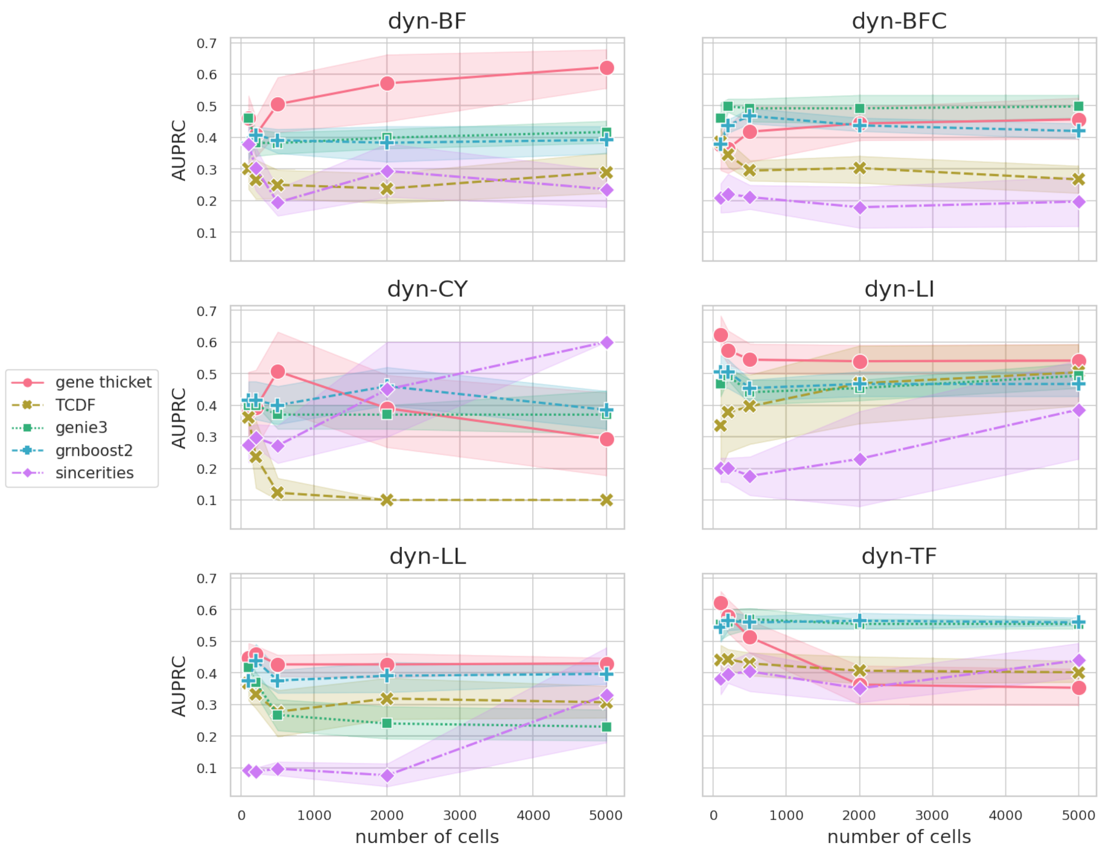


Figure 6.6: Effect of number of cells on the performance of GRN inference algorithms.

6.3 Scalability

An important characteristic of a GRN inference method is its *running time*, or the time that the algorithm takes to infer the gene regulatory network. We compared the running time of the *gene thicket* against the running time of other inference methods. Figure 6.7 shows the running time of the different GRN inference methods on the curated and synthetic datasets. The time that the *gene thicket* takes to infer the gene regulatory networks is better than the running time of TCDF and GENIE3 in all cases. However, GRNBoost2 is the fastest method among all because of its boosting nature.

We take advantage of how the synthetic datasets were generated to assess the scalability of the GRN inference methods. We computed the running time of all GRN inference methods and observed their behavior when the number of cells increases. In Figure 6.8, we can note that the running time of Genie3 grows exponentially with respect to the number of cells, for GRNBoost2 the running time increment is linear, while for the deep

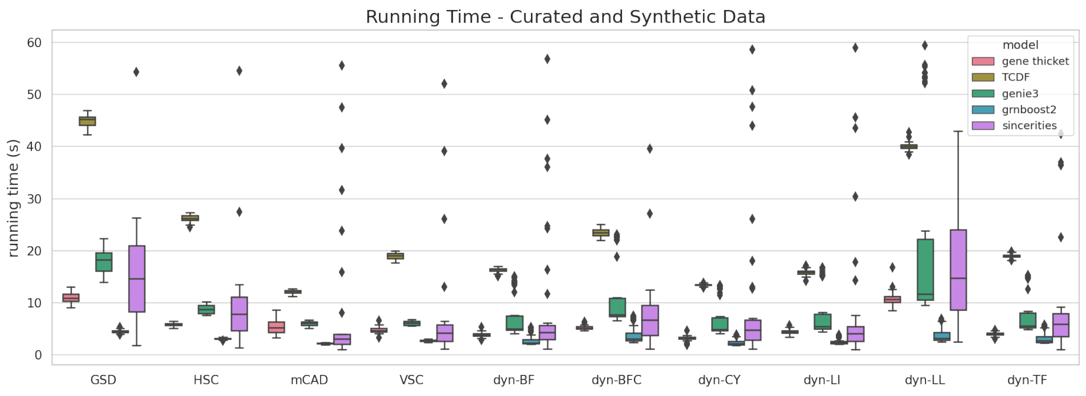


Figure 6.7: Running time of GRN inference methods across all datasets.

learning temporal approaches we observe more stability. TCDF is the slowest method and GRNBoost2 the fastest, followed by our method the *gene thicket*.

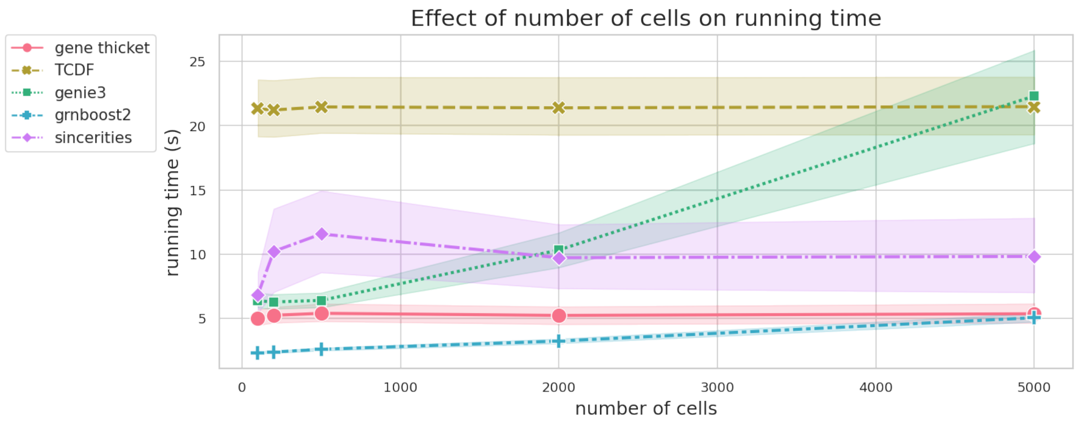


Figure 6.8: Running time of deep learning methods is similar across the number of cells. However, for tree ensemble methods, the number of cells has a great impact in the running time.

Figure 6.9 shows how the running time of all GRN inference methods changes with respect to the number of cells, in each synthetic dataset type. It is evident that the dyn-LL requires more running time from all the algorithms than the other datasets. We believe that this is because the number of genes of the dyn-LL dataset is 18, while for the rest of the datasets the number of genes lies between 6 and 10. This means that for each cell we add to the dyn-LL dataset, we are adding 18 values, which will be equivalent to add 3 cells to the dyn-CY dataset with only 6 genes.

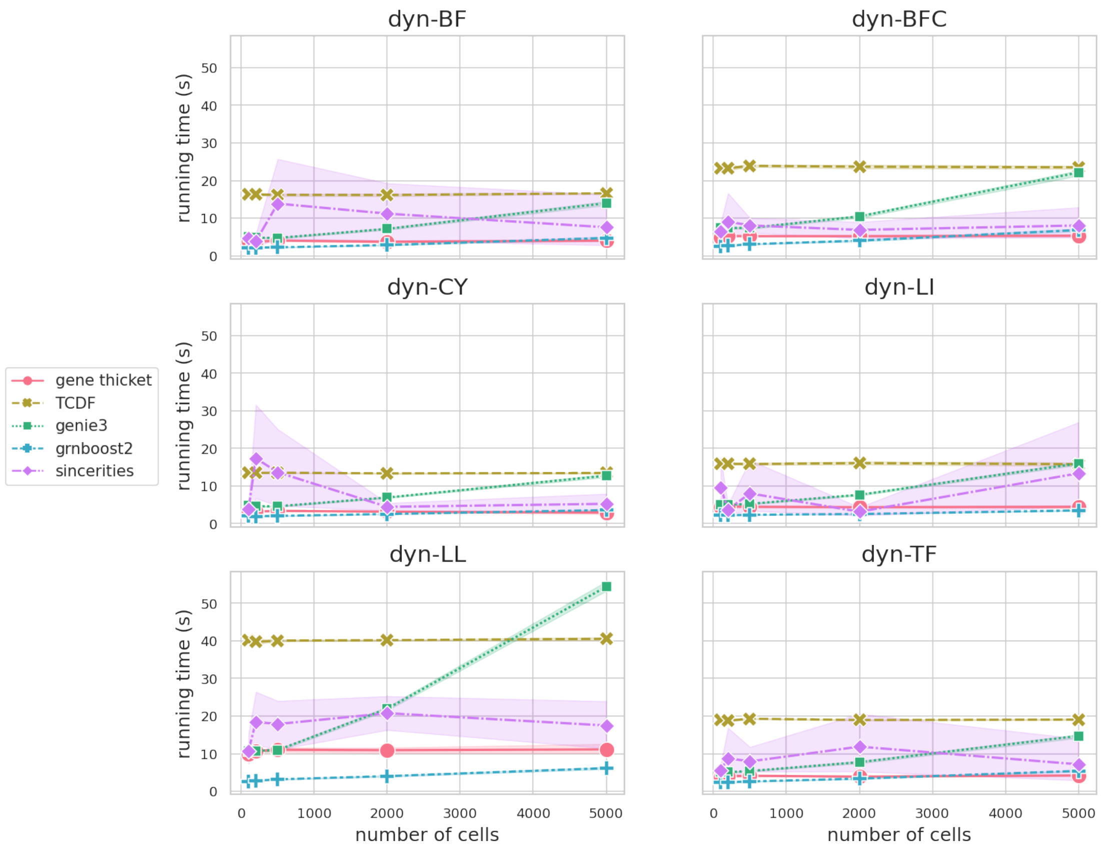


Figure 6.9: The running time of *gene thicket* is consistently low through datasets and number of cells. SINCERITIES shows significantly more running time variance, independent of the number of cells. The number of cells has greater impact on the running time for tree ensemble methods.

6.4 Pancreatic Endocrinogenesis

We applied the *gene thicket* to biological data to decipher the underlying regulatory mechanisms of the system. Since the inspiration to build the *gene thicket* came from scVelo [Ber+20], we wanted to infer a GRN and compute displacements of cells to compare them with scVelo's velocities. Therefore, we applied the *gene thicket* to the Pancreatic Endocrinogenesis dataset [BP+19], one of the main datasets analyzed using scVelo.

This particular dataset focused on the expression of neurogenin 3 (Ngn3) and its change over time. The authors identified genes with a behavior similar to Ngn3, associated with cell-fate allocation towards the distinct endocrine cell types: glucacon-producing α-cells, insulin-producing β-cells, somatostatin-producing δ-cells and ghrelin-producing ϵ-cells.

The dataset has 3696 cells and 27998 genes grouped into eight clusters: ductal, Ngn3 low EP, Ngn3 high EP, pre-endocrine, alpha, beta, delta and epsilon cells. We used scVelo's preprocessing function *filter_and_normalize* with minimum shared counts as 20 and selecting the 2000 top variable genes. Then, we computed the velocities using the dynamical model as it is shown in scVelo's documentation. At the end we obtained a dataset with 3696 cells and 2000 genes with latent time estimation and velocities.

Since no scATAC-seq data of pancreatic endocrinogenesis were available, we use the cisTarget [Imr+15] ranking databases to obtain a list of potential target genes for each transcription factor. In particular, we selected the first 1500 target genes from the mm9-500bp-upstream-10species table that compresses information about 10 orthologous species, scanning 500bp upstream of the gene bodies. More information about the ranking databases can be found on [Imr+15].

Figure 6.10 shows the UMAP of the pancreatic endocrinogenesis dataset and the vector fields generated using the inferred GRN by the *gene thicket* on the left and using the dynamical model of scVelo on the right. Although the vector fields do not resemble each other, we believe that this is a first step to infer velocities from a GRN structure.

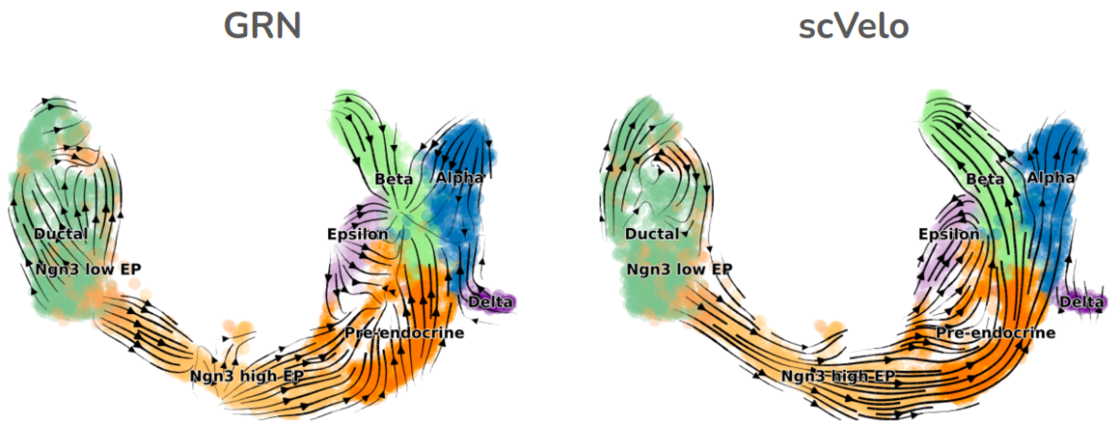


Figure 6.10: Comparison between GRN displacements computed with the *gene thicket* and RNA velocities from scVelo in the pancreatic endocrinogenesis dataset.

We also wondered which gene expression patterns are difficult for the *gene thicket* to predict. We discovered that abrupt changes in expression present a particular challenge for the *gene thicket* as it can be observed in Figure 6.11. However, for genes with a smooth expression pattern, the *gene thicket* manages to capture the tendency.

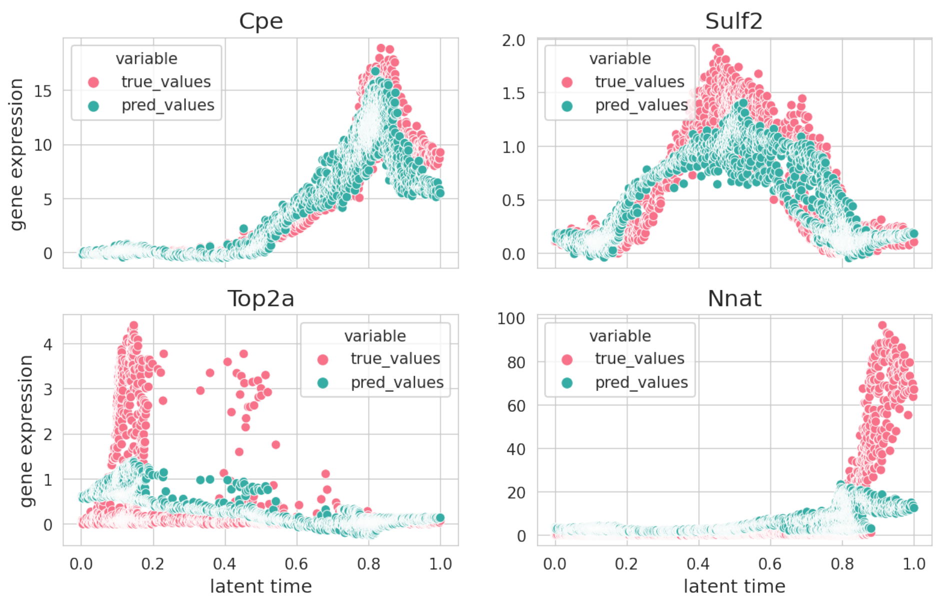


Figure 6.11: Gene expression predictions using the *gene thicket*. The model predicts the tendency for genes like Cpe and Sulf2, however struggles to capture abrupt changes in gene expression patterns as in Top2a and Nnat.

Chapter 7

Conclusion and outlook

In this work, we developed a new method to infer gene regulatory networks from scRNA-seq and scATAC-seq data using deep learning called the *gene thicket*. The *gene thicket* is based on temporal convolutional neural networks with an attention mechanism for interpretability. Our model was inspired by the TCDF architecture from [NBS19], used in the context of stock market prediction.

What sets the *gene thicket* apart from other GRN inference methods is its ability to predict the future state of each cell. To do that, the *gene thicket* leverages scATAC-seq information to identify putative transcription factor - target gene pairs and then uses a convolutional neural network to predict the expression of a gene using only previous time steps from its putative transcription factors. With the attention mechanism, we can construct a directed and weighted graph. To add the signs, we compute the Pearson correlation coefficient between the transcription factor - target gene pairs.

The ability of predicting the future state of each cell, is an extremely powerful characteristic of the *gene thicket* that enables us to estimate cell displacements, comparable to RNA velocities. This is a first step to obtain velocities from a GRN structure.

We evaluated our method against other state-of-the-art GRN inference methods and an adaptation of the TCDF architecture to infer GRNs that we implemented. For the evaluation, we considered the area under the precision recall curve (AUPRC) and the area under the receiver operating characteristic curve (AUROC) as metrics. The AUROC allows us to compare the models against a random predictor, while the AUPRC is suitable for classification problems with unbalanced classes. We used the curated and synthetic datasets from [Pra+20] to evaluate the models, these were created specifically to benchmark GRN inference methods. Our model, the *gene thicket* competes against other state-of-the-art GRN methods in terms of AUPRC, AUROC and scalability (running time).

Additionally, we presented applications on biological data. We decided to use the pancreatic endocrinogenesis dataset from [BP+19] to have a better comparison of the *gene thicket's* cell displacements against scVelo's velocities. However, complementary scATAC-seq data were not yet available and we computed a putative transcription factor - target gene list from cisTarget databases [Imr+15]. For future work, it is remaining to apply the *gene thicket* on biological data, where both, scRNA-seq and scATAC.seq are available.

As far as we are aware, GRN inference methods include scATAC-seq data either as a hard prior (CellOracle [KHM20], the *gene thicket*) or as an additional information to prune an inferred graph (SCENIC [Aib+17]). Nevertheless, we believe that scATAC-seq data information should be included as a soft prior to enable the discovery of additional regulatory links that are not necessarily captured by this type of data. For example, the list of transcription factor - gene target pairs can be expressed as a penalty term in the loss function of the models.

The *gene thicket* uses an attention mechanism to identify the relevant transcription factor - target gene pairs, but still does not consider any type of uncertainty estimation nor distribution. The use of a statistical method to identify transcription factor - target gene pairs will be more robust and may lead to a different model interpretation.

The sign of an edge in a GRN describes whether a transcription factor activates or represses a particular target gene. The *gene thicket* infers the type of relationship according to the Pearson correlation coefficient sign. If the sign is negative, the relationship will be considered repression, and if the sign is positive, the relationship will be considered activation. We believe that this estimation can be improved by taking into account the temporal trends of transcription factor - target gene pairs.

Finally, one of the biggest pitfalls of the *gene thicket* is its lack of ability to handle different cell trajectories. We observed that this severely affects its capacity to estimate future gene expression patterns, leading to spurious link predictions, for which it is necessary to adapt the model.

Appendix A

Metrics for network evaluation

We can evaluate GRN inference using: the Area Under the Precision Recall Curve (AUPRC) and the Area Under the Receiver Operating Characteristic Curve (AUROC). These metrics are commonly used to assess classifiers. In the case of GRN inference, we will not take into account signs (activation or repression), self-loops nor weights. We will only consider if a relationship between two genes was predicted. Then, we can think of the problem as a binary classification task: if *geneA*→*geneB* exists, then (*geneA*, *geneB*) should be of class 1; and if such relationship does not exist, (*geneA*, *geneB*) should be of class 0.

A.1 Area Under the Precision Recall Curve

The AUPRC is a very useful metric when the classes that we are predicted are imbalanced and we care about predicting the positive examples correctly, which makes it a suitable metric to assess GRN's. The *Precision Recall Curve* (PR curve) is a curve that shows the trade-off between *Precision* and *Recall* across different decision thresholds of the classifier. *Precision* (also known as *Positive Predictive Value*) is the ability of the model to classify all positive examples as positive and it is defined as:

$$Precision = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \tag{A.1}$$

while *Recall* (also known as *Sensitivity*) is the ability of the model not to wrongly classify a negative example as positive, and it is defined as:

$$Recall = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \tag{A.2}$$

The x-axis of a PR curve is the *Recall* and the y-axis is the *Precision.* The PR curve starts at the point $(\text{recall} = 0, \text{precision} = 1)$, where everything is classified as negative; and it ends at a point where recall= 1 and every example is classified as positive. The rest of the points from the PR curve correspond to the precision and recall of different decision thresholds.

The AUPRC is a measure of how good is the model to predict the positive examples correctly, considering also the false positives. It can take values from zero to one, where one is the score for perfect precision and recall. To calculate the AUPRC, we use a method called *average precision* (AP) from the scikit-learn package [Ped+11]. The AP is a weighted sum of the precisions at each threshold, where the weights are the increases of recall. It is defined as:

$$AP = \sum_n (R_n - R_{n-1}) P_n \tag{A.3}$$

where R_n and R_{n-1} are the recall scores at points n and $n-1$, respectively, and P_n is the precision at point n.

A.2 Area Under the Receiver Operating Characteristic Curve

The ROC curve is a probability curve, that can be interpreted as the probability the classifier has to make a correct prediction. Then, the area under the ROC curve (AUROC) represents how much a classifier can distinguish between classes. The x-axis of the ROC curve is the false positive rate, and the y-axis is the recall, given by equation A.2. The *False Positive Rate* (FPR) is defined as:

$$FPR = \frac{\text{false positives}}{\text{false positives} + \text{true negatives}} \tag{A.4}$$

The baseline for the AUROC is 0.5, which means that the model does not have the ability to distinguish between two classes. An AUROC of 1 will represent a perfect classifier, which can distinguish the two classes with no error. On the contrary, an AUROC of 0 means that the classifier is predicting all positive samples as negative and vice versa.

To calculate the AUROC, we use the scikit-learn package [Ped+11].

Appendix B

Datasets

B.1 Curated Datasets

The curated data was created to avoid the situation of having a large-scale regulatory network that may not capture the regulatory interactions of a specific developmental process, as mentioned in [CM18]. This data was generated with BoolODE, a method developed by them. BoolODE transforms boolean functions into ordinary differential equations, adds noise, and simulates stochastic time. The simulations were performed using four selected published boolean models: Mammalian Cortical Area Development (mCAD) [GG10], Ventral Spinal Cord Development (VSC)[Lov+14], Hematopoietic Stem Cell Differentiation (HSC) [Kru+11], and Gonadal Sex Determination (GSD) [Río+15]. The authors ensure that all the simulated datasets have the same number of steady states and gene expression patterns that characterize them, as it is reported in their own respective literature. A summary of the ground truth networks for each of the models can be found in table B.1.

Model	Number of genes	Number of activation edges	Number of inhibition edges	Network density	Number of steady states
mCAD	5	5	9	0.65	2
VSC	8	0	15	0.27	5
HSC	11	15	15	0.24	4
GSD	19	27	59	0.25	2

Table B.1: Summary of curated datasets from selected boolean models [Pra+20]. Self-loops were ignored for the computation of Network density.

For each boolean model, the authors of [Pra+20] created ten different datasets with two thousand cells, and later generated three different versions of each one, according to distinct dropout rates (0, 50 and 70). In total we have 30 datasets for each boolean model. Pseudotime values for each dataset were computed using Slingshot [Str+18]. Figure B.1 shows a clear visualization of the data and the ground truth network of each of the models. More information about the datasets can be found in [Pra+20].

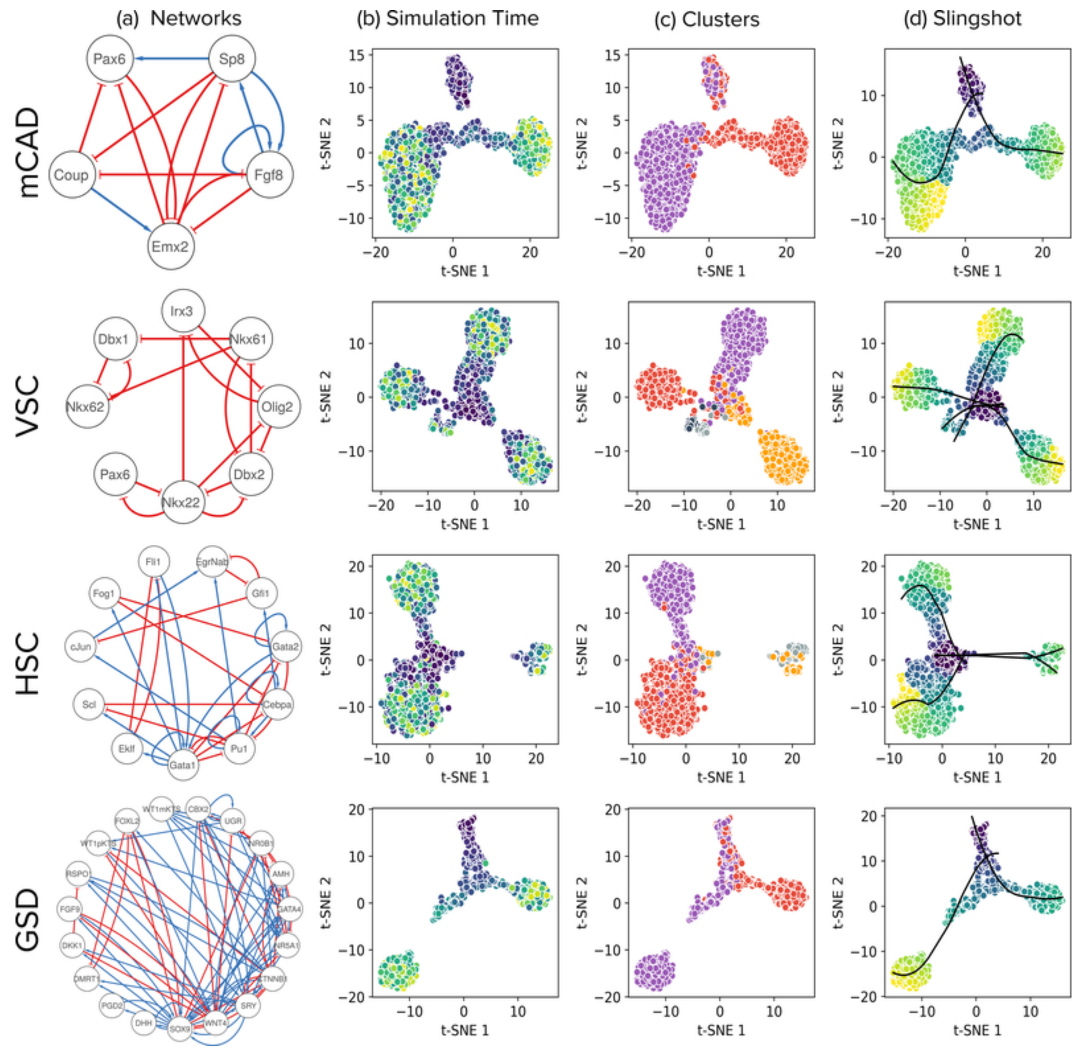


Figure B.1: Visualization of curated models. (a) Network diagrams. (b) t-SNE maps colored by simulation time. (c) t-SNE maps colored by different cell subsets obtained using k-means clustering of simulations, where k is the number of steady states for each model (two for mCAD, five for VSC, four for HSC, and two for GSD). (d) t-SNE maps colored by simulation time and pseudotime computed with Slingshot [Str+18]. Principal curves are shown in black. Source: [Pra+20].

B.2 Synthetic Datasets

The motivation of having synthetic data is to have a ground truth GRN to evaluate the performance of GRN inference methods, and also to have a dataset that does not rely on pseudotime inference algorithms. Six different types of synthetic data were created using BoolODE [Pra+20]:

- **Linear:** a cascade of gene activations in one single trajectory, where the initial and final states are different.
- **Linear long:** follows the same principle as the Linear data, but with more genes.
- **Cycle:** gene activations and repressions in a single trajectory, where the initial and final state are the same.
- **Bifurcating:** a network with a mutual repression relationship between two genes, resulting in two different branches, leading to two different final states.
- **Bifurcating converging:** similar to the bifurcating, but with the two branches converging to a single final state.
- **Trifurcating:** a network that contains mutual repression relationships between three genes, that results in three different final states.

Dataset	Number of genes	Number of activation edges	Number of inhibition edges	Number of steady states
Linear	7	7	1	1
Linear long	18	18	1	1
Cycle	6	3	3	0
Bifurcating	8	7	5	2
Bifurcating converging	10	13	5	1
Trifurcating	8	10	10	3

Table B.2: Summary of synthetic datasets [Pra+20].

For each one of the synthetic dataset types, the authors selected 10 different parameter sets. For each parameter set, they performed 5000 simulations and generated five distinct datasets with different number of cells (100, 200, 500, 2000, and 5000). In total, we have 50 different datasets of each type. Figure B.2 and Table B.2 show an overview of the synthetic networks. More information aboout the synthetic datasets can be found in [Pra+20].

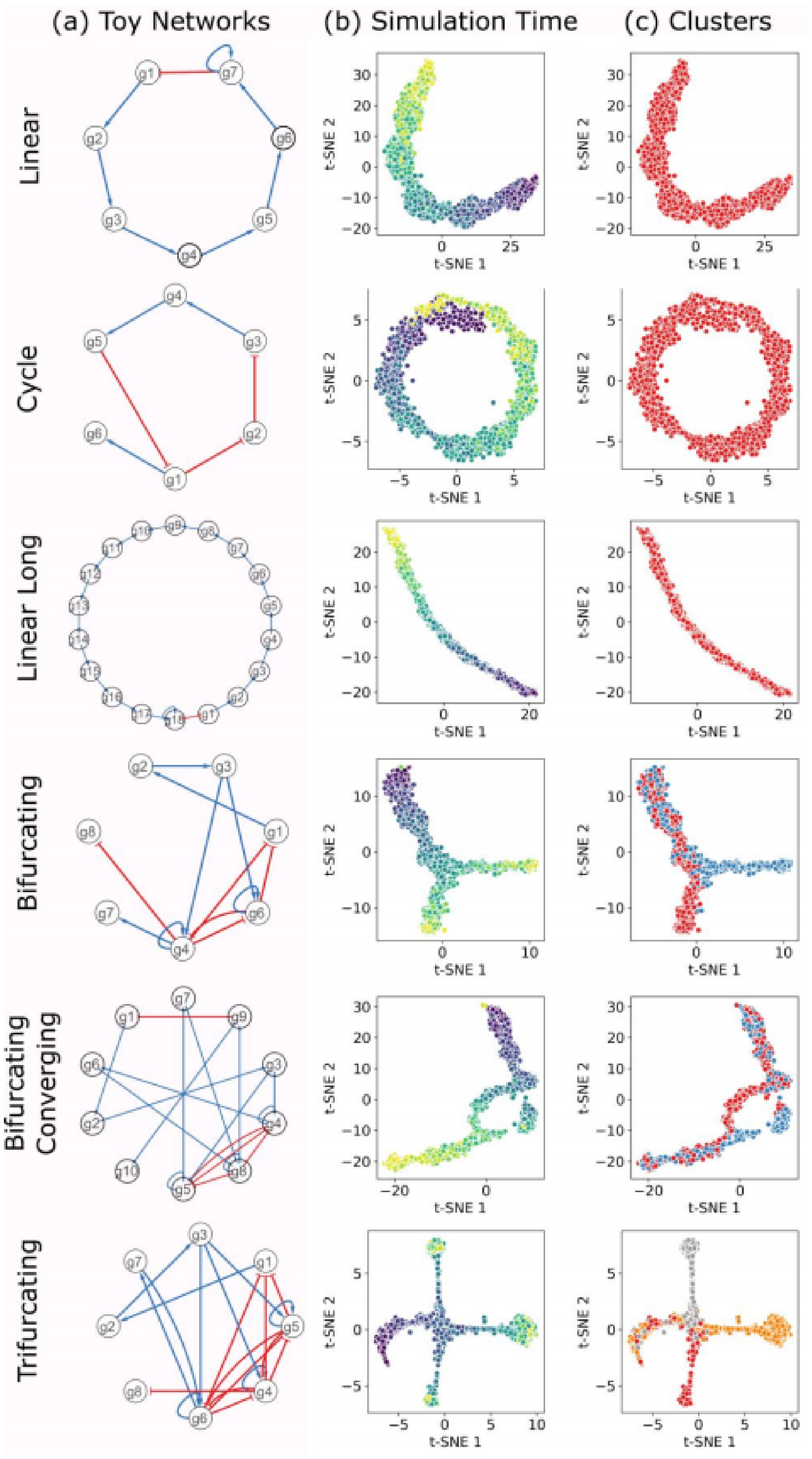


Figure B.2: Visualization of synthetic networks. (a) Network diagrams. (b) t-SNE maps colored by simulation time for 2000 cells. Dark points represent early stages, and light points final states. (c) Each color corresponds to a different cluster inferred using k-means, where k was specified as the number of expected final steady states. Source: [Pra+20].

Appendix C

The gene thicket: number of blocks

The number of blocks that conform the *gene thicket* is perhaps the most important hyperparameter among all. To select the number of blocks we evaluate the performance of the *gene thicket* on the curated and synthetic data using the AUPRC and AUROC metrics described on Appendix A.

In Figures C.1 and C.2, we can observe the performance of the *gene thicket* with one, two and three blocks on all curated and synthetic datasets. We noticed that for curated datasets the architecture with one block seemed to have the best behavior, while for the synthetic data we obtained the best results when the architecture has three blocks.

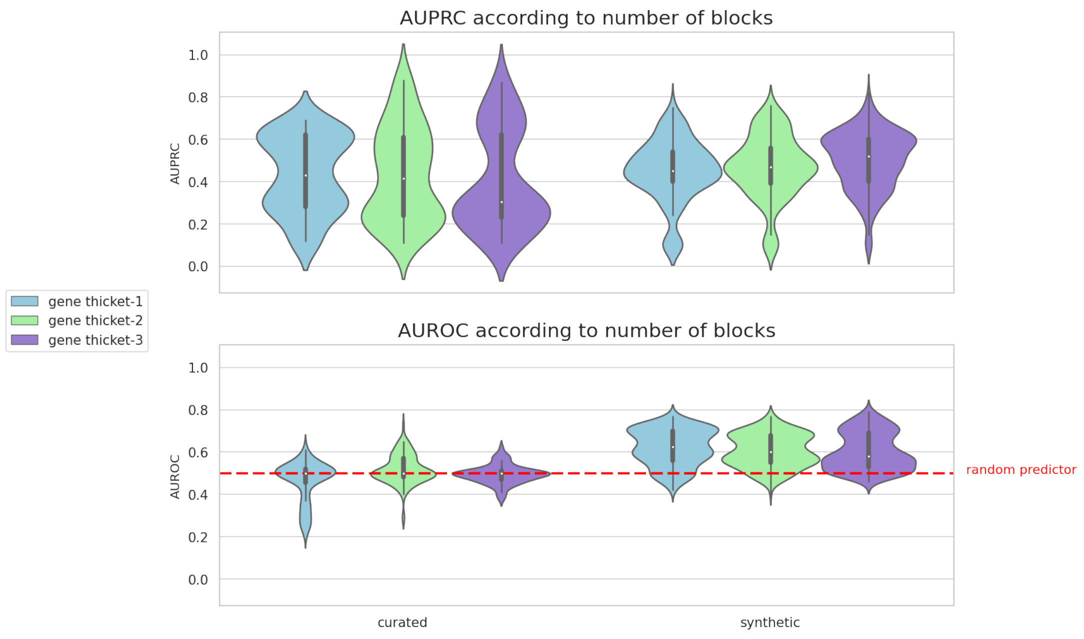


Figure C.1: AUPRC and AUROC according to different number of blocks in the distinct dataset types. The *gene thicket's* architecture shows the best performance on the curated data when having two blocks, while for synthetic data when having only one.

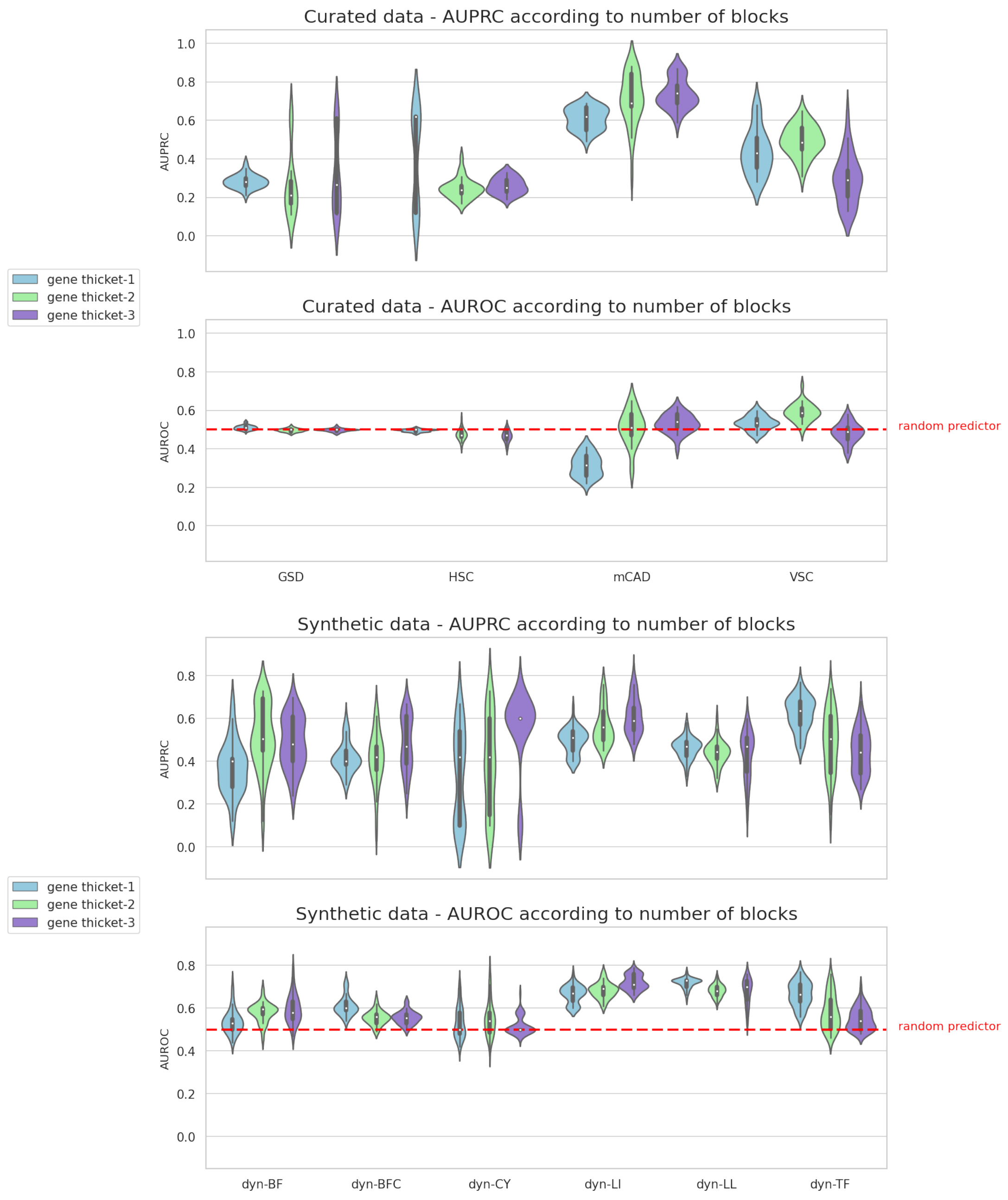


Figure C.2: AUPRC and AUROC according to different number of blocks on curated and synthetic data. The *gene thicket's* architecture with two blocks has the best performance on curated data, while for synthetic data the architecture with one block shows the best results.

Since we have to select a specific number of blocks to infer all the GRNs, we decided to take the architecture with one block since it has the best performance on the curated data, and takes considerably less time to run compared with the other architectures as it is shown in Figure C.3. However, the number of blocks is a hyperparameter that can be adjusted according to the data we will use.

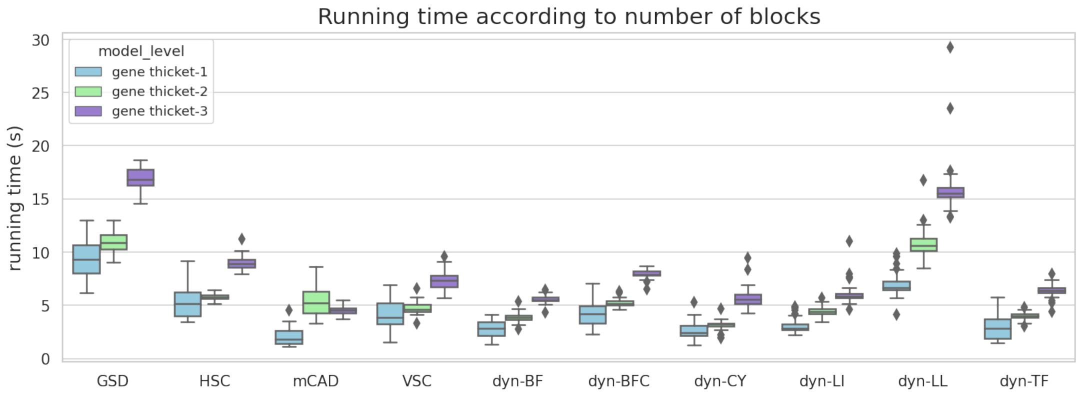


Figure C.3: Time tend to increase when the architecture gets more complex.

List of Figures

2.1 Transcription and translation [Bro] 3

2.2 RNA-seq quality control distributions with filtering decisions for a mouse epithelium dataset from [Hab+17]. Image source: [LT19] 5

2.3 Regulatory interactions and GRNs [San+19] 7

3.1 Tn5 (green) with sequencing adapters (red and blue) binds to the open chromatin region and cuts the DNA into fragments [Bue+13]. 10

3.2 a ATAC-seq datasets and publications are increasing exponentially b approximate input material and preparation time of open chromatin assays c comparison of open chromatin assays at a locus GM12878 lymphoblastoid. ([Yan+20] [Bue+13]) 11

3.3 Successfull ATAC-seq length distribution from [Bue+13]. 12

3.4 TSS enrichment plot from [Yan+20]. Nucleosome fragments are enriched at TSS, while mono-nucleosome fragments are enriched at flanking regions. 13

3.5 Tn5 transposase fragments and tags the accessible regions with sequencing adapters to identify open chromatin regions (black) and transcription factor footprints (blue). The nucleosome-free region (NFR) fragments are represented by peaks [Yan+20]. 13

3.6 Comparison of distinct peak calling algorithms from [Che+19]. 14

3.7 Overview of Cicero by [Pli+18]. Cicero finds putative cis-regulatory interactions through co-accessibility. 18

3.8 Example of Cicero co-accessibility links between MYOG promoter and surrounding distal sites. Red links correspond to phase 1, while blue links to phase 2. The height of connections indicates the magnitude of the Cicero co-accessibility score between the connected peaks ([Pli+18]). . . . 18

3.9 Trajectories on CD34+ hematopoietic stem and progenitor cells using Monocle3 [Tra+14]. Erythroid cells are earlier in time than Lymphoid ([SSb]). 19

3.10 Differential analysis of gene Neurod6 peaks across different cell types of the mouse brain ([SSa]). 20

3.11 Change of promoter accessibility through time in human skeletal muscle myoblasts (HSMM). Black line indicates the pseudotime-dependent average from a smoothed binomial regression ([Pli+18]). 21

3.12 a Aggregate ATAC-seq footprint of CTCF transcription factor. b CTCF footprints observed through different assays. ([Bue+13]) 21
3.13 Gene activities estimated using scATAC-seq data of cells from the mouse brain ([Stu+20]). 23
3.14 scRNA-seq and scATAC-seq joint embedding of cells from the mouse brain using Signac [Stu+20]. 24

4.1 Graphical representation of a neural network. 26
4.2 PReLU activation function. 27
4.3 Structure of a residual block. [Bec21] 29
4.4 Training and validation loss through epochs. Early stopping (dotted line), indicates the point where we should stop the training before overfitting the data. [Vog18] 29
4.5 Convolutional layer. The input image is represented with its three channels: red, green and blue; the convolution kernel of size 2×2 is shown in orange; the feature map is represented in gray. [Bec21] 30
4.6 Padding. A channel of the input image with zero padding is shown in blue. The feature map, represented in gray, has the same size of the original image as indicated by the green outline. [Bec21] 31
4.7 Comparison between a standard convolution (a) and a causal convolution (b) [Kla19] 31
4.8 Dilated convolution. The receptive field expands without loss of resolution (without parameter increasing).[YK15] 32
4.9 Causal dilated convolution. The combination of a causal convolution and dilations [Kla19]. 32
4.10 Comparison between a standard convolution (a) and a depthwise separable convolution (b) [Ben]. 33

5.1 The gene thicket's workflow overview. scATAC-seq data is used as a prior to obtain directed regulatory gene connections. Then, pseudotime ordered expression profiles of transcription factors are used to predict the pseudotime ordered expression profile of its putative target genes. The output is a refined network with signed and weighted connections. 36
5.2 Overview of a network that predicts a target gene expression through time. Each transcription factor time series is multiplied by an attention constant σa_i, and then used as an input for a residual block denoted by $\bigoplus$. The residual block consists of convolutional layers with dilations, which have PReLU activation functions and weight normalizations in between. At the end a pointwise convolution is used to combine separable outputs into a single future prediction for the target gene. 38

5.3 The threshold value τ_j is the attention score on the left side of the largest gap g_k, with $k \neq 0$. 40

6.1 AUPRC and AUROC of GRN Inference methods evaluated on the curated datasets. The *gene thicket* has the best performance on the mCAD dataset, while for the rest, its performance lies on the bottom two. 44
6.2 The *gene thicket* struggles to predict the expression of Cebpa, Gfi1, Gata1 and Gata2 in version eight from HSC dataset with 2000 cells and drop out rate of 70. 44
6.3 Effect of dropouts on the performance of GRN inference methods. 45
6.4 AUPRC and AUROC of GRN Inference methods evaluted on the synthetic datasets. 46
6.5 The *gene thicket* predicts the expression of genes of version two from the dyn-LI dataset with 2000 cells. 47
6.6 Effect of number of cells on the performance of GRN inference algorithms. 48
6.7 Running time of GRN inference methods across all datasets. 49
6.8 Running time of deep learning methods is similar across the number of cells. However, for tree ensemble methods, the number of cells has a great impact in the running time. 49
6.9 The running time of *gene thicket* is consistently low through datasets and number of cells. SINCERITIES shows significantly more running time variance, independent of the number of cells. The number of cells has greater impact on the running time for tree ensemble methods. 50
6.10 Comparison between GRN displacements computed with the *gene thicket* and RNA velocities from scVelo in the pancreatic endocrinogenesis dataset. 51
6.11 Gene expression predictions using the *gene thicket*. The model predicts the tendency for genes like Cpe and Sulf2, however struggles to capture abrupt changes in gene expression patterns as in Top2a and Nnat. 52

B.1 Visualization of curated models. (a) Network diagrams. (b) t-SNE maps colored by simulation time. (c) t-SNE maps colored by different cell subsets obtained using k-means clustering of simulations, where k is the number of steady states for each model (two for mCAD, five for VSC, four for HSC, and two for GSD). (d) t-SNE maps colored by simulation time and pseudotime computed with Slingshot [Str+18]. Principal curves are shown in black. Source: [Pra+20]. 58
B.2 Visualization of synthetic networks. (a) Network diagrams. (b) t-SNE maps colored by simulation time for 2000 cells. Dark points represent early stages, and light points final states. (c) Each color corresponds to a different cluster inferred using k-means, where k was specified as the number of expected final steady states. Source: [Pra+20]. 60

C.1 AUPRC and AUROC according to different number of blocks in the distinct dataset types. The *gene thicket's* architecture shows the best performance on the curated data when having two blocks, while for synthetic data when having only one. 61
C.2 AUPRC and AUROC according to different number of blocks on curated and synthetic data. The *gene thicket's* architecture with two blocks has the best performance on curated data, while for synthetic data the architecture with one block shows the best results. 62
C.3 Time tend to increase when the architecture gets more complex. 63

List of Tables

B.1 Summary of curated datasets from selected boolean models [Pra+20]. Self-loops were ignored for the computation of Network density. 57
B.2 Summary of synthetic datasets [Pra+20]. 59

Bibliography

[Aib+17] S. Aibar, C. B. González-Blas, T. Moerman, H. Imrichova, G. Hulselmans, F. Rambow, J.-C. Marine, P. Geurts, J. Aerts, J. van den Oord, et al. “SCENIC: single-cell regulatory network inference and clustering”. In: *Nature methods* 14.11 (2017), pp. 1083–1086.

[AKB19] H. M. Amemiya, A. Kundaje, and A. P. Boyle. “The ENCODE blacklist: identification of problematic regions of the genome”. In: *Scientific reports* 9.1 (2019), pp. 1–5.

[And10] S. Andrews. “Babraham bioinformatics-FastQC a quality control tool for high throughput sequence data”. In: *Babraham Inst* (2010).

[AFV20] P.-C. Aubin-Frankowski and J.-P. Vert. “Gene regulation inference from single-cell RNA-seq data with linear differential equations and velocity inference”. In: *Bioinformatics* 36.18 (2020), pp. 4774–4780.

[AAG14] E. Azizi, E. Airoldi, and J. Galagan. “Learning modular structures from network data and node variables”. In: *International conference on machine learning*. PMLR. 2014, pp. 1440–1448.

[Azo+18] J. G. Azofeifa, M. A. Allen, J. R. Hendrix, J. D. Rubin, and R. D. Dowell. “Enhancer RNA profiling predicts transcription factor activity”. In: *Genome research* 28.3 (2018), pp. 334–344.

[BRL19] J Baglama, L Reichel, and B. Lewis. *Irlba: Fast Truncated Singular Value Decomposition and Principal Components Analysis for Large Dense and Sparse Matrices, R package version 2.3. 3.* 2019.

[BR05] J. Baglama and L. Reichel. “Augmented implicitly restarted Lanczos bidiagonalization methods”. In: *SIAM Journal on Scientific Computing* 27.1 (2005), pp. 19–42.

[BKK18a] S. Bai, J. Z. Kolter, and V. Koltun. “An empirical evaluation of generic convolutional and recurrent networks for sequence modeling”. In: *arXiv preprint arXiv:1803.01271* (2018).

[BKK18b] S. Bai, J. Z. Kolter, and V. Koltun. “Convolutional sequence modeling revisited”. In: (2018).

[BP+19] A. Bastidas-Ponce, S. Tritschler, L. Dony, K. Scheibner, M. Tarquis-Medina, C. Salinno, S. Schirge, I. Burtscher, A. Böttcher, F. J. Theis, et al. "Comprehensive single cell mRNA profiling reveals a detailed roadmap for pancreatic endocrinogenesis". In: *Development* 146.12 (2019).

[Bec21] A. Becker. "Predicting transcription rate from multiplexed protein maps using deep learning". MA thesis. Technische Universität München, May 2021.

[Ben+14] S. C. Bendall, K. L. Davis, E.-a. D. Amir, M. D. Tadmor, E. F. Simonds, T. J. Chen, D. K. Shenfeld, G. P. Nolan, and D. Pe'er. "Single-cell trajectory detection uncovers progression and regulatory coordination in human B cell development". In: *Cell* 157.3 (2014), pp. 714–725.

[Ben] E. Bendersky. *Depthwise separable convolutions for machine learning.* URL: `https://eli.thegreenplace.net/2018/depthwise-separable-convolutions-for-machine-learning/`. (published: 04.04.2018).

[Ben+20] M. Bentsen, P. Goymann, H. Schultheis, K. Klee, A. Petrova, R. Wiegandt, A. Fust, J. Preussner, C. Kuenne, T. Braun, et al. "ATAC-seq footprinting unravels kinetics of transcription factor binding during zygotic genome activation". In: *Nature communications* 11.1 (2020), pp. 1–11.

[Ber+20] V. Bergen, M. Lange, S. Peidli, F. A. Wolf, and F. J. Theis. "Generalizing RNA velocity to transient cell states through dynamical modeling". In: *Nature biotechnology* 38.12 (2020), pp. 1408–1414.

[Bis14] C. Bishop. "Bishop-Pattern Recognition and Machine Learning-Springer 2006". In: *Antimicrob. Agents Chemother* (2014), pp. 03728–14.

[Blo+08] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. "Fast unfolding of communities in large networks". In: *Journal of statistical mechanics: theory and experiment* 2008.10 (2008), P10008.

[Bog12] D. F. Bogenhagen. "Mitochondrial DNA nucleoid structure". In: *Biochimica et Biophysica Acta (BBA)-Gene Regulatory Mechanisms* 1819.9-10 (2012), pp. 914–920.

[BLU14] A. M. Bolger, M. Lohse, and B. Usadel. "Trimmomatic: a flexible trimmer for Illumina sequence data". In: *Bioinformatics* 30.15 (2014), pp. 2114–2120.

[Boy+08] A. P. Boyle, J. Guinney, G. E. Crawford, and T. S. Furey. "F-Seq: a feature density estimator for high-throughput sequence tags". In: *Bioinformatics* 24.21 (2008), pp. 2537–2538.

[Bre01] L. Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[Bre+13] P. Brennecke, S. Anders, J. K. Kim, A. A. Kołodziejczyk, X. Zhang, V. Proserpio, B. Baying, V. Benes, S. A. Teichmann, J. C. Marioni, et al. “Accounting for technical noise in single-cell RNA-seq experiments”. In: *Nature methods* 10.11 (2013), pp. 1093–1095.

[Bro] T. Brown. *Simple representation of the gene expression process.* URL: https://www.atdbio.com/content/14/Transcription-Translation-and-Replication. (accessed: July 2021).

[BVH18] N. Bruse and S. J. Van Heeringen. “GimmeMotifs: an analysis framework for transcription factor motif analysis”. In: *BioRxiv* (2018), p. 474403.

[Bue+13] J. D. Buenrostro, P. G. Giresi, L. C. Zaba, H. Y. Chang, and W. J. Greenleaf. “Transposition of native chromatin for fast and sensitive epigenomic profiling of open chromatin, DNA-binding proteins and nucleosome position”. In: *Nature methods* 10.12 (2013), p. 1213.

[Bue+15] F. Buettner, K. N. Natarajan, F. P. Casale, V. Proserpio, A. Scialdone, F. J. Theis, S. A. Teichmann, J. C. Marioni, and O. Stegle. “Computational analysis of cell-to-cell heterogeneity in single-cell RNA-sequencing data reveals hidden subpopulations of cells”. In: *Nature biotechnology* 33.2 (2015), pp. 155–160.

[Bue+17] F. Buettner, N. Pratanwanich, D. J. McCarthy, J. C. Marioni, and O. Stegle. “f-scLVM: scalable and versatile factor analysis for single-cell RNA-seq”. In: *Genome biology* 18.1 (2017), pp. 1–13.

[But+18] A. Butler, P. Hoffman, P. Smibert, E. Papalexi, and R. Satija. “Integrating single-cell transcriptomic data across different conditions, technologies, and species”. In: *Nature biotechnology* 36.5 (2018), pp. 411–420.

[CSB17] T. E. Chan, M. P. Stumpf, and A. C. Babtie. “Gene regulatory network inference from single-cell data using multivariate information measures”. In: *Cell systems* 5.3 (2017), pp. 251–267.

[Che+19] H. Chen, C. Lareau, T. Andreani, M. E. Vinyard, S. P. Garcia, K. Clement, M. A. Andrade-Navarro, J. D. Buenrostro, and L. Pinello. “Assessment of computational methods for the analysis of single-cell ATAC-seq data”. In: *Genome biology* 20.1 (2019), pp. 1–25.

[Che+21] J. Chen, C. Cheong, L. Lan, X. M. Zhou, J. Liu, A. Lu, W. K. Cheung, and L. Zhang. “DeepDRIM: a deep neural network to reconstruct cell-type-specific gene regulatory network using single-cell RNA-seq data”. In: *bioRxiv* (2021).

[CM18] S. Chen and J. C. Mar. “Evaluating methods of inferring gene regulatory networks highlights their lack of performance for single cell gene expression data”. In: *BMC bioinformatics* 19.1 (2018), pp. 1–21.

[Cho16] F. Chollet. “Xception: deep learning with depthwise separable convolutions. CoRR abs/1610.02357 (2016)”. In: *arXiv preprint arXiv:1610.02357* (2016).

[Coi+05] R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner, and S. W. Zucker. “Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps”. In: *Proceedings of the national academy of sciences* 102.21 (2005), pp. 7426–7431.

[Col+19] M. B. Cole, D. Risso, A. Wagner, D. DeTomaso, J. Ngai, E. Purdom, S. Dudoit, and N. Yosef. “Performance assessment and selection of normalization procedures for single-cell RNA-seq”. In: *Cell systems* 8.4 (2019), pp. 315–328.

[Con+12] E. P. Consortium et al. “An integrated encyclopedia of DNA elements in the human genome”. In: *Nature* 489.7414 (2012), p. 57.

[Cus+15] D. A. Cusanovich, R. Daza, A. Adey, H. A. Pliner, L. Christiansen, K. L. Gunderson, F. J. Steemers, C. Trapnell, and J. Shendure. “Multiplex single-cell profiling of chromatin accessibility by combinatorial cellular indexing”. In: *Science* 348.6237 (2015), pp. 910–914.

[Cus+18] D. A. Cusanovich, A. J. Hill, D. Aghamirzaie, R. M. Daza, H. A. Pliner, J. B. Berletch, G. N. Filippova, X. Huang, L. Christiansen, W. S. DeWitt, et al. “A single-cell atlas of in vivo mammalian chromatin accessibility”. In: *Cell* 174.5 (2018), pp. 1309–1324.

[Des+21] A. Deshpande, L.-F. Chu, R. Stewart, and A. Gitter. “Network inference with granger causality ensembles on single-cell transcriptomic data”. In: *BioRxiv* (2021), p. 534834.

[Dut+19] S. H. Duttke, M. W. Chang, S. Heinz, and C. Benner. “Identification and dynamic quantification of regulatory elements using total RNA”. In: *Genome research* 29.11 (2019), pp. 1836–1846.

[EY36] C. Eckart and G. Young. “The approximation of one matrix by another of lower rank”. In: *Psychometrika* 1.3 (1936), pp. 211–218.

[Eic12] M. Eichler. *Causal inference in time series analysis.* na, 2012.

[Fan+21] R. Fang, S. Preissl, Y. Li, X. Hou, J. Lucero, X. Wang, A. Motamedi, A. K. Shiau, X. Zhou, F. Xie, et al. “Comprehensive analysis of single cell ATAC-seq data with SnapATAC”. In: *Nature communications* 12.1 (2021), pp. 1–15.

[FHT08] J. Friedman, T. Hastie, and R. Tibshirani. “Sparse inverse covariance estimation with the graphical lasso”. In: *Biostatistics* 9.3 (2008), pp. 432–441.

[Gasa] J. M. Gaspar. *ATAC-seq guidelines.* URL: https://informatics.fas.harvard.edu/atac-seq-guidelines.html. (published: 18.01.2019).

[Gasb] J. M. Gaspar. *Genrich*. URL: https://github.com/jsh58/Genrich. (published: 2018).

[Gas18a] J. M. Gaspar. “Improved peak-calling with MACS2”. In: *BioRxiv* (2018), p. 496521.

[Gas18b] J. M. Gaspar. “NGmerge: merging paired-end reads via novel empirically-derived models of sequencing errors”. In: *BMC bioinformatics* 19.1 (2018), pp. 1–9.

[GG10] C. E. Giacomantonio and G. J. Goodhill. “A Boolean model of the gene regulatory network underlying Mammalian cortical area development”. In: *PLoS computational biology* 6.9 (2010), e1000936.

[GB10] X. Glorot and Y. Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 249–256.

[GBC16] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[Hab+17] A. L. Haber, M. Biton, N. Rogel, R. H. Herbst, K. Shekhar, C. Smillie, G. Burgin, T. M. Delorey, M. R. Howitt, Y. Katz, et al. “A single-cell survey of the small intestinal epithelium”. In: *Nature* 551.7680 (2017), pp. 333–339.

[Hao+21] Y. Hao, S. Hao, E. Andersen-Nissen, W. M. Mauck III, S. Zheng, A. Butler, M. J. Lee, A. J. Wilk, C. Darby, M. Zager, et al. “Integrated analysis of multimodal single-cell data”. In: *Cell* (2021).

[He+15] K. He, X. Zhang, S. Ren, and J. Sun. “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.

[He+16] K. He, X. Zhang, S. Ren, and J. Sun. “Identity mappings in deep residual networks”. In: *European conference on computer vision*. Springer. 2016, pp. 630–645.

[Hec+09] M. Hecker, S. Lambeck, S. Toepfer, E. Van Someren, and R. Guthke. “Gene regulatory network inference: data integration in dynamic models—a review”. In: *Biosystems* 96.1 (2009), pp. 86–103.

[Hei+10] S. Heinz, C. Benner, N. Spann, E. Bertolino, Y. C. Lin, P. Laslo, J. X. Cheng, C. Murre, H. Singh, and C. K. Glass. “Simple combinations of lineage-determining transcription factors prime cis-regulatory elements required for macrophage and B cell identities”. In: *Molecular cell* 38.4 (2010), pp. 576–589.

[ILO15] M. M. Ibrahim, S. A. Lacadie, and U. Ohler. “JAMM: a peak finder for joint analysis of NGS replicates”. In: *Bioinformatics* 31.1 (2015), pp. 48–55.

[Imr+15] H. Imrichová, G. Hulselmans, Z. Kalender Atak, D. Potier, and S. Aerts. “i-cisTarget 2015 update: generalized cis-regulatory enrichment analysis in human, mouse and fly”. In: *Nucleic acids research* 43.W1 (2015), W57–W64.

[IS15] S. Ioffe and C. Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *International conference on machine learning*. PMLR. 2015, pp. 448–456.

[IWG+10] A. Irrthum, L. Wehenkel, P. Geurts, et al. “Inferring regulatory networks from expression data using tree-based methods”. In: *PloS one* 5.9 (2010), e12776.

[JLR07] W. E. Johnson, C. Li, and A. Rabinovic. “Adjusting batch effects in microarray expression data using empirical Bayes methods”. In: *Biostatistics* 8.1 (2007), pp. 118–127.

[KHM20] K. Kamimoto, C. M. Hoffmann, and S. A. Morris. “CellOracle: Dissecting cell identity via network inference and in silico gene perturbation”. In: *bioRxiv* (2020).

[Kha+18] A. Khan, O. Fornes, A. Stigliani, M. Gheorghe, J. A. Castro-Mondragon, R. Van Der Lee, A. Bessy, J. Cheneby, S. R. Kulkarni, G. Tan, et al. “JASPAR 2018: update of the open-access database of transcription factor binding profiles and its web framework”. In: *Nucleic acids research* 46.D1 (2018), pp. D260–D266.

[KSS14] P. V. Kharchenko, L. Silberstein, and D. T. Scadden. “Bayesian approach to single-cell differential expression analysis”. In: *Nature methods* 11.7 (2014), pp. 740–742.

[Kim15] S. Kim. “ppcor: an R package for a fast calculation to semi-partial correlation coefficients”. In: *Communications for statistical applications and methods* 22.6 (2015), p. 665.

[KB14] D. P. Kingma and J. Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[Kla19] J. Klaas. *Machine learning for finance: principles and practice for financial insiders*. Packt Publishing Ltd, 2019.

[Kor+19] I. Korsunsky, N. Millard, J. Fan, K. Slowikowski, F. Zhang, K. Wei, Y. Baglaenko, M. Brenner, P.-r. Loh, and S. Raychaudhuri. “Fast, sensitive and accurate integration of single-cell data with Harmony”. In: *Nature methods* 16.12 (2019), pp. 1289–1296.

[Kru+11] J. Krumsiek, C. Marr, T. Schroeder, and F. J. Theis. “Hierarchical differentiation of myeloid progenitors is encoded in the transcription factor network”. In: *PloS one* 6.8 (2011), e22649.

[Kul+18] I. V. Kulakovskiy, I. E. Vorontsov, I. S. Yevshin, R. N. Sharipov, A. D. Fedorova, E. I. Rumynskiy, Y. A. Medvedeva, A. Magana-Mora, V. B. Bajic, D. A. Papatsenko, et al. “HOCOMOCO: towards a complete collection of transcription factor binding models for human and mouse via large-scale ChIP-Seq analysis”. In: *Nucleic acids research* 46.D1 (2018), pp. D252–D259.

[LS12] B. Langmead and S. L. Salzberg. “Fast gapped-read alignment with Bowtie 2”. In: *Nature methods* 9.4 (2012), p. 357.

[Lar+19] C. A. Lareau, F. M. Duarte, J. G. Chew, V. K. Kartha, Z. D. Burkett, A. S. Kohlway, D. Pokholok, M. J. Aryee, F. J. Steemers, R. Lebofsky, et al. “Droplet-based combinatorial indexing for massive-scale single-cell chromatin accessibility”. In: *Nature Biotechnology* 37.8 (2019), pp. 916–924.

[LD09] H. Li and R. Durbin. “Fast and accurate short read alignment with Burrows–Wheeler transform”. In: *bioinformatics* 25.14 (2009), pp. 1754–1760.

[Lin+21] Y. Lin, T.-Y. Wu, S. Wan, J. Y. Yang, Y. R. Wang, and W. H. Wong. “scJoint: transfer learning for data integration of single-cell RNA-seq and ATAC-seq”. In: *bioRxiv* (2021), pp. 2020–12.

[LWT18] M. Lotfollahi, F. A. Wolf, and F. J. Theis. “Generative modeling and latent space arithmetics predict single-cell perturbation response across cell types, studies and species”. In: *bioRxiv* (2018), p. 478503.

[Lov+14] A. Lovrics, Y. Gao, B. Juhász, I. Bock, H. M. Byrne, A. Dinnyés, and K. A. Kovács. “Boolean modelling reveals new regulatory connections between transcription factors orchestrating the development of the ventral spinal cord”. In: *PloS one* 9.11 (2014), e111430.

[LT19] M. D. Luecken and F. J. Theis. “Current best practices in single-cell RNA-seq analysis: a tutorial”. In: *Molecular systems biology* 15.6 (2019), e8746.

[LB+] A. Lun, K Bach, et al. “(2016) Pooling across cells to normalize single-cell RNA sequencing data with many zero counts”. In: *Genome Biol* 17 (), p. 75.

[Mac+15] E. Z. Macosko, A. Basu, R. Satija, J. Nemesh, K. Shekhar, M. Goldman, I. Tirosh, A. R. Bialas, N. Kamitaki, E. M. Martersteck, et al. “Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets”. In: *Cell* 161.5 (2015), pp. 1202–1214.

[Mao+16] Q. Mao, L. Wang, I. W. Tsang, and Y. Sun. “Principal graph and structure learning based on reversed graph embedding”. In: *IEEE transactions on pattern analysis and machine intelligence* 39.11 (2016), pp. 2227–2241.

[Mar11] M. Martin. “Cutadapt removes adapter sequences from high-throughput sequencing reads”. In: *EMBnet. journal* 17.1 (2011), pp. 10–12.

[Mat+17] H. Matsumoto, H. Kiryu, C. Furusawa, M. S. Ko, S. B. Ko, N. Gouda, T. Hayashi, and I. Nikaido. “SCODE: an efficient regulatory network inference algorithm from single-cell RNA-Seq during differentiation”. In: *Bioinformatics* 33.15 (2017), pp. 2314–2321.

[MHM18] L. McInnes, J. Healy, and J. Melville. “Umap: Uniform manifold approximation and projection for dimension reduction”. In: *arXiv preprint arXiv:1802.03426* (2018).

[MB10] R. C. McLeay and T. L. Bailey. “Motif Enrichment Analysis: a unified framework and an evaluation on ChIP data”. In: *BMC bioinformatics* 11.1 (2010), pp. 1–11.

[Moe+19] T. Moerman, S. Aibar Santos, C. Bravo González-Blas, J. Simm, Y. Moreau, J. Aerts, and S. Aerts. “GRNBoost2 and Arboreto: efficient and scalable inference of gene regulatory networks”. In: *Bioinformatics* 35.12 (2019), pp. 2159–2161.

[NBS19] M. Nauta, D. Bucur, and C. Seifert. “Causal discovery with attention-based convolutional neural networks”. In: *Machine Learning and Knowledge Extraction* 1.1 (2019), pp. 312–340.

[Ntr+19] V. Ntranos, L. Yi, P. Melsted, and L. Pachter. “A discriminative learning approach to differential expression analysis for single-cell RNA-seq”. In: *Nature methods* 16.2 (2019), pp. 163–166.

[Ou+18] J. Ou, H. Liu, J. Yu, M. A. Kelliher, L. H. Castilla, N. D. Lawson, and L. J. Zhu. “ATACseqQC: a Bioconductor package for post-alignment quality assessment of ATAC-seq data”. In: *BMC genomics* 19.1 (2018), pp. 1–13.

[Pac+07] M. Pachkov, I. Erb, N. Molina, and E. Van Nimwegen. “SwissRegulon: a database of genome-wide annotations of regulatory sites”. In: *Nucleic acids research* 35.suppl_1 (2007), pp. D127–D131.

[PG+18] N. Papili Gao, S. M. Ud-Dean, O. Gandrillon, and R. Gunawan. “SINCERITIES: inferring gene regulatory networks from time-stamped single cell transcriptional expression profiles”. In: *Bioinformatics* 34.2 (2018), pp. 258–266.

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

[Pic+14] S. Picelli, Å. K. Björklund, B. Reinius, S. Sagasser, G. Winberg, and R. Sandberg. “Tn5 transposase and tagmentation procedures for massively scaled sequencing projects”. In: *Genome research* 24.12 (2014), pp. 2033–2040.

[Pli+18] H. A. Pliner, J. S. Packer, J. L. McFaline-Figueroa, D. A. Cusanovich, R. M. Daza, D. Aghamirzaie, S. Srivatsan, X. Qiu, D. Jackson, A. Minkina, et al. “Cicero predicts cis-regulatory DNA interactions from single-cell chromatin accessibility data”. In: *Molecular cell* 71.5 (2018), pp. 858–871.

[Pol+20] K. Polański, M. D. Young, Z. Miao, K. B. Meyer, S. A. Teichmann, and J.-E. Park. “BBKNN: fast batch alignment of single cell transcriptomes”. In: *Bioinformatics* 36.3 (2020), pp. 964–965.

[PMC18] T. J. Pranzatelli, D. G. Michael, and J. A. Chiorini. “ATAC2GRN: optimized ATAC-seq and DNase1-seq pipelines for rapid and accurate genome regulatory network inference”. In: *BMC genomics* 19.1 (2018), pp. 1–13.

[Pra+20] A. Pratapa, A. P. Jalihal, J. N. Law, A. Bharadwaj, and T. Murali. “Benchmarking algorithms for gene regulatory network inference from single-cell transcriptomic data”. In: *Nature methods* 17.2 (2020), pp. 147–154.

[Qiu+18] X. Qiu, A. Rahimzamani, L. Wang, Q. Mao, T. Durham, J. L. McFaline-Figueroa, L. Saunders, C. Trapnell, and S. Kannan. “Towards inferring causal gene regulatory networks from single cell expression measurements”. In: *BioRxiv* (2018), p. 426981.

[QH10] A. R. Quinlan and I. M. Hall. “BEDTools: a flexible suite of utilities for comparing genomic features”. In: *Bioinformatics* 26.6 (2010), pp. 841–842.

[Reg+17] A. Regev, S. A. Teichmann, E. S. Lander, I. Amit, C. Benoist, E. Birney, B. Bodenmiller, P. Campbell, P. Carninci, M. Clatworthy, et al. “Science forum: the human cell atlas”. In: *elife* 6 (2017), e27041.

[Río+15] O. Ríos, S. Frias, A. Rodríguez, S. Kofman, H. Merchant, L. Torres, and L. Mendoza. “A Boolean network model of human gonadal sex determination”. In: *Theoretical Biology and Medical Modelling* 12.1 (2015), pp. 1–18.

[SK16] T. Salimans and D. P. Kingma. “Weight normalization: A simple reparameterization to accelerate training of deep neural networks”. In: *Advances in neural information processing systems* 29 (2016), pp. 901–909.

[SC+18] M. Sanchez-Castillo, D. Blanco, I. M. Tienda-Luna, M. Carrion, and Y. Huang. “A Bayesian framework for the inference of gene regulatory networks from time and pseudo-time series data”. In: *Bioinformatics* 34.6 (2018), pp. 964–970.

[San+19] G. Sanguinetti et al. “Gene regulatory network inference: an introductory survey”. In: *Gene Regulatory Networks*. Springer, 2019, pp. 1–23.

[STB20] N. P. D. Santos, L. Texari, and C. Benner. “MEIRLOP: improving score-based motif enrichment by incorporating sequence bias covariates”. In: *BMC bioinformatics* 21.1 (2020), pp. 1–22.

[Sch+17] A. N. Schep, B. Wu, J. D. Buenrostro, and W. J. Greenleaf. “chromVAR: inferring transcription-factor-associated accessibility from single-cell epigenomic data”. In: *Nature methods* 14.10 (2017), pp. 975–978.

[She+18] T. Shen, T. Zhou, G. Long, J. Jiang, S. Wang, and C. Zhang. “Reinforced self-attention network: a hybrid of hard and soft attention for sequence modeling”. In: *arXiv preprint arXiv:1801.10296* (2018).

[SL17] A. T. Specht and J. Li. “LEAP: constructing gene co-expression networks for single-cell RNA-sequencing data using pseudotime ordering”. In: *Bioinformatics* 33.5 (2017), pp. 764–766.

[SS19] E. B. Stovner and P. Sætrom. “epic2 efficiently finds diffuse domains in ChIP-seq data”. In: *Bioinformatics* 35.21 (2019), pp. 4392–4393.

[Str+18] K. Street, D. Risso, R. B. Fletcher, D. Das, J. Ngai, N. Yosef, E. Purdom, and S. Dudoit. “Slingshot: cell lineage and pseudotime inference for single-cell transcriptomics”. In: *BMC genomics* 19.1 (2018), pp. 1–16.

[Stu+19] T. Stuart, A. Butler, P. Hoffman, C. Hafemeister, E. Papalexi, W. M. Mauck III, Y. Hao, M. Stoeckius, P. Smibert, and R. Satija. “Comprehensive integration of single-cell data”. In: *Cell* 177.7 (2019), pp. 1888–1902.

[SSa] T. Stuart and A. Srivastava. *Analyzing adult mouse brain scATAC-seq*. URL: https://satijalab.org/signac/articles/mouse_brain_vignette.html. (published: 10.05.2021).

[SSb] T. Stuart and A. Srivastava. *Building trajectories with Monocle3*. URL: https://satijalab.org/signac/articles/monocle.html. (published: 10.05.2021).

[Stu+20] T. Stuart, A. Srivastava, C. Lareau, and R. Satija. “Multimodal single-cell chromatin analysis with Signac”. In: *bioRxiv* (2020).

[Sve+17] V. Svensson, K. N. Natarajan, L.-H. Ly, R. J. Miragaia, C. Labalette, I. C. Macaulay, A. Cvejic, and S. A. Teichmann. “Power analysis of single-cell RNA-sequencing experiments”. In: *Nature methods* 14.4 (2017), pp. 381–387.

[Tan+18] A. Tank, I. Covert, N. Foti, A. Shojaie, and E. Fox. “Neural Granger Causality”. In: *arXiv preprint arXiv:1802.05842* (2018).

[TL19] E. D. Tarbell and T. Liu. “HMMRATAC: a Hidden Markov ModeleR for ATAC-seq”. In: *Nucleic acids research* 47.16 (2019), e91–e91.

[Tra+14] C. Trapnell, D. Cacchiarelli, J. Grimsby, P. Pokharel, S. Li, M. Morse, N. J. Lennon, K. J. Livak, T. S. Mikkelsen, and J. L. Rinn. “The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells”. In: *Nature biotechnology* 32.4 (2014), p. 381.

[TAD18] I. J. Tripodi, M. A. Allen, and R. D. Dowell. “Detecting differential transcription factor activity from ATAC-seq data”. In: *Molecules* 23.5 (2018), p. 1136.

[VDM09] L. Van Der Maaten. “Learning a parametric embedding by preserving local structure”. In: *Artificial Intelligence and Statistics*. PMLR. 2009, pp. 384–391.

[Vas+17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. “Attention is all you need”. In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[Ver+15] A. Verfaillie, H. Imrichova, R. Janky, and S. Aerts. “iRegulon and i-cisTarget: Reconstructing Regulatory Networks Using Motif and Track Enrichment”. In: *Current protocols in bioinformatics* 52.1 (2015), pp. 2–16.

[Vog18] R. Vogl. “Deep learning methods for drum transcription and drum pattern generation”. In: *Johannes Kepler University Linz, Linz* (2018).

[WRY16] A. Wagner, A. Regev, and N. Yosef. “Revealing the vectors of cellular identity with single-cell genomics”. In: *Nature biotechnology* 34.11 (2016), pp. 1145–1160.

[WVE13] L. Waltman and N. J. Van Eck. “A smart local moving algorithm for large-scale modularity-based community detection”. In: *The European physical journal B* 86.11 (2013), pp. 1–14.

[Wan+12] J. Wang, J. Zhuang, S. Iyer, X.-Y. Lin, M. C. Greven, B.-H. Kim, J. Moore, B. G. Pierce, X. Dong, D. Virgil, et al. “Factorbook. org: a Wiki-based database for transcription factor-binding data generated by the ENCODE consortium”. In: *Nucleic acids research* 41.D1 (2012), pp. D171–D176.

[Wan+20] J. Wang, A. Ma, Q. Ma, D. Xu, and T. Joshi. “Inductive inference of gene regulatory network using supervised and semi-supervised graph neural networks”. In: *Computational and Structural Biotechnology Journal* 18 (2020), pp. 3335–3343.

[WWK18] C. Weinreb, S. Wolock, and A. M. Klein. “SPRING: a kinetic interface for visualizing high dimensional single-cell expression data”. In: *Bioinformatics* 34.7 (2018), pp. 1246–1248.

[WAT18] F. A. Wolf, P. Angerer, and F. J. Theis. “SCANPY: large-scale single-cell gene expression data analysis”. In: *Genome biology* 19.1 (2018), pp. 1–5.

[Yan+20] F. Yan, D. R. Powell, D. J. Curtis, and N. C. Wong. “From reads to insight: a hitchhiker’s guide to ATAC-seq data analysis”. In: *Genome biology* 21.1 (2020), p. 22.

[Yin+16] W. Yin, H. Schütze, B. Xiang, and B. Zhou. “Abcnn: Attention-based convolutional neural network for modeling sentence pairs”. In: *Transactions of the Association for Computational Linguistics* 4 (2016), pp. 259–272.

[YK15] F. Yu and V. Koltun. “Multi-scale context aggregation by dilated convolutions”. In: *arXiv preprint arXiv:1511.07122* (2015).

[YBJ19] Y. Yuan and Z. Bar-Joseph. “Deep learning for inferring gene relationships from single-cell expression data”. In: *Proceedings of the National Academy of Sciences* 116.52 (2019), pp. 27151–27158.

[Zei+18] A. Zeisel, H. Hochgerner, P. Lönnerberg, A. Johnsson, F. Memic, J. Van Der Zwan, M. Häring, E. Braun, L. E. Borm, G. La Manno, et al. “Molecular architecture of the mouse nervous system”. In: *Cell* 174.4 (2018), pp. 999–1014.

[Zha+88] W. Zhang et al. “Shift-invariant pattern recognition neural network and its optical architecture”. In: *Proceedings of annual conference of the Japan Society of Applied Physics*. 1988.

[Zha+11] X. Zhang, G. Robertson, M. Krzywinski, K. Ning, A. Droit, S. Jones, and R. Gottardo. “PICS: probabilistic inference for ChIP-seq”. In: *Biometrics* 67.1 (2011), pp. 151–163.

[Zha+08] Y. Zhang, T. Liu, C. A. Meyer, J. Eeckhoute, D. S. Johnson, B. E. Bernstein, C. Nusbaum, R. M. Myers, M. Brown, W. Li, et al. “Model-based analysis of ChIP-Seq (MACS)”. In: *Genome biology* 9.9 (2008), pp. 1–9.

[Zhu+08] J. Zhu, B. Zhang, E. N. Smith, B. Drees, R. B. Brem, L. Kruglyak, R. E. Bumgarner, and E. E. Schadt. “Integrating large-scale functional genomic data to dissect the complexity of yeast regulatory networks”. In: *Nature genetics* 40.7 (2008), pp. 854–861.

[Zie+17] C. Ziegenhain, B. Vieth, S. Parekh, B. Reinius, A. Guillaumet-Adkins, M. Smets, H. Leonhardt, H. Heyn, I. Hellmann, and W. Enard. “Comparative analysis of single-cell RNA sequencing methods”. In: *Molecular cell* 65.4 (2017), pp. 631–643.

[ZC20] C. Zuo and L. Chen. “Deep-joint-learning analysis model of single cell transcriptome and open chromatin accessibility data”. In: *Briefings in Bioinformatics* (2020).