

Master thesis : Gene regulatory network inference using deep learning-based feature scores

Auteur : Vilour, Lucas

Promoteur(s) : Huynh-Thu, Vân Anh; Geurts, Pierre

Faculté : Faculté des Sciences appliquées

Diplôme : Master en ingénieur civil en informatique, à finalité spécialisée en "intelligent systems"

Année académique : 2025-2026

URI/URL : <http://hdl.handle.net/2268.2/26967>

Avertissement à l'attention des usagers :

Tous les documents placés en accès ouvert sur le site le site MatheO sont protégés par le droit d'auteur. Conformément aux principes énoncés par la "Budapest Open Access Initiative"(BOAI, 2002), l'utilisateur du site peut lire, télécharger, copier, transmettre, imprimer, chercher ou faire un lien vers le texte intégral de ces documents, les disséquer pour les indexer, s'en servir de données pour un logiciel, ou s'en servir à toute autre fin légale (ou prévue par la réglementation relative au droit d'auteur). Toute utilisation du document à des fins commerciales est strictement interdite.

Par ailleurs, l'utilisateur s'engage à respecter les droits moraux de l'auteur, principalement le droit à l'intégrité de l'oeuvre et le droit de paternité et ce dans toute utilisation que l'utilisateur entreprend. Ainsi, à titre d'exemple, lorsqu'il reproduira un document par extrait ou dans son intégralité, l'utilisateur citera de manière complète les sources telles que mentionnées ci-dessus. Toute utilisation non explicitement autorisée ci-avant (telle que par exemple, la modification du document ou son résumé) nécessite l'autorisation préalable et expresse des auteurs ou de leurs ayants droit.

Gene regulatory network inference using deep learning-based feature scores

VILOUR Lucas

Thesis presented to obtain the degree of:
Master of Science in Computer Science and Engineering

Thesis supervisors:
HUYNH-THU Vân Anh, GEURTS Pierre

Academic year: **2025 - 2026**

Acknowledgments

First of all, I would like to thank my supervisors, Vân Anh Huynh-Thu and Pierre Geurts, for their guidance and support during this study, but also for their high availability and their help in the writing of this thesis.

I would also like to thank my friends for their help, for their support during my academic journey and especially this year.

I would like to thank my family for their unconditional support in both the best and the worst moments and for helping me make the choices that led me to the writing of this thesis.

Finally, I would also like to thank the members of the jury for their time and their interest in this work.

Abstract

Reconstructing gene regulatory networks (GRNs) from expression data remains a central challenge in computational biology. While tree-based ensemble methods such as GENIE3 and GRNBoost2 have long been the gold standard, deep learning offers the potential to capture more complex regulatory patterns, provided its predictions can be made interpretable. This thesis investigates whether feature-importance mechanisms embedded in deep learning architectures can be exploited for GRN inference. Two methods embedding feature scores in their architectures, LassoNet and TabNet, are integrated into a GENIE3-like per-gene regression framework, and a stability-selection procedure is used for LassoNet to aggregate feature-importance scores across resampled regularization paths into more robust interaction ranking. These methods are benchmarked against GENIE3 and GRNBoost2 but also against other deep learning-based methods such as DeepSEM or scGeneRAI on the DREAM and BEELINE datasets using AUPRC and Early Precision Ratio. Results show tree-based ensembles retain a clear advantage on synthetic and smaller curated networks, while somehow trailing DeepSEM in experimental and bigger networks. No single method dominates uniformly across biological contexts and network scales.

Contents

Introduction	1
1 State of the art	3
1.1 Machine-learning-based methods	3
1.1.1 Decision Tree (DT)	3
1.1.2 Random Forest (RF)	4
1.1.3 Extra-Trees (ET)	5
1.1.4 Boosting	5
1.2 Deep learning	6
1.2.1 Multi-Layer Perceptron (MLP)	6
1.2.2 Convolutional Neural Networks (CNNs)	7
1.2.3 Autoencoders and Variational Autoencoders (VAEs)	8
1.2.4 Explainable AI and Layer-wise Relevance Propagation	10
2 Methods and datasets	12
2.1 Architectures	12
2.1.1 LassoNet	12
2.1.2 TabNet	14
2.2 Methodology	17
2.2.1 Network inference using simple Feature Importance	17
2.2.2 Network inference using Stability Selection	17
2.2.3 Benchmarks	21
2.2.4 Performance metrics	22
3 Results and discussions	25
3.1 DREAM4 datasets	25
3.2 DREAM5 dataset	27
3.3 Curated datasets from BEELINE	28
3.3.1 HSC dataset	28
3.3.2 VSC dataset	29
3.3.3 GSD dataset	30
3.3.4 Cross-dataset comparison	31
3.4 Experimental datasets from BEELINE	31
3.4.1 hHep dataset	32
3.4.2 mDC dataset	33
3.4.3 mESC dataset	34
3.4.4 hESC dataset	34
3.4.5 mHSC-E dataset	36
3.4.6 mHSC-GM dataset	36
3.4.7 mHSC-L dataset	38
3.4.8 Cross-dataset comparison	38

TABLE OF CONTENTS

3.5	LassoNet's computation time	39
4	Conclusions and perspectives	42
4.1	Conclusions	42
4.2	Perspectives	44
	Additional content	46
	Code availability	46
	Data availability	46
	Artificial intelligence tools usage	46
	Bibliography	50

Introduction

A Genetic Regulatory Network (GRN) is an interpretable graph structure that indicates a set of intrinsic rules of gene expression in living organisms [1]. It is formally described as the graph $G = (V, E)$ where nodes or vertices V represent individual genes and edges E represent the interactions or relationships between these genes [2]. The graph can be undirected (Gene A and Gene B interact), or directed (Gene A regulates Gene B) [3]. The latter is more expressive and is generally preferred when the algorithm allows it. The goal of GRNs is to have a fully predictive explanation of how information encoded in DNA underlies biological processes. These biological processes are responsible for controlling cell development, regulating cell differentiation, and determining cells responses to the surrounding environment. Deciphering GRNs is therefore crucial to understand the biological mechanisms that drive every biological organism [4], with direct implications for developmental biology and clinical medicine. However, gene expression regulation is inherently complex [5], and reconstructing gene regulatory networks from high-throughput data remains a long-standing challenge [6].

Direct observation of gene interactions in organisms is difficult, but it is feasible to measure element abundance by experimental means, and the expression pattern of these elements often reflects the regulatory role of genes [5]. The advent of microarray technology provided a systematic and comprehensive vehicle for exploring the genome [7] enabling the simultaneous measurement of all RNA transcripts in a cell [8]. This abundance of data made inference or reverse-engineering approaches conceivable. Since the emergence of DNA microarray technology, researchers have proposed methods to reconstruct GRNs by inferring the mechanisms behind transcription control in cells. The subsequent development of RNA-seq, an application of Next-Generation Sequencing (NGS)¹ to detect RNA abundance in samples through reverse transcription and fragmentation further increased data throughput. The current state of the art in sequencing is single-cell RNA-sequencing (scRNA-seq), which achieved unprecedented resolution. While bulk RNA-seq only measured the population-average expression level of all cells in a tissue, masking the unique molecular properties of rare cell states [1, 9], scRNA-seq resolves individual cells, making it especially promising for computing GRNs because it does not obscure biological signals by averaging [10].

However, scRNA-seq data introduce their own set of difficulties. These challenges range from substantial cellular heterogeneity and cell-to-cell variation in sequencing depth to high sparsity caused by technical dropouts² and cell-cycle-related effects [10]. While traditional bulk transcriptomic data provide a higher signal-to-noise ratio, they are typically high-dimensional and significantly undersampled [1, 8]. scRNA-seq addresses the sample-size limitations of bulk data but introduces dropouts and a lower signal-to-noise ratio at the individual cell level [9] and raises serious computational scalability concerns, as the calculation of gene interactions often scales quadratically with the number of genes. However, even with the increase in observations provided by scRNA-seq, biological data volume remains limited compared to fields like Computer Vision or Natural Language Processing, where deep learning is the gold standard. Taken together, these properties necessitate the development of specialized architectures capable of robust representation learning despite the inherent noise and relative sparsity of biological datasets [11].

From a data-representation standpoint, gene expression matrices are a specific instance of *tabular*

¹An advance in sequencing technology, which greatly improves throughput by parallelizing sequence reactions. [5]

²Some genes show zero expression due to low capture efficiency or amplification bias [1].

data, i.e. data organized as a two-dimensional array in which rows correspond to independent observations and columns correspond to descriptive features. In the case of microarray, bulk RNA-seq, or scRNA-seq experiments, rows correspond to samples or individual cells and columns correspond to genes, with each entry quantifying the expression level of a gene in a given sample or in a given cell [12,13]. This structure differs fundamentally from the data types that have driven the recent success of deep learning: unlike images, which exhibit strong local spatial correlation between neighbouring pixels, or natural language, which exhibits sequential and syntactic dependencies between tokens, tabular gene expression data has no intrinsic ordering among its columns, and permuting genes does not alter the biological meaning of a sample. Gene expression data is also characterized by a specific mix of homogeneity and heterogeneity. At the feature level, it is largely *homogeneous*: every column encodes the same type of quantitative measurement (an expression value), unlike typical tabular datasets that combine categorical, ordinal, and continuous variables. At the sample level, however, it is often strongly *heterogeneous*, since observations may originate from different tissues, cell types, disease states, or experimental batches, introducing biological and technical variability that must be accounted for during preprocessing [1,10]. Gene expression tables are further distinguished by their extreme dimensionality: the number of genes (columns) typically reaches tens of thousands, while the number of samples (rows) rarely exceeds a few thousand, even for scRNA-seq datasets, yielding the classical high-dimensional, low-sample-size regime [14]. These properties place gene expression data among the most challenging instances of tabular data for deep learning, and directly motivate the search for architectures able to exploit, rather than ignore, this structure.

Deep learning has emerged as the leading methodology across many data-intensive fields since 2012, and its application to GRN inference is particularly compelling. Unlike classical machine learning approaches that rely on heavy feature engineering, deep learning adopts a data-driven approach in which lower layers learn feature representations and higher layers summarize the representations from the lower layers and produce the inference result [11]. This makes deep learning especially suited to omics, where it can discover regulatory patterns in biological sequences that would otherwise remain hidden. However, deep learning models are by nature black boxes: they yield predictions without transparent explanations of the underlying reasoning. In the context of GRN inference, this poses a fundamental problem. GRNs are themselves interpretability tools, structured, human-readable representations of regulatory logic. The goal of GRNs, which is to gain explanation of the information encoded in DNA, motivates the use of Explainable Artificial Intelligence (XAI) techniques or of architectures embedding interpretability mechanisms, which aim to make the reasoning of deep learning models transparent and biologically meaningful.

Understanding the advantages and limitations of different network inference methods is critical for their effective application in a given biological context [6]. Network inference methods have complementary advantages and limitations under different contexts. Given the biological variation among organisms and the experimental variation among gene-expression data sets, it is difficult to determine beforehand which methods will perform optimally for reconstructing an unknown regulatory network. Methods have "inherent biases" toward certain connectivity patterns, and no single inference method performs optimally across all biological contexts or network motifs. Furthermore, while integrated community networks were the gold standard for bulk transcriptomics, recent single-cell benchmarks suggest that ensembles do not yet systematically outperform the best individual methods, making the search for highly accurate, individual deep learning architectures even more critical [10]. The development of new, diverse deep learning methodologies is therefore essential to capture the full complexity of the regulatory landscape.

In this thesis, we benchmark multiple deep learning architectures embedding feature score mechanisms or using XAI for GRN inference on single-cell RNA-sequencing and bulk sequencing data.

Chapter 1

State of the art

"Reverse-engineering", or inferring gene networks is the process of identifying gene interactions from experimental data through computational analysis [3]. Gene expression data from microarrays, bulk RNA-sequencing, or single-cell RNA-sequencing are typically used for this purpose. A lot of computational methods have been proposed to infer GRNs but their ideal scenario can differ a lot [5]. In this chapter, we will first focus on machine-learning-based methods that have been the state of the art for many years and that are still widely used in GRN inference. Then, deep learning method implemented especially for inferring GRN will be explored.

1.1 Machine-learning-based methods

As stated in the name, this family of methods uses machine learning algorithms to infer the regulatory network. These kinds of algorithms can produce directed regulatory network by identifying the direction of the regulation, and some of them provide ways to measure feature importance that can be exploited for this purpose [5]. For this tabular data, tree-based machine learning methods have been the gold standard outperforming deep learning and other machine learning architectures for a long time [15]. In fact, tree-based-ensemble methods are particularly well-suited for this kind of data because such methods do not need any assumption about the target function. A function can thus be learned independently of its nature even non-linearity can be tackled. Tree-based methods also work in the presence of a large number of features and since tabular data often provides more features than examples, this once again shows that tree-based method seems to be perfectly suited. Moreover, tree-based methods can deal with interacting features which is particularly interesting in the task of gene regulatory network inference. In general, for this task, machine learning-based methods divide the problem into subproblems. Each gene has its subproblem that tries to infer its regulator based on the expression data [4]. Multiple algorithms are commonly used in the task of reverse-engineering gene regulatory networks such as Random Forest (RF), Extra-Trees (ET), or Boosting [5].

1.1.1 Decision Tree (DT)

To define the Random Forest, one must first define a decision or regression tree. Trees divide the samples based on decision made over features. A decision tree is a succession of decision leading to a "terminal" node. Each node starting from the "root" will be divided into two child nodes, one to the left and one to the right. This division is made upon a particular variable. The split is chosen based on the minimisation of the criterion called Q_{split} . This criterion is defined by

$$Q_{\text{split}} = n_L Q_L + n_R Q_R,$$

where Q_L and Q_R are the criteria for the two candidates descendants and n_L and n_R are their sample sizes. The forms of these criteria are based on the performed task. For a regression, the criterion is the

following:

$$Q = \frac{1}{n} \sum_{i=1}^n (y_i - \bar{y})^2,$$

where $y_1, \dots, y_n$ are the output value at the node and $\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$ is the predicted value at the node. This criterion gives a measure of "goodness of fit", with large values representing poor fit. If the task is a classification, the criterion is of the form

$$Q = \sum_{k \neq k'}^K \hat{p}_k \hat{p}_{k'},$$

where there are K classes denoted $1, \dots, K$ and $\hat{p}_k$ is the proportion of class k observations in the node:

$$\hat{p}_k = \frac{1}{n} \sum_{i=1}^n I(y_i = k).$$

This criterion gives a measure "purity", with large values representing impure node.

Then the evaluation of the split variable on a sample will decide which of the two created nodes will inherit this particular sample. Finally, when all remaining nodes are "final" nodes, the decision tree is complete. The evaluation of a new sample will follow each decision step upon a "terminal" node. Let l denote the terminal node and let $y_{l_1}, \dots, y_{l_n}$ denote the response values of the training data in node l . The predicted value of the sample is either $\hat{h}(x) = \bar{y}_l = \frac{1}{n} \sum_{i=1}^n y_{l_i}$ for regression, or $\hat{h}(x) = \arg \max_y \sum_{i=1}^n I(y_{l_i} = y)$ for classification, where $I(y_{l_i} = y) = 1$ if $y_{l_i} = y$ and 0 otherwise [16].

1.1.2 Random Forest (RF)

A Random Forest is an ensemble method based on the aforementioned decision tree. The random forest builds a collection of trees, each grown using a bootstrap sample from the original data. A second particularity of these trees is their decision. When splitting a node, the best split is not found over the whole set of features but over a randomly selected subset. These particularities induce randomness such that each tree can learn different patterns. Finally, the output of the random forest is a voting in a classification task and an averaging in a regression task [16].

GENIE3 [4] The first machine learning-based GRN inference algorithm studied in this thesis is the well-known GENIE3 algorithm. This algorithm is mainly based on the feature selection facility of the Random Forest. GENIE3 decomposes the GRN inference problem into d independent regression subproblems, one per target gene, where each gene's expression is predicted from the expression of all other candidate regulators. Each subproblem is solved by a Random Forest, and the variable importance measures extracted from the learned forest are used to assign a confidence score $w_{i,j}$ representing the predicted strength of the regulatory effect of each candidate regulator i on the target gene j . Aggregating these scores across all target genes yields a weighted adjacency matrix over all gene pairs, from which the regulatory network is derived [4].

Tree-based methods allow the computation of multiple importance measures in order to rank the features. The measure used in the GENIE3 algorithm is a measure that computes the total reduction of the variance of the output variable due to the split at each node $\mathcal{N}$. This reduction is given by

$$I(\mathcal{N}) = \#\mathcal{S}\text{Var}(\mathcal{S}) - \#\mathcal{S}_t\text{Var}(\mathcal{S}_t) - \#\mathcal{S}_f\text{Var}(\mathcal{S}_f), \quad (1.1)$$

where $\mathcal{S}$ denotes the set of samples that reach node $\mathcal{N}$, $\mathcal{S}_t$ (resp. $\mathcal{S}_f$) denotes its subset for which the test is true (resp. false), $\text{Var}(\cdot)$ is the variance of the output variable in a subset, and $\#$ denotes the cardinality of a set of samples. To compute the importance of a feature in a tree, one sums this measure for all nodes where this feature has been used to make the split. Then one averages this

feature importance measure over every tree in the forest to get the final gene importance measure [4]. It is worth noting, however, that despite being frequently described as interpretable, ensemble methods like Random Forest operate as aggregates of hundreds of individual trees, and the resulting feature importance scores are statistical summaries rather than direct causal statements. A high importance score for gene i in the regression of gene j indicates predictive relevance, not necessarily a direct regulatory relationship. Indirect influences mediated through correlated genes can inflate importance scores in ways that are difficult to disentangle. This limitation motivates the search for methods that perform more principled feature selection, explicitly identifying a sparse set of true regulators rather than ranking all candidates by an aggregated proxy score [5, 17, 18]. However, despite this limitation, this algorithm’s performance was validated in DREAM4 and DREAM5 challenges, discussed further in Chapter 2 [4, 10].

1.1.3 Extra-Trees (ET)

Extra-Trees is another tree-based ensemble algorithm, building on the decision and regression tree framework defined in Section 1.1.1. Like Random Forest, Extra-Trees constructs an ensemble of multiple trees. However, it differs in two important ways. First, each tree is trained on the whole learning sample rather than on a bootstrap replicate, which aims to reduce bias. Second, and more distinctively, Extra-Trees introduces randomization at the splitting stage: rather than searching for the optimal split among a subset of K randomly selected features as Random Forest does, Extra-Trees additionally randomizes the split threshold itself, selecting it uniformly at random within the range of each candidate feature and retaining the best among these fully random splits [19]. This double source of randomization results in a stronger variance reduction through averaging. While Random Forest tends to achieve lower bias on the training data, Extra-Trees generally exhibits stronger randomization, which can translate into better generalization on unseen data [5].

GENIE3 [4] GENIE3 has also been applied using the Extra-Trees algorithm. The two versions of GENIE3 gave comparable results, with however slightly better results obtained with Random Forest.

1.1.4 Boosting

While GENIE3, both the Random Forest version and the Extra-Trees version, represented a significant step forward in GRN inference quality, their computational cost becomes a practical bottleneck as dataset size grows. The rise of single-cell RNA sequencing technologies, capable of profiling the transcriptional state of tens of thousands of individual cells simultaneously, dramatically increased both the number of samples and the number of candidate regulators that inference algorithms must handle. A Random Forest or Extra-Trees model trained independently for each target gene scales poorly under these conditions, making genome-wide inference on large scRNA-seq datasets prohibitively slow. This scalability challenge motivated the development of new algorithms which replaced the forest-based learner with a gradient boosting framework that achieves comparable network recovery performance at a fraction of the computational cost, making high quality GRN inference tractable at single-cell scale [20].

GRNBoost2 [20] GRNBoost2 keeps the same decomposition strategy as GENIE3: the GRN inference problem is split into d independent regression subproblems, one per target gene, and a feature importance score is extracted from the fitted model to build the weighted adjacency matrix over candidate regulator-target pairs. What changes is the underlying learner. Instead of an ensemble of independently grown trees whose predictions are averaged, GRNBoost2 relies on a stochastic Gradient Boosting Machine (GBM) regressor [21], in which trees are grown sequentially, each one fitted on the residual errors left by the previous trees, so that the ensemble is built as an additive expansion rather than as a simple average. This sequential, error-connecting construction lets GRNBoost2 reach

competitive predictive performance with markedly fewer and shallower trees than GENIE3, and the training procedure is further regularized through early stopping, which halts the addition of new trees once the validation error stops improving, limiting overfitting and reducing training time. Feature importances are extracted from the boosted model in the same way as GENIE3 on Random Forest, by accumulating the variance-reduction criterion $I(\mathcal{N})$ (Equation 1.1) over every node where a given regulator was selected for splitting, and then aggregating this score across all target-gene subproblems. Because each subproblem requires substantially fewer trees to converge, GRNBoost2 offers large speed gains over GENIE3 while achieving comparable accuracy on the DREAM5 benchmark, which makes it particularly well suited to the sample sizes and regulator counts encountered in single-cell RNA-seq datasets [20].

1.2 Deep learning

The advent of single-cell RNA sequencing (scRNA-seq) provides massive datasets that traditional bulk-expression methods struggle to process due to high dimensionality, noise and data sparsity. This is the primary driver for deep learning in this field [1]. In fact, in contrast to "shallow learning" algorithms, deep learning performs inference tasks with a deep and hierarchical architecture where lower layers learn the representations while higher layers summarize these representations and produce the result of the inference. This makes deep learning especially useful in omics. These kind of algorithms could discover patterns in biological sequences or measurements that are yet unknown.

Moreover, deep learning has achieved remarkable success in various other fields such as computer vision or natural language processing thanks to data-driven approaches [11]. This success is in part due to the learning capability, generalizability, and computational efficiency and more importantly its representation learning ability. In addition, there is a similarity between biological sequences and natural languages [22]. For example, some sequence motifs serve as regulatory codes, and the interactions between such codes serve as the regulatory grammar.

Deep learning is a general term for machine-learning algorithms that are made up of deep neural networks (DNNs) [11]. They consist of multiple artificial neural network layers which are data-processing units that serve as non-linear transformation functions of their inputs. As each layer takes as input the result from the previous layer, the transformation becomes increasingly complex when the number of layers increases. These models are then trained by fitting themselves on the training data. This is done thanks to the optimization of an objective function generally called the "loss function" using gradient descent.

The following paragraphs will talk about the different families of deep neural networks and whether or not promising architectures based on these families have been developed for the application of gene regulatory network inference. We start with the simplest DNN, the MLP which underlies most of the more specialized architectures described below.

1.2.1 Multi-Layer Perceptron (MLP)

Multi-Layer Perceptron or MLP, also called the fully connected feedforward network, is the most popular family of deep neural networks [11]. The multi-layer perceptron is a succession of layers composed of multiple neurons. Each neuron takes as input a weighted sum of the outputs of the previous layer. The first layer is called the input layer and is fed with the input to be processed while the last layer is called the output layer and gives us the prediction over the input data. An example of a Multi-Layer Perceptron is depicted in Figure 1.1. The multi-layer perceptron directly takes the data values as input and subsequently processes this input in one or more hidden layers. The output layer finally summarizes the processed informations of the earlier hidden layers to produce the final prediction.

There are no promising architectures using the plain multi-layer perceptron. However, this is the base block for more complex architectures that will be talked about later.

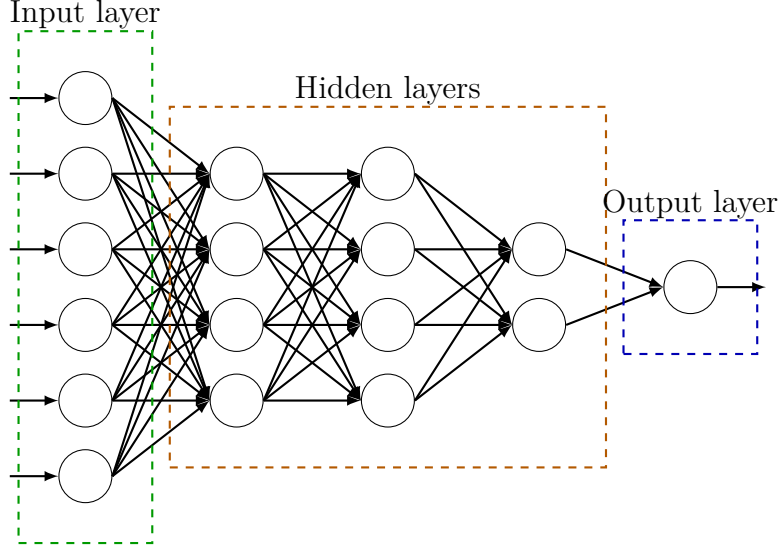


Figure 1.1: **Multi-Layer Perceptron** with three hidden layers.

1.2.2 Convolutional Neural Networks (CNNs)

After the success in image recognition, convolutional neural networks or CNNs have been tested on raw biological sequence data such as DNA, RNA, or protein sequences. CNNs have proven to be very useful in these types of data [11].

A CNN is generally used on a matrix and is (mainly) composed of convolutional layers. A convolutional layer is defined by a set of kernel $\mathbf{u}_k$ of size $C \times h \times w$ where $k \in K$. This kernel applies a 2d convolution operation to the input tensor $\mathbf{x}$ of size $C \times H \times W$ to produce feature maps o_k with $k \in K$ [23]. Mathematically, this 2d convolution with C channels can be defined for the 3d input tensor $\mathbf{x} \in \mathbb{R}^{C \times H \times W}$ and the 3d convolutional kernel $\mathbf{u} \in \mathbb{R}^{C \times h \times w}$ where $h \leq H$ and $w \leq W$ (if there is no padding), as

$$(\mathbf{x} \otimes \mathbf{u})[j, i] = \sum_{c=0}^{C-1} \sum_{n=0}^{h-1} \sum_{m=0}^{w-1} \mathbf{x}_{c, n+j, m+i} \mathbf{u}_{c, n, m},$$

C denotes the number of channels, $H \times W$ denotes the spatial dimensions and $h \times w$ denotes the kernel size. An example of a convolutional neural network is depicted on the Figure 1.2. For a more readable figure, C , H , h and K have all been set to 1.

There are multiple convolutional neural network-based GRN inference algorithms such as Convolutional Neural Network for Coexpression (CNNC) [24]. However, these algorithms and in particular CNNC are mainly supervised which is not the scope of this thesis. Indeed, comparing the performance of supervised and unsupervised methods is methodologically problematic. This motivates restricting the scope of this thesis to the latter. *Supervised* methods such as CNNC are trained on a subset of known interactions and evaluated on a held-out subset of the same reference network, meaning that part of the ground-truth is, by construction, already available to the model during training. *Unsupervised* methods, in contrast, receive no such information and must infer the entire network structure from expression data alone, with the reference network used only post hoc for evaluation. As a result, a direct comparison of performance metrics between the two paradigms tends to favor supervised methods unfairly, since they are partially evaluated on a task closer to interpolation within a known network rather than genuine discovery [10]. This discrepancy makes benchmark results across paradigms difficult to interpret on equal footing, and is a further reason why this thesis concentrates on comparing unsupervised approaches among themselves rather than against supervised baselines.

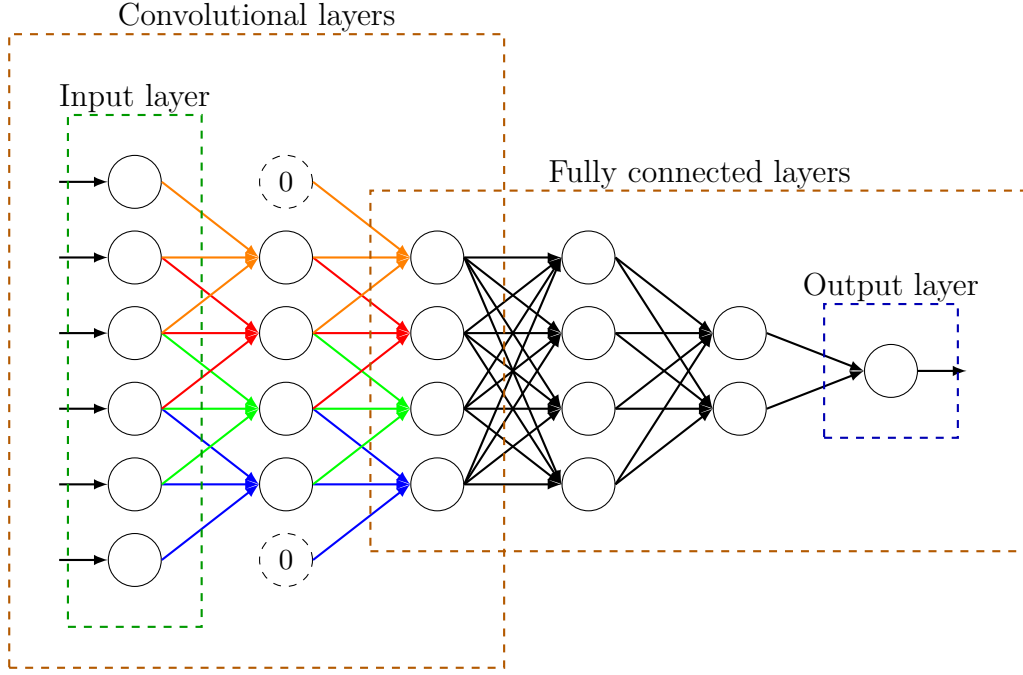


Figure 1.2: **Convolutional Neural Network** with the input layer $\mathbf{x} \in \mathbb{R}^{1 \times 1 \times 6}$ and the convolutional kernel $\mathbf{u} \in \mathbb{R}^{1 \times 1 \times 3}$. The first convolutional layer is a simple one without padding but the second one has a padding of one represented by the two dashed neurons which value is set to 0. There is also 3 fully connected layers leading to the output layer.

1.2.3 Autoencoders and Variational Autoencoders (VAEs)

An autoencoder (AE) is a type of neural network designed to learn a compressed, lower dimensional representation of its input in an unsupervised manner [25–27]. Unlike the MLP and CNN architectures described above, which are typically trained to map an input to an external label, an autoencoder is trained to reconstruct its own input. It is composed of two parts: an *encoder* (f), which maps the input $\mathbf{x}$ to a lower-dimensional latent representation $\mathbf{z}$, and a *decoder* (g), which attempts to reconstruct the original input $\hat{\mathbf{x}}$ from $\mathbf{z}$. Both the encoder and the decoder are generally implemented as multi-layer perceptrons. Encoders composed of convolutional layers can also be combined with decoders composed of transposed convolutional layers. The network is trained by minimizing a reconstruction loss, typically the mean squared error between the input and its reconstruction,

$$\mathcal{L}_{AE}(\mathbf{x}, \hat{\mathbf{x}}) = \|\mathbf{x} - \hat{\mathbf{x}}\|^2,$$

where $\hat{\mathbf{x}}$ is the result of input $\mathbf{x}$ after evaluation by the autoencoder. More formally given the encoder f and the decoder g , the loss is given by

$$\mathcal{L}_{AE} = \|\mathbf{x} - g \circ f(\mathbf{x})\|^2.$$

Because the latent space $\mathbf{z}$ is of much lower dimension than the input, the network is forced to learn a compact representation that captures the most salient features of the data, which makes autoencoders particularly attractive for high-dimensional and noisy data such as scRNA-seq counts.

A variational autoencoder (VAE) extends this idea by treating the latent representation probabilistically rather than deterministically [28, 29]. Instead of encoding an input as a single point $\mathbf{z}$, the encoder outputs the parameters (mean μ and variance σ^2) of a probability distribution $q_\phi(\mathbf{z} | \mathbf{x})$, typically assumed Gaussian, from which a latent vector is sampled before being passed to the decoder. This turns the autoencoder into a generative model: once trained, new data points can be generated by sampling $\mathbf{z}$ directly from a prior distribution $p_\theta(\mathbf{z})$ (usually $\mathcal{N}(0, I)$) and decoding it. To

make this sampling step compatible with gradient-based training, the reparameterization trick is used, whereby $\mathbf{z} = \mu + \sigma \odot \epsilon$ with $\epsilon \sim \mathcal{N}(0, I)$ [28]. The VAE is trained by maximizing the evidence lower bound (ELBO), which combines the reconstruction term with a regularization term that encourages the approximate posterior $q_\phi(\mathbf{z} | \mathbf{x})$ to remain close to the prior $p_\theta(\mathbf{z})$:

$$\mathcal{L}_{VAE} = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log p_\theta(\mathbf{x} | \mathbf{z})] - D_{KL}(q_\phi(\mathbf{z} | \mathbf{x}) \parallel p_\theta(\mathbf{z})),$$

where ϕ and θ denote the parameters of the encoder and decoder networks respectively, and D_{KL} denotes the Kullback-Leibler divergence, a measure of how much one probability distribution differs from another [30]. Intuitively, the first term encourages accurate reconstruction of the input while the second regularizes the latent space by keeping the encoder’s output distribution close to the prior $p_\theta(\mathbf{z})$, which prevents the latent space from fragmenting into disconnected regions. This is what gives the VAE its generative capability. This probabilistic formulation gives VAEs two properties that are particularly relevant for scRNA-seq analysis: a smoother, more structured latent space than that of a plain AE, and an explicit generative mechanism that naturally accounts for the noise and dropout events characteristic of single-cell measurements [31, 32].

Autoencoders and VAEs have accordingly been used for dimensionality reduction, denoising, and visualization of scRNA-seq data [33]. However, in their standard form, no architecture explicitly models directed relationships between genes: the latent space captures a compressed representation of the overall expression profile of a cell, but does not by itself indicate which genes regulate which. This limitation motivates architectures that embed an explicit model of gene-gene regulatory structure directly into the encoder or decoder. This approach is adopted by DeepSEM [32] described in the next paragraph.

DeepSEM [32] DeepSEM adapts the variational autoencoder framework described above to explicitly infer a gene regulatory network, rather than only a generic latent representation of the data. It builds on the linear structural equation model (SEM), a classical formalism from causal discovery in which each observed variable is expressed as a weighted linear combination of the others, plus an independent noise term. For a set of d genes, this can be written as

$$\mathbf{x} = \mathbf{A}^T \mathbf{x} + \mathbf{z},$$

where $\mathbf{x}$ is the vector of observed gene expression values, $\mathbf{z}$ is a vector of independent noise terms following a Gaussian distribution, and $\mathbf{A} \in \mathbb{R}^{d \times d}$ is a weighted adjacency matrix in which entry A_{ij} represents the strength of the regulatory effect of gene i on gene j . Rearranging this expression gives

$$\mathbf{x} = (\mathbf{I} - \mathbf{A}^T)^{-1} \mathbf{z},$$

meaning that $\mathbf{A}$ fully determines how the noise variables $\mathbf{z}$ are transformed into the observed expression profile $\mathbf{x}$: inferring $\mathbf{A}$ is therefore equivalent to inferring the gene regulatory network itself [34].

DeepSEM embeds this transformation directly inside a VAE. Instead of using a standard prior $p_\theta(\mathbf{z})$ and a neural network decoder as described in the previous section, the decoder is replaced by the SEM transformation $(\mathbf{I} - \mathbf{A}^T)^{-1}$ applied to a latent noise vector $\mathbf{z}$ sampled by the encoder, with $\mathbf{A}$ treated as an additional set of learnable parameters shared across all cells. The encoder, conversely, takes the expression profile of a gene as input and produces the parameters of the approximate posterior $q_\phi(\mathbf{z} | \mathbf{x})$, following the same reparameterization scheme as a standard VAE. The full model is trained by maximizing a modified ELBO, weighted by a coefficient β on the KL term (a β -VAE [35]) to better balance reconstruction accuracy against the regularization of the latent space, and augmented with an ℓ_1 sparsity penalty on $\mathbf{A}$, which reflects the biological prior that regulatory networks are sparse (i.e., that most gene pairs do not directly regulate one another). Concretely, this modifies the original ELBO from the previous section by introducing a weight β on the KL term:

$$\mathcal{L}_{VAE}(\beta) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log p_\theta(\mathbf{x} | \mathbf{z})] - \beta D_{KL}(q_\phi(\mathbf{z} | \mathbf{x}) \parallel p_\theta(\mathbf{z})),$$

which reduces to the standard VAE objective $\mathcal{L}_{VAE}$ when $\beta = 1$. Increasing β places more emphasis on matching the prior, encouraging a smoother and more disentangled latent space at the cost of reconstruction accuracy, while decreasing it favors accurate reconstruction over regularization [35]. We can thus define the modified ELBO to minimize:

$$\mathcal{L}_{DeepSEM} = -\mathcal{L}_{VAE}(\beta) + \lambda \|\mathbf{A}\|,$$

where λ controls the strength of the sparsity constraint. Notably, and unlike other neural SEM-based causal discovery methods such as DAG-GNN [34], DeepSEM does not enforce an explicit acyclicity constraint on $\mathbf{A}$, which simplifies training at the cost of allowing the inferred network to contain cycles. This is a reasonable trade-off in the context of gene regulation, where feedback loops are biologically common.

Once trained, the learned matrix $\mathbf{A}$ directly provides a weighted, directed gene regulatory network, while the latent space learned by the encoder simultaneously offers a denoised, lower-dimensional representation of the scRNA-seq data that can be used for downstream tasks such as clustering or visualization, making DeepSEM an all-in-one framework for single-cell data analysis rather than a GRN-inference method alone [32]. DeepSEM was shown to outperform earlier unsupervised GRN inference methods such as GENIE3, GRNBoost2, PIDC and SCODE on the BEELINE benchmark [10]. However, subsequent work has reported that its performance can vary substantially across independent training runs, an instability attributed to the interaction between the ℓ_1 sparsity term and the optimization dynamics of $\mathbf{A}$ early in training [36].

1.2.4 Explainable AI and Layer-wise Relevance Propagation

As deep neural networks have grown deeper and more complex, understanding *why* a given model produces a particular prediction has become increasingly difficult, since the transformation from input to output results from the composition of many non-linear layers. Explainable AI (XAI) refers to a family of methods designed to address this issue, by attributing a model’s prediction back to its input features, thereby indicating which parts of the input were most responsible for a given output [37]. In the context of gene regulatory network inference, this attribution can be repurposed: if a neural network is trained to predict the expression of one gene from the expression of others, an XAI method can reveal which of the other genes contributed most to that prediction, and can therefore be interpreted as a proxy for a regulatory relationship.

Among the various XAI techniques developed for neural networks, Layer-wise Relevance Propagation (LRP) [38] is a backward decomposition method originally introduced to explain individual predictions of feedforward and convolutional neural networks. Rather than relying on gradients, LRP operates by propagating a *relevance score* backward from the output layer to the input layer. The propagation obeys a conservation property, meaning that the relevance received by a neuron must be redistributed to the lower layer: for layer $l + 1$ propagating relevance back onto layer l ,

$$R_i^{(l)} = \sum_{j \in I(l+1)} \frac{z_{ij}}{\sum_{k \in I(l)} z_{kj}} R_j^{(l+1)},$$

where $I(l)$ is the index set of neurons in layer l , $R_i^{(l)}$ is the relevance assigned to neuron i in layer l , and z_{ij} models the contribution of neuron i to make neuron j relevant in the next layer, typically, computed from the corresponding weight and activation values [37, 38]. This ensures that $\sum_{i \in I(l)} R_i^{(l)} = f(\mathbf{x})$ at every layer l , where $f(\mathbf{x})$ is the network’s output for input $\mathbf{x}$: no relevance is created or destroyed during propagation; it is only redistributed among the neurons of the previous layer. Starting from the output neuron of interest, whose relevance is initialized to the network’s predicted value, this process is repeated layer by layer until the input feature has been assigned a relevance score reflecting its contribution to the prediction.

This property makes LRP particularly attractive for GRN inference: applied to a network trained to predict gene expression, it yields relevance scores for each input gene with respect to a given predicted target gene, and these scores can be interpreted as the strength of a directed regulatory relationship between them, as adopted by scGeneRAI, described below.

scGeneRAI [9] scGeneRAI applies the LRP framework described above to infer gene regulatory networks at the resolution of individual cells, rather than only at the level of an entire cell population. The core idea is to train a single multi-layer perceptron to predict the expression of an arbitrary subset of genes from the expression of the remaining genes for a given cell, and then use LRP to attribute this prediction back to the individual input genes.

Concretely, for a dataset of M cells, and d genes, the input to the network for a given cell is its full expression profile, in which a randomly selected subset of genes is masked at each training iteration; the network is trained to reconstruct the expression values of the masked genes from the remaining, unmasked ones. Because the choice of masked genes is randomized across iterations, a single trained network learns to predict any gene from any arbitrary combination of other genes, rather than requiring one dedicated model per target gene. Once trained, LRP is applied for each cell and for each target gene in turn: a random subsample of the genes containing the target gene is masked, the expression of the latter is predicted, and LRP propagates the corresponding relevance score back to every other gene used as input, yielding a relevance value between the target gene and each candidate regulator. This procedure is repeated for many random masks and for every gene taken as target, producing a full matrix of pairwise relevance scores for that cell, which is interpreted as a cell-specific, directed gene regulatory network.

Because this procedure is repeated independently for every cell, scGeneRAI is able to reconstruct a distinct regulatory network for each individual cell in a dataset, rather than a single network averaged over a population, as is the case for most other unsupervised GRN inference methods, including DeepSEM. The authors validated this approach on synthetic scRNA-seq data with known ground-truth networks derived from the BEELINE framework [9].

Chapter 2

Methods and datasets

This chapter presents the methods and data used to address the problem of network inference from tabular data. The two neural network architectures at the core of this work, LassoNet and TabNet, both designed to combine predictive performance with built-in feature selection are first introduced. Then, the methodology developed to translate the feature importance produced by these models into inferred network structure are described. First through a direct feature-importance approach and then through a more robust stability selection framework. Finally, the benchmark methods used for comparison and the performance metrics used to evaluate and compare all approaches are presented.

2.1 Architectures

This section introduces the two neural network architectures used throughout this thesis: LassoNet and TabNet. Both models were selected for their ability to perform automatic feature selection during training, which makes them particularly well suited to the task of network inference, where identifying the most relevant input variables is as important as achieving good predictive performance. We describe the theoretical foundations of each architectures, their main components, and the mechanisms by which they produce feature importance scores.

2.1.1 LassoNet

LassoNet is a feed-forward neural network architecture that incorporates a sparsity-inducing mechanism inspired by Lasso regression, enabling simultaneous learning and feature selection [39]. The architecture, shown in Figure 2.1, is built such that only a subset of the input features are used by the network. This sparsity is enforced through an input-to-output skip-layer connection: a feature is allowed to have non-zero weight in the network only if its associated skip-layer connection is active.

The LassoNet objective function is the following:

$$\min_{\theta, W} L(\theta, W) + \lambda \|\theta\|_1, \quad (2.1)$$

where $\theta \in \mathbb{R}^d$ denotes the weights in the residual layer, W denotes the remaining weights and $L(\theta, W)$ is the empirical loss. It is defined as

$$L(\theta, W) = \frac{1}{n} \sum_{i=1}^n \ell(f_{\theta, W}(\mathbf{x}_i), y_i),$$

where the dataset consists of n samples $\{\mathbf{x}_i, y_i\}_1^n$ and ℓ is a loss function such as squared-error, cross-entropy or misclassification rate.

The sparsification of the network is achieved by enforcing the following constraint during the minimization of the loss:

$$|W_{jk}^{(1)}| \leq M \cdot |\theta_j|, k = 1, \dots, K,$$

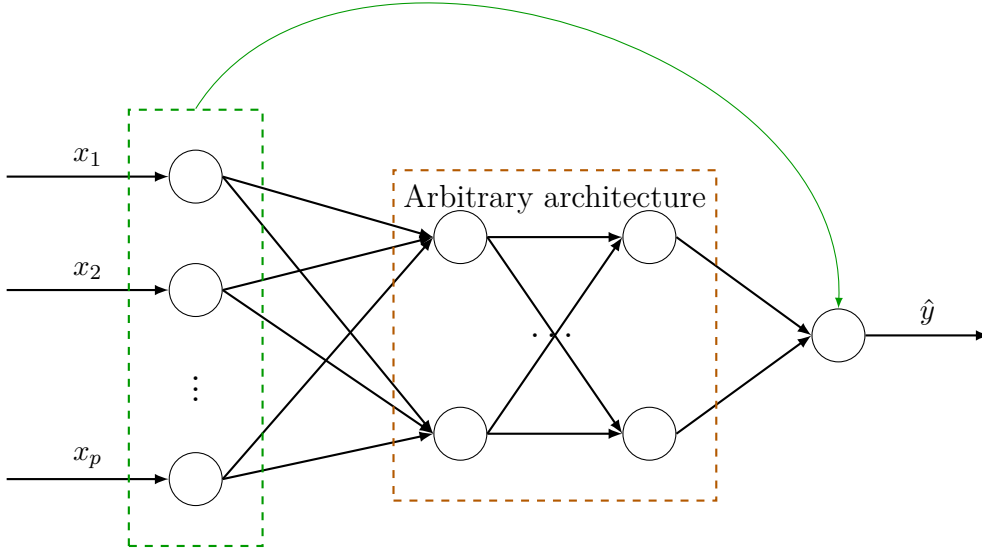


Figure 2.1: **LassoNet architecture** with the single residual connection in green and an arbitrary feed-forward network in black. In this thesis the feed-forward network is a simple MLP. Figure taken from [39].

where $W_j^{(1)}$ denotes the weight for feature j in the first hidden layer, K denotes the number of units in the first hidden layer, and θ_j denotes the residual weight associated with feature j . M is a hyperparameter governing the trade-off between sparsity and model flexibility. This constraint ensures that a feature can only influence the hidden layers if its associated residual weight θ_j is non-zero, effectively tying the activity of each feature across the entire network to a single coefficient.

In the context of gene regulatory network inference, the magnitude $|\theta_j|$ of the residual weight associated with each input gene j serves as a natural feature importance score. A larger value of $|\theta_j|$ indicates that gene j has a stronger influence on the prediction of the target gene's expression, and can therefore be considered a stronger candidate regulator. This score can directly be used within a GENIE3-like framework described in Section 2.2.1 to rank candidate interactions.

The level of sparsity is controlled by λ in Equation 2.1, λ is the ℓ_1 -penalty coefficient: higher values of λ encourage sparser models by penalising non-zero residual weights more strongly. Some regularization paths produced across multiple stability selection experiments are illustrated in Figure 2.4 in which one can see multiple λ values as well as the associated loss and number of selected feature.

All the result for the LassoNet architecture have been computed with the following hyperparameters:

- *hidden_dims*: the number of neurons in each hidden layer has been set to (5, 5), which corresponds to a small network. The default value is (100,) but smaller networks were preferred here to limit computational cost as the GENIE3-like methodology requires training one model per target gene.
- *path_multiplier*: LassoNet computes the regularization path by starting from λ_{start} and multiplying the current λ value by *path_multiplier* at each step. This value has been kept to its default of 1.1, which offers a reasonable trade-off between training speed and path precision. Smaller values such as 1.01 would yield a finer regularization path at the cost of increased computation time.
- *lambda_start*: kept to its default value "auto", which allows the model to select a starting regularization value automatically thanks to a heuristic unpublished by the authors of [39]. A value that is too small results in the model remaining dense for an unnecessarily long time, while

a value that is too large causes the number of selected features to decrease too rapidly, reducing the accuracy of the regularization path.

- *gamma*: kept to its default value of 0.0, meaning no L2 penalty is applied to the network weights. A small non-zero value could improve performance on small datasets. However, some different values for *gamma* have been tested with no significant improvement of the model.
- other standard MLP training parameters (*dropout*, *batch_size*, *patience*, ...) have been kept to their default values.

2.1.2 TabNet

Despite the success of deep learning in many domains, its application to tabular data remains limited, and tree-based ensemble methods continue to dominate most practical applications [40]. To address the limitations of standard deep learning architectures on tabular data, TabNet was proposed as a dedicated neural network architecture. Its design departs from conventional approaches in several important ways.

Rather than requiring feature engineering or preprocessing, TabNet operates directly on raw tabular inputs and is trained end-to-end using gradient descent. Its most distinctive feature is the use of sequential attention, which allows the model to select, at each step of its decision process, which subset of features to focus on. This selection is instance-wise meaning it can differ from one sample to another, allowing the model to adapt its reasoning to the specific characteristics of each input. By concentrating its learning capacity on the most relevant features at each step, the model avoids wasting representational power on uninformative inputs.

This design also confers interpretability properties at two levels. Locally, one can inspect which features were selected for a given prediction and how they were combined, providing insight into individual decisions. Globally, the contribution of each feature can be aggregated across all samples to quantify its overall importance to the trained model. These properties are particularly relevant in the context of gene regulatory network inference as knowing which genes drive a prediction is what makes the gene regulatory network prediction [4].

The key insight behind TabNet is to reproduce a decision-tree-like process within a deep learning framework by handling individual feature selection carefully. In a decision tree, the decision boundary at each node can be seen as a linear combination of a small subset of features, where most coefficients are zero. TabNet is designed to replicate this behaviour while overcoming the limitations of decision trees, through four core design choices.

First, feature selection is sparse and instance-wise, meaning that for each input, only a small and potentially different subset of features is selected, mimicking the local nature of decision tree splits.

Second, the architecture is sequential and multi-step, where each step focuses on a different portion of the decision, allowing the model to decompose complex decisions into simpler sub-decisions in a manner analogous to the successive splits of a decision tree.

Third, the selected features at each step are processed through non-linear transformations, giving the model greater expressive power than a standard decision tree while retaining its interpretable structure.

Fourth, the use of multiple steps combined with higher-dimensional representations introduces a form of implicit ensembling, capturing the variance reduction effect that makes ensemble methods such as Random Forest and Extra-Trees so effective on tabular data.

The overall TabNet architecture for encoding tabular data is shown in Figure 2.2. The input to the network consists of raw numerical features, while categorical features are handled through trainable embeddings. Rather than applying global feature normalization, TabNet relies on batch normalization (BN)¹ applied locally. The same d -dimensional feature vector $\mathbf{f} \in \mathbb{R}^{B \times d}$ is passed to each decision

¹Batch normalization works by re-centering and re-scaling the input values.

step, where B denotes the batch size, and d the number of genes. The encoding process is sequential, consisting of N_{steps} decision steps, where each step receives the processed representation from the previous step to determine which features to attend to, and produces an output that is aggregated into the final decision.

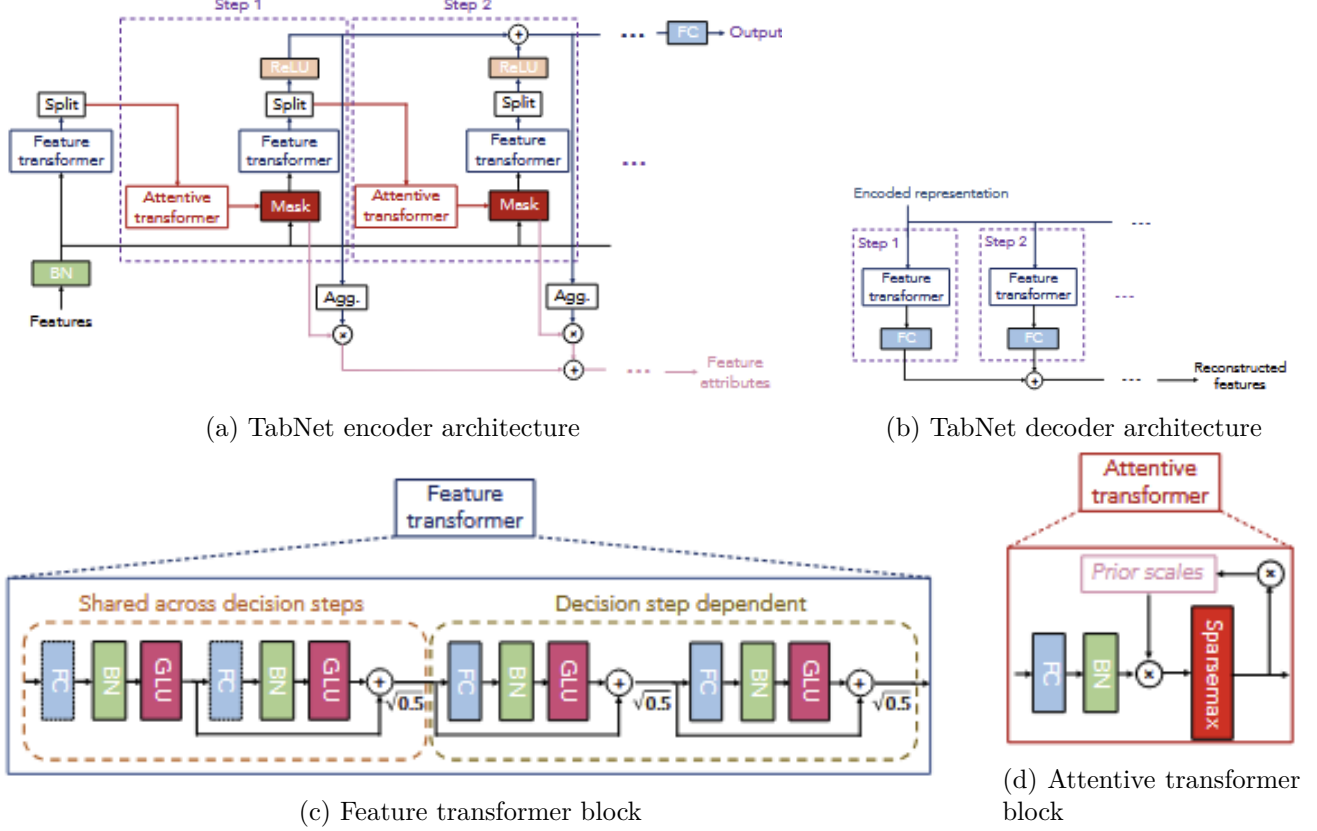


Figure 2.2: Overview of the TabNet architecture, (a) the encoder, (b) the decoder, (c) the feature transformer block, and (d) the attentive transformer block. Each component is described in detail in the text. Figures have been taken from [40].

Each decision step is composed of two main building blocks, illustrated in Figure 2.2. The feature transformer (Figure 2.2c) processes the selected features through a four-layer network, where two layers are shared across all decision steps and two are step-specific, each consisting of a fully-connected layer followed by batch normalization and a Gated Linear Unit (GLU) nonlinearity. Sharing a subset of layers across steps allows the model to retain common representations while adapting to the specific features selected at each step. The attentive transformer (Figure 2.2d) is responsible for feature selection: it produces a sparse mask over the input features using sparsemax normalization, modulated by a prior scale term that keeps track of how much each feature has already been used in previous steps, encouraging the model to spread its attention across different features rather than repeatedly selecting the same ones.

A split block then divides the output of the feature transformer into two parts: one is passed to the attentive transformer of the next step to inform subsequent feature selection, and the other contributes to the overall output representation. The masks produced at each step also serve an interpretability purpose, as they can be inspected individually to understand which features were used at each stage of the decision process, or aggregated across steps to obtain a global measure of feature importance.

Feature selection At each decision step i , a learnable mask $M[i] \in \mathbb{R}^{B \times D}$ is applied multiplicatively to the input features, selecting only the most relevant ones for that step. This sparse selection ensures

that the learning capacity of each step is focused on the most salient features, making the model more parameter efficient. The mask is computed by the attentive transformer using the processed representation from the preceding step $a[i-1]$:

$$M[i] = \text{sparsemax}(P[i-1] \cdot h_i(a[i-1])),$$

where h_i is a trainable function implemented as a fully-connected layer followed by batch normalization, and sparsemax normalization encourages sparsity by projecting onto the probabilistic simplex, ensuring that $\sum_{j=1}^D M[i]_{b,j} = 1$ while pushing most coefficients towards zero.

The prior scale term $P[i]$ controls how much each feature has already been used across previous steps:

$$\begin{aligned} P[i] &= \prod_{j=1}^i (\gamma - M[j]) \\ &= (\gamma - M[i]) \prod_{j=1}^{i-1} (\gamma - M[j]) \\ &= (\gamma - M[i]) \cdot P[i-1], \end{aligned}$$

where γ is a relaxation parameter. When $\gamma = 1$, each feature can only be selected at a single decision step, enforcing strict feature specialisation across steps. As γ increases, features are allowed to appear at multiple steps, providing more flexibility. $P[0]$ is initialised as all ones, reflecting the absence of any prior preference over features at the first step.

To further encourage sparsity, an entropy-based regularization term is added to the training loss:

$$L_{\text{sparse}} = \sum_{i=1}^{N_{\text{steps}}} \sum_{b=1}^B \sum_{j=1}^D \frac{-M_{b,j}[i] \log(M_{b,j}[i] + \epsilon)}{N_{\text{steps}} \cdot B},$$

where ϵ is a small constant added for numerical stability. This term is weighted by a coefficient λ_{sparse} and added to the overall loss, allowing the degree of sparsity to be controlled during training. This sparsity-inducing inductive bias is particularly relevant in the context of gene regulatory network inference, where the number of true regulatory interactions is small relative to the total number of possible gene pairs, and where selecting only a handful of relevant input genes at each step mirrors the sparse nature of the underlying biological regulatory mechanism [4, 10].

Feature processing After feature selection by the attentive transformer, the filtered features are processed by the feature transformer and its output is split into two parts:

$$[d[i], a[i]] = f_i(M[i] \cdot \mathbf{f}),$$

where $\mathbf{f} \in \mathbb{R}^{B \times D}$ still denotes the D -dimensional feature vector, $d[i] \in \mathbb{R}^{B \times N_d}$ contributes to the overall output of the encoder at step i , and $a[i] \in \mathbb{R}^{B \times N_a}$ is passed to the attention transformer of the next step to inform subsequent feature selection. To stabilise training and prevent variance from accumulating across the network, the outputs are normalised by a factor of $\sqrt{0.5}$.

Interpretability The feature selection masks produced by the attentive transformer at each decision step provide a natural interpretability mechanism. For a given sample b and feature j , a mask value of $M_{b,j}[i] = 0$ at step i indicates that feature j has no contribution to the decision at that step. While each decision step applies non-linear transformations to the selected features, the outputs of all steps are combined linearly, which makes it possible to define a meaningful measure of global feature importance.

To aggregate feature importance across all decision steps, a coefficient $\eta_b[i]$ is introduced to weight the relative contribution of each step to the overall decision:

$$\eta_b[i] = \sum_{c=1}^{N_d} \text{ReLU}(d_{b,c}[i]).$$

Intuitively, if all outputs $d_{b,c}[i]$ at a given step are negative, that step contributes nothing to the final decision and its mask should be discounted accordingly. The aggregate feature importance mask is then defined as:

$$M_{\text{agg}-b,j} = \frac{\sum_{i=1}^{N_{\text{steps}}} \eta_b[i] M_{b,j}[i]}{\sum_{j=1}^D \sum_{i=1}^{N_{\text{steps}}} \eta_b[i] M_{b,j}[i]}.$$

This quantity measures, for each sample b , the relative importance of feature j across all decision steps, normalised so that importances sum to one. In the context of gene regulatory network inference, $M_{\text{agg}-b,j}$ can be interpreted as a measure of how much gene j contribute to predicting the expression of the target gene for sample b . By averaging this quantity across all samples, one obtains a single importance score per input gene, which can then be used to rank candidate pairs of genes in a GENIE3-like framework described in Section 2.2.1.

2.2 Methodology

This section describes the methodology developed to infer networks from the feature importance scores produced by LassoNet and TabNet. Two complementary approaches are presented: a straightforward method based directly on the feature importance output by the models, and a more robust approach based on stability selection, which aims to improve the reliability of the inferred connections. The benchmark methods used to contextualize the results, as well as the performance metrics used to evaluate all approaches, are also introduced.

2.2.1 Network inference using simple Feature Importance

The first idea is based on the GENIE3 [4] method that decomposes the problem of gene regulatory network inference involving d genes into d regression problems. Then one exploits a feature importance measure of the method to build the overall pairwise interaction ranking. A summary of the methodology can be found in Figure 2.3. In this thesis, both the LassoNet and TabNet architectures have been tested using this methodology, exploiting the feature importance directly derived from their respective architectures. The results of these approaches will be analysed in Chapter 3. TabNet, despite its attention-based feature selection mechanism designed for tabular data, yielded significantly less promising results than LassoNet in the context of gene regulatory network inference. This may be attributed to the fact that TabNet’s instance-wise feature selection, while powerful for classification and regression tasks on tabular data, does not translate straightforwardly into a reliable gene importance score when aggregated across the d subproblems of the network inference methodology. For this reason, no further methodology was developed around TabNet, and only the results obtained using its built-in feature importance mechanism will be reported in Chapter 3. In contrast, the feature importance derived from LassoNet’s skip-layer connection weights showed more consistent results, motivating the development of a more stable methodology built around it. Indeed, a single LassoNet training’s feature importance score can lack robustness, since it depends on one particular realisation of the training process. This motivated the more stable methodology described in the following section.

2.2.2 Network inference using Stability Selection

Stability selection [41] Stability selection is a general framework designed to enhance and improve the reliability of feature selection methods. It does so by quantifying how consistently each feature is

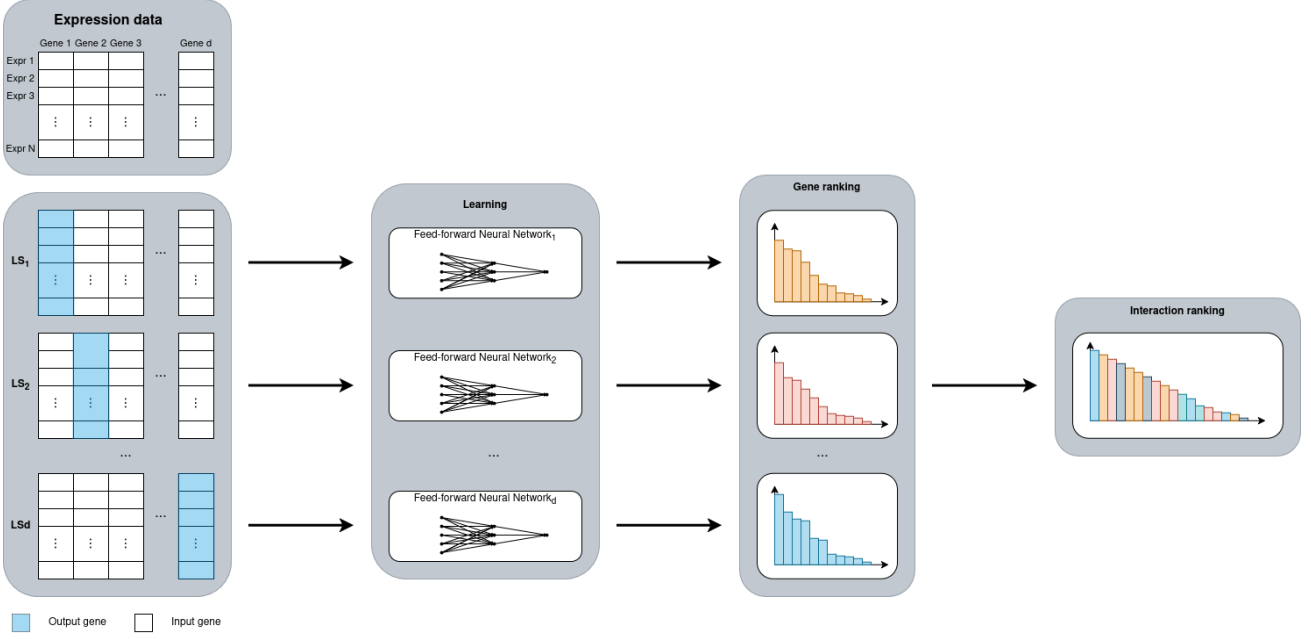


Figure 2.3: **Network inference using Feature Importance.** For each gene $j = 1, \dots, d$, a learning sample LS_j is created with gene j as output and every other genes as input. A local ranking of all genes except j is computed from the learning of LS_j . The d local rankings are then aggregated to get a global ranking of all interactions. The feed-forward neural network can be any architecture that provides a way to compute feature importance. Figure adapted from [4].

selected across different resamples of the data. It builds on the concept of regularization paths, defined as the evolution of each variable's coefficient as the regularization parameter λ varies over a set $\Lambda \subseteq \mathbb{R}^+$:

$$\{\hat{\beta}_k^\lambda; \lambda \in \Lambda, k = 1, \dots, d\},$$

where λ controls the amount of regularization and $k = 1, \dots, d$ indexes the d input features, corresponding to the d genes in the gene regulatory network inference setting and $\hat{\beta}_k^\lambda$ denotes the estimated coefficient of variable k under regularization level λ .

Rather than relying on a single regularization path computed on the full dataset, stability selection repeatedly resamples the data and runs the feature selection method on each subsample. Specifically, for each resample, a random subset $I \subseteq \{1, \dots, n\}$ of size $\lfloor \frac{n}{2} \rfloor$ is drawn becoming an element of the set $\mathcal{I}$, the set of all resamples. On each subset I , a feature selection model is trained, and the selected set $\hat{S}^\lambda(I)$ denotes the set of features whose estimated coefficient is non-zero when the model is trained on subsample I at regularization level λ :

$$\hat{S}^\lambda(I) = \{k \in \{1, \dots, d\} : \hat{\beta}_k^\lambda(I) \neq 0\}.$$

This selected set is computed for each value of $\lambda \in \Lambda$ and the procedure is repeated across many independent resamples. Using half the data at each resample is a deliberate choice made by the authors of [41] that introduces sufficient variability between experiments while retaining enough observations for reliable estimation. The stability path for each variable k is then defined as its selection probability across resamples:

$$\hat{\Pi}_k^\lambda = P^*(k \in \hat{S}^\lambda(I)),$$

for $\lambda \in \Lambda$, where P^* denotes the probability taken over all random subsamples $I \in \mathcal{I}$. A variable that is consistently selected across resamples will have a high selection probability $\hat{\Pi}_k^\lambda$, while a variable whose selection is sensitive to the particular subsample used will have a low and unstable selection probability. This provides a more robust measure of feature importance than a single regularization

path, as it filters out features whose selection is an artefact of the specific data realisation rather than a genuine signal.

Throughout this section, the hat notation, $\hat{\cdot}$, denotes an estimated quantity rather than a true population quantity. Specifically, $\hat{\beta}_k^\lambda$ denotes the estimated coefficient of variable k at regularization level λ as produced by the feature selection algorithm on a finite dataset, as opposed to the true unknown coefficient β_k^λ that would be obtained with infinite data. Similarly, $\hat{\Pi}_k^\lambda$ denotes the empirically estimated selection probability of variable k at regularization level λ , computed from a finite number of resamples, as opposed to the true selection probability Π_k^λ that would be obtained with infinitely many resamples.

In the standard stability selection framework, λ is a fixed external hyperparameter and the set Λ of values tested is defined by the practitioner. However, we proceed as in [39]: starting from λ_{start} , the algorithm automatically explores a sequence of increasing λ values by multiplying λ by *path_multiplier* at each step, defining its own Λ internally. Stability selection is therefore at the level of the resampling only; for each subsample $I \in \mathcal{I}$, LassoNet is run independently and produces its own regularization path over its own set Λ , which may differ slightly between experiments due to the stochastic nature of the training process. As illustrated in Figure 2.4, each stability selection experiment produces a different path, both in terms of the λ values explored and the order in which features are removed as regularization increases. Despite this difference from the standard formulation, the core benefit of stability selection is preserved: by aggregating results across multiple independent resamples, one obtains selection probabilities $\hat{\Pi}_k^\lambda$ that are more robust to the randomness of any individual training run. These probabilities are then used to define the feature importance scores described in the following paragraphs.

Base feature order A first natural idea would be to use the feature rank directly output by LassoNet’s stability selection procedure as a gene importance score. However, this approach does not extend to the network inference setting, where d separate experiments are conducted, one per target gene. Since each rank maps to a different gene depending on the experiment considered, there is no natural way to aggregate these per-experiment ranks into a single global ranking without introducing an arbitrary choice. This limitation motivates the use a continuous importance score rather than a discrete rank, as introduced below.

Feature score using probability Two variants of a continuous feature importance score are introduced, both based on the selection probabilities produced by LassoNet’s stability selection procedure. For each gene $k = 1, \dots, d$ and each subset $I \in \mathcal{I}$, one defines the empirical selection probability:

$$p_{k,I} = \frac{n(S_I^k)}{n(\Lambda_I)}$$

where $n(S_I^k)$ denotes the number of λ values in the resample I for which gene k was selected, and $n(\Lambda_I)$ denotes the total number of λ values tested during that resample, so that $p_{k,I} \in [0, 1]$.

The mean probability score for gene k is then defined as the average of $p_{k,I}$ across all stability selection resamples:

$$\text{MeanProb}_k = \frac{1}{|\mathcal{I}|} \sum_{I \in \mathcal{I}} p_{k,I}. \quad (2.2)$$

where $|\mathcal{I}|$ denotes the number of resamples in the stability selection process. A high value of MeanProb_k indicates that gene k is consistently selected across resamples and regularization levels, suggesting a robust regulatory relationship with the target gene. This quantity is closely related to the selection probability $\hat{\Pi}_k^\lambda$ introduced in the stability selection framework of Section 2.2.2. While $\hat{\Pi}_k^\lambda$ measures the probability of gene k being selected at a specific regularization level λ , MeanProb_k summarises this information across the entire regularization path by counting the proportion of λ values for which gene k was selected. It can therefore be seen as an average of $\hat{\Pi}_k^\lambda$ over all λ considered in all the resamples.

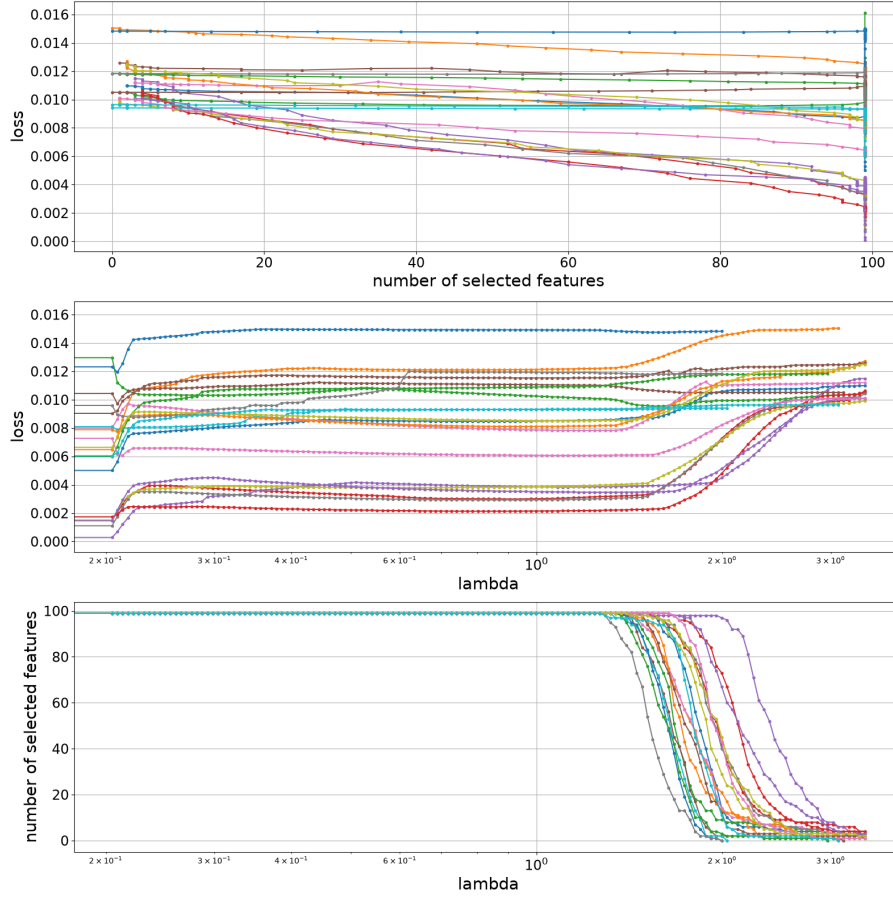


Figure 2.4: **Stability selection paths.** Each colour corresponds to a different stability selection experiment, i.e. a different random subsample of the data. The top panel shows the empirical loss as a function of the number of selected features. The middle panel shows the loss as a function of λ . The bottom panel shows the number of selected features as a function of λ , illustrating how increasing regularization progressively removes features from the model. Note that the set of λ values explored differs between experiments, reflecting the stochastic nature of the training process.

A normalised variant of this score is also defined as:

$$MeanProbNorm_k = \frac{MeanProb_k}{\sum_{k=1}^d MeanProb_k}. \quad (2.3)$$

The normalization of the mean probability score is motivated by the need to ensure comparability across target genes. Since each target gene experiment may be conducted over different sets of regularization parameter Λ , the raw mean probability scores $MeanProb_k$ may systematically differ in magnitude depending on the range of the regularization paths considered. By normalising with respect to the sum of all feature scores, $MeanProbNorm_k$ places each target gene on a common scale, allowing for a fair comparison of variable importance across experimental settings with potentially differing regularization scales. However, this normalization may come at a cost when aggregating variable importances across the d experiments. If a variable is consistently assigned high selection probability across the regularization path in a given target gene experiment, its normalised score is diluted by the sum of all other feature scores within that experiment. When these normalised scores are then aggregated across target genes, this dilution effect accumulates, potentially reducing the discriminant power of $MeanProbNorm_k$ precisely for the variables the method selects with higher confidence. This is an undesirable property in the context of gene regulatory network inference. Both scores are compared empirically in Chapter 3.

2.2.3 Benchmarks

In this thesis, three benchmarks were used to assess the quality of the different methods. The first two come from the DREAM project. The datasets are called DREAM4 and DREAM5. The last one is the BEELINE benchmark. These benchmarks will be talked about in the next paragraphs.

DREAM

DREAM or Dialogue for Reverse Engineering Assessments and Methods is a community-designed project run by researchers to find solutions for biomedical problems and promote collaboration [4–6]. It takes place as annual competition during which researchers compete and exchange on biomedical problems.

DREAM4 During the DREAM4 edition of this competition, one challenge concerned in silico regulatory network inference. The results the tested methods have achieved in the In Silico Size 100 Multifactorial challenge will be reported in the Chapter 3.

The goal of the In Size 100 Multifactorial challenge was to infer five networks of $d = 100$ genes each from multifactorial perturbation data [4].

The data provided by DREAM4 comes from a software called GeneNetWeaver. This software can generate benchmark networks and datasets and analyze the performance of GRN inference methods. It works by extracting the topological structure of regulatory networks in *Escherichia coli* and *Saccharomyces cerevisiae* to form the benchmark network [5]. The benchmark is then simulated to generate steady-state expression data.

DREAM5 The second dataset used is one coming from the DREAM5 edition of the DREAM competition. During this edition, experts were asked to infer genome-scale transcriptional regulatory networks from gene-expression microarray data sets for a prokaryotic model organism (*E. coli*), a eukaryotic model organism (*S. cerevisiae*), a human pathogen (*S. aureus*) and an *in silico* benchmark [6]. The latter is the DREAM5 dataset used in this thesis to assess the result of the tested algorithms. This dataset is composed of 1643 genes among which there are 195 regulators.

These datasets have been the gold standard benchmark used to assess methods' performance for a long time. Born from a competition, these datasets have outlasted the event itself and become the de facto standard against which any new inference method must prove itself.

BEELINE

BEELINE is a comprehensive evaluation framework to assess the accuracy, robustness and efficiency of GRN inference methods for single-cell gene expression [10]. It was developed to address the unique challenges of single-cell transcriptomics, such as high dimensionality and data sparsity, which traditional bulk benchmarks like DREAM4 and DREAM5 do not fully cover [1]. It directly incorporates diverse GRN inference algorithms such as GENIE3 or GRNBoost2. It also allow to add new methods using a uniform interface for each one of them in the form of a Docker image. The goal of BEELINE is to enable reproducible and rigorous evaluations of GRN inference methods for single-cell gene expression data.

Simulated datasets from curated models The first few datasets in the BEELINE framework are three published Boolean models: ventral spinal cord (VSC) development, hematopoietic stem cell (HSC) differentiation and gonadal sex determination (GSD). The performances of each methods over ten datasets of each model are summarized in the Chapter 3.

Experimental single-cell RNA-seq datasets The last set of datasets on which methods have been assessed on are five experimental single-cell RNA-seq datasets, two in human and three in mouse cells [10]. These five datasets are diluted in seven cell types. The seven cell types are human mature hepatocytes (hHEP), human embryonic stem cells (hESC), mouse embryonic stem cells (mESC), mouse Dendritic Cells (mDC), mouse Hematopoietic Stem Cells - Erythroid lineage (mHSC-E), mouse Hematopoietic Stem Cells - Granulocyte/Macrophage lineage (mHSC-GM), and mouse Hematopoietic Stem Cells - Lymphoid lineage (mHSC-L). As one can see, the datasets span different species, distinct cell types and also different differentiation among the mouse hematopoietic stem cells. BEELINE evaluates methods across biologically diverse regulatory contexts. This is one of its key strengths over DREAM4 and DREAM5. Following the gene selection procedure provided in the BEELINE codebase, and also adopted in the DeepSEM paper [32], the expression data for each dataset is restricted to the transcription factors present in the data together with the 500 most variable genes, rather than the full set of measured genes. This truncation reduces the dimensionality of the inference problem for every method evaluated and directly determines the gene counts reported for each dataset in the Results chapter 3.

However, a major challenge arising from the evaluation of GRN inference algorithms for single-cell RNA-seq data is that the "ground truth" is usually unknown [10]. The "ground truth" represents the network of regulatory interactions governing the dynamics of genes of interest. The BEELINE paper addresses this challenge by matching each scRNA-seq dataset with an appropriate ChIP-seq-derived network connecting TFs to their target. Additionally, noncell-type-specific transcriptional regulatory networks as well as the functional protein-protein interactions in the STRING database were used as the ground truth. Of these options, only the STRING-derived ground truth is used in this work.

2.2.4 Performance metrics

Among the challenges of benchmarking gene regulatory network inference algorithms, the choice of performance metric is one of the most critical since classical classification metrics are not always well suited to the structural properties of these networks. In this thesis, three metrics were considered. The Area Under the Receiver Operator Characteristic (AUROC), the Area Under the Precision Recall Curve (AUPRC) and the Early Precision Ratio (EPR).

Class imbalance in gene regulatory networks Before defining the aforementioned metrics, it is important to recall an important property of gene regulatory networks: they are extremely sparse. For d genes, the number of possible interactions is $d(d-1)$, yet only a small fraction of these pairs correspond to true regulatory relationships. This strong class imbalance between the few positive (regulatory) and

many negative (non-regulatory) has a direct consequence on the choice of evaluation metrics, as some classical metrics become uninformative or misleadingly optimistic under such imbalance.

Notation in the GRN inference context In the context of gene regulatory network inference, the standard classification notions of positive and negative examples translate as follows. A *positive* example corresponds to a gene pair (j, k) for which a directed regulatory interaction from gene j to gene k exists in the gold standard network, while a *negative* example corresponds to a gene pair for which no such interaction exists. Given a predicted ranked list of interactions, applying a threshold produces a binary prediction for each pair:

- a *true positive* (TP) is a gene pair correctly predicted as regulatory,
- a *false positive* (FP) is a gene pair incorrectly predicted as regulatory,
- a *true negative* (TN) is a gene pair correctly predicted as non-regulatory,
- a *false negative* (FN) is a gene pair incorrectly predicted as non-regulatory.

These four quantities, summarised in the confusion matrix of Table 2.1, form the basis of all metrics defined below. Given the sparsity of regulatory networks, the number of true negatives TN vastly exceeds the number of true positives TP , which is precisely what makes naive metrics such as accuracy misleading in this setting.

	Actual positive	Actual negative
Predicted positive	TP	FP
Predicted negative	FN	TN

Table 2.1: Confusion Matrix [42]. In the GRN inference context, positive examples are true regulatory interactions and negative examples are non-regulatory gene pairs.

Area Under the Receiver Operator Characteristic (AUROC) In machine learning or deep learning evaluation, research has shifted away from simple accuracy results to other metrics such as Receiver Operator Characteristic (ROC) analysis [42, 43]. ROC space denotes a coordinate system used for visualizing classifier performance. In this space, the true positive rate (TPR) is plotted on the Y axis and the false positive rate (FPR) is plotted on the X axis. They are defined as

$$TPR = \frac{TP}{TP + FN},$$

$$FPR = \frac{FP}{TN + FP},$$

The TPR measures the fraction of true regulatory interactions that are correctly retrieved, while the FPR measures the fraction of non-regulatory pairs that are incorrectly predicted as regulatory. Each threshold value defines a point in the ROC space, and the curve is traced by varying the threshold continuously over all predicted scores.

The area under this curve (AUROC) summarizes the ability of the method to rank true interaction (regulatory) above non-interactions (non-regulatory) across all thresholds, with a value of 0.5 corresponding to a random predictor and a value of 1.0 corresponding to a perfect one. However, because of the high class imbalance described above, the FPR remains low even when a large number of false positives are predicted, since the number of true negatives TN is very large. AUROC can therefore appear artificially high and fail to discriminate between methods with meaningfully different performance on sparse networks [9, 42].

Area Under the Precision Recall Curve (AUPRC) Precision-Recall (PR) curves have been cited as an alternative to ROC curves for tasks with high class imbalance [42]. In the PR space, the *Recall* is plotted on the X axis and the *Precision* is plotted on the Y axis. The *Recall* is the same as the *TPR* while the *Precision* measures the fraction of predicted regulatory interactions that are truly regulatory. They are thus defined as

$$\begin{aligned} \text{Recall} = \text{TPR} &= \frac{TP}{TP + FN}, \\ \text{Precision} &= \frac{TP}{TP + FP}. \end{aligned}$$

Each model can be plotted in the PR space the same way they are plotted in the ROC space.

Unlike the ROC curve, Precision is directly sensitive to the number of false positives relative to true positives, without being diluted by the large number of true negatives. The Area Under this curve (AUPRC) is therefore generally considered more informative than AUROC for gene regulatory network inference, and is the metric most commonly retained in benchmarking studies such as BEELINE and DREAM4 [9, 42].

Early Precision Ratio (EPR) Finally, since in practice a biologist is typically interested only in the top-ranked predicted interactions (those most likely to be validated experimentally), a third metric focusing on early retrieval performance is considered [10]. The Early Precision is computed by taking the top- k predicted interactions and computing the precision restricted to this subset. k is set to $\bar{E}$ which is the number of true edges in the gold standard network. Since this value is highly dependent on the dataset, the Early Precision is then normalised by the precision expected under a random predictor, giving the Early Precision Ratio (EPR):

$$\text{EPR} = \frac{\text{Early Precision}}{\text{Edge Density}},$$

where

$$\text{Edge Density} = \frac{\bar{E}}{d(d-1)},$$

and $\bar{E}$ denotes the number of true edges in the gold standard network and $d(d-1)$ is the total number of possible directed interactions among d genes. An EPR value greater than 1 indicates that the method recovers true regulatory interactions among its most confident predictions at a rate exceeding random chance.

EPR is particularly well suited to the sparse GRN inference setting, as it focuses precisely on the regime where a biologist would act on the predictions, and has been shown to be sensitive to the ability of a method to distinguish direct regulatory relationships from indirect ones [10].

Chapter 3

Results and discussions

This chapter presents and discusses the results of the different experiments conducted in this thesis.

These results are presented in the same order as in the methods section 2.2.3: starting with the DREAM4 datasets, followed by DREAM5, then the curated datasets from BEELINE, and finally the experimental datasets of BEELINE.

Throughout this chapter, the methods are referred to by the labels used on the x-axis of the figures. Table 3.1 provides a summary of all labels and their corresponding methods.

Label	Description
DEEPSEM	DeepSEM algorithm [32]
SCGENERA1	scGENERA1 algorithm [9]
GENIE3	GENIE3 algorithm [4]
GRNBOOST2	GRNBoost2 algorithm [20]
TABNET	TabNet base feature importance embedded in the GENIE3-like framework (Section 2.2.1)
LASSONET	LassoNet base feature importance embedded in the GENIE3-like framework (Section 2.2.1)
LASSONETPROB	$MeanProb_k$ score (Equation 2.2) embedded in the GENIE3-like framework, with 20 stability selection resamples
LASSONETPROBNORM	$MeanProbNorm_k$ score (Equation 2.3) embedded in the GENIE3-like framework, with 20 stability selection resamples
LASSONETPROBNORM N	Equivalent to LASSONETPROBNORM but with N stability selection resamples, where $N \in \{5, 10, 15, 25, 30, 35\}$

Table 3.1: Description of the method labels used throughout Chapter 3.

3.1 DREAM4 datasets

In this section, the results of the algorithms on all the datasets in the In Silico Size 100 Multifactorial challenge are presented.

The AUPRC results are shown in Figure 3.1. GENIE3 achieves the best AUPRC among all evaluated methods, followed by LassoNetProbNorm and GRNBoost2 which achieved comparable performances. LassoNetProbNorm is the best performer among LassoNet-based methods. Other deep learning-based approaches perform comparatively poorly on the DREAM4 datasets.

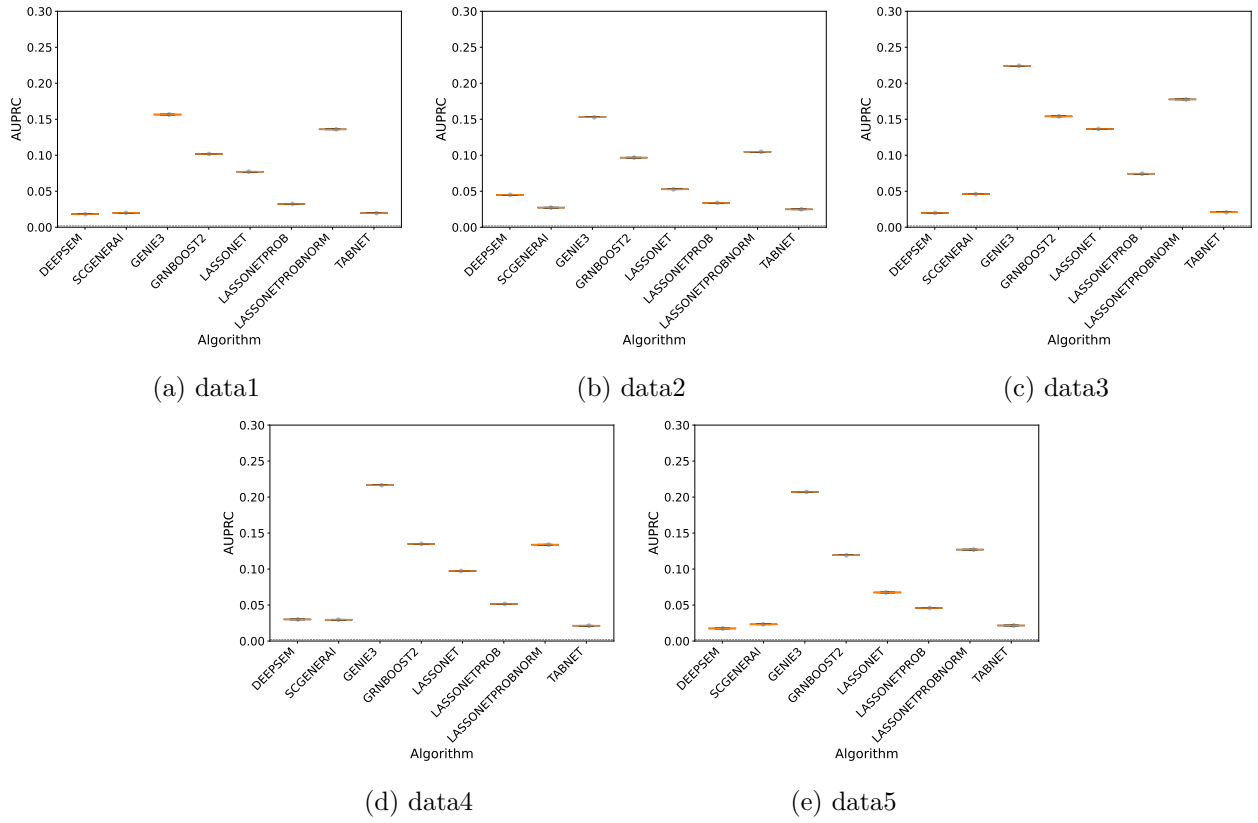


Figure 3.1: AUPRC score of the different algorithms on the 5 DREAM4 datasets.

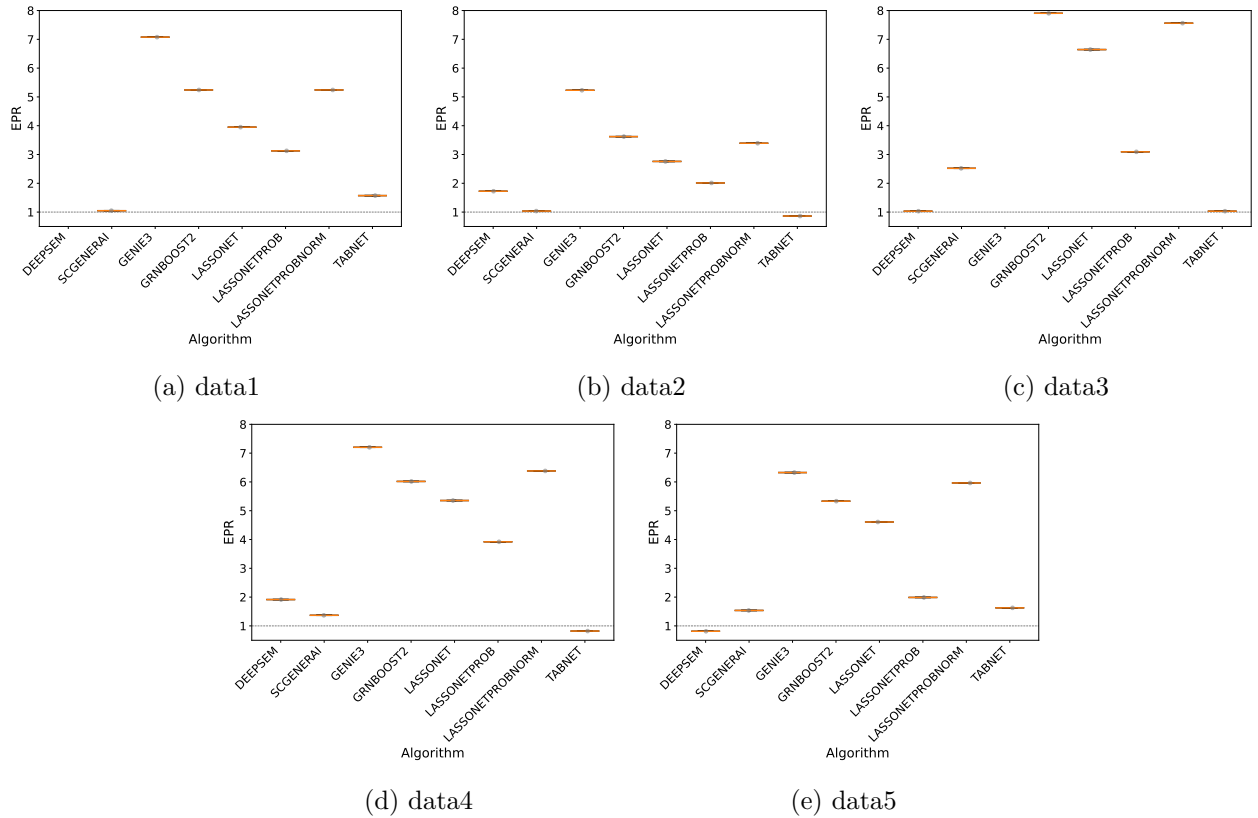


Figure 3.2: EPR score of the different algorithms on the 5 DREAM4 datasets.

This gap can be explained by both the nature of the methods and the nature of the benchmark itself. Tree-based ensemble approaches such as GENIE3 and GRNBoost2 require comparatively little training data and are less prone to overfitting on datasets of the size provided by DREAM4 [1, 44], whereas deep learning methods generally require larger sample sizes than bulk benchmarks such as DREAM4 provide [11]. This limitation is compounded by the nature of the DREAM4 data itself: the multifactorial data consist of static, steady-state expression measurements generated by simulating gene perturbations on a known synthetic network, rather than being sampled from real, noisy biological systems [4]. Because the underlying network and simulation model are fully known and controlled, this data lacks much of the complexity, noise, and higher-order structure typically present in real biological expression data, precisely the kind of structure deep learning models are designed to exploit. As a result, on this particular benchmark, simpler regression- or tree-based approaches such as GENIE3 and GRNBoost2 appear to face little disadvantage relative to deep learning models [1, 44]. LassoNet-based methods also work relatively well and sometimes even better than GRNBoost2. This can be explained by the chosen network size which is relatively small. A small network makes LassoNet methods less prone to overfitting than other deep learning methods.

The EPR results are shown in Figure 3.2. As stated earlier, the strength of the EPR lies in its focus on early retrieval rather than on the full set of retrieved edges.

For this dataset, the EPR mainly serves as a sanity check, confirming that this metric produces results consistent with the AUPRC on small datasets. In line with the AUPRC results, GENIE3 remains the best performer under EPR.

For both EPR and AUPRC, some figures do not show the same magnitude. This is due to differences between the individual experiments: each dataset in the DREAM4 In Silico Size 100 Multifactorial challenge has a different ground truth network and, more importantly, a different network density. These differences affect not only the difficulty of the inference task but also the overall magnitude of the scores achieved by every algorithm on a given dataset, regardless of their relative ranking. Especially, across nearly all algorithms, scores on data3 are consistently higher than on data2. For instance, LassoNetProbNorm achieves an EPR of approximately 3.4 on data2 compared to approximately 7.6 on data3, more than doubling despite the algorithm itself being unchanged. These two datasets are the most relevant because they are synthesized over the regulatory networks of different organisms: data1 and data2 come from *E. coli* while data3, data4 and data5 come from *S. cerevisiae*. The same pattern, though somewhat less pronounced, is also visible in the AUPRC results. This effect is most visible among the stronger-performing methods (GENIE3, GRNBoost2, and the LassoNet variants); it is less consistent for the weakest performers, such as DeepSEM and TabNet, whose scores remain close to random across all datasets and are therefore less sensitive to this magnitude effect.

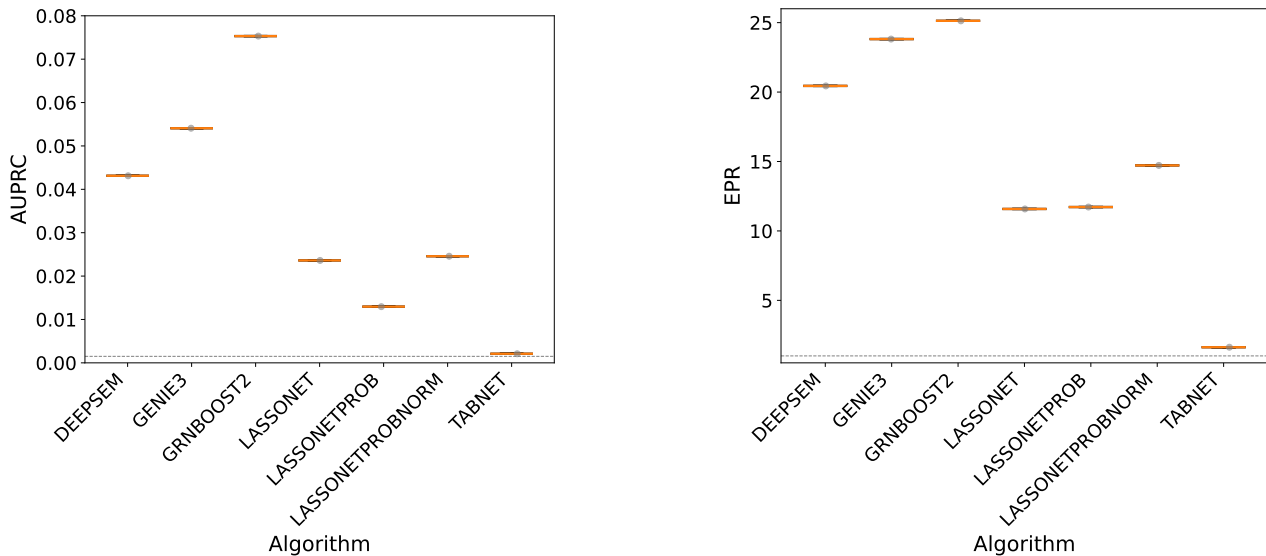
3.2 DREAM5 dataset

As with the experimental BEELINE datasets discussed later in this chapter, only a single inferred network is available per algorithm for DREAM5, rather than a set of inferred networks as for DREAM4; no measure of variance or robustness across repeated runs is therefore available here either.

The DREAM5 *in silico* network consists of 1643 genes and 4012 gold-standard edges, measured across 805 samples, giving a ground-truth density of approximately 0.0015. This is far sparser than any of the five DREAM4 networks, which are considerably smaller in scale (100 samples each); this larger and sparser network is expected to yield substantially lower achievable AUPRC than DREAM4, independently of any difference in the algorithms' actual inference quality since the AUPRC value that a random predictor would have is approximately the ground-truth density.

scGeneRAI could not be run on the DREAM5 dataset within the scope of this thesis. Given its larger size compared to the DREAM4 datasets, and the correspondingly longer computation time required, its execution was not launched early enough in the project timeline to complete before the completion of this thesis. scGeneRAI is therefore absent from the results presented in this section.

The AUPRC and EPR results for the DREAM5 *in silico* network are shown in Figure 3.3a and Figure 3.3b, respectively.



(a) AUPRC score of the different algorithms on the DREAM5 *in silico* network.

(b) EPR score of the different algorithms on the DREAM5 *in silico* network.

Figure 3.3: Performance metrics of the different algorithms on the DREAM5 *in silico* network.

Consistent with the DREAM4 results, GRNBoost2 and GENIE3 are the top two performers under AUPRC, followed by DeepSEM, with a clear drop to LassoNetProbNorm and LassoNet, and a further drop to LassoNetProb and, most markedly, TabNet, which achieves the lowest AUPRC of any algorithm on this dataset by a wide margin. This ranking closely mirrors the pattern observed on DREAM4, reinforcing the hypothesis raised there: because both DREAM4 and DREAM5’s *in silico* networks are generated through steady-state simulation of a known network topology using GeneNetWeaver, rather than sampled from real, noisy biological systems, they lack much of the complexity and higher-order structure that deep learning-based methods are designed to exploit leaving tree-based ensemble methods such as GENIE3 and GRNBoost2 at an advantage [1, 11, 44]. In this dataset, DeepSEM is the best performing method among the deep learning-based ones.

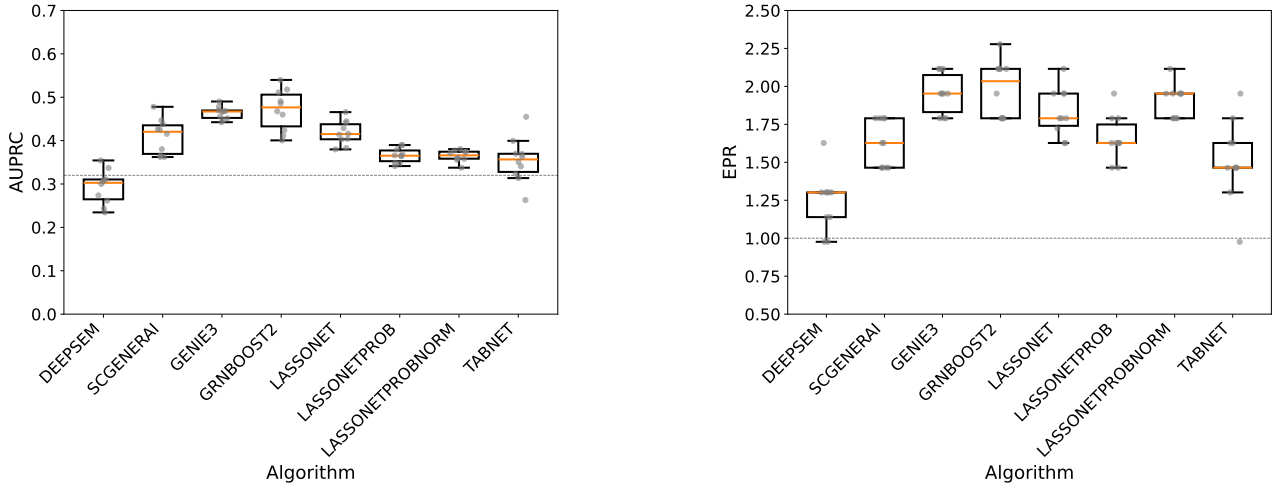
The EPR results agree closely with the AUPRC ranking: GRNBoost2 leads narrowly over GENIE3, followed by DeepSEM, LassoNetProbNorm, LassoNetProb, and LassoNet, with TabNet again clearly the weakest performer.

3.3 Curated datasets from BEELINE

This section examines the curated datasets from the BEELINE benchmark [10]: Gonadal Sex Determination (GSD), Hematopoietic Stem Cell (HSC) differentiation, and Ventral Spinal Cord (VSC) development. Unlike the DREAM4 datasets, which consist of several resampled instances of both *E. coli* and *S. cerevisiae* coming from the software GeneNetWeaver, these three datasets correspond to distinct, literature-curated Boolean models of real biological differentiation processes, each simulated with BoolODE to generate ten single-cell expression replicates [10].

3.3.1 HSC dataset

The AUPRC results for the HSC dataset are shown in Figure 3.4a, and the corresponding EPR results are shown in Figure 3.4b.



(a) AUPRC score of the different algorithms on the HSC datasets.

(b) EPR score of the different algorithms on the HSC datasets.

Figure 3.4: Performance metrics of the different algorithms on the HSC datasets. Boxplot computed over 10 datasets.

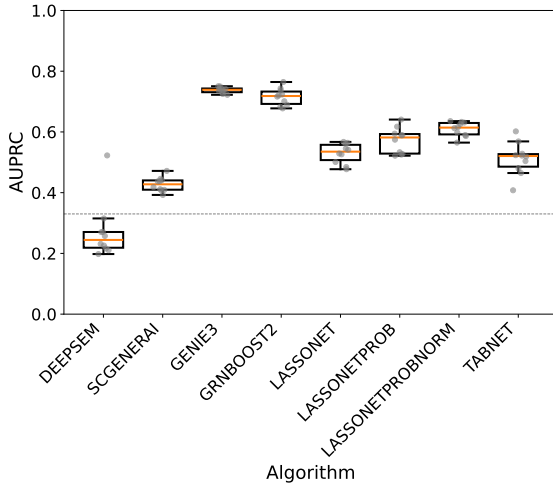
On the HSC dataset, GRNBoost2 and GENIE3 again achieve the highest scores, consistent with their performance on the DREAM4 datasets. However, the gap with the deep learning-based methods is noticeably smaller here: scGeneRAI, in particular, ranks third overall, ahead of LassoNet, LassoNetProb, LassoNetProbNorm, and TabNet. DeepSEM, by contrast, is the weakest performer on this dataset. This narrower gap may be explained by the nature of the HSC data itself: because it is derived from a curated Boolean model simulated with realistic single-cell dropout and noise, rather than from a purely deterministic steady-state system as in DREAM4, it more closely resembles the kind of noisy structured data that deep learning-based approaches are designed to exploit [11].

The EPR results on HSC are consistent with the AUPRC results: GRNBoost2 and GENIE3 remain the top two performers, followed by LassoNetProbNorm, while DeepSEM and TabNet remain the weakest. scGeneRAI’s relative advantage over other deep learning-based methods is somewhat less pronounced under EPR than under AUPRC, which may suggest that it retrieves true edges reliably across the full ranking rather than being especially strong at early retrieval specifically. Conversely, LassoNetProbNorm moves markedly closer to GENIE3 and GRNBoost2 under EPR than under AUPRC, nearly matching their scores despite trailing them by a clear margin on AUPRC. LassoNetProb shows the same shift, but far less markedly, remaining closer to scGeneRAI than to the leading methods under EPR. This asymmetry between the two LassoNet variants suggests that whichever mechanism drives LassoNetProbNorm’s stronger early retrieval is not simply a general property of the LassoNetProb-based approaches, but may instead be specific to the additional normalization step distinguishing the two variants.

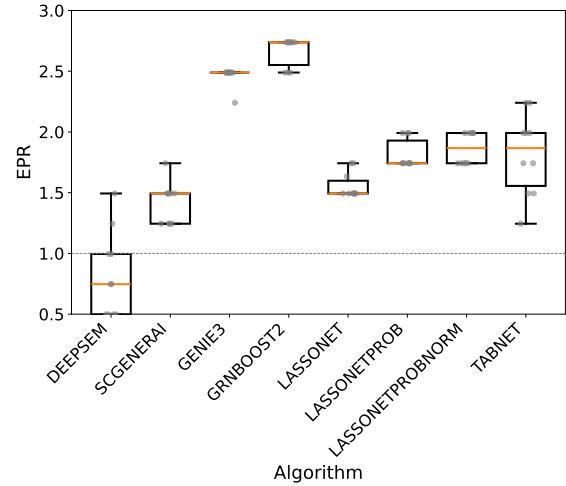
3.3.2 VSC dataset

The AUPRC results for the VSC dataset are shown in Figure 3.5a, and the corresponding EPR results are shown in Figure 3.5b.

The VSC dataset shows the clearest gap between tree-based and deep learning-based methods among the three curated datasets. GENIE3 and GRNBoost2 substantially outperform every other method, while DeepSEM is consistently the weakest performer, with one notable outlier (on VSC-2000-7) reaching a markedly higher score than its other nine replicates. LassoNetProbNorm, LassoNetProb, LassoNet and TabNet form an intermediate group, ahead of scGeneRAI and DeepSEM. This pattern is closer to what was observed on the DREAM4 datasets than to the HSC results, which may indicate



(a) AUPRC score of the different algorithms on the VSC datasets.



(b) EPR score of the different algorithms on the VSC datasets.

Figure 3.5: Performance metrics of the different algorithms on the VSC datasets. Boxplot computed over 10 datasets.

that the VSC network’s dynamics are comparatively easier for tree-based feature-importance methods to exploit, or alternatively that the noise and dropout levels used in this particular simulation were less favorable to the deep learning-based methods than in the HSC case.

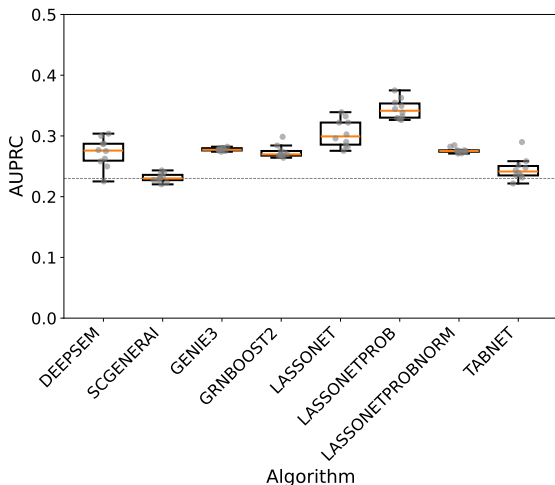
The EPR results largely mirror the AUPRC results, with GRNBoost2 and GENIE3 again clearly ahead of all other methods. One notable difference is TabNet, which ranks above the LassoNet variants and scGeneRAI under EPR despite ranking below them under AUPRC. This suggests that TabNet may retrieve a number of true edges early in its ranking, even though its overall precision-recall performance across the full ranking remains comparatively weak.

3.3.3 GSD dataset

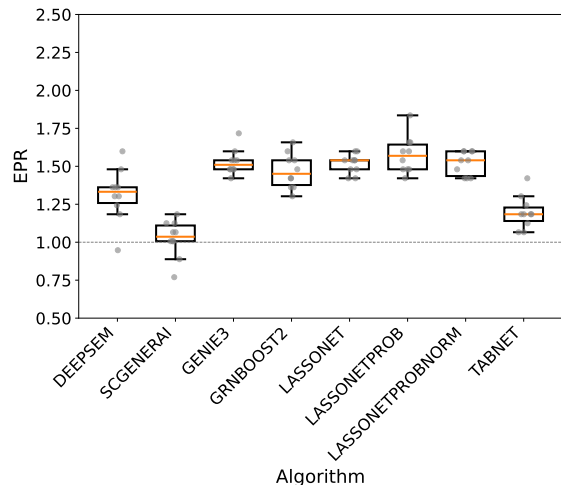
The AUPRC results for the GSD dataset are shown in Figure 3.6a, and the corresponding EPR results are shown in Figure 3.6b.

Unlike on the previous datasets and methods, the tree-based ensemble methods do not clearly dominate on GSD. GENIE3 and GRNBoost2 achieve broadly average scores, while LassoNetProb is the best-performing method overall, and LassoNet also outperforms both tree-based methods. All algorithms nonetheless remain within a comparatively narrow score range, with scores for every method well below those observed on the other curated datasets discussed below.

The EPR results on GSD are consistent with the AUPRC results: LassoNetProb remains the top performer, and the LassoNet variants and GENIE3 form a closely clustered leading group, with GRNBoost2 trailing slightly behind them. A reversal of the ranking observed on the DREAM4, DREAM5, HSC, and VSC datasets, where the tree-based methods consistently led. GSD is also the largest of the three curated networks considered here, with 19 genes and 79 ground-truth edges, compared to 10 genes and 29 edges for HSC and 7 genes and 14 edges for VSC. Since the number of samples is held constant at 2000 across all three datasets, this larger and denser network provides comparatively more structure per sample for a regularized regression-based method such as LassoNet to exploit, which may explain why it is competitive with, or even surpasses, the tree-based ensembles specifically on this dataset [11].



(a) AUPRC score of the different algorithms on the GSD datasets.



(b) EPR score of the different algorithms on the GSD datasets.

Figure 3.6: Performance metrics of the different algorithms on the GSD datasets. Boxplot computed over 10 datasets.

3.3.4 Cross-dataset comparison

Taken together, the three curated datasets show a consistent lead for GENIE3 and GRNBoost2 on VSC and HSC, but not on GSD, where LassoNetProb and the other LassoNet variants take the lead instead. This reversal coincides with a clear difference in network size: GSD (19 genes, 79 edges) is substantially larger and denser than HSC (10 genes, 29 edges) and VSC (7 genes, 14 edges), while the number of simulated samples is held constant at 2000 across all three. One possible explanation is that, as the number of genes increases relative to a fixed sample size, the network becomes sparser, a direction that favors regularized regression-based methods such as LassoNet over tree-based feature importance methods. This is consistent with the broader observation, discussed for the DREAM4 datasets, that deep learning and regression-based approaches tend to require comparatively more structure or dimensionality to perform competitively [11]. Table 3.2 summarizes the size of each curated dataset for reference. This hypothesis would need to be tested more systematically, for instance by evaluating the same algorithms on synthetic networks of varying size while holding the underlying dynamics fixed, before it could be considered more than a plausible explanation for the pattern observed here.

Dataset	Genes	Ground-truth edges	Samples	Density
HSC	10	29	2000	0.32
VSC	7	14	2000	0.33
GSD	19	79	2000	0.23

Table 3.2: Size of the three curated BEELINE datasets used in this section.

3.4 Experimental datasets from BEELINE

Unlike the DREAM and curated BEELINE datasets, which are generated from synthetic or literature-curated simulation models, the experimental datasets from BEELINE consist of real single-cell RNA-seq measurements collected from actual biological systems [10]. As a result, each dataset here corresponds to a single inferred network rather than a set of stochastic simulation replicates, and no measure of variance or robustness across repeated runs is available. For the datasets considered in this section, the

ground truth network is derived from the STRING database [45] rather than from a known simulation model or a curated Boolean network.

scGeneRAI could not be run on the experimental BEELINE datasets within the scope of this thesis. Given their larger size compared to the curated BEELINE datasets and the DREAM4 dataset, and the correspondingly longer computation time required, its execution was not launched early enough in the project timeline to complete before the completion of this thesis. scGeneRAI is therefore absent from the results presented in this section.

The datasets in this section are presented in order of decreasing ground-truth density, from the dataset most similar in this aspect to the curated BEELINE networks discussed in the previous section (hHep) to the least similar (mHSC-L), as summarized in Table 3.3. This ordering is intended to make it easier to track how the algorithms’ relative and absolute performance evolves as the underlying network moves further from the density range observed in the curated datasets (0.23 to 0.33, Table 3.2).

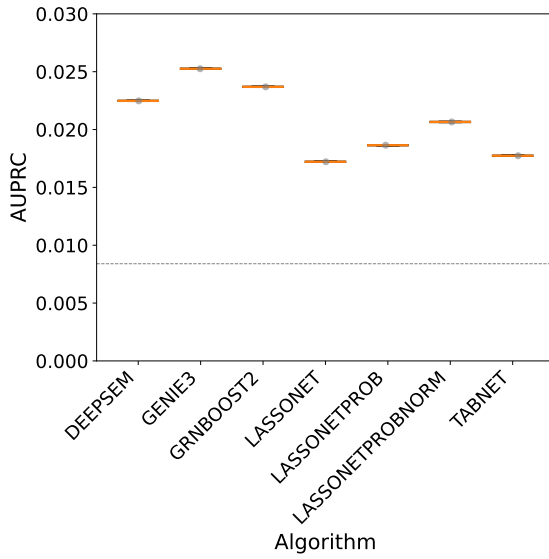
Dataset	Genes	Ground-truth edges	Cells	Density
hHep	948	7523	425	0.0084
mDC	821	4815	383	0.0072
mESC	1120	7763	421	0.0062
hESC	910	4257	758	0.0051
mHSC-E	704	1371	1071	0.0028
mHSC-GM	632	748	889	0.0019
mHSC-L	560	137	847	0.00044

Table 3.3: Size and ground-truth density of the experimental BEELINE datasets used in this section, ordered by decreasing density. Density is computed as the number of ground-truth edges divided by the number of possible directed gene pairs.

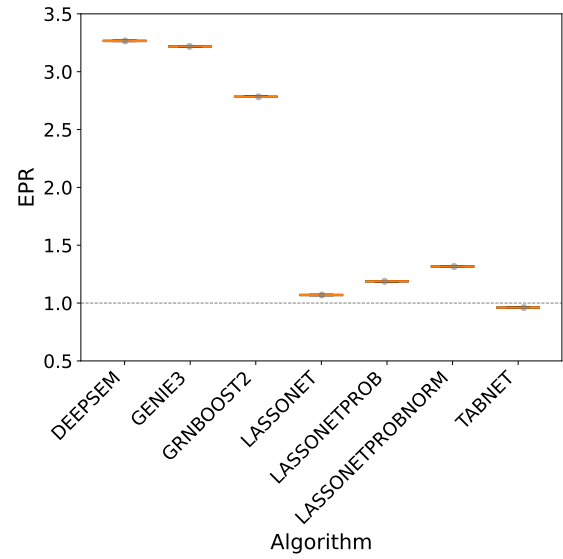
3.4.1 hHep dataset

The AUPRC and EPR results for the human hepatocyte (hHep) dataset are shown in Figure 3.7a and Figure 3.7b, respectively.

The first observation is that the absolute AUPRC values, ranging from approximately 0.017 to 0.025, are markedly lower than those observed on the DREAM4 and curated BEELINE datasets, where AUPRC values generally ranged from 0.2 to 0.7. This drop in magnitude is mainly attributable to the size and sparsity of the hHep network relative to the curated BEELINE networks: with 948 genes and 7523 ground-truth edges, hHep has a density of 0.0084, the highest among the seven experimental datasets, and therefore the closest of the group to the density range observed for the curated GSD, HSC, and VSC networks (0.23 to 0.33, Table 3.2), even though it remains roughly 27 to 40 times sparser than any of them. Since the expected AUPRC of a random predictor is approximately equal to the ground-truth density, this difference alone predicts an order-of-magnitude gap in achievable AUPRC, closely matching what is observed here, independently of any difference in the algorithms’ actual ability to rank true edges above false ones. GENIE3, GRNBoost2, and DeepSEM form a tight leading group under AUPRC, with GENIE3 marginally ahead. They are followed by LassoNetProbNorm, LassoNetProb, TabNet, and LassoNet with LassoNetProbNorm being quite close to DeepSEM. Despite being the densest experimental dataset, hHep does not show the highest achievable AUPRC among this group; on the contrary, its best score (GENIE3, approximately 0.0253) is the lowest maximum AUPRC observed across all seven experimental datasets, well below the values reached on the considerably sparser mHSC-GM (approximately 0.108) and mHSC-L (approximately 0.111) datasets discussed later in this section. This is the opposite of what a straightforward density-based explanation of AUPRC magnitude, as discussed for the curated BEELINE datasets, would predict. This pattern is examined systematically across all seven experimental datasets in the second cross-dataset comparison at the end of this section.



(a) AUPRC scores of the different algorithms on the hHep dataset.



(b) EPR scores of the different algorithms on the hHep dataset.

Figure 3.7: Performance metrics of the different algorithms on the hHep dataset.

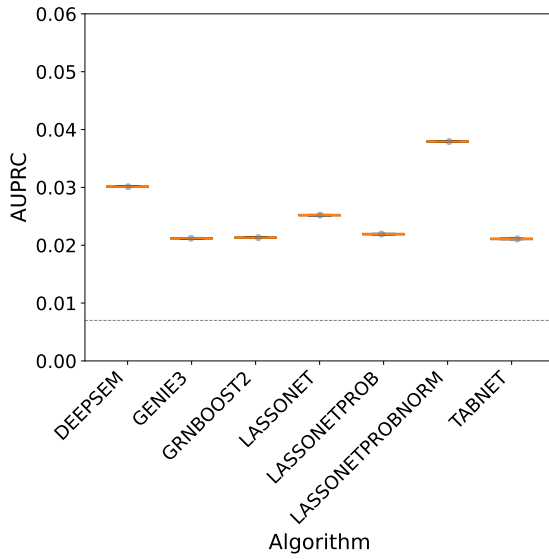
The EPR results again agree closely with the AUPRC ranking: DeepSEM leads narrowly over GENIE3, followed by GRNBoost2, with a clear drop to LassoNetProbNorm and the remaining methods, though this relative gap is considerably larger under EPR than under AUPRC. The absolute EPR values on hHep also remain comparatively close to the EPR values reported for the curated BEELINE datasets in the previous section, consistent with EPR’s normalization by the random-predictor baseline (see Section 2.2.4).

3.4.2 mDC dataset

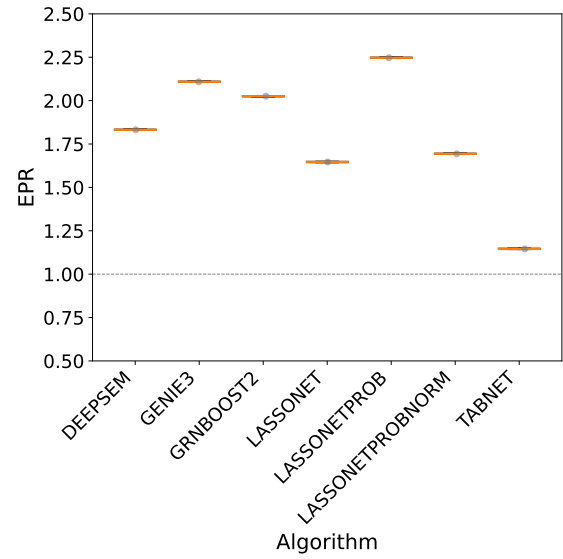
The AUPRC and EPR results for the mouse dendritic cell (mDC) dataset are shown in Figure 3.8a and Figure 3.8b, respectively.

An interesting observation made upon the AUPRC values on mDC dataset is the relative ranking of the algorithms that departs sharply from the pattern observed on hHep, curated BEELINE datasets and other experimental datasets discussed in the following sections. GENIE3 and GRNBoost2, which generally were among the top performers, are here among the weakest, both trailing LassoNet, DeepSEM, and LassoNetProbNorm, the latter being the best-performing method overall. This reversal is consistent with the hypothesis raised for the curated HSC dataset: because mDC is derived from real single-cell measurements rather than a synthetic or curated simulation, it contains noise, dropout, and non-linear structure that tree-based ensemble methods are not specifically designed to exploit, whereas deep learning-based approaches may be better suited to exploit [11]. However, as discussed in the following subsections, this reversal is not observed on any of the other experimental datasets examined in this thesis, which consistently show GENIE3, GRNBoost2, and DeepSEM as the top three performers; mDC should therefore be treated as a notable exception rather than as representative of the experimental datasets as a group.

The EPR results complicate this picture even more. GENIE3 and GRNBoost2 rank second and third under EPR, respectively, well ahead of their AUPRC ranking, while LassoNetProbNorm and LassoNet, two of the top three methods under AUPRC, both drop to the lower half of the ranking. LassoNetProb shows the opposite shift, becoming the top performer under EPR despite ranking in the middle of the pack under AUPRC. A possible interpretation is that GENIE3 and GRNBoost2 continue to rank a substantial number of true edges early in their output, but accumulate comparatively more false



(a) AUPRC scores of the different algorithms on the mDC dataset.



(b) EPR scores of the different algorithms on the mDC dataset.

Figure 3.8: Performance metrics of the different algorithms on the mDC dataset.

positives later in the ranking on this dataset, which would depress their AUPRC without necessarily depressing their EPR.

Given that these results rest on a single inferred network per algorithm, with no repeated runs to assess variability, these interpretations on AUPRC and EPR values should be treated as tentative hypotheses rather than a firm conclusion.

3.4.3 mESC dataset

The AUPRC and EPR results for the mouse embryonic stem cell (mESC) dataset are shown in Figure 3.9a and Figure 3.9b, respectively.

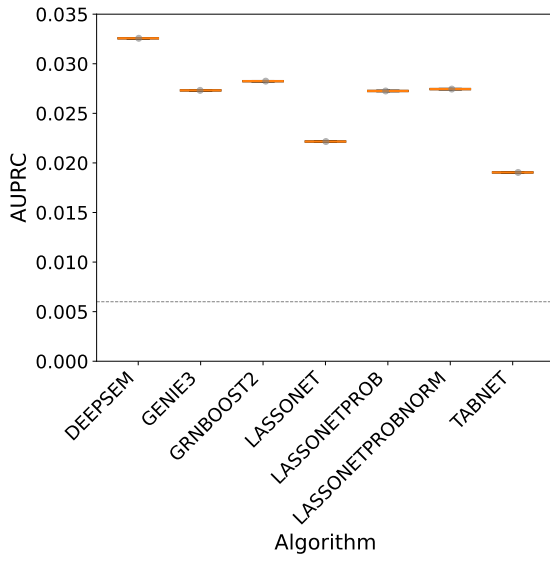
mESC is, at 1120 genes, the largest network among the experimental datasets, with a density of 0.0062. DeepSEM leads under AUPRC. Notably, GRNBoost2, GENIE3, LassoNetProbNorm, and LassoNetProb are nearly indistinguishable from each other in second, third, fourth, and fifth position, meaning the usual gap between the leading tree-based and DeepSEM group and the regularized deep learning-based methods narrows considerably on this dataset, specifically with respect to GENIE3 and GRNBoost2, rather than DeepSEM. LassoNet and TabNet remain the weakest performers.

However, this narrowing does not carry over to EPR. Under EPR, GRNBoost2, DeepSEM, and GENIE3 reassert a clear lead over LassoNetProbNorm, LassoNetProb, and LassoNet, which fall back to a distinctly lower tier, with TabNet remaining the weakest method. This is the second instance, after mDC, of the AUPRC and EPR rankings disagreeing considerably for a subset of methods on a given dataset, though the specific methods and direction of the disagreement differ between the two cases.

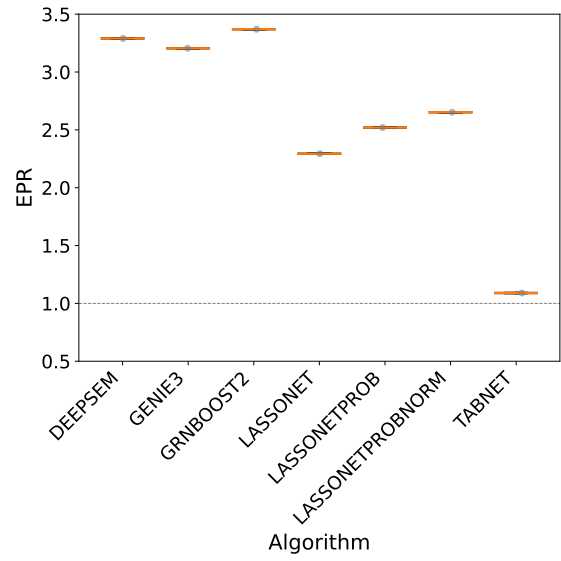
3.4.4 hESC dataset

The AUPRC and EPR results for the human embryonic stem cell (hESC) dataset are shown in Figure 3.10a and Figure 3.10b, respectively.

With 910 genes and a density of 0.0051, hESC shows the same pattern as hHep: GENIE3, GRNBoost2, and DeepSEM form a tightly clustered leading group under AUPRC, with GENIE3 markedly

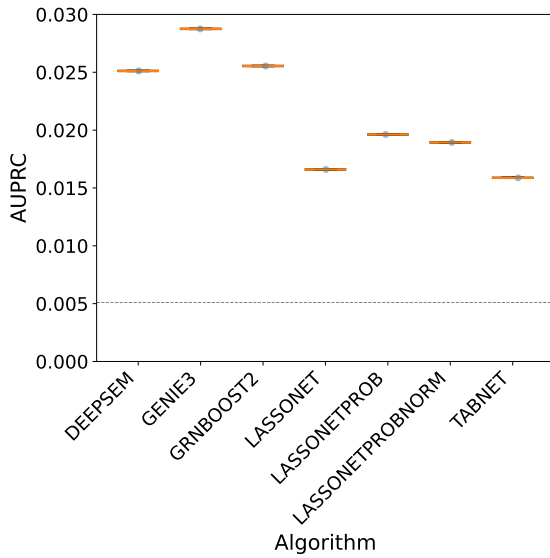


(a) AUPRC scores of the different algorithms on the mESC dataset.

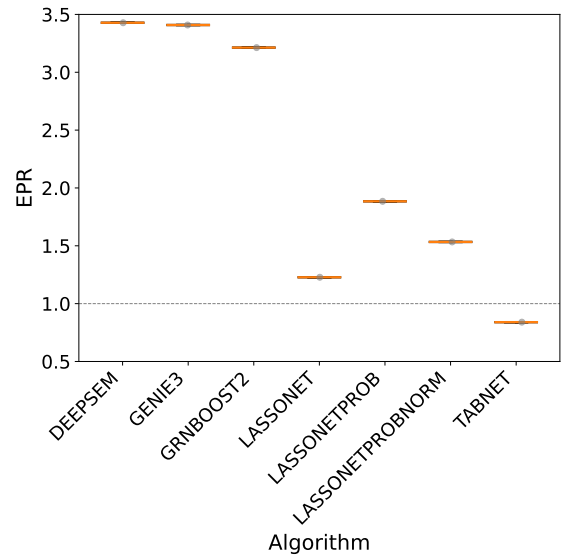


(b) EPR scores of the different algorithms on the mESC dataset.

Figure 3.9: Performance metrics of the different algorithms on the mESC dataset.



(a) AUPRC scores of the different algorithms on the hESC dataset.



(b) EPR scores of the different algorithms on the hESC dataset.

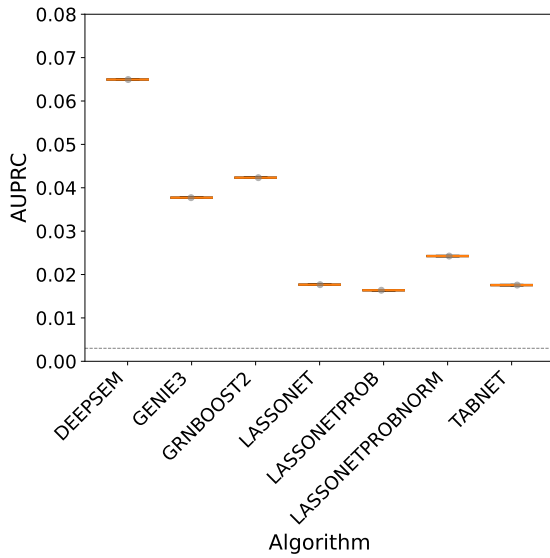
Figure 3.10: Performance metrics of the different algorithms on the hESC dataset.

ahead. They are followed by LassoNetProb, LassoNetProbNorm, LassoNet and TabNet at a clear distance below.

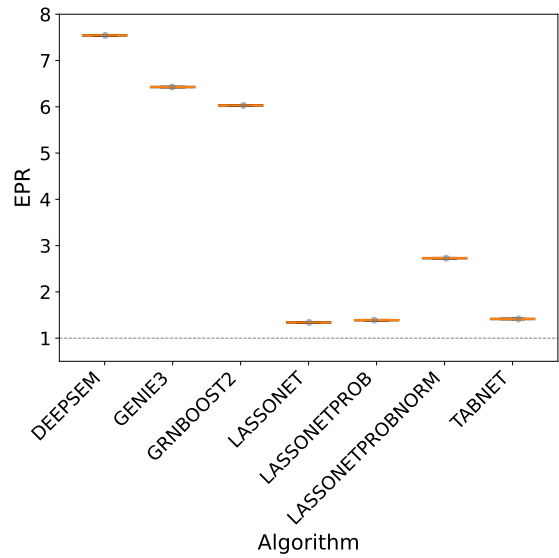
The EPR results are consistent with the AUPRC ranking, with DeepSEM narrowly ahead of GENIE3 and GRNBoost2 at the top, and the same clear separation from LassoNetProb, LassoNetProbNorm, LassoNet, and TabNet below.

3.4.5 mHSC-E dataset

The AUPRC and EPR results for the mouse hematopoietic stem cell, erythroid lineage (mHSC-E) dataset are shown in Figure 3.11a and Figure 3.11b, respectively.



(a) AUPRC scores of the different algorithms on the mHSC-E dataset.



(b) EPR scores of the different algorithms on the mHSC-E dataset.

Figure 3.11: Performance metrics of the different algorithms on the mHSC-E dataset.

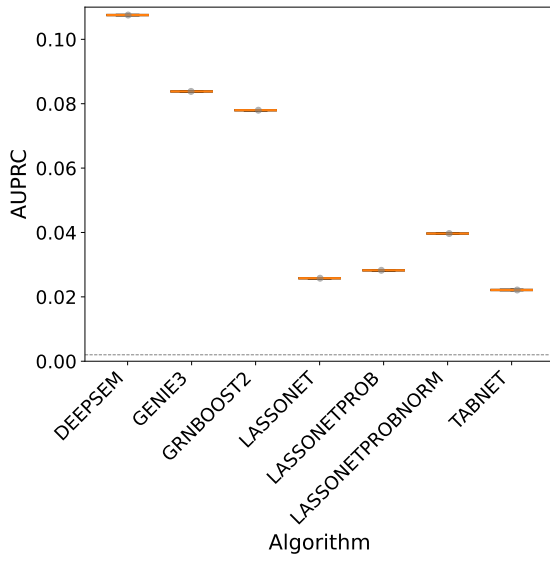
With 704 genes, 1371 ground-truth edges, and 1071 cells, mHSC-E has a density of 0.0028. DeepSEM is the clear top performer under AUPRC, ahead of GRNBoost2 and GENIE3, with a marked drop to LassoNetProbNorm, and a further drop to LassoNet, TabNet, and LassoNetProb.

The EPR results reinforce the AUPRC ranking rather than diverging from it. DeepSEM again leads by a wide margin, followed by GENIE3 and GRNBoost2, while LassoNetProbNorm, TabNet, LassoNetProb, and LassoNet form a clearly separated lower group.

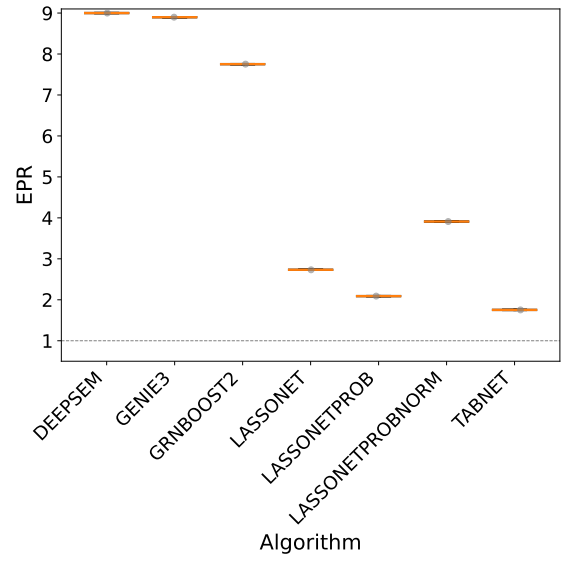
3.4.6 mHSC-GM dataset

The AUPRC and EPR results for the mouse hematopoietic stem cell, granulocyte-monocyte lineage (mHSC-GM) dataset are shown in Figure 3.12a and Figure 3.12b, respectively.

mHSC-GM shows the same leading group as mHSC-E, with DeepSEM ahead of GENIE3 and GRNBoost2, followed by a clear drop to LassoNetProbNorm and a further drop to LassoNetProb, LassoNet, and TabNet. With a density of 0.0019 (632 genes, 748 ground-truth edges), mHSC-GM is the second sparsest dataset in this section, yet it produces among the highest absolute AUPRC scores observed here, with DeepSEM reaching approximately 0.108. As with hHep above, this pattern (lower density coinciding with higher, not lower, achievable AUPRC) runs counter to a simple density-based explanation of AUPRC magnitude. This is examined systematically across all seven datasets in the cross-dataset comparison below.

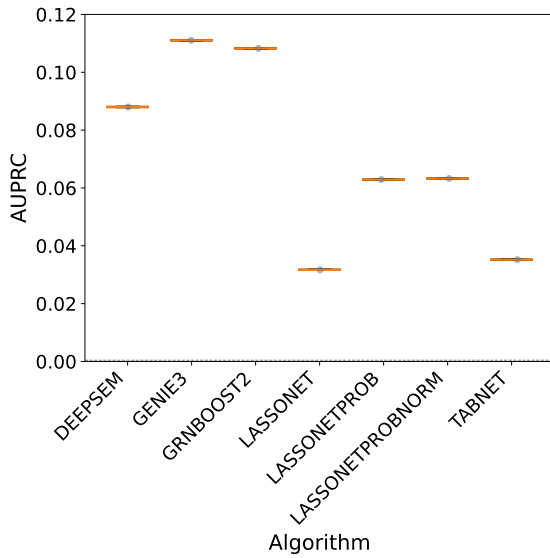


(a) AUPRC scores of the different algorithms on the mHSC-GM dataset.

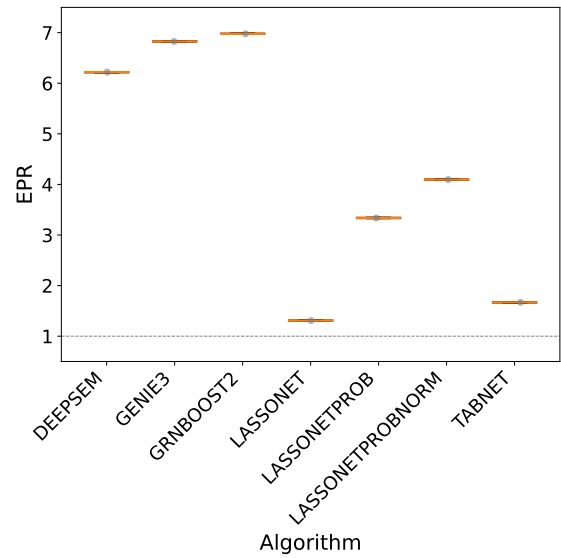


(b) EPR scores of the different algorithms on the mHSC-GM dataset.

Figure 3.12: Performance metrics of the different algorithms on the mHSC-GM dataset.



(a) AUPRC scores of the different algorithms on the mHSC-L dataset.



(b) EPR scores of the different algorithms on the mHSC-L dataset.

Figure 3.13: Performance metrics of the different algorithms on the mHSC-L dataset.

The EPR results follow the same ranking as AUPRC, with DeepSEM, GENIE3, and GRNBoost2 clearly ahead of LassoNetProbNorm, LassoNet, LassoNetProb, and TabNet.

3.4.7 mHSC-L dataset

The AUPRC and EPR results for the mouse hematopoietic stem cell, lymphoid lineage (mHSC-L) dataset are shown in Figure 3.13a and Figure 3.13b, respectively.

With 560 genes and only 137 ground-truth edges, mHSC-L has a density of 0.00044, the lowest of all seven experimental datasets, roughly four times sparser than mHSC-GM and nearly twenty times sparser than hHep. Despite this, mHSC-L’s AUPRC scores are the highest observed in this section. GENIE3 leads narrowly over GRNBoost2, with DeepSEM in third position. LassoNetProbNorm and LassoNetProb form a second group before the third group containing TabNet and LassoNet.

The EPR results are consistent with this picture: GRNBoost2 and GENIE3 lead narrowly ahead of DeepSEM, followed by LassoNetProbNorm and LassoNetProb at a comparatively small distance, mirroring the AUPRC ranking. Then TabNet and LassoNet come last.

3.4.8 Cross-dataset comparison

Table 3.4 summarizes, for each experimental dataset, the ground-truth density, the number of cells, and the best-performing algorithm under AUPRC and under EPR.

Dataset	Density	Cells	Best AUPRC	Best EPR
hHep	0.0084	425	GENIE3 (0.025)	DEEPSEM (3.27)
mDC	0.0072	383	LassoNetProbNorm (0.038)	LassoNetProb (2.25)
mESC	0.0062	421	DeepSEM (0.033)	GRNBoost2 (3.37)
hESC	0.0051	758	GENIE3 (0.029)	DeepSEM (3.43)
mHSC-E	0.0028	1071	DeepSEM (0.065)	DeepSEM (7.54)
mHSC-GM	0.0019	889	DeepSEM (0.108)	DeepSEM (9.00)
mHSC-L	0.00044	847	GENIE3 (0.111)	GRNBoost2 (6.98)

Table 3.4: Best-performing algorithm and score under AUPRC and EPR for each experimental dataset, alongside ground-truth density and cell count.

Leading group. Across the seven experimental datasets, the best-performing algorithm under AUPRC is always one of GENIE3, GRNBoost2, or DeepSEM, with the single exception of mDC, where LassoNetProbNorm leads. GENIE3 is the top AUPRC performer on three datasets (hHep, hESC, mHSC-L), DeepSEM on three (mESC, mHSC-E, mHSC-GM), and LassoNetProbNorm on the one exception. The same three algorithms also dominate under EPR, with DeepSEM leading on four datasets (hHep, hESC, mHSC-E, mHSC-GM), GRNBoost2 on two (mESC, mHSC-L), and LassoNetProb leading only on the exceptional mDC dataset. In both metrics, LassoNet, and TabNet consistently occupy the bottom of the ranking outside of mDC. This confirms, with the full set of seven datasets now available, that mDC is a genuine outlier rather than representative of how these algorithms perform on real experimental data.

Density and achievable AUPRC magnitude. The individual dataset discussions above repeatedly noted that lower ground-truth density did not translate into lower achievable AUPRC, contrary to the naive expectation that AUPRC magnitude should scale with the density of the ground truth. Across the seven datasets, comparison between the best-achieved AUPRC and the density shows a strong negative association rather than a positive one: denser ground truths are associated with lower achievable AUPRC. The three densest datasets (hHep, mDC, mESC) all have comparatively few cells (383 to 425), while the four sparsest (hESC, mHSC-E, mHSC-GM, mHSC-L) have considerably more

(758 to 1071). Cell count also correlates with achievable AUPRC; however, neither factor can be fully isolated from the other with the data available here.

The comparison with the DREAM4 datasets helps disentangle the two at a larger scale. DREAM4 achieves substantially higher AUPRC than any of the experimental datasets despite having only 100 samples per network, far fewer than any of the seven experimental datasets considered here (383 to 1071 cells). If cell count were the dominant driver of achievable AUPRC, DREAM4’s far smaller sample size would predict a lower AUPRC than the experimental datasets, not a higher one. This is inconsistent with what is observed, and instead points to density as the stronger candidate explanation across large density gaps such as the one separating DREAM4 from the experimental datasets. Across the comparatively smaller density differences observed among the experimental BEELINE datasets themselves, however, cell count appears to be the dominant factor associated with achievable AUPRC. With only seven experimental datasets, these observations should be treated as descriptive and hypothesis-generating rather than as statistically confirmed relationships; testing this properly would require datasets in which density and cell count vary independently of one another, or a larger set of datasets than is currently available.

Agreement between AUPRC and EPR. Of the seven datasets, five (hHep, hESC, mHSC-E, mHSC-GM, mHSC-L) show broadly consistent rankings between AUPRC and EPR, with at most minor reordering within the leading group. The remaining two, mDC and mESC, show substantial disagreement between the two metrics for a subset of algorithms: on mDC, GENIE3 and GRNBoost2 rank far higher under EPR than under AUPRC, while LassoNetProbNorm and LassoNet show the opposite shift; on mESC, the LassoNet variants lose the near-tie with GENIE3 observed under AUPRC once EPR is considered. GENIE3 reasserts a clear separation from the LassoNet variants. No shared property of mDC and mESC (density, cell count, or gene count) obviously distinguishes them from the other five datasets, so what drives this divergence on these two datasets specifically remains an open question.

Overall. Taken together, these results suggest that GENIE3 and GRNBoost2 remain the most reliable performers, joined here by DeepSEM, which was not part of the leading group on the DREAM4 or curated BEELINE datasets but emerges as one of the strongest algorithms specifically on real experimental data, with mDC standing out as a clear exception. The relationship between dataset properties and achievable performance is more complex than for the DREAM4 and curated BEELINE datasets: density appears to dominate across large density gaps, such as between DREAM4 and the experimental datasets, while cell count appears more influential across the smaller density differences observed among the experimental datasets themselves. Given that all of these results rest on a single inferred network per algorithm and dataset, with no repeated runs to assess variability, and that the dataset-level correlations reported here are based on only seven data points, these conclusions should be interpreted with appropriate caution given the limited number of datasets.

3.5 LassoNet’s computation time

One of the main limitations of the LassoNet-based methods is their relatively high computational cost¹.

Several factors may contribute to this computational overhead. As an initial investigation, the impact of the number of models learned during each stability selection experiment on the inference performance is examined. Increasing the number of models corresponds to increasing the number of random subsamples drawn from the data, which in principle should yield more stable and reliable

¹No dedicated benchmarking study was conducted to precisely measure the execution time of the different algorithms. However, for the single-cell RNA-seq datasets considered in this work, the LassoNet-based GRN inference methods required approximately 5 to 10 days to complete, whereas running GENIE3, GRNBoost2, DeepSEM, and TabNet sequentially required up to 4 days.

selection probability estimates. However, it also increases the computational cost of the procedure proportionally. It is therefore important to determine whether a small number of models is sufficient to obtain reliable results or whether performance continues to improve as more models are added.

The results are shown in Figure 3.14 for the AUPRC metric and in Figure 3.15 for the EPR metric, evaluated on three datasets from the BEELINE benchmark: the Gonadal Sex Determination (GSD), Hematopoietic Stem Cell (HSC), and Ventral Spinal Cord (VSC) datasets.

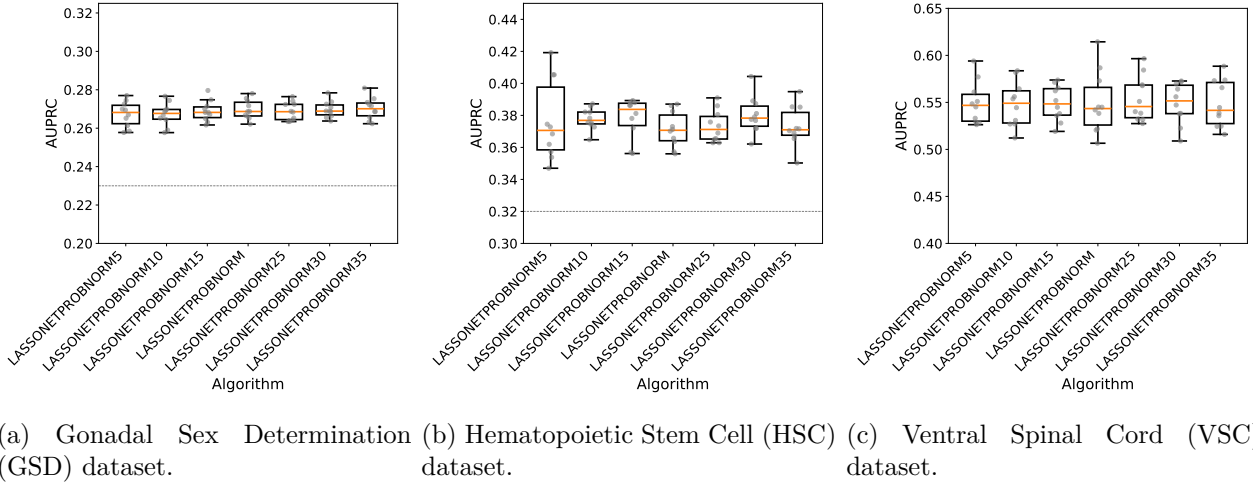


Figure 3.14: Impact of the number of models learned during a stability selection experiment on the AUPRC score, evaluated on three BEELINE datasets.

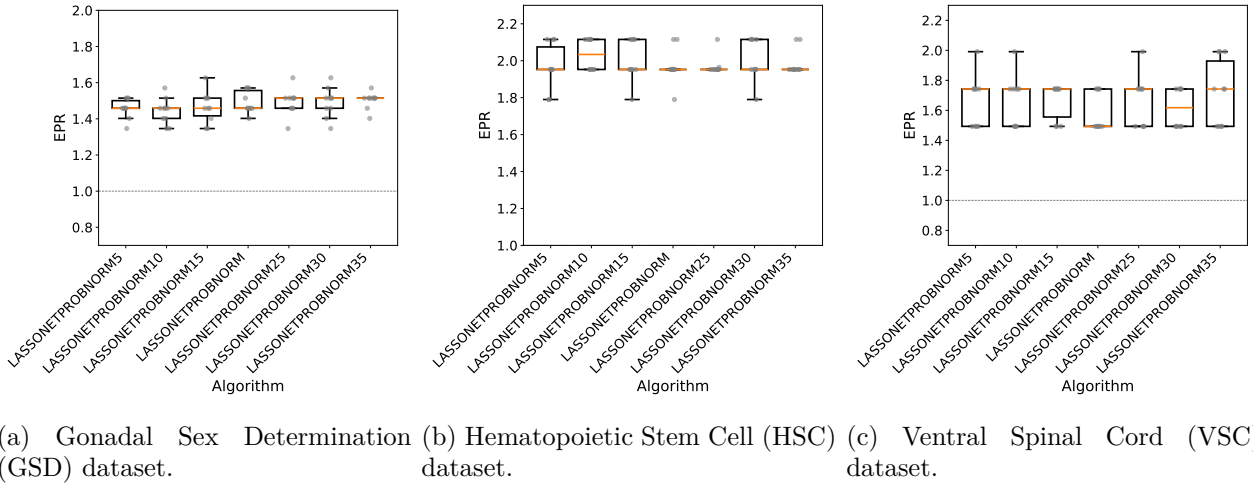


Figure 3.15: Impact of the number of models learned during a stability selection experiment on the EPR score, evaluated on three BEELINE datasets.

Across all three datasets and both metrics, the number of models learned during a stability selection experiment does not significantly affect the inference performance. For instance, on GSD, the median AUPRC ranges from approximately 0.268 to 0.270 as the number of models increases from 5 to 35, and the median EPR ranges from approximately 1.46 to 1.51 over the same range (both well within the variability already observed between the ten replicates of a single configuration). The same holds on HSC and VSC. Both the AUPRC and EPR scores remain stable as the number of models increases, suggesting that the selection probability estimates converge rapidly and that a relatively small number of models is sufficient for reliable estimation. These results indicate that using fewer models during a stability selection experiment maintains comparable inference performance while potentially reducing

the computational cost of the method. This observation is particularly relevant since the LassoNet-based approaches are the most computationally demanding methods evaluated in this thesis.

Another likely contributor to the high computational cost of the LassoNet-based methods is the lack of parallelization across the models learned during a stability selection experiment. In the current implementation, each of the models is trained sequentially, one after the other, rather than concurrently. This is in contrast to GENIE3 and GRNBoost2, which parallelize across trees and, in the case of GRNBoost2, across the underlying gradient boosting framework, allowing them to make full use of the available computational resources. Since stability selection itself does not require any interaction between the models learned across different subsamples, the LassoNet-based methods could in principle also be parallelized across models, or even across genes, without altering the underlying method or its results. The absence of such parallelization in the current implementation therefore represents a straightforward avenue for future optimization, rather than an inherent limitation of the stability selection procedure itself.

Chapter 4

Conclusions and perspectives

4.1 Conclusions

This thesis set out to investigate whether the feature-importance mechanisms embedded directly in certain deep learning architectures could be exploited for gene regulatory network inference, as an alternative to tree-based ensemble method or to post-hoc explainability tools applied to otherwise opaque models. Two such architectures, LassoNet and TabNet, were integrated into a GENIE3-like per-gene regression framework, in which the importance a model assigns to each candidate regulator when predicting a target gene’s expression is reused directly as a confidence score for a regulatory interaction. For LassoNet, whose residual-weight-based importance proved considerably more informative than TabNet’s attention-based masks, a stability-selection procedure was further developed to aggregate feature-importance scores across multiple resampled regularisation paths, in an attempt to obtain a more robust ranking of candidate interactions than a single training run could provide.

These methods were benchmarked against two established tree-based ensemble methods, GENIE3 and GRNBoost2, and two other deep-learning-based GRN inference approaches, DeepSEM and scGenerAI, across a broad range of benchmarks: the synthetic DREAM4 and DREAM5 datasets, three curated Boolean-model datasets from BEELINE (HSC, VSC, GSD), and seven experimental single-cell RNA-seq datasets from BEELINE, evaluated using AUPRC and the Early Precision Ratio.

Across this benchmark, no single method dominates uniformly, and the relative ranking of algorithms depends strongly on the nature of the data being analysed. On the synthetic DREAM4 and DREAM5 networks, and on two of the three curated BEELINE datasets (HSC and VSC), the tree-based ensembles methods GENIE3 and GRNBoost2 retain a clear advantage over every deep-learning-based method tested, including the architectures developed in this thesis. This advantage appears linked to the deterministic, noise-free nature of these simulated datasets, which do not exhibit the kind of higher-order structure and noise that deep learning methods are designed to exploit, combined with the comparatively small sample sizes these benchmarks provide. On the denser and larger GSD dataset, however, and on several of the experimental BEELINE datasets, this advantage narrows or reverses: DeepSEM emerges as the strongest performer on most experimental datasets, and the LassoNet-based methods developed in this thesis, particularly the stability-selection variants, become competitive with, and in some cases exceed, the tree-based ensembles. TabNet, despite its purpose-built sequential attention mechanism for tabular data, consistently underperformed relative to every other method across all benchmarks, suggesting that its instance-wise feature selection does not translate straightforwardly into a reliable gene-level importance score once aggregated across the many per-gene subproblems that the GENIE3-like framework requires.

Taken together, these results suggest that feature-importance mechanisms embedded in deep learning architectures can indeed be exploited for GRN inference, and that LassoNet in particular, combined with a stability-selection procedure, is able to match or outperform the long-standing tree-based gold standards on some real, biologically diverse datasets. However, this potential is not realised uniformly:

it depends on the architecture (LassoNet succeeding where TabNet does not), on the methodology used to extract a stable score from the model (stability selection outperforming a single training run), and on properties of the dataset itself, notably its size, density, and its nature: whether it originates from a simulated or an experimentally measured biological system, or whether the gene expression comes from bulk RNA-sequencing, single-cell RNA-sequencing or other sequencing methods. Moreover, DeepSEM more consistently outperforms GENIE3 and GRNBoost2 on experimental datasets. However, it does not work better than a random predictor on the synthetic datasets from DREAM4. This context-dependence echoes a recurring observation in the GRN inference literature: methods have inherent biases toward certain data regimes, and no method should be expected to perform optimally across every biological context [6, 10].

This work has several limitations that should be kept in mind when interpreting these conclusions. First, for the DREAM5 *in silico* network and for all seven experimental BEELINE datasets, only a single inferred network was produced per algorithm, since these datasets do not come with multiple stochastic replicates as the DREAM4 and curated BEELINE datasets do; no measure of variance or robustness across repeated runs is therefore available for roughly half of the benchmarks used in this thesis, and the corresponding conclusions should be read with the appropriate caution. Second, the observations relating achievable AUPRC to dataset density and cell count discussed in Section 3.4.8, rest on only seven experimental datasets and should be treated as hypothesis-generating rather than statistically established. Third, the LassoNet-based methods and the stability-selection procedure in particular, remain considerably more computationally expensive than tree-based ensembles. A limitation that is at least partly attributable to the current implementation training the resampled models sequentially rather than in parallel. This current limitation limited the exploration of different hyperparameters and, in particular, different network sizes that could have helped on larger, noisier and higher-order networks.

4.2 Perspectives

Several directions could extend and consolidate the work presented in this thesis.

Parallelising the stability-selection procedure. Since the models trained on different resamples do not depend on one another, and since the number of resamples was shown in Section 3.5 to have little impact on inference performance beyond a relatively small number of models on curated BEELINE datasets, training these models concurrently, either across resamples or across target genes, should substantially reduce the computational cost of the LassoNet-based methods without altering their results, making them more practical for genome-scale application.

Exploring LassoNet’s hyperparameters more thoroughly. Parallelising the stability-selection procedure would also free up computational budget for a more intensive hyperparameter search than was possible in this thesis. In particular, the hidden-layer width, the `path_multiplier` governing granularity of the regularisation path, and the L2 penalty gamma that were kept close to their default values in Section 2.1.1 to limit training cost could be explored much deeply.

Understanding TabNet’s underperformance. TabNet’s instance-wise, sample-specific feature selection is well suited to standard tabular prediction tasks, but did not translate into a useful gene-level importance score in this thesis’ GENIE3-like framework. Further work could investigate alternative ways of aggregating TabNet’s per-sample masks into a global importance score, or explore whether TabNet’s sequential decision-step structure could be repurposed more directly to recover regulatory interactions, before concluding that the architecture does not suit this task.

Disentangling the effects of network density and sample size. The relationship observed between dataset density, cell count, and achievable AUPRC could be tested more systematically by generating synthetic networks in which these two properties are varied independently while the underlying network dynamics are held fixed, for instance using GeneNetWeaver or BoolODE with controlled parameters. This would allow the hypotheses raised in Section 3.4.8 to be tested with statistical rigour rather than inferred descriptively from seven experimental datasets.

Assessing robustness on experimental data. Because the experimental BEELINE datasets and the DREAM5 *in silico* network each yield only a single inferred network algorithm, an important next step would be to assess the stability of these methods, particularly the LassoNet-based ones, across multiple datasets of each cell type considered in the experimental data itself, in order to obtain confidence intervals on their reported performance rather than point estimates. However, experimental datasets are generally hard to gather. To avoid the need for new sequencing, one could use bootstrapping or resampling for a first investigation of the robustness on experimental data.

Combining tree-based and deep-learning-based scores. Since GENIE3, GRNBoost2, DeepSEM, and the LassoNet-based methods appear to have complementary strengths depending on the dataset, an ensemble approach that aggregates rankings across several of these methods (such a community approach is described in [6]) could be investigated as a way of obtaining more consistently strong performance across biological contexts. This is in line with the BEELINE authors’ own observation that no single method is uniformly best [10], while being mindful of their finding that naive ensembling does not automatically outperform the best individual method.

Extending the benchmark. Finally, now that a GENIE3-like, feature-importance-based framework has been established for LassoNet, it would be worth evaluating it on a wider range of experimental

single-cell datasets, additional species, and larger gene sets, to determine whether the trends observed here persist at greater scale.

Additional content

This chapter provides the information needed to access and reproduce the work presented in this thesis. It describes where the code developed and used throughout the project can be found, as well as the practical steps required to set it up and run it, in particular on the CECI clusters. It also indicates where the datasets used for benchmarking can be obtained and explains how artificial intelligence tools were used in the preparation of this thesis.

Code availability

All code is available in the following GitHub repository: <https://github.com/luvilour/Beeline>. This repository is a fork of the original BEELINE GitHub repository¹, to which the newly tested algorithms (DeepSEM, SCGeneRAI, LassoNet, and TabNet) were added.

A method for launching `BLRunner.py` on the CECI clusters has also been added, by defining containers using Apptainer instead of Docker. The environment can be set up by running `setup_beeline_container.sh`. The Apptainer images can then be built by running `initialize_ceci.sh` for the algorithms already implemented, or `initialize_ceci.sh -b` to build the newly added algorithms. Finally, `docker_shim` must be set as the alias for the `docker` command. Once this is done, the original `ReadMe.md` can be followed to run the different components of the code and benchmark the algorithms. A Slurm script has also been added to launch the BLRunner Slurm job.

Data availability

All the BEELINE datasets used to assess the results of the different algorithms in this thesis are available at the following URL: <https://zenodo.org/records/19009603>.

Artificial intelligence tools usage

Artificial intelligence tools were used throughout this work to support translation, the understanding of scientific papers, and research within and across the literature. Google NotebookLM was used to explore papers and access relevant information more efficiently. Claude was used to rephrase certain sections in order to improve the overall linguistic quality of this work. Translation tools such as DeepL were also employed.

In addition, Claude was used as an assistant to help integrate the newly implemented algorithms within the BEELINE framework, as well as to develop the capabilities required to run BEELINE on the CECI clusters.

However, the use of these tools in no way influenced the technical content of this thesis. All analysis, discussions, and conclusions remain entirely my own responsibility.

¹<https://github.com/Murali-group/Beeline>

Bibliography

- [1] Jiayi Dong, Jiahao Li, and Fei Wang. Deep Learning in Gene Regulatory Network Inference: A Survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 21(6):2089–2101, November 2024.
- [2] Hamid Bolouri. *Computational modeling of gene regulatory networks: a primer*. Imperial College Press, London, reprint edition, 2011.
- [3] Mukesh Bansal, Vincenzo Belcastro, Alberto Ambesi-Impiombato, and Diego Di Bernardo. How to infer gene networks from expression profiles. *Molecular Systems Biology*, 3(1):78, January 2007.
- [4] Vân Anh Huynh-Thu, Alexandre Irrthum, Louis Wehenkel, and Pierre Geurts. Inferring Regulatory Networks from Expression Data Using Tree-Based Methods. *PLoS ONE*, 5(9):e12776, September 2010.
- [5] Mengyuan Zhao, Wenying He, Jijun Tang, Quan Zou, and Fei Guo. A comprehensive overview and critical evaluation of gene regulatory network inference technologies. *Briefings in Bioinformatics*, 22(5):bbab009, September 2021.
- [6] The DREAM5 Consortium, Daniel Marbach, James C Costello, Robert Küffner, Nicole M Vega, Robert J Prill, Diogo M Camacho, Kyle R Allison, Manolis Kellis, James J Collins, and Gustavo Stolovitzky. Wisdom of crowds for robust gene network inference. *Nature Methods*, 9(8):796–804, August 2012.
- [7] Patrick O. Brown and David Botstein. Exploring the new world of the genome with DNA microarrays. *Nature Genetics*, 21(S1):33–37, January 1999.
- [8] T Gardner and J Faith. Reverse-engineering transcription control networks. *Physics of Life Reviews*, 2(1):65–88, March 2005.
- [9] Philipp Keyl, Philip Bischoff, Gabriel Dernbach, Michael Bockmayr, Rebecca Fritz, Nils Blüthgen, Grégoire Montavon, Klaus-Robert Müller, David Horst, and Frederick Klauschen. Single-cell gene regulatory network prediction by explainable AI. *Nucleic Acids Research*, 51(4):e20–e20, February 2023.
- [10] Aditya Pratapa, Amogh P. Jalihal, Jeffrey N. Law, Aditya Bharadwaj, and T. M. Murali. Benchmarking algorithms for gene regulatory network inference from single-cell transcriptomic data. *Nature Methods*, 17(2):147–154, February 2020.
- [11] Zhongxiao Li, Elva Gao, Juexiao Zhou, Wenkai Han, Xiaopeng Xu, and Xin Gao. Applications of deep learning in understanding gene regulation. *Cell Reports Methods*, 3(1):100384, January 2023.
- [12] Jun-Peng Jiang, Si-Yang Liu, Hao-Run Cai, Qi-Le Zhou, and Han-Jia Ye. Representation Learning for Tabular Data: A Comprehensive Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 48(6):6488–6508, June 2026.

- [13] Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep Neural Networks and Tabular Data: A Survey. *IEEE Transactions on Neural Networks and Learning Systems*, 35(6):7499–7519, June 2024.
- [14] Timothy S. Gardner, Diego Di Bernardo, David Lorenz, and James J. Collins. Inferring Genetic Networks and Identifying Compound Mode of Action via Expression Profiling. *Science*, 301(5629):102–105, July 2003.
- [15] Salvatore Raieli. Tabular Deep Learning: A Survey from Small Neural Networks to Large Language Models, September 2025.
- [16] Cha Zhang and Yunqian Ma, editors. *Ensemble Machine Learning: Methods and Applications*. Springer New York, New York, NY, 2012.
- [17] Vân Anh Huynh-Thu and Pierre Geurts. Unsupervised Gene Network Inference with Decision Trees and Random Forests. In Guido Sanguinetti and Vân Anh Huynh-Thu, editors, *Gene Regulatory Networks*, volume 1883, pages 195–215. Springer New York, New York, NY, 2019. Series Title: Methods in Molecular Biology.
- [18] Anne-Claire Haury, Fantine Mordelet, Paola Vera-Licona, and Jean-Philippe Vert. TIGRESS: Trustful Inference of Gene REgulation using Stability Selection. *BMC Systems Biology*, 6(1):145, December 2012.
- [19] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, April 2006.
- [20] Thomas Moerman, Sara Aibar Santos, Carmen Bravo González-Blas, Jaak Simm, Yves Moreau, Jan Aerts, and Stein Aerts. GRNBoost2 and Arboreto: efficient and scalable inference of gene regulatory networks. *Bioinformatics*, 35(12):2159–2161, June 2019.
- [21] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5), October 2001.
- [22] Yanrong Ji, Zhihan Zhou, Han Liu, and Ramana V Davuluri. DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome. *Bioinformatics*, 37(15):2112–2120, August 2021.
- [23] Gilles Louppe. Lecture 5: Convolutional networks.
- [24] Ye Yuan and Ziv Bar-Joseph. Deep learning for inferring gene relationships from single-cell expression data. *Proceedings of the National Academy of Sciences*, 116(52):27151–27158, December 2019.
- [25] G. E. Hinton and R. R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504–507, July 2006.
- [26] Gilles Louppe. Lecture 11: Auto-encoders and variational auto-encoders.
- [27] Francois Fleuret. Deep autoencoders.
- [28] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes, December 2022. arXiv:1312.6114 [stat.ML].
- [29] Francois Fleuret. Deep learning 7.4. Variational Autoencoder.
- [30] Carl Doersch. Tutorial on Variational Autoencoders, January 2021. arXiv:1606.05908 [stat.ML].

- [31] Junliang Zhou, Ningji Gong, Yanjun Hu, Hong Yu, Guoyin Wang, and Hao Wu. GRANet: a graph residual attention network for gene regulatory network inference. *Briefings in Bioinformatics*, 26(4):bbaf349, July 2025.
- [32] Hantao Shu, Jingtian Zhou, Qiuyu Lian, Han Li, Dan Zhao, Jianyang Zeng, and Jianzhu Ma. Modeling gene regulatory networks using neural network architectures. *Nature Computational Science*, 1(7):491–501, July 2021.
- [33] Romain Lopez, Jeffrey Regier, Michael B. Cole, Michael I. Jordan, and Nir Yosef. Deep generative modeling for single-cell transcriptomics. *Nature Methods*, 15(12):1053–1058, December 2018.
- [34] Yue Yu, Jie Chen, Tian Gao, and Mo Yu. DAG-GNN: DAG Structure Learning with Graph Neural Networks.
- [35] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. β -VAE: LEARNING BASIC VISUAL CONCEPTS WITH A CONSTRAINED VARIATIONAL FRAMEWORK. 2017.
- [36] Hao Zhu and Donna Slonim. GRN-VAE: A simplified and stabilized SEM model for GRN Inference.
- [37] Grégoire Montavon, Alexander Binder, Sebastian Lapuschkin, Wojciech Samek, and Klaus-Robert Müller. Layer-Wise Relevance Propagation: An Overview. In Wojciech Samek, Grégoire Montavon, Andrea Vedaldi, Lars Kai Hansen, and Klaus-Robert Müller, editors, *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, volume 11700, pages 193–209. Springer International Publishing, Cham, 2019. Series Title: Lecture Notes in Computer Science.
- [38] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. *PLOS ONE*, 10(7):e0130140, July 2015.
- [39] Ismael Lemhadri, Feng Ruan, Louis Abraham, and Robert Tibshirani. LassoNet: A Neural Network with Feature Sparsity.
- [40] Sercan O. Arik and Tomas Pfister. TabNet: Attentive Interpretable Tabular Learning, December 2020. arXiv:1908.07442 [cs].
- [41] Nicolai Meinshausen and Peter Buehlmann. Stability Selection, May 2009. arXiv:0809.2932 [stat.ME].
- [42] Jesse Davis and Mark Goadrich. The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd international conference on Machine learning - ICML '06*, pages 233–240, Pittsburgh, Pennsylvania, 2006. ACM Press.
- [43] Foster Provost, White Plains, and Ron Kohavi. The Case Against Accuracy Estimation for Comparing Induction Algorithms.
- [44] Nicolas Vecoven, Jean-Michel Begon, Antonio Sutera, Pierre Geurts, and Vân Anh Huynh-Thu. Nets Versus Trees for Feature Ranking and Gene Network Inference. In Annalisa Appice, Grigorios Tsoumakas, Yannis Manolopoulos, and Stan Matwin, editors, *Discovery Science*, volume 12323, pages 231–245. Springer International Publishing, Cham, 2020. Series Title: Lecture Notes in Computer Science.
- [45] Damian Szklarczyk, Annika L Gable, David Lyon, Alexander Junge, Stefan Wyder, Jaime Huerta-Cepas, Milan Simonovic, Nadezhda T Doncheva, John H Morris, Peer Bork, Lars J Jensen, and Christian von Mering. STRING v11: protein-protein association networks with increased coverage,

supporting functional discovery in genome-wide experimental datasets. *Nucleic Acids Research*, 47(D1):D607–D613, January 2019.