

https://dspace.org/

# Neural Graph Scoring

**Gathoo, Mohammed**

**2025**
University of the Witwatersrand, Johannesburg
https://hdl.handle.net/10539/50280
Gathoo, Mohammed. (2026. Neural Graph Scoring [Master's dissertation, University of the Witwatersrand, Johannesburg]. WIReDSpace. https://hdl.handle.net/10539/50280

Downloaded from a DSpace repository. https://dspace.org/

# Neural graph scoring

**School of Computer Science & Applied Mathematics**
**University of the Witwatersrand**

**Mohammed Gathoo**
**2089236**

**Supervised by Prof. Ritesh Ajoodha**

**February 13, 2026**




A dissertation submitted to the Faculty of Science, University of the Witwatersrand, Johannesburg, in partial fulfilment of the requirements for the degree of Master of Science

**Abstract**

Learning graph structures has long been a focal point of graphical modelling, serving as a way of analyzing the relationship between concepts and observations and using that information to make predictions or to reason about some situation. Not only will more efficient solutions benefit graphical modelling but can broaden the applicability of graphical models as a whole. But this problem is $\mathcal{NP}$-hard and many solutions put forward are either too inefficient (spatially or temporally) or do not produce optimal approximations. This necessitates the need for novel and innovative solutions that can circumvent these issues faced by traditional structure learning methods.

With the continuing development of deep learning technologies, an increase in the popularity of structure discovery using deep learning has surface. This led to the development of continuous optimization based approaches, using techniques such as Graph Neural Networks and Reinforcement Learning. This dissertation, motivated by the use of deep learning for causal discovery in directed acyclic graphs, proposes the integration of deep learning techniques for the association of graphical structures with their numerical scores. Conducted on synthetically generated data, this framework shows to accurately score graphs of varying sizes with data belonging to a variety of data distributions. This mitigates the issue experienced by many continuous optimization methods of causal discovery that are held back by inefficient scoring methods.

**Declaration**

I, Mohammed Gathoo, hereby declare the contents of this research dissertation to be my own work. This dissertation is submitted for the degree of Masters of Science in Computer Science at the University of the Witwatersrand. This work has not been submitted to any other university, or for any other degree.

February 13, 2026



# Contents

**Preface**
Abstract . . . . . i
Declaration . . . . . ii
Table of Contents . . . . . iii
List of Figures . . . . . v
List of Tables . . . . . vi
Abbreviations . . . . . vii

**1 Introduction** **1**

**2 Background** **3**
2.1 Probabilistic Graphical Models . . . . . 3
2.1.1 Early Ages of Artificial Intelligence and Probability Theory . . . . 3
2.1.2 Graphical Models . . . . . 4
2.1.3 Representation . . . . . 4
2.1.4 Inference . . . . . 6
2.2 Deep Learning . . . . . 7
2.2.1 What is it? . . . . . 7
2.2.2 Neural Networks . . . . . 7

**3 Related Work** **9**
3.1 Introduction . . . . . 9
3.2 Conceptual Framework . . . . . 9
3.3 Structure Learning as a Combinatorial Problem . . . . . 10
3.4 Continuous Optimization . . . . . 12
3.5 Graph Neural Networks and Expressive Power . . . . . 16
3.6 Conclusion . . . . . 17

**4 Methodology** **18**
4.1 Problem Statement . . . . . 18
4.2 Experimental Setup . . . . . 19
4.3 Data Generation and Ground Truth Networks . . . . . 21
4.3.1 Generating Data Samples . . . . . 21
4.3.2 Erdõs-Rényi Model . . . . . 21
4.3.3 Benchmark Datasets . . . . . 22
4.3.4 Linear Structural Equation Model . . . . . 22

4.3.5 Comparative Metrics . . . 23
4.4 Structure Learning . . . 25
4.4.1 BIC Score Function . . . 25
4.4.2 Search Algorithm . . . 27
4.5 Structure- and Position-aware GNN . . . 29
4.5.1 Preliminaries . . . 29
4.5.2 Overview of SP-GNN Architecture and Training . . . 30
4.5.3 Thresholding . . . 32

**5 Results** **33**
5.1 Graph Scoring using SP-GNN . . . 33
5.1.1 *k*-Fold Cross-Validation . . . 33
5.1.2 Predictive Capabilities . . . 34
5.1.3 Scalability . . . 38
5.2 Effect of Data Sample Sizes . . . 40
5.2.1 Hyperparameter Tuning . . . 40
5.2.2 Encoder/Decoder . . . 40
5.2.3 Results . . . 41
5.3 Model Selection Guidance . . . 48
5.3.1 Learning Process . . . 48
5.3.2 Results . . . 48

**6 Impact and Implications** **50**

**7 Limitations and Future Work** **52**

**8 Conclusion** **54**

**References** **61**

# List of Figures


1.1 Structure Learning Problem . . . . . 2

2.1 Neuron Diagram . . . . . 8

3.1 Bayesian Network . . . . . 10
3.2 Traditional Algorithm Comparison . . . . . 11
3.3 RL-BIC . . . . . 13
3.4 RL-BIC . . . . . 14
3.5 P-GNN . . . . . 16

4.1 Structure Learning Problem . . . . . 19
4.2 SP-GNN . . . . . 29

5.1 k-Fold Cross-Validation . . . . . 34
5.2 Average Score Difference . . . . . 39
5.3 Sample Size Results Comparison . . . . . 42
5.4 25 Nodes . . . . . 43
5.5 50 Nodes . . . . . 44
5.6 100 Nodes . . . . . 45
5.7 SP-GNN . . . . . 48

# List of Tables


4.1 Graph Characteristics . . . . . 22
4.2 Benchmark BN Characteristics . . . . . 22

5.1 GNN Results . . . . . 35
5.2 Raw Data . . . . . 36
5.3 BIC Scores vs SP-GNN Scores on Benchmark Networks . . . . . 37
5.4 Awareness Scores . . . . . 37
5.5 Score Ranges . . . . . 38
5.6 Heatmap . . . . . 49

## List of Abbreviations

| Abbreviation | Meaning | Page |
|---|---|---|
| AI | Artificial Intelligence | 3 |
| AIC | Akaike Information Criterion | 26 |
| ANN | Artificial Neural Network | 1 |
| BDe | Bayesian Dirichlet equivalent | 26 |
| BDeu | Bayesian Dirichlet equivalence uniform | 10 |
| BIC | Bayesian Information Criterion | 26 |
| BN | Bayesian Network | 1 |
| CNN | Convolutional Neural Network | 2 |
| GGNN | Causal Generative Neural Network | 12 |
| CPD | Conditional Probability Distribution | 4 |
| CPT | Conditional Probability Table | 4 |
| DAG | Directed Acyclic Graph | 1 |
| DAGMA | Directed Acyclis Grahs via M-matrices for Acyclicity | 15 |
| DAGs with NO TEARS | DAGs with Non-combinatorial Optimization via Trace Exponential and Augmented lagRangian for Structure Learning | 12 |
| ELBO | Evidence Lower BOund | 13 |
| GAN | Graph Attention Network | 1 |
| GFlowNets | Generative Flow Networks | 14 |
| GNN | Graph Neural Network | 1 |
| GPU | Graphical Processing Unit | 1 |
| I-GNN | Identity-aware Graph Neural Network | 17 |
| IID | Independently and Identically Distributed | 1 |
| L-BFGS | Limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm | 12 |
| MAE | Mean Absolute Error | 23 |
| MAP | Maximum a Posteriori | 6 |
| MCMC | Markov Chain Monte Carlo | 14 |
| MLP | Multilayer Perceptron | 1 |
| MMHC | Max-Min Hill Climbing | 11 |
| MMPC | Max-Min Parent-Child | 11 |
| MPE | Most Probable Explanation | 6 |
| MRF | Markov Random Fields | 4 |
| NP-hard | Non-deterministic Polynomial-time hard | 1 |
| PGM | Probabilistic Graphical Model(ling) | 3 |
| P-GNN | Position-aware Graph Neural Network | 17 |
| ReLU | Rectified Linear Unit | 1 |
| RL | Reinforcement Learning | 1 |
| RMSE | Root Mean Squared Error | 23 |
| RNN | Recurrent Neural Network | 16 |
| SEM | Structural Equation Model | 13 |
| SHD | Structural Hamming Distance | 23 |
| SP-GNN | Structure- and Position-Aware Graph Neural Network | 29 |
| STD (Std Dev) | Standard Deviation | 23 |
| TDPA | Three Phase Dependency Analysis | 12 |

# Chapter 1

# Introduction

Graphical models have become popular and widely used in the field of machine learning owing to its ability to model uncertainty using probability theory and confront complexity using graph theory. One of the most common types of graphical models, Bayesian Networks (BNs), is represented with a directed acyclic graph (DAG) $\mathcal{G}$, where each node is a random variable within a domain and the edges correspond to the influence one random variable has over another. Learning the structure of DAGs from data however, is an $\mathcal{NP}$-hard problem due to an enforcement of acyclicity constraint and a super exponentially growing search space with increasing numbers of random variables as shown in Figure 1.1 on the following page.

The structure learning problem has consequently stimulated a proliferation of literature. Score-based methods present this problem as the optimization of a score function with respect to some model $G$ and a dataset of observed samples. Super exponential search spaces are intractable and have rendered exhaustive techniques infeasible, posing a challenge for optimized searching. Much research has taken up the challenge of devising novel and creative strategies to circumvent the presented challenges, such as adding background structural assumptions, placing constraints on either the search space or graph structure, or the employment of distributed computing for parallel algorithms.

With the advent of GPUs and increased computing power, there has been an increase in popularity of using black-box deep learning models for the development of continuous optimization approaches for structural and causal discovery. This was demonstrated by changing its nature from a combinatorial one over search spaces to a continuous one over weighted adjacency matrices. Despite its inability to return a global optimum, the demonstrated success of such a local solver being comparable to more expensive global combinatorial searches has inspired research in this field, including the use of Graph Neural Networks (GNNs) for causal discovery in DAGs, or more intensive reinforcement learning techniques and actor-critic algorithms.

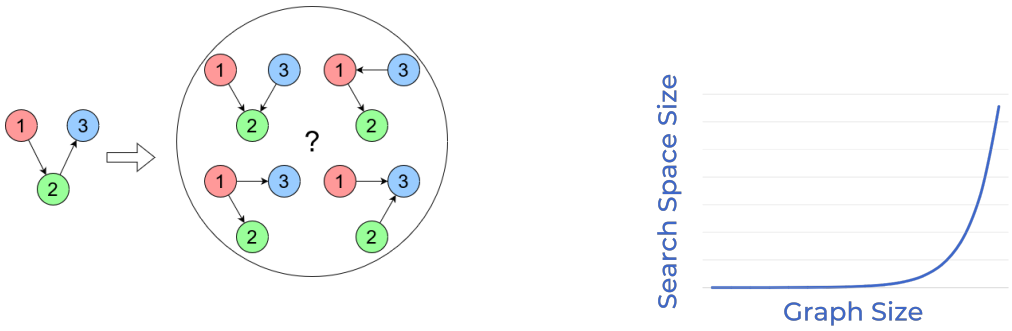



Figure 1.1: Combinatorial structure learning problem search space growing superexponential in number of nodes.

A non-trivial factor when considering the increased popularity of continuous optimization approaches is the tremendous amount of progress made in the deep learning paradigm. Works such as those presented by Yu *et al.* [2019] are rooted in the growing advancement of the aforementioned GNNs. Motivated by the want for a more generic framework suitable for non-Euclidean graph structured data than that of Convolutional Neural Networks (CNNs), GNNs have proven to be effective for the representation of graph structured data and, subsequently, representation learning of graphs.

Despite empirically achieving state-of-the-art performance in tasks such as graph classification, link prediction and node classification, many GNN variants lack discriminative power with regards to graph topology. This became an active area of research to more adequately capture structural/positional information of nodes within graphs to increase expressiveness. Position-aware GNNs use randomly sampled anchor nodes to learn a distance-weighted aggregation scheme with target nodes, while Identity-aware GNNs extends structural information gathering by encoding node identities during message passing.

Between the rapidly growing variants of GNNs displaying more discriminative power and the continued success of continuous optimization approaches for causal discovery, this paper proposes Neural Graph Scoring, a new approach that introduces the end-to-end deep learning paradigm to the structure learning problem of graphical models, utilizing the highly expressive Structure- and Position-aware GNN (SP-GNN) variant to learn to score graph structures based on the Bayesian Information Criterion (BIC).

The graph-scoring framework has several distinct advantages. Firstly, it incorporates deep learning models (specifically SP-GNN built on Graph Attention Network architecture) for increased structural and positional discrimination of graphs, improving the model selection process when scoring models. Secondly, the framework is distribution-agnostic, meaning it is capable of processing data generated from multiple distributions (to varying efficacies). Third, the constructed model is independent of the approach used for data aggregation. This approach makes use of pair encoder/decoder modules, however, should a more suitable option arise, it can be easily implemented with no loss in performance.

# Chapter 2

# Background

## 2.1 Probabilistic Graphical Models

This section explores the basics of Probabilistic Graphical Models (PGMs), types of PGMs that are commonly used and core concepts.

### 2.1.1 Early Ages of Artificial Intelligence and Probability Theory

Probability Theory [Rényi 2007] and Graph Theory [Bondy *et al.* 1976] have become crucial building blocks in modern intelligent systems, but this was not always the case. In the early years of artificial intelligence (AI), researchers were more concerned with the development of general purpose procedures primarily based on intuition as opposed to probability and statistics. Although probability theory was seen to be of use for procedures that handle uncertainty, it was ruled out due to the assumption that a complete set of conditional parameters were required but rarely available [Shortliffe and Buchanan 1975].

This lead to the creation of a new formalism called *certainty factors* which aimed at providing reasoning with available uncertainty rules, but was not subject to the requirements of the Theory of Probability. This was used by expert systems such as INTERNIST-1 [Miller *et al.* 1985] and INFERNO [Quinlan 1982], but proved to produce important inconsistencies as a result of using wrong information contexts.

This changed when it was realized that while this approach was adequate for simple problems, it was unsuitable for more complex situations [Buchanan and Shortliffe 1984]. The focus changed to the adoption of well-founded theories in general sciences, a point at which probability and graph theory began contributing to the field in a more substantial manner. It was not until the 1980s that researchers began working on probabilistic models based on knowledge in statistics. The work of Pearl [2022] is considered the inception of probabilistic models in AI wherein he argued that structuring knowledge pertaining to a problem as a tree can be used for reasoning without being susceptible to circular inferences.

### 2.1.2 Graphical Models

After the book *Probabilistic Reasoning in Intelligent Systems* [Pearl 2014], the viability of probability in AI accelerated development of graphical models. They provided a language for representing complex situations with efficient computation and the estimation of only necessary parameters based on observations. Employing graphs to describe these models had been a long-standing practice, but most relevant literature highlighted its benefits when using graphs to represent relationships [Darroch *et al.* 1980; Wermuth and Lauritzen 1982; Dawid 1979; Cannings *et al.* 1978].

PGMs are the marriage of the two fields in a manner that allows us to reason about uncertainty in complex intelligent systems. They are a natural tool for dealing with two consistently reoccurring problems throughout applied sciences such as mathematics and engineering: uncertainty and complexity. Comprised of these two fields, this contributes to the modular nature of graphical models: combining simpler elements to create a complex system. Probability theory ensures that systems are consistent and provides a way of interfacing models with data, while graph theory offers an intuitive human-legible method of modelling variables and identifies as a data structure perfectly suited for efficient general-purpose algorithms.

PGMs are graphs where nodes represents random variables and and the egdes between them represent conditional dependence assumptions, providing a compact representation of a joint probability distribution. PGMs are comprised of two variants: directed and undirected. Directed models (BNs, causal models) are the more popular option amongst machine learning and AI communities. Undirected models (Markov networks or Markov Random Fields - MRF) are predominantly used by physics communities and image-related domains. There does exist a combination of the two wherein graphs possess both dirceted and undirected edges, also known as chain graphs.

### 2.1.3 Representation

#### Directed Models

In directed models, edges between nodes can be interpretted as one random variable depending on another, or one influencing the outcome of another. This encoding of conditional independence asusmptions within directed models can be stated simply as follows: a node is said to be independent of its ancestors given its parents, where the ancestor/parent relationships corresponds to some fixed topological ordering of the nodes [Murphy 2001].

At the center of BNs are DAGs where, as stated previously, each node corresponds to a random variable and each edge the influence between them. Let us consider a BN graph $\mathcal{G}$ over a set of random variables $\mathbf{X} = [X_1, \ldots, X_n]$. Each random variable $X_i$ has a corresponding *conditional probability distribution* (CPD) $P(X_i|\text{Pa}_{X_i})$ where $\text{Pa}_{X_i}$ are the parent nodes of $X_i$ in the graph. CPDs have a variety of ways in which they can be represented, but the most common is by way of using tables, known as *conditional probability tables* (CPTs). Each row in the table corresponds to a possible configuration of the values of the parents of $X_i$, and each column the values that $X_i$ can assume.

A distribution $P_{\mathcal{B}}$ over **X** specifies the joint distribution - the probability of every possible event occuring according to the values of variables - given by the product

$$P_{\mathcal{B}}(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | \text{Pa}_{X_i}). \tag{2.1}$$

Representation of a joint distribution, either implicitly or explicitly, is a must for any probabilistic model. Given their could be exponentially many events, BNs factor the joint distribution for each variable according to $\mathcal{G}$. A BN can therefore be described as a pair $(\mathcal{G}, \theta_{\mathcal{G}})$ where $P_{\mathcal{B}}$ is a set of CPDs for each node in $\mathcal{G}$, denoted by $\theta_{\mathcal{G}}$, factored over $\mathcal{G}$. This is known as the *global semantics* of a BN and equips us with the power to determine the possibility of any possible permutation of variable assignment. There is also an equivalent *local semantics* encoded by a BN which asserts that every variable is independent of its nondescendents given its parents (also called independence assertions or Markov assumptions).

### Undirected Models

Undirected models on the other hand model a variety of phenomena in which directionality cannot be described. This offers a simpler perspective with respect to independence assertions and inference tasks. *Markov networks* or *Markov random fields* are the most common undirected graphical models and parameterization thereof is done in a vain similar to how CPDs parameterize directed graphical models, though less intuitive. Definition of a joint distribution in MRFs consists of the maximal cliques in the graph, a potential function on the clique (clique potentials) and a normalization factor. It is easy to misconstrue clique potentials as representing marginal probabilities, but this is incorrect.

Independence assumptions in MRFs can be separated into 2 classes: local Markov properties and global Markov properties. Local Markov properties corresponds to each individual node and the conditioning of influence based on its immediate neighbours which can be potentially blocked off. The Markov assumption would state that every node is independent of any other node that is not its immediate neighbour. Global Markov properties are based on active paths within the network and nodes are separated if there does not exist an active path between them. While undirected graphs will not be explored further given that they were not used during the research, this serves as a soft reminder of the other facet of PGMs.

### 2.1.4 Inference

Given some set of observed nodes, the goal of inference is to estimate the values of hidden nodes given the values ascribed to the observed nodes. If we observe the roots of a model and try to infer the leaves, this is known as top-down reasoning, and the inverse of observing the leaves attempting to infer the roots is known as bottom-up reasoning. Directed and undirected models alike face the same queries they must answer with their respective joint distributions. Inference is another $\mathcal{NP}$-hard problem where computing posterior estimates is computationally intractable.

There are two types of queries commonly directed at graphical models: (1) conditional probability queries, and (2) most probable assignment. The conditional probability query can be mathematically described as follows:

$$P(\mathbf{Y}|E = e) = \frac{P(\mathbf{Y}, e)}{e}. \tag{2.2}$$

This query consists of two parts: the evidence $E$ which is a subset of observed random variables within the network, and $e$ which is an instantiation of $E$; and $\mathbf{Y}$ which is a subset of the remaining unobserved variables in the network. The task is to then compute the probability distribution over the values of $\mathbf{Y}$ given that $E = e$. Variable elimination is one of the exact inference algorithms used to compute equation (2.2).

Most probable assignment is concerned with determining what the assignment of values to variables would be most probable to obtain a specific outcome. Once again we have $E = e$, but on this occasion we are interested in determining the most likely assignment of values to the subset of remaining variables. This problem can be broken into two variants: *most probable explanation* (MPE) queries and *maximum a posteriori* (MAP) queries. The former is the simpler of the two but serves as a special case of the latter. MPE queries attempts finding the most likely assignment for every non-evidence variable. In the event that there are multiple assignments with the highest posterior, either a single assignment can be chosen arbitrarily or the entire set of assignments can be returned.

In the case of MAP queries, we begin with a subset of variables $\mathbf{Y}$ that serves as the query. The task then shifts to finding the most likely assignment of variables in $\mathbf{Y}$, which can clearly be seen as a more general framing compared to MPE queries. Ultimately, MAP queries contain both elements of conditional probability queries and MPE queries, making it harder since it is computing a more complex function. Certain techniques and analysis applicable to MPE cannot be generalized to the MPE task either. These differences provides adequate reasoning for viewing MAP as a separate task alltogether.

Learning in PGMs is another crcuial task on the same level of importance and complexity as inference. This will be covered extensively in Section 3.3.

## 2.2 Deep Learning

This section explores deep learning and how it came to exist as a solution for complex hierarchical learning, as well as a breakdown of what neural networks are, how they are constructuted and how they function.

### 2.2.1 What is it?

As seen from the gradual implementation of probability theory, the early days of AI were a much different landscape compared to what it is today. The same can be said for how deep learning arose in the quest for more powerful mechanisms capable of learning from raw data.

For decades, machine learning techniques had to have their input data carefully engineered into more suitable representations (feature vectors) from which they could classify or detect patterns. Raw data being fed into these algorithms produced discouraging results, making representation learning infeasible. Deep learning methods were specifically designed to adapt existing learning systems for raw data, comprising of simple non-linear modules that would transform the representation of an input from one level to the next with each subsequent level being slightly more abstract than the previous. Sufficient transformations allows the learning of complex functions that can be used for classification and prediction.

More formally presented - deep learning is a hierarchy of concepts that allows computers to learn complex and complicated concepts by building them out of simpler ones [Bengio *et al.* 2017]. Visualizing these concepts as a graph will show that it is deep and complex with multiple layers to it, allowing for the gathering of knowledge from experience. Deep learning has proven to be revolutionary in solving problems that have otherwise stumped the AI community for years given its aptitude for discovering intricate structures in complex data. It has shown remarkable success in image recognition [Krizhevsky *et al.* 2012; Tompson *et al.* 2014], speech recognition [Mikolov *et al.* 2011; Hinton *et al.* 2012] and has produced extremely strong results in the field of natural language processing [Vaswani *et al.* 2017].

### 2.2.2 Neural Networks

Neural networks, the backbone of what makes deep learning possible, were originally inspired by the functioning of the human nervous system. First, let us take a look at some basic neurology: the human brain consists of approximately $10^{11}$ or 100 billion *neurons*, each of which communicate via electric signals within the cell walls or *membranes*. *Synapses* mediate interneuron connections at electrochemical junctions at the branches of a cell, called *dendrites*. Each neuron receives thousands of signals from other neurons and must therefore be aggregated or summed together. The resulting aggregated signal determines if the neuron will generate a response (fire its own signal) based on whether it exceeds some threshold. Should it exceed that threshold, a signal is then transmitted via a branching fibre known as the *axon*.

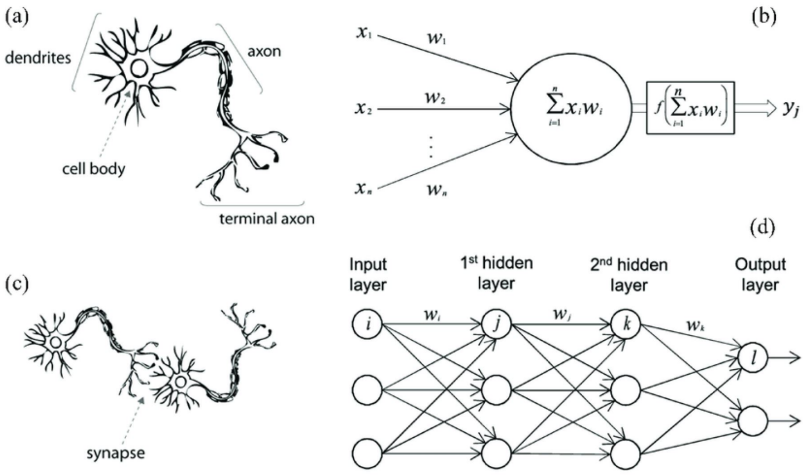



Figure 2.1: Example of a Neural Network and its Resemblance to Human Neurons

Figure 2.1 shows just how closely the structure of an *artificial neural network* (ANN) resembles the structure of neurons in the human brain. Consisting of an input layer, one or more hidden layers and an output layer, this layered architecture is made up of neurons that receives some input $X = [x_1, \ldots, x_n]$ (except the input layer), processes said input by applying weights $W = [w_1, \ldots, w_n]$ to each input as well as any potential bias the neuron has, performs some mathematical operation on this aggregated sum (known as an activation function), and passes the resulting output to the next layer (except the output layer). This can be mathematically described as:

$$y_k = F\left(\sum_{i=0}^{m} x_i \cdot w_i + b\right), \tag{2.3}$$

where $x_i$ is the $i$-th input for the current neuron, $w_i$ is the $i$-th weight that will be applied to the respective input, $b$ is the bias of that neuron if any, and $F$ is the activation function applied to the output.

This is known as a feed-forward network (sometimes called *multilayer perceptrons* - MLPs) wherein input is propogated forward through the network to obtain, sometimes multiple, outputs (forward propagation). Back propagation is intuitively the reverse process: once a forward pass has been completed and an output is obtained, the "blame" for the output being incorrect (error) is propagated back to the provider wherein its weights and biases are adjusted according to the strength of the connection between the provider and itself. This algorithm is used to train neural networks and reduce the difference between actual and predicted outputs.

Although the design and ruleset of neural networks appears to be simple, the potential of this model is put on full display when its architecture is expanded upon. Increasing the number of hidden layers, the number of neurons within each layer, using different activation functions and adding additional convolution layers are just some of the ways the standard neural network architecture has been expanded upon to meet the needs of different computational tasks. As will be seen later, GNNs are just on of the many configurations developed for the purpose of learning from graph data.

# Chapter 3

# Related Work

## 3.1 Introduction

The structure learning problem has been a focal point of graphical models, attempting to discover causal relationships present within raw data. The need to find effective and efficient approaches to do so is the cornerstone of much of the literature surrounding this problem, ranging from traditional combinatorial search and score-based methods to continuous optimization. Challenges arising with methods in the former have inspired the search for solutions in the latter based on the modern widespread success of black-box deep learning models.

This section examines some of the vast amount of literature pertaining to this problem: Section 3.2 describes the motivating reasons behind turning to deep learning for graph scoring, Section 3.3 explains the problem of structural learning in its traditional combinatorial nature and some of the issues posed by solutions to combat it. Section 3.4 discusses the turn to continuous optimization seen in recent years, followed by the technologies that have made it possible in Section 3.5.

## 3.2 Conceptual Framework

GNNs have demonstrated significant capability in learning and optimizing graph structures through their advanced mechanisms for graph representation, message passing, and aggregation [Hamilton *et al.* 2017]. By leveraging supervised approaches, GNNs can effectively associate graph structures with numerical scores, optimizing the network structure during the learning process. The combination of empirical success through tasks such as link prediction [Zhang and Chen 2018] and graph classification [Hu *et al.* 2020] in the field of chemistry for molecular property prediction (toxicity, binding affinity, etc.) [Yang and Buehler 2022] through its expressive power by way of approximation and information propagation [Chen *et al.* 2023] highlights the potential of GNNs in the structure learning problem.

## 3.3 Structure Learning as a Combinatorial Problem

Graphical models have become popular and widely used in the field of machine learning owing to its ability to model uncertainty using probability theory and confront complexity using graph theory. The applications of graphical models span far and wide, from image processing [Zhang and Ji 2009] and medical applications [Kalisch *et al.* 2010] to risk modelling [Wang and Williams 2010] and genetics [Dobra *et al.* 2004]. One of the most common types of graphical model, BNs, are represented with a DAG, wherein each node is a random variable within a domain and the edges correspond to the influence one random variable has over another.

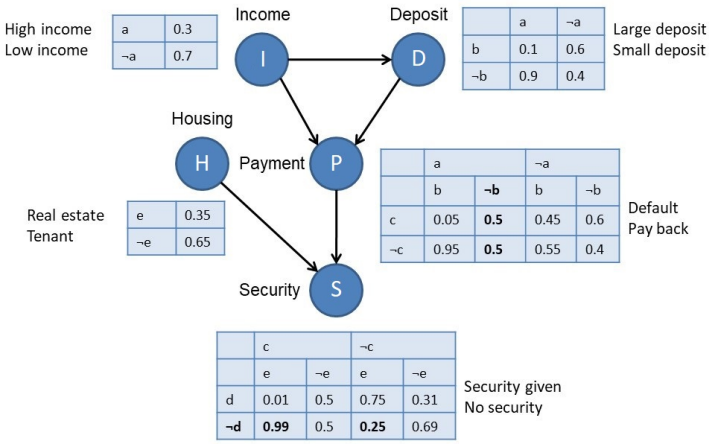



Figure 3.1: Simple Bayesian Newtork with Conditional Probability Tables for every Random Variable

Consider some DAG $G$ and a joint probability distribution $\mathcal{P}$. $G$ and $\mathcal{P}$ are said to be *faithful* to each other if $G$ encodes every conditional independency present in $\mathcal{P}$. This faithfulness condition [Uhler *et al.* 2013] allows the recovery of $G$ from $\mathcal{P}$. However, learning the structure of $G$ from data $D$ sampled identically and independently from an unknown $\mathcal{P}$ is an $\mathcal{NP}$-hard problem [Chickering *et al.* 2004] due to the constraint of enforcing acyclicity and a super exponentially growing search space with increasing numbers of random variables. Simply put, the structure learning problem is one in which one attempts to recover $G$ from $D$ by employing some combinatorial search.

Traditionally, the structure learning problem has been viewed as a combinatorial optimization problem in which a graph is assigned a score $\mathcal{S}(G)$ based on observed data and then search for the best scoring graph:

$$\min_G \mathcal{S}(G), \text{ subject to } \mathcal{G} \in \text{DAGs}, \tag{3.1}$$

where $\mathcal{S} : \mathbb{R}^{n \times n} \to \mathbb{R}$ is a score function and $n$ is the number of nodes in $G$. Mathematicians and statisticians alike have developed many well-defined score functions [Liu *et al.* 2012], including the Bayesian Information Criterion (BIC) score [Schwarz 1978], Akaike Information Criterion (AIC) score, [Akaike 1973], Bayesian Dirichlet equivalent (BDe) score [Heckerman *et al.* 1995], Bayesian Dirichlet equivalence uniform (BDeu) score [Buntine 1991] and others [De Campos and Friedman 2006].

| Algorithm | Sample size (SS) 500 | 1000 | 5000 | Average over SS |
|---|---|---|---|---|
| MMHC | 1.000 (22) | 1.000 (22) | 1.000 (22) | 1.000 |
| OR1 $k = 5$ | 1.016 (19) | 1.019 (18) | 1.021 (17) | 1.019 |
| OR1 $k = 10$ | 1.014 (19) | 1.016 (18) | 1.023 (16) | 1.018 |
| OR1 $k = 20$ | 1.016 (19) | 1.022 (18) | 1.022 (16) | 1.020 |
| OR2 $k = 5$ | 1.005 (19) | 1.009 (18) | 1.011 (16) | 1.008 |
| OR2 $k = 10$ | 1.000 (18) | 1.009 (18) | 1.003 (16) | 1.004 |
| OR2 $k = 20$ | 1.009 (18) | 1.010 (18) | 1.006 (16) | 1.009 |
| SC $k = 5$ | 0.999 (21) | 0.996 (22) | 1.016 (18) | 1.004 |
| SC $k = 10$ | 1.010 (13) | 1.016 (13) | 1.014 (13) | 1.013 |
| GS | 0.976 (20) | 0.983 (20) | 0.991 (20) | 0.984 |
| PC | 1.275 (18) | 1.234 (18) | 1.195 (20) | 1.235 |
| TPDA | 1.406 (21) | 1.367 (21) | 1.206 (22) | 1.326 |
| GES | 1.083 ( 7) | 1.007 ( 6) | 1.118 ( 6) | 1.070 |
| Empty Graph | 1.498 (22) | 1.520 (22) | 1.546 (22) | 1.521 |
| True Graph | 0.977 (22) | 0.990 (22) | 1.007 (22) | 0.991 |

Figure 3.2: BIC scores normalized for MMHC algorithm in comparison with Sparse Candidate (SC), Optimal Reinsertion (OR), Greedy Search (GS), PC, Greedy Equivalence Search (GES) and Three Phase Dependency Analysis (TDPA) algorithms (see [Tsamardinos *et al.* 2006] for more).

Many model selection algorithms to learn DAGs [Drton and Maathuis 2017] highlights the challenges of structure learning. One class of causal discovery is search and score-based methods such as greedy hill-climbing which focuses on local heuristics to search for the best scoring graph using the aforementioned score functions, making incremental modifications that preserve acyclicity. Unfortunately, this algorithm tends to locate a local maximum with infinite samples, and has been modified to be better at finding the global maximum. This includes random restarts which, when stuck, applies several random operations on the graph and restarts the greedy search.

A common problem search and score-based methods run into is intractable search spaces. The class of constraint-based methods seeks to circumvent this by concentrating on testing the conditional independence of random variables against each other. This places constraints on candidate graphs, and the graph chosen is one that maximizes the number of constraints satisfied. Constraints include assumptions such as tree-like structures [Chow and Liu 1968], limiting subsets of viable parent nodes [Tsamardinos *et al.* 2003] and more.

There has also been interest in hybrid algorithms that combine the principles of both classes such as the Max-Min Hill-Climbing (MMHC) algorithm proposed by Tsamardinos *et al.* [2006]. This algorithm estimates a number of parent nodes on a node-to-node basis and utilizes the Max-Min Parents and Children (MMPC) algorithm [Tsamardinos *et al.* 2003] for structure discovery and Markov blankets for edge orientation. Another more recent approach is distributed parallel computing using GPUs to computation-

ally accelerate existing algorithms such as the common PC algorithm [Zarebavani *et al.* 2019; Spirtes *et al.* 2001].

While these solutions have displayed marked success over the years, many problems still persist such as a tendency to find a local maximum rather than a global, a need for graphs faithful to a distribution, models specific to discrete variables or powerful computational hardware inaccessible to the average individual. As such, a turn to the recently widely successful black-box models for novel approaches has attracted the interest of researchers.

## 3.4 Continuous Optimization

Attempting to directly solve optimization problems using iterative techniques such as gradient descent to optimize highly parameterized networks has become widespread in the field of deep learning [Andréasson *et al.* 2020], in no small part due to the advent of GPUs and the increased computing power they offer. This method of problem solving has led to the confluence of structure discovery and black-box deep learning models, recasting the traditionally combinatorial model selection problem to a continuous optimization one.

Zheng *et al.* [2018] is generally considered the first to propose this approach with DAGs with NO TEARS (*Non-combinatorial Optimization via Trace Exponential and Augmented lagRangian for Structure Learning*), transforming the discrete search procedure into a continuous equality constraint by introducing a new smooth characterization for acyclicity. Equation 3.1 is the reformulated as a continuous optimization problem with respect to a weighted adjacency matrix that represents graphs:

$$\begin{aligned} &\min_{W \in \mathbb{R}^{n\times n}} F(W) \\ &\text{subject to } \mathrm{G}(W) \in \text{DAGs} \end{aligned} \quad \Longleftrightarrow \quad \begin{aligned} &\min_{W \in \mathbb{R}^{n\times n}} F(W) \\ &\text{subject to } h(W) = 0, \end{aligned} \tag{3.2}$$

where $\mathrm{G}(W)$ represents the $n$-node graph of weighted adjacency matrix $W$, $F$ is a score function and $h : \mathbb{R}^{n\times n} \to \mathbb{R}$ is a smooth function over real matrices which, when enforcing acyclicity, is derived as:

$$h(W) = \mathrm{tr}(\exp^{W\circ W}) - n = 0, \tag{3.3}$$

where $\circ$ is the Hadamard product and $\exp^A$ is the matrix exponential evaluated using the $O(n^3)$ algorithm developed by Al-Mohy and Higham [2010], a computational disadvantage that sought to be remedied in future methods. DAGs with NO TEARS also utilizes a loss function appropriate for encouraging sparsity, in this case least-squares with an $l1$ penalty, an objective maximized by, as the name implies, the Augmented Lagrangian method [Nemirovsky 1999] with L-BFGS matrices [Byrd *et al.* 1995].

The integration of neural networks surfaced recently with Goudet *et al.* [2018] proposing a generative adversarial network (GAN) - CGNN [Goodfellow *et al.* 2014] - to learn the distinction of joint distributions on a node-to-node basis. A significant motivator for neural network use is the absence of restrictions on the form of a priori, "letting the

data speak for itself". This approach also incorporates a search algorithm such as hill-climbing or TABU search for structure optimization. Much like DAGs with NO TEARS, thresholding functions are used but to regularize the number of edges in a graph in this instance, as opposed to preventing the addition of cycle-inducing edges. A failure to enforce acyclicity proved to be a substantial drawback and a future point of research.

Subsequent works such as the DAG Graph Neural Network (DAG-GNN) [Yu *et al.* 2019] extends both DAGs with NO TEARS and CGNN, adopting an Evidence Lower BOund (ELBO) based score function, while integrating a novel GNN architecture:

$$X = f_2((I - W^T)^{-1} f_1(Z)), \tag{3.4}$$

where $X \in \mathbb{R}^{n \times d}$ is a sample of a joint distribution and $Z \in \mathbb{R}^{n \times d}$ is a noise matrix. This equation becomes a generalized version of the linear structural equation model (SEM) if the parameterized function $f_2$ is invertible, allowing the recovery of samples through the parameterization of the variational autoencoder model. DAG-GNN uses this neural network architecture to model causal relationships exhibited in observations and boasts its ability to handle vector-valued random variables just as well as discrete random variables. This is demonstrated on synthetic data sampled using the linear SEM, as well as the protein and phospholipid datasets.

Although works such as the DAG-GNN show impressive success in learning DAG structures not only for both linearly and nonlinearly generated samples but also closer to global optima, effective score functions have very complicated equivalent loss functions or cannot be represented in closed forms. This challenge inhibits assimilation with these proposed models and has promoted the search for solutions in the world of machine learning outside of deep learning.

The Reinforcement Learning (RL) paradigm is an area of machine learning that focuses on maximizing reward through the execution of a specific sequence of actions with a focus on unlabelled training data [Kaelbling *et al.* 1996]. It is about learning an optimal behavioural policy for an unknown environment to maximize rewards or returns. Zhu *et al.* [2019] opted for this approach to causal discovery with RL-BIC, adopting the actor-critic REINFORCE algorithm [Sutton *et al.* 1999] in which directed graphs are generated using an encoder-decoder neural network (actor) and updated while fine-tuning the parameters of the neural network as a means of optimization (critic). Candidate graphs are scored using the BIC score function to generate reward signals and are penalized if they are not acyclic.

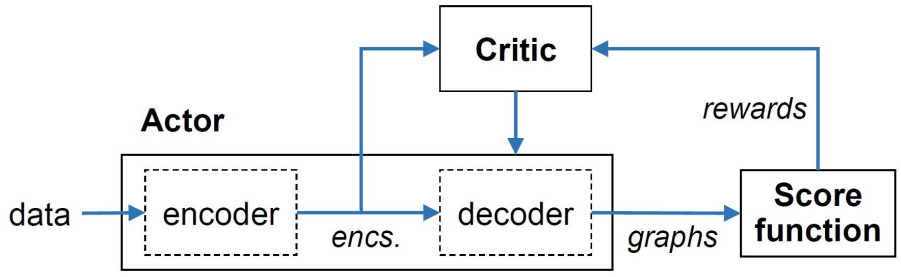



Figure 3.3: Reinforcement Learning for Score-Based Causal Discovery

The main motivation behind the turn towards RL for causal discovery lies in the ability of an RL agent with stochastic policies to determine the optimal search location amongst all generated graphs, using RL as a search strategy rather than the traditional application of learning a policy. Policy gradient and stochastic methods are employed for weight training, and the output is the best scoring DAG from all those generated during training represented with a binary adjacency matrix. An unfortunate shortcoming of RL-BIC is the lack of an efficient method of scoring graphs, often taking longer than actual training time, as well as poor scalability.

Some have even opted for utilizing probabilistic models themselves during the structure learning process. Generative Flow Networks (GFlowNets) are one such example introduced by Bengio *et al.* [2021]. GFlowNets are generative models over composite and discrete objects for "building objects". Rather than generating a sequence of actions that will maximize the return, the philosophy is to sample a number of action sequences that have a probability proportional to the possible return: generating compositional objects at a frequency proportional to their associated reward. Structurally they are similarly equivalent to Markov Reward Processes (decision processes with deterministic dynamics - [Puterman 2014; Hastings 1973]).

Generation of samples in GFlowNets is treated as a sequential decision problem, particularly appealing for graph reconstruction when modelling distributions over graphs. Deleu *et al.* [2022] presented the use of GFlowNets as an alternative to Markov Chain Monte Carlo (MCMC) where graph construction is treated as a sequential task (sequentially adding edges to an empty graph) based on learned transition probabilities - DAG-GFlowNets. Despite adopting RL principles in its work, this sequential approach differs from that of Zhu *et al.* [2019] attempting to learn and predict whole graphs.

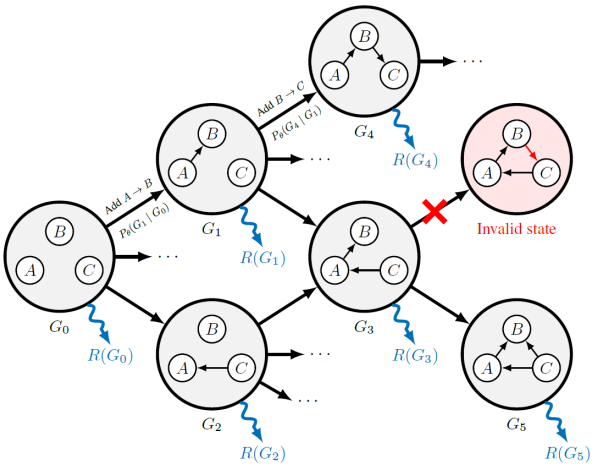



Figure 3.4: States of a GFlowNet corresponding to DAGs

The GFlowNet used is parameterized by forward transition probabilities (similar transition functions in a Turing machine) $P_\theta(G_{t+1}|G_t)$ where $G_t$ is the current state and $G_{t+1}$ is the next (potentially terminal) state. The probabilities are defined using 2 neural networks and the probability of transition from one state to the next $G \to G'$ is given by:

$$P_\theta(G'|G) = (1 - P_\theta(s_f|G))P_\theta(G'|G, \neg s_f) \tag{3.5}$$

where $P_\theta(s_f|G)$ is one neural network modelling the probability of termination, and $P_\theta(G'|G, \neg s_f)$ is the other neural network modelling the probability of state transitions. DAG-GFlowNets are shown to be consistently competitive against other MCMC methods without a loss of computational efficiency or predictive accuracy, with a predictive log-likelihood concentrated near that of the ground truth DAG. Unfortunately, the GFlowNet has less than ideal scalability due to the difficulty experienced when fitting the detail-balance loss with larger graphs.

A common thread present in all the literature mentioned regarding the use of continuous optimization for the prediction of DAGs from data is their differentiable characterization of acyclicity using the traces of matrix exponential functions [Zheng *et al.* 2018]. This is based on the idea that adjacency matrix powers encodes walks and cycles information. Mathematically defined as

$$h_{\text{expm}}(W) = \text{Tr}(e^{W \circ W}) - d, \tag{3.6}$$

this was later expanded on by Yu *et al.* [2019] to improve computational speed using the identity matrix $I$ as follows:

$$h_{\text{poly}}(W) = ((I + \frac{1}{d} W \circ W)^d) - d. \tag{3.7}$$

As noted by Wei *et al.* [2020] both formulas are of the form $\text{Tr}(\sum_d^{k=0} c_p (W \circ W)^k) - d$ for some $c_k > 0$ and both adhere to the same property where $h_{\text{expm}}(W) = \text{Tr}(e^{W \circ W}) - d = 0 = h_{\text{poly}}(W) = ((I + \frac{1}{d} W \circ W)^d) - d$ if and only if $W$ corresponds to a DAG. This characterization of acyclicity using trace power matrices was used by all subsequent research in the field until Bello *et al.* [2022] proposed a new function based on the log-determinant function, which proved to be both smooth and exact. With a domain defined over the set of M-matrices [Ostrowski 1937] (hence *Directed Acyclic Graphs via M-matrices for Acyclicity* or DAGMA), this function was shown to be invex with its stationary points (global minima) corresponding to DAGs. The domain of M-matrices can be enlarged or shrunk using parameter $s$ according to the definition:

$$\mathbb{W}^s = \{W \in \mathbb{R}^{d \times d} | s > p(W \circ W)\} \tag{3.8}$$

This sets the ground for the log-determinant characterization as follows:

$$h^s_{ldet} = -\text{log det}(sI - W \circ W) + d\text{log}s, \tag{3.9}$$

where the DAG being searched for, $W$, is found within $\mathbb{W}^s$, i.e. $DAGs \subset \mathbb{W}^s$ and presents 3 arguments highlighting why it is preferable over existing acyclicity regularizers:

1. it does not diminish arbitrary length cycles.
2. it has better behaving gradients.
3. it has a faster computational speed.

DAGMA shows significant improvements in the speed and structural accuracy of learned DAGs compared to its competitors for linear and nonlinear SEMs with graph sizes up to 1000 nodes.

Despite producing remarkable results, many neural network-based approaches place a strong emphasis on the manner in which acyclicity is enforced, the recovery of data through the SEM model or the adjustment of weights and parameters in the neural networks used for encoding and decoding. There has been a distinct lack in the analysis of neural network models used in the proposed methods and the effectiveness at which they can express the graphs they intend to predict.

## 3.5 Graph Neural Networks and Expressive Power

The success of CNNs with data in Euclidean spaces sparked interest in a similar framework adapted for data in non-Euclidean spaces: graph data. The complexity of graph data imposed a significant challenge on existing machine learning techniques, given instances of graph data were no longer independent of each other, as is with 2D image data. Sperduti and Starita [1997] developed the first implementation of the application of recurrent neural networks (RNNs) to DAGs, a framework based on the idea of applying deep learning to non-Euclidean spaces [Bronstein *et al.* 2017].

Based on graph embeddings and CNNs, GNNs are capable of learning highly expressive low-dimensional representations of data through the extraction of multi-scaled localized features. Providing output information on multiple levels has been of great use for tasks involving node classification [Kipf and Welling 2016], link prediction [Zhang and Chen 2018] and graph classification [Zhang *et al.* 2018] offering substantial successful applications in fields such as physics [Sanchez-Gonzalez *et al.* 2018; Kipf *et al.* 2018], biochemistry [Yang and Buehler 2022; Zitnik and Leskovec 2017], information networks [Berg *et al.* 2017; Fan *et al.* 2019] and many more.

In recent years, there has been a surge of interest in GNNs for the representation learning of graph-structured data through advanced mechanisms involving graph-level pooling, message passing, and neighbourhood aggregation [Hamilton *et al.* 2017]. This is in stark contrast to traditional graph embedding learning methods such as matrix factorization using Laplacian eigenmaps [Anderson Jr and Morley 1985] and random walks [Grover and Leskovec 2016], offering greater abstraction of semantic information such as topological information and improved scalability [Ju *et al.* 2024].

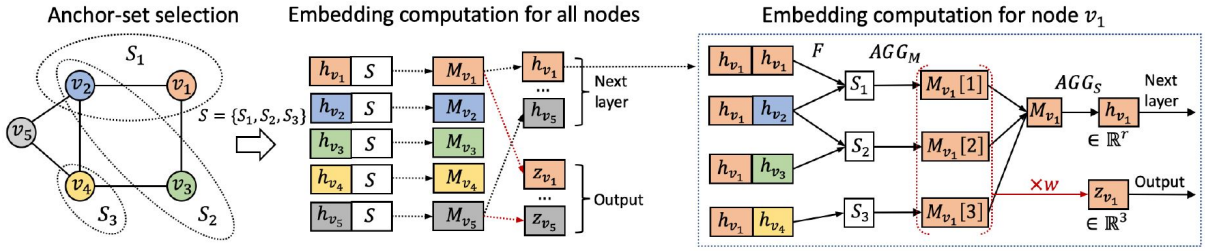



Figure 3.5: Position-Aware Graph Neural Network [You *et al.* 2019] architecture sampling anchor sets for node embeddings and aggregation using message passing functions.

Empirically, this approach has achieved state-of-the-art performance; however, many of these variants lack discriminative power with regards to graph topology. A framework designed by Xu *et al.* [2018] adopts the Weisfeiler-Leman graph isomorphism test Weisfeiler and Leman [1968] as a means of analysing and characterising the expressive power of GNN variants, formulating a mathematical study of their properties and establishing conditions for the creation of maximally powerful GNNs.

Upper-bounded by the 1-WL test, 2 limitations of GNNs arise: (1) the ego network of 2 nodes may reside in different local neighbourhoods and are indistinguishable; and (2) 2 nodes with identical local neighbourhoods but in different locations within a graph are indistinguishable. Variants that extend beyond the 1-WL test have thus become an active area of research to more adequately capture positional/structural information. Position-aware GNNs (P-GNNs) [You *et al.* 2019] use randomly sampled anchor nodes to learn a distance-weighted aggregation scheme with target nodes, while Identity-aware GNNs (I-GNNs) [You *et al.* 2021] extends structural information gathering by encoding node identities during message passing.

## 3.6 Conclusion

The vast amount of literature surrounding structure learning is not only an indication of it being an ongoing an active field of research in probabilistic graphical models and the vast amount of progress made, but also the magnitude of the challenges posed within the field and the room for improvement still obtainable.

Traditional combinatorial approaches to tackling the structure learning problem have yielded the most optimal results for many decades. However, they are not without issues, ranging from intractable super exponential search spaces resulting from high-dimensional data, constraints that become less effective with a relaxing of the faithfulness assumption, or the difficulty with efficiently enforcing acyclicity. The recasting of this problem to a continuous optimization one over real matrices has yielded promising results leveraging black-box deep learning models for structural causal discovery in directed graphs.

While it is an increasingly appealing framework, much of the innovation lies in the fine-tuning of parameters of models and adapted score functions utilized in these approaches, neglecting the abundance of deep learning models that provide increased expressive power. Subsequently, resorting to less powerful score functions has created a void necessitating improved graph scoring.

These challenges are ever present and ever growing, and the evolution of the statistical and machine learning world creates the opportunity of newer, more effective and efficient solutions that exploits these technologies in novel and innovative ways. The issues addressed having a more diversified array of potential solutions paves the way for broader applications and grows the number of domains in which they are viable.

# Chapter 4

# Methodology

This section goes into greater detail with regards to the experiments that were carried out over the course of this research to investigate the use of neural networks for graph scoring in structure learning. This will involve exploring the proposed design and architecture of GNNs to be integrated, the traditional algorithms that will serve as a benchmark for performance and the evaluation metrics for assessment of quality.

## 4.1 Problem Statement

Consider a dataset $\mathcal{D} = \{\xi[1], \ldots, \xi[M]\}$ where each data point is generated independently and identically distributed (IID) from some probability distribution $P^*(\mathcal{X})$ induced by a BN $\mathcal{G}^*$ over the random variables $\mathcal{X}$. The learning problem is to find the extent to which the independencies in $\mathcal{G}^*$ are manifested in the data $\mathcal{D}$. Figure 4.1 on the facing page provides a brief look at the process.

However, this problem is $\mathcal{NP}$-hard as the search space of BNs is combinatorial with a super exponential scaling with larger number of nodes —$2^{O(n^2)}$. This scaling proves a significant limitation on the size of models that can be efficiently learned, reducing the domains in which PGMs can be applied. Treading the line between reliable and accurate results, and the efficiency at which those results are produced is a fine one, and developing frameworks that do so effectively is a difficult but necessary task.

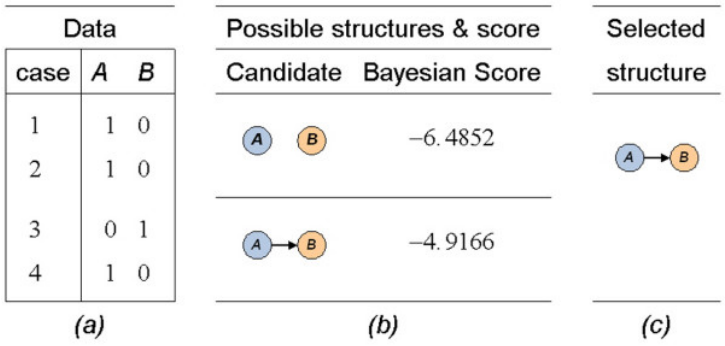



Figure 4.1: Basic representation of the structure learning process for Bayesian Networks.

## 4.2 Experimental Setup

This section gives a rundown of the experiments that will be conducted, as well as the manner in which they will be validated. Three research question are posed.

**Can deep learning be used for graph scoring in cases of structure learning that deal with high dimensional data compared to traditional approaches?**

To answer this research question, a dataset of DAGs of varying dimensionality (10/25/50/100 nodes) and sparsity will be generated using the Erdõs-Rényi model which involves obtaining weighted adjacency matrices. This will serve as the ground truth models ($\mathcal{M}^*$). Once generated, data will be sampled from the DAGs using a generalized linear model with a standard identity mapping for scalar-valued discrete random variables. The set of data samples may belong to any one of 5 distributions. The DAGs and their respective data samples will be split according to the 80-20 training-testing data split respectively for the training of the GNN and evaluated using k-fold cross validation.

Next a popular search algorithm: greedy hill-climbing with random restarts; and a popular score function: the BIC score function; will be implemented as benchmark approaches for comparison with the proposed approach. Then, an SP-GNN will be implemented as the proposed deep learning alternative where structural and positional encoders compute their respective awareness scores using ReLU activation function and CONCAT message passing functions with two 3 hidden-layer MLPs and 64 neurons per layer for feature fusion.

The trained SP-GNN will be tasked with scoring graphs learned using greedy hill-climbing with random restarts and compared with the scores of those using the BIC score function. Each respective score will be compared for an assessment of accuracy. The SP-GNN will also be used to guide the greedy hill-climbing process to determine the quality of models learned using the SP-GNN compared to the use of the BIC Score.

**What effect does the amount of training data available have on the ability of the GNN to classify and predict graphs in the case of structure learning?**

To evaluate this research question, a total of 5000 data points will be sampled for every ground truth network per distribution. For each category of ground truth networks, the SP-GNN will be given different amounts of data samples to train on per network (e.g. for 10-node graphs, the SP-GNN will train on 1000, 2000, 3500 and 5000 samples per graph, producing 4 different learned models). The results of each learned model will be compared with each other to gauge the importance of the amount of data available when scoring graphs, as well as determining which prior distributions require more data samples.

**How similar are models learned using the SP-GNN compared to models learned using the BIC score function?**

If a focal point is the development of a system that is capable of scoring graphs at a faster rate than traditional graph scoring methods, the quality of models learned using the SP-GNN must be evaluated. The SP-GNN will be tasked with scoring graphs produced during the greedy hill-climbing search algorithm in an attempt to learn a model. The learned model will then be compared to that of one which was learned using the BIC score function, evaluated using the Structural Hamming Distance (SHD) metric.

## 4.3 Data Generation and Ground Truth Networks

This section will outline the methods to be used for the generation of training and testing data. This involves the random generation of DAGs and the sampling of data points for each generated DAG. Pre-existing datasets will also be tested alongside the synthetic datasets. The ground truth models $\mathcal{P}^*$ represents the benchmark models against which the proposed deep learning model will be compared.

### 4.3.1 Generating Data Samples

#### Ground Truth Networks

This research involves the utilization of several different categories of DAGs of different complexities. This includes a variation in the number of nodes and edges within each DAG.

#### Generation and Sampling

The Erdõs-Rényi model for DAG generation will be used in which each edge will be assigned a random weight to obtain a weighted adjacency matrix $A$. The samples $\mathcal{D}^*$ will be generated using a generalized linear model:

$$\mathcal{D}^* = g(A^T\mathcal{D}^*) + Z \tag{4.1}$$

where $g$ is an identity mapping. $Z$ will serve as noise following a standard matrix normal to replicate real-world data.

#### Data Splitting

The data will be split according to the standard 80%/20% training and testing split respectively. It will follow the $k$-fold cross-validation for validation of models, with 10 folds being used (see Section 5.1.1 on page 33 for more on this process).

### 4.3.2 Erdõs-Rényi Model

There exists 2 variants of the Erdõs-Rényi model for DAG generation, the one to be used in this research will be based on the random reconstruction of a graph with edges added independently according to the probability of:

$$p^M(1-p)^{\binom{n}{2}-M}, \tag{4.2}$$

where $n$ is the number of nodes in the graph and $p$ is the probability that an edge is added. For an 80% connectivity, $p = 0.8$. Therefore, every graph generated has an average of $\binom{n}{2}p$ edges. As the degree of the graphs increases in node count, the distribution changes from Binomial to Poisson. The following table shows the configuration of the graphs to be generated.

| # Variables | In-Degree | Out-Degree | # Approx Edges | Data Type |
|---|---|---|---|---|
| 10 | 3 | 1 | 15 | Scalar-valued |
| 25 | 3 | 1 | 50 | Scalar-valued |
| 50 | 5 | 2 | 100 | Scalar-valued |
| 100 | 5 | 2 | 200 | Scalar-valued |

Table 4.1: Characteristics of ground truth networks to be generated.

### 4.3.3 Benchmark Datasets

Three common BNs will serve as a benchmark for the trained SP-GNN as a more concrete demonstration of the model's capabilities and in contrast to the synthetic datasets described previously. The following table presents the configuration of the benchmark networks:

| Network | # Nodes | # Edges | Average Degree | Maximum In-Degree |
|---|---|---|---|---|
| Asia | 8 | 8 | 2 | 2 |
| Alarm | 37 | 46 | 2.49 | 4 |
| Pathfinder | 109 | 195 | 3.58 | 5 |

Table 4.2: Characteristics of benchmark ground truth networks.

### 4.3.4 Linear Structural Equation Model

Let $W \in \mathbb{R}^{m \times m}$ represent the weighted adjacency matrix for a graph with uniformly assigned random edge weights, and $X \in \mathbb{R}^{m \times d}$ be a set of $d$ data points sampled from a joint distribution. Each variable $m$ can then be generalized to a $d$-dimensional vector given the current conditions, with the linear SEM model as follows:

$$X = A^T X + Z, \tag{4.3}$$

where $Z$ functions as the noise matrix following a standard matrix normal. The generated graphs will be used for the sampling of datasets $X^*$ using a generalized linear model $\mathcal{D}^* = g(A^T \mathcal{D}^*) + Z$ where $g$ is an identity mapping, generating a sample of size $d = 5000$.

Once the data has been sampled from the synthetic networks, partitions of the data will be made when conducting the experiments to analyse how the peformance of the SP-GNN differs when given varying amounts of training data, specifically $d \in \{1000, 2000, 3500, 5000\}$.

### 4.3.5 Comparative Metrics

For comparison of the proposed models $\mathcal{P}$ with the ground truth models $\mathcal{P}^*$, the following metrics will be used:

#### Structural Hamming Distance

Measurement of graph structure deviation between learned models and ground-truth models using Structural Hamming Distance (SHD). The SHD is, as implied by the name, a structural distance metric that quantifies the differences between 2 graphs by means of their adjacency matrices. Specifically, it is a count of the number of edits required to transform one graph into another, written mathematically as:

$$SHD(G_1, G_2) = |Additions| + |Removals| + |Reversals|. \tag{4.4}$$

Evidently, the parameters are the common graph edge operators. This is a critical metric for quantifying the faithfulness of reconstructed graphs and evaluating the effectiveness of the proposed deep learning approach to structure learning.

#### Mean Absolute Error

Used for a quick and intuitive assessment of a regression model, the Mean Absolute Error (MAE) is the average size of error exhibited by predictions. MAE has a low sensitivity to outliers, excluding the direction towards which the error lies and avoiding the penalization of observations with larger errors.

$$MAE = \frac{\sum_{i=1}^{d} |y_i - \hat{y}_i|}{d}, \tag{4.5}$$

is thus an arithmetic average of absolute errors, where $y_i$ is the actual value and $\hat{y}_i$ is the predicted value. This is an easy way to simply gauge the effectiveness of the GNN and its ability to score graphs relative to their true scores.

#### Root Mean Squared Error

A commonly used metric in the evaluation of regression models, the Root Mean Squared Error (RMSE) is a statistical measure used to investigate the disparity between actual and predicted values. This metric is ideal for continuous values and other prediction-based models as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{d} (y_i - \hat{y}_i)^2}{d}}, \tag{4.6}$$

where $y_i$ is the actual observation and $\hat{y}_i$ predicted observation. The error is calculated based on the scores predicted by the GNN model.

**Standard Deviation**

An important metric to observe when analysing predictive models and their capabilities is the amount of variation in their output, providing insight into the average margin of error with regards to the predicted values and how far they deviate from the mean. The formula for standard deviation is as follows:

$$\sigma = \sqrt{\frac{\sum(x_i - \mu)}{d}} \tag{4.7}$$

This metric allows for the examination of how often the GNN model deviates from the RMSE, generally preferring models that have a lower Std Dev indicating a more consistent prediction of values, in this case the scores of each graph. This is a key factor when discussing convergence and thresholding during search algorithms.

## 4.4 Structure Learning

This section will outline the benchmark algorithms that will be implemented for comparison against the proposed deep learning approach, that is, a greedy search-and-score based algorithm and the GNN containing encoders for structure and position embeddings.

### 4.4.1 BIC Score Function

The choice of a scoring function is one of the most important when discussing score-based methods. These functions score candidate structures according to the training data and are used to search for the highest scoring candidate.

The **Likelihood Score** function is a natural and commonly chosen scoring function, measuring the probability that a model will generate the data on which it is being trained. To maximize the likelihood of the model, the goal is to find some graph $\mathcal{G}$ and the parameters $\theta_{\mathcal{G}}$. This can be defined as follows:

$$\text{score}_L(\mathcal{G} : \mathcal{D}) = \ell(\langle \mathcal{G}, \hat{\theta}_{\mathcal{G}} \rangle : \mathcal{D}), \tag{4.8}$$

where $\ell(\langle \mathcal{G}, \hat{\theta}_{\mathcal{G}} \rangle : \mathcal{D})$ is the logarithm of the likelihood function and $\hat{\theta}_{\mathcal{G}}$ are the parameters of graph $\mathcal{G}$ according to its maximum likelihood estimations. The problem with the likelihood score function is that it tends to favour graphs that produce more edges, which may lead to overfitting the training data due to any noise present. A reasonable mitigation strategy is to deploy additional mechanisms that restricts complex and dense graph structures, for example, restricting the in-degree of nodes. Another pitfall is that it may also learn empirical distributions which do not match those true distributions that produce the training data, failing to generalise any new data sampled.

Given these limitations, an alternative scoring function based on Bayesian principles was developed. The Bayesian Score function is one that takes Bayesian techniques into account, using the uncertainty on the nature of the true network structure to establish a structural prior $P(\mathcal{G})$ that places a prior probability on different structures, and a parameter prior $P(\theta_{\mathcal{G}}|\ \mathcal{G})$ that places a prior on different parameter choices on a given graph. Using Bayes rule, the *maximum posteriori* (MAP) model will maximize

$$P(\mathcal{G}|\mathcal{D}) = \frac{P(\mathcal{D}|\mathcal{G})P(\mathcal{G})}{P(\mathcal{D})}, \tag{4.9}$$

where $P(\mathcal{D})$ is the normalizing factor. Taking the logs of this equation, it can be rewritten as:

$$\text{score}_B(\mathcal{G} : \mathcal{D}) = \log P(\mathcal{G}|\mathcal{D}) = \log P(\mathcal{D}|\mathcal{G}) + \log P(\mathcal{G}) + c, \tag{4.10}$$

where $c = -\log P(\mathcal{D})$ is some constant independent of graph structure $\mathcal{G}$. The benefit of having a prior is that it allows for the penalization of structures that deviate from the prior probability, creating a preference for some structures over others. The term $P(\mathcal{D}|\mathcal{G})$, the *marginal likelihood* can be written as

$$P(\mathcal{D}|\mathcal{G}) = \int P(\mathcal{D}|\mathcal{G}, \theta_{\mathcal{G}}) P(\theta_{\mathcal{G}}|\ \mathcal{G}) d\theta_{\mathcal{G}}, \tag{4.11}$$

where $P(\mathcal{D}|\mathcal{G},\theta_\mathcal{G})$ is the likelihood of the training data given network $\langle\mathcal{G},\theta_\mathcal{G}\rangle$, and $P(\theta_\mathcal{G}|\mathcal{G})$ is the prior distribution over various parameter estimations for graph $\mathcal{G}$, which will automatically penalize structures with the incorrect amount of parameters as the probability mass will not be given to regions where the data is not found. This is a phenomenon known as *Ockham's razor*, defined as an explanation with the least amount of assumptions needed having a higher probability of being correct.

This differs from the likelihood score in that instead of trying to maximize this function, marginal likelihood opts to average this function based on the prior. If we consider a graph structure $\mathcal{G}$ given an $n \times d$ data matrix $\mathbf{x}$ where every row $\{x_{i1},\ldots,x_{id}\}$ for $i = 1,\ldots,n$ is an instantiation vector of the random variable $\underline{X} = (X_1,\ldots,X_d)$, the likelihood function is given by the formula:

$$L(\mathcal{G}|\mathbf{x}) = \prod_{j=1}^{d}\prod_{l=1}^{q_j}\frac{\Gamma(\sum_{i=1}^{k_j}\alpha_{jil})}{\Gamma(n(\pi_j^l)+\sum_{i=1}^{k_j}\alpha_{jil})}\prod_{i=1}^{k_j}\frac{\Gamma(n(x_j^i,\pi_j^l)+\alpha_{jil})}{\Gamma(\alpha_{jil})}, \tag{4.12}$$

where $x_j^{(k_j)}$ is some instantiation of a random variable $X_j$ in some state space $\mathcal{X}_j = (x_j^{(1)},\ldots,x_j^{(k_j)})$, $\prod_j$ denotes the parent sets of variable $X_j$ with state space $\mathcal{X}_{\prod_j} = (\pi_j^{(1)},\ldots,\pi_j^{(q_j)})$. The number of occurrences of a particular variable/parent configuration can then be denoted as $n(x_j^{(i)},\pi_j^{(l)})$ ($\pi_j^{(l)}$ denotes the instantiation of a parent set independent of its child variables). $\alpha_{jil}$ represents hyperparameters specified by a network with Dirichlet priors, giving weight to variable/parent configurations.

The formula in Equation 4.12 is known as the Cooper-Herzkovitz, or K2 (if hyperparameters $\alpha_{jil} = 1$) likelihood, a score function made appealing by its ability to input prior information. If the prior $\alpha$ is assumed to be such that $\alpha_{jil} \doteq \alpha^* \dot{P}(\theta_{jil}|\mathcal{G})$ where $\alpha^*$ is the prior belief strength and the equation is multiplied by the prior probability of some graph $P(\mathcal{G})$, the resultant is known as the *Bayesian Dirichlet equivalent* (BDe) score function.

But the need to specify a large number of hyperparameters makes this formula impractical; therefore, to have a more manageable numerical computation, the logarithm of this formula is used:

$$LL(\mathcal{G}|\mathbf{x}) = \sum_{j=1}^{d}\sum_{i=1}^{k_j}\sum_{l=1}^{q_j} n(\pi_j^l,x_j^i)\ln\frac{n(\pi_j^l,x_j^i)}{n(\pi_j^l)}. \tag{4.13}$$

Due to the nature of the Bayesian score tending to favour more complex graphs with fewer parameters, a penalization is added in such a way that varied structures are accommodated for at the arrival of new data if necessitated. It is this trade-off between likelihood and complexity that gives rise to what is called the *Bayesian Information Criterion* (BIC) scoring function:

$$\text{score}_{BIC} = LL(\mathcal{G}|\mathbf{x}) - \frac{\ln M}{2}\dim[\mathcal{G}], \tag{4.14}$$

where $\dim[\mathcal{G}]$ is the number of independent parameters in the graph $\mathcal{G}$. This scoring function was constructed using Bayesian principles to define the trade-off mentioned above.

### 4.4.2 Search Algorithm

As mentioned previously, exhaustively searching graph structures is computationally infeasible according to the function devised to determine the number of possible graphs for an arbitrary set of nodes, showing that the search space grows super exponentially. To combat this, heuristic algorithms are used to attempt to find the best network structure; however, this is not guaranteed.

Traditional search algorithms begin by defining a search space of graphs $\mathcal{M}$ containing all candidate structures (subject to constraints if any). A neighbourhood of $M$ is defined for each $M \in \mathcal{M}$ which is a set of all models that can be reached from $M$ that differs by the presence or absence of a single link, including itself. This can be thought of as a graph structure of its own, with each vertex in this graph representing a candidate network structure and the edges between them the moves that can be made from the structure at the current node.

Let $G = (V, E)$ be a DAG where vertices $v_i, v_j \in V$. An *incidence matrix* $I$ where there exists a row and column for every vertex in $V$ to denote an arc between two vertices: $I(i, j) = 1, 0$ otherwise. An *ancestor matrix* $A$ is used to denote whether there exists a directed path between 2 vertices, i.e. if $A(i, j) = 1$ then it is said that $v_j$ is an ancestor of $v_i$. Any proposed move resulting in the modification of $G$ must be tested to ensure the move is legal before it can be accepted as a potential move. In the case of DAGs, three conditions of legality must be met, each specific to the operation being carried out. The three conditions are:

- *Addition*: if we consider the addition of an arc from $v_j$ to $v_i$: $v_j \rightarrow v_i$; this addition is legal if and only if $A(j, i) = 0$ (there is no directed path from $v_j$ to $v_i$). $A(j, i) = 1$ shows the existence of a directed path from $v_i$ to $v_j$ and the addition of $v_j \rightarrow v_i$ would create a cycle (constant time complexity $O(1)$).
- *Removal*: removing an arc will never create a cycle, therefore, it is always legal.
- *Reversal*: this is a two-step process in which the existing arc is removed and the arc in the opposite direction is added. Although removing an arc is always legal, the addition must be checked to ensure that no directed cycles were introduced. Unfortunately, this cannot be as easily detected through the ancestor matrix, but can be determined by checking the ancestry of the node the arc will come from all the way to the vertex to which the arc is meant to lead (linear time complexity $O(n)$).

Only after a move is deemed legal is it accepted and the graph undergoes some local modification. Intuitively, it is possible that a search algorithm may never visit some, if not most, candidate structures, and so a trade-off needs to be made between how densely connected each node in the search space is and how to effectively search the neighbourhood of the current node. Many local-search heuristic algorithms do this, though they tend to get stuck at some local maximum.

For the purposes of this paper a common and powerful search algorithm, greedy hill-climbing, will be employed (for this discussion, a specific choice of scoring function is negligible) as the search algorithm of choice through which the BIC score and GNN scoring will guide learning of a graph structure.

### Greedy Hill-Climbing Search Algorithm (Local Search method)

**Algorithm 1:** Greedy Hill-Climbing Algorithm

**Input :** initial structure $\mathcal{G}_\emptyset$, observed dataset $\mathcal{D}$, scoring function score, some stopping criteria $\epsilon$, search operators $\mathcal{O}$
**Output:** $\mathcal{G}_*$ best network structure

```
1  G_* ← G_∅
2  Δ ← inf
3  while Δ > ε do
4      G ← G_*
5      for operator o ∈ O do
6          G_o ← o(G)
7          if G_o is legal structure then
8              if score(G_o : D) > score(G_* : D) then
9                  Δ ← score(G_o : D) − score(G_* : D)
10                 G_* ← G_o
11             end
12         end
13     end
14 end
15 return G_*
```

This heuristic search algorithm is one of the most frequently used search procedures. Once an initial graph structure is chosen and scored, all its neighbours and their respective scores are considered and the one producing the highest score is chosen. This is done incrementally until such a graph structure is reached that no neighbours have a higher score. However, given this is an optimization heuristic, there is no guarantee that a global maximum will be found.

The pseudocode in Algorithm 1 gives an overview of how this procedure is carried out. Legal structures mentioned in Line 8 specifically refers to moves which do not create such a graph that it is no longer a DAG, i.e. moves that create directed cycles. As previously stated, variations of this procedure have been developed to sidestep the tendency of greedy hill-climbing to work towards a local maximum such random restarts. This modification incorporates multiple restarts from randomly generated initial solutions once the algorithm reaches a local maximum or some predefined number of iterations. This process is repeated for $k$ restarts or terminates on the discovery of a satisfactory solution.

## 4.5 Structure- and Position-aware GNN

Despite its continued development, numerous innovations have been made in the domain such as the specific class of GNNs that will be implemented known as *Structure- and Position-Aware Graph Neural Networks* (SP-GNN). Developed by Chen *et al.* [Chen *et al.* 2023], the crux of the SP-GNN is to use position and structure encodings to capture structural and positional information from graph data in a generic and expressive manner (see Figure 4.2). Utilizing this power will enable us to provide useful insight into the structure of data modelling graph structures.

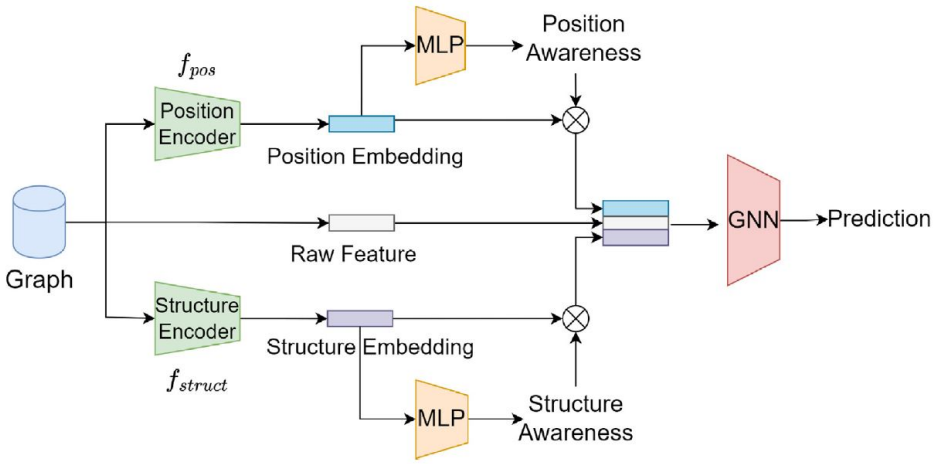



Figure 4.2: An overview of the architecture of SP-GNN.

### 4.5.1 Preliminaries

Before the architecture can be discussed, several preliminary concepts need to be established and understood first. In GNNs graphs contain an extra parameter alongside vertices and edges which are feature vectors $F = \{\mathbf{f}_v, v \in V\}$ for every node, containing data such as the closest and furthest nodes, distances between itself and other nodes and number of possible values. Embeddings within the GNN collectively aggregates this information about the graph structure through the use of message passing and aggregation functions. The $k$-th layer of a GNN can be computed as follows:

$$\begin{aligned} \mathbf{m}_u^{(k)} &= \text{RELU}(W^{(k)}\mathbf{h}_u^{k-1}) \\ \mathbf{h}_v^{(k)} &= \text{U}^{(k)}\text{CONCAT}(\text{MAX}(\{\mathbf{m}_u^{(k)}, u \in N(v)\}), \mathbf{h}_v^{k-1}) \\ \mathbf{h}_v^{(0)} &= \mathbf{f}_v, \end{aligned} \tag{4.15}$$

where $N(v)$ are the neighbours of vertex $v$, $\mathbf{m}_u^{(k)}$ the message embedding and $\mathbf{h}_v^{(k)}$ the node embedding. In this equation, ReLU and CONCAT MAX are chosen as the respective message passing and aggregation functions, with $W^k$ and $\text{U}^{(k)}$ being trainable parameters. According to You *et al.* [2019], the structural and positional awareness of node embeddings can be defined as follows:

**Definition 4.5.1** (Position Awareness)**.** A node embedding $z_i = f_p(v_i), \forall v_i \in V$ is said to be position aware if there exists a function $g_p(\cdot, \cdot)$ such that $d_{sp}(v_i, v_j) = g_p(z_i, z_j)$ where $d_{sp}(v_i, v_j)$ is the shortest path distance in a graph $\mathcal{G}$.

**Definition 4.5.2** (Structure Awareness)**.** A node embedding $z_i = f_{s_q}(v_i), \forall v_i \in V$ is said to be structure aware if $z_i$ is a function involving a $q$-hop network neighbourhood of a node $v_i$ i.e. $z_i = g_s(N_1(v_i), \ldots, N_q(v_i))$ where any $N_k(v_i)$ is some set of nodes $k$ hops away from node $v_i$.

Using these definitions, the architecture of an SP-GNN can be outlined.

### 4.5.2 Overview of SP-GNN Architecture and Training

The architecture of the proposed model is based on the work of Velickovic *et al.* [2017], using a three-layer GAN with four attention heads with 64 hidden units for graph classification.

As indicated in Figure 4.2 on the previous page, the training process is divided into 2 stages:

1. Awareness Scoring Stage: the SP-GNN makes use of its structural and positional encoders to compute its respective embeddings and their corresponding awareness scores per node feature.

2. Fusion Stage: the computed embeddings are fused with raw features that are weighted according to the calculated awareness scores.

The purpose of the position encoder is to learn a position embedding that respects the distances between points in the original graph (having low complexity and distortion), computing a 128-dimensional embedding using the $q$-hop sampling method. A *position loss function* helps in the description of gaps between the similarity of two nodes in both embedding and graph spaces. The function is as follows:

$$L_{pos} = \sum_{v_i \neq v_j} \left[ \frac{1 - d_{cos}(f_{pos}(v_i), f_{pos}(v_j))}{2} - (1 - \frac{1}{d_{spd}(v_i, v_j)}) \right]^2, \tag{4.16}$$

where $d_{cos}(\cdot, \cdot)$ is the measure of cosine similarity between two position embeddings $f_{pos}(v_i)$ and $f_{pos}(v_j)$, and $d_{cos}(\cdot, \cdot)$ is the shortest-path-distance between the two nodes. As this function value approaches 0, it is an indication of the position embedding becoming a near-isometric embedding (distance preserving map) of the original graph space.

The structure encoder has a focus on two types of structural information: (1) **multi-hop neighbourhood sequences** that involves the embedding of the egocentric network of each node into a vector, and (2) **identities** which counts the cycle number from the root of the egocentric network (node clustering property). These 2 types of information aid in the discrimination capabilities of nodes residing in similar structures, encoding 5-hop neighbour sequences and identity information. The computation of this embedding is

as follows:

$$\begin{aligned} f_{neigh} &= [A \cdot 1, A \cdot (A \cdot 1), \dots]^T \\ f_{identity} &= [diag(A), diag(A^2), \dots] \\ f_{struct} &= CONCAT(f_{neigh}, f_{identity}), \end{aligned} \tag{4.17}$$

where $A$ is a graph adjacency matrix and $diag(\cdot)$ contains the diagonal of matrix $A$. This encoder scales well with structure depth and can easily cope with multiple hops without neighbour expulsion.

**Algorithm 2:** SP-GNN Training Framework

**Input :** graph $\mathcal{G}$, trainable messaging passing and aggregation function MSG$(\cdot)$ and AGG$(\cdot)$, $L$-hop computational graph extractor SAMPLER$(\cdot)$, layer $L$, encoders for structure and position embeddings StructEncoder$(\cdot)$ and PosEncoder$(\cdot)$, awareness score generator AnalyzeAware$(\cdot)$

**Output:** node embeddings $\mathbf{h}$, for all $v \in V$

$\mathbf{f}_{pos}(v) \leftarrow$ PosEncoder$(v)$
$\mathbf{f}_{struct}(v) \leftarrow$ StructEncoder$(v)$
$w_{pos} \leftarrow$ AnalyzeAware$(\mathbf{f}_{pos}(v))$
$w_{struct} \leftarrow$ AnalyzeAware$(\mathbf{f}_{struct}(v))$
$w_{raw} \leftarrow$ AnalyzeAware$(\mathbf{f}_v)$
$\mathbf{h}_v^0 \leftarrow$ CONCAT$(w_{raw} \cdot \mathbf{f}_v, w_{pos} \cdot \mathbf{f}_{pos}(v), w_{struct} \cdot \mathbf{f}_{struct}(v)), \forall v \in V$

**for** $l \leftarrow 1, \dots, L$ **do**
  **for** $v \in V$ **do**
    $\mathbf{m}_u^{(k)} =$ RELU$(W^{(k)}\mathbf{h}_u^{k-1})$
    $\mathbf{h}_v^{(k)} =$ U$^{(k)}$CONCAT(MAX$(\{\mathbf{m}_u^{(k)}, u \in N(v)\}), \mathbf{h}_v^{k-1})$
  **end**
**end**
$\mathbf{h}_v \leftarrow \mathbf{h}_v^k$

With these embeddings being computed, they are analysed by an MLP to compute their respective awareness through the prediction of pairwise label equivalence. The MLP model is trained with a cross-entropy loss function and testing results represent the corresponding awareness scores for the structure and position embeddings.

Intuitively, fusion fuses information in feature space, combining the computed embeddings with raw node feature vectors. While the computation of structure and position embeddings are computed independently, this is the stage where they are combined with features. This is achieved through the use of message passing and aggregation functions which, as is evident in Equation (4.15) on page 29, tend to be ReLU and CONCAT. A fusion function can be written as follows:

$$h_v^{(0)} = Fuse(w_{pos} \cdot f_{pos}(v), w_{struct} \cdot f_{struct}(v), w_{raw} \cdot f_v), \tag{4.18}$$

where $w_{pos}, w_{struct}$ and $w_{raw}$ are embedding weights, specifically awareness scores for the former two terms. Algorithm 2 illustrates this process through the pseudocode intended to train the SP-GNN.

The intent behind the implementation of this class of deep learning and GNNs is to have it trained on many graph structures that model distributions, learning how to represent complex relationships that exist within the data. The expressive nature of SP-GNNs to learn these relationships can be used to accurately score graphs based on structural and positional information through the use of structural and positional embeddings. Optimization encoders by sampling $k$-hop neighbours provide an efficient solution to scalability problems, reducing the complexity from $O(|V|^2)$ to $O(k(|V|))$.

### 4.5.3 Thresholding

Obtaining a stationary point during the search procedure within some parameter $\epsilon$ is difficult to accomplish when the model used to score graphs provides estimates as opposed to calculating true scores. This does not bode well for heuristic algorithms creating the possibility for search algorithms to run indefinitely.

As such, a simple solution to mitigate this is devised. The $\epsilon$ value will be set to the standard deviation of the GNN model used to score graphs belonging to a distribution. Search will continue until the GNN model begins scoring graphs within the threshold $F(W) \leq \epsilon$, marking a soft convergence on the learned model. Once this threshold has been reached, $l$ modifications will be performed with each being scored multiple times and averaged for a more accurate prediction of the graphs score.

The graph at the end of the search with the lowest score will be chosen as the learned graph, terminating the search process.

# Chapter 5

# Results

In this section the results of the experiments performed, based on the research questions outlined in Chapter 4 on page 18, demonstrating the effectiveness of the proposed method. The implementation is based on PyTorch and the code is publicly available online at https://github.com/moeGathoo/MSc-Research2024.

## 5.1 Graph Scoring using SP-GNN

Accurately approximating graph scores is the cornerstone of this paper. This section presents the ability of the framework described to score graphs and the success with which this is accomplished based on varying graph sizes and data distributions.

### 5.1.1 *k*-Fold Cross-Validation

Cross-validation techniques are essential in evaluating models on unseen data and to test their efficacy on real world data, assessing their generalizability. This involves dividing data samples into multiple folds, using one as an evaluation subset and the remaining as training subsets, repeating this process until every fold has been used as an evaluation subset once. The results from each validation step are then averaged for a more robust estimate of the model's performance.

Various cross-validation techniques are plentiful [Berrar and others 2019], such as the single holdout method which involves splitting a given dataset into equal parts training and testing subsets (a 50/50 split), or the $k$-fold random subsampling method that performs the Holdout method $k$ times over $k$ pairs of training and test sets sampled randomly. Immediately, many issues arise with these methods, such as the potential for the training half of the data to contain little to no important information that will not be captured by the model in the former method, or the potential overlapping of subsets in the latter method.

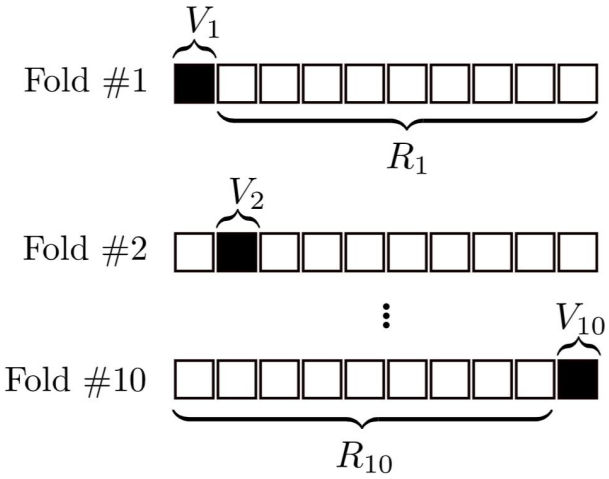



Figure 5.1: $k$-Fold Cross-Validation with $k - 10$

The method chosen is a variant of *k*-fold random subsampling, modified to ensure that no two test sets overlap. For a given dataset $\mathcal{D}$ containing $n$ samples, *k*-fold cross-validation divides $\mathcal{D}$ into *k* disjoint sets (folds) of data of approximately the same size $P = \{P_1, \ldots, P_k\}$. If we let $V_i = \mathcal{D} \setminus P_k$ denote the test set for the $i$-th iteration of the process and the remaining sets $R_i$ as the training set, the model is applied to $R_i$ which can be written as $\hat{f}_{-P_i}(R_i)$ This process terminates upon every fold having been used as a training set with accuracies averaged across every iteration:

$$\text{score}_{k-\text{cv}} = \frac{1}{k} \sum_{i=1}^{k} \text{score}(\hat{f}_{-P_i}(R_i), V_i), \tag{5.1}$$

and is illustrated by Figure 5.1. This technique also involves stratified data sampling: a process that ensures that each fold contains class proportions that reflect the original dataset in its entirety. This is essential in ensuring folds are not biased towards relationships that may not exist in the complete dataset.

During the training and testing stages, $k = 10$ was chosen as the number of folds created. In the final stages of testing model success, however, 10 individual graphs unknown to the model was given as test graphs, and the resulting metrics provided are based off the results obtained from these graphs.

### 5.1.2 Predictive Capabilities

#### Synthetic Datasets

The results presented in Table 5.1 on the facing page represent the baseline of 1000 data samples per random graph.

Table 5.1 shows the results of the experiments carried out to evaluate the predictive capabilities of the developed framework. The first column indicates the size of the graphs used during training, testing, and validation, with each following column representing the distribution to which the dataset sampled from the respective graphs belongs (see

| | Uniform | | | Normal | | | Dirichlet | | | Gumbel | | | Binomial | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MAE | RMSE | STD | MAE | RMSE | STD | MAE | RMSE | STD | MAE | RMSE | STD | MAE | RMSE | STD |
| 10 Nodes | 57.21 | 58.78 | 32.76 | 46.63 | 49.20 | 46.54 | 38.94 | 40.67 | 43.56 | 74.65 | 75.96 | 53.39 | 126.69 | 127.15 | 62.32 |
| 25 Nodes | 102.27 | 102.50 | 49.89 | 84.41 | 85.86 | 65.52 | 71.71 | 92.89 | 49.99 | 175.07 | 175.66 | 69.72 | 278.70 | 276.98 | 76.08 |
| 50 Nodes | 174.46 | 175.23 | 90.26 | 150.28 | 150.88 | 74.85 | 119.63 | 120.03 | 51.75 | 220.16 | 320.48 | 82.91 | 399.29 | 399.77 | 94.96 |
| 100 Nodes | 289.89 | 290.20 | 113.18 | 251.90 | 252.40 | 77.71 | 202.78 | 203.24 | 57.78 | 352.58 | 502.73 | 97.90 | 602.68 | 602.85 | 139.05 |

Table 5.1: Performance Metrics Evaluating Model

Section 4.3.2 on page 22 for more information on the graphs). Each sub-column is the metric used to evaluate the model: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and Standard Deviation (STD). These results are based on an initial use of 1000 samples per graph.

A few observations can be made by examining the results. The MAE provides a simple yet effective look at how far the predicted score is from the true score of a graph. It can be seen that the GNN's ability to score graphs is dependent on both the size of graphs and the distribution from which the data is sampled. The best performing models are, intuitively, the ones trained and validated on smaller graphs, with an increase in inaccuracy as dimensionality increases. This trend can be seen across every distribution and will be discussed in more detail in the next section.

Another important metric is the STD which gives us an indication, on average, how far off each prediction is from the mean error. One observation is that distributions that have a more random nature to its sampling process show to have an increased deviation from the mean, such as the Uniform and Gumbel. In contrast, distributions with lower overall randomness, such as Dirichlet and Binomial, have a lower deviation. This can be attributed to the arbitrary nature of values in the Uniform distribution and the sensitivity to outliers found in the Gumbel distribution. Overall, the GNN model trained on Dirichlet data outperforms every other model across all graph sizes.

### Benchmark Datasets

The benchmark datasets used in this experiment were specifically chosen for having discrete data, an intrinsic accommodation of the GNN model used. To preserve consistency with synthetic datasets, each benchmark network uses 1000 data samples as well. See Section 4.3.3 on page 22 for a breakdown of each benchmark network configuration.

The SP-GNN scores in Table 5.3 in this instance are not averaged over a specified number of iterations. This is to demonstrate the predictive capabilities of the model under circumstances where there is a lack of time to obtain multiple scores and subsequently average to achieve a truer reflection of the network score.

It can be seen that the SP-GNN performed best with smaller networks, with the Asia dataset having the lowest disparity between predicted and true scores. The Pathfinder network also shows a relatively low score disparity between true and predicted scores. This can be explained by a similarity in network architecture, both in size and density, compared to the training data of the models used for scoring, respectively. Despite differences in architecture, the ability to accommodate them is attributed to the encoders and their ability to retain positional and structural information.

| Uniform | | Normal | | Dirichlet | | Binomial | | Gumbel | |
|---|---|---|---|---|---|---|---|---|---|
| Actual | Predicted | Actual | Predicted | Actual | Predicted | Actual | Predicted | Actual | Predicted |
| 10 Nodes | | | | | | | | | |
| -5357.73 | -5393.57 | -5632.97 | -5581.42 | -6016.24 | -6050.47 | -4479.30 | -4355.79 | -3544.53 | -3607.06 |
| -6044.55 | -5979.89 | -4999.33 | -4972.33 | -5691.32 | -5660.24 | -4089.63 | -4223.31 | -2384.97 | -2297.67 |
| -5856.67 | -5793.29 | -5339.05 | -5311.11 | -5797.26 | -5856.48 | -3326.24 | -3205.82 | -2114.25 | -2040.76 |
| -5788.19 | -5859.61 | -6165.92 | -6200.55 | -5815.39 | -5789.83 | -2808.61 | -2917.37 | -2359.67 | -2422.88 |
| -6157.82 | -6227.45 | -5675.04 | -5607.66 | -5804.93 | -5766.61 | -5106.85 | -5242.69 | -2484.47 | -2573.74 |
| -6035.36 | -6067.75 | -5879.70 | -5923.90 | -5649.29 | -5622.29 | -3395.16 | -3252.90 | -3099.32 | -3197.30 |
| -6368.70 | -6438.77 | -4914.75 | -4841.65 | -5925.79 | -5867.96 | -3379.41 | -3517.05 | -3986.06 | -3926.21 |
| -6188.17 | -6251.20 | -5844.49 | -5784.82 | -4930.87 | -4902.87 | -5475.04 | -5344.25 | -3264.72 | -3211.56 |
| -5342.02 | -5390.19 | -5658.77 | -5708.53 | -5807.64 | -5767.32 | -2816.50 | -2692.51 | -2535.66 | -2460.52 |
| -6082.95 | -6029.40 | -5268.85 | -5237.77 | -5572.48 | -5620.32 | -2192.03 | -2302.02 | -2955.82 | -2871.24 |
| 25 Nodes | | | | | | | | | |
| -14344.40 | -14242.49 | -14708.44 | -14776.20 | -14308.34 | -14364.37 | -7069.79 | -7354.78 | -8154.54 | -8335.68 |
| -14428.36 | -14338.17 | -13099.59 | -13045.54 | -14495.65 | -14554.72 | -10127.45 | -10402.16 | -8062.67 | -8218.86 |
| -15238.09 | -15141.61 | -14565.80 | -14664.60 | -14619.08 | -14556.56 | -7159.86 | -7428.18 | -8030.06 | -7874.09 |
| -14877.73 | -14768.13 | -14604.98 | -14704.52 | -14647.66 | -14726.00 | -8903.16 | -9161.93 | -6025.78 | -5837.21 |
| -14399.94 | -14503.18 | -14249.00 | -14323.87 | -14365.94 | -14294.32 | -9147.80 | -8892.08 | -7790.09 | -7630.58 |
| -15194.16 | -15086.19 | -14078.68 | -14009.47 | -14969.07 | -14881.38 | -11082.71 | -11361.34 | -6406.87 | -6600.31 |
| -14719.91 | -14826.28 | -13727.83 | -13823.61 | -14640.57 | -14570.57 | -8664.24 | -8387.31 | -4675.92 | -4500.43 |
| -14615.24 | -14725.13 | -13846.49 | -13944.10 | -13502.47 | -13561.19 | -6444.87 | -6147.68 | -7325.30 | -7520.39 |
| -14584.54 | -14479.25 | -14793.68 | -14695.02 | -15169.74 | -15070.87 | -9687.51 | -9977.76 | -8151.01 | -7986.19 |
| -14967.08 | -15058.85 | -14152.88 | -14065.12 | -13540.20 | -13614.47 | -7465.15 | -7183.67 | -6596.78 | -6416.32 |
| 50 Nodes | | | | | | | | | |
| -29685.39 | -29489.34 | -27645.11 | -27480.07 | -29428.77 | -29307.10 | -17554.25 | -17179.39 | -14137.21 | -13902.59 |
| -29911.64 | -29721.26 | -29078.51 | -29223.85 | -30414.97 | -30520.62 | -17906.86 | -17516.68 | -12740.00 | -12946.34 |
| -28714.33 | -28897.67 | -28958.94 | -28815.39 | -28801.52 | -28677.40 | -18558.80 | -18972.56 | -16755.92 | -16996.88 |
| -28788.71 | -28638.33 | -28971.77 | -29121.24 | -29794.19 | -29657.06 | -19579.87 | -19967.32 | -13376.35 | -13170.43 |
| -28751.36 | -28941.65 | -27136.48 | -26974.91 | -29919.76 | -30042.02 | -15917.04 | -15512.45 | -14898.74 | -14676.15 |
| -29715.07 | -29878.04 | -27811.51 | -27937.75 | -28871.72 | -28754.23 | -20197.69 | -19774.78 | -13011.18 | -12792.37 |
| -29771.94 | -29936.02 | -27164.94 | -27028.18 | -30690.68 | -30575.93 | -15302.42 | -15701.07 | -15866.87 | -15645.43 |
| -30250.23 | -30059.19 | -28911.93 | -29055.23 | -30130.89 | -30264.41 | -17069.22 | -16655.82 | -14721.99 | -14939.49 |
| -29910.70 | -30072.04 | -29309.58 | -29482.43 | -29730.77 | -29622.73 | -17416.15 | -17778.79 | -12747.30 | -12951.48 |
| -29437.71 | -29592.46 | -28405.46 | -28246.79 | -29508.56 | -29620.22 | -14753.70 | -14329.19 | -13141.79 | -13371.03 |
| 100 Nodes | | | | | | | | | |
| -58492.81 | -58207.61 | -56451.03 | -56713.21 | -58444.13 | -58234.59 | -30622.80 | -31242.61 | -28072.18 | -27736.12 |
| -58975.05 | -59253.91 | -59095.38 | -58827.94 | -58237.80 | -58030.68 | -39931.29 | -39354.96 | -29304.93 | -29668.51 |
| -59591.71 | -59298.01 | -56237.88 | -55996.47 | -59538.62 | -59731.90 | -36154.12 | -35550.10 | -28020.08 | -27665.20 |
| -59958.90 | -60252.37 | -56595.65 | -56326.22 | -59713.81 | -59519.35 | -40991.20 | -40372.73 | -29651.98 | -29289.37 |
| -59012.45 | -59330.68 | -56257.29 | -56513.21 | -59019.68 | -58810.13 | -32430.68 | -33015.97 | -28149.65 | -28487.10 |
| -59618.88 | -59334.81 | -57214.47 | -57484.72 | -59379.54 | -59168.18 | -35012.48 | -34394.52 | -25594.07 | -25951.72 |
| -58954.60 | -59264.67 | -54853.91 | -54628.53 | -58577.31 | -58755.72 | -32849.22 | -32244.41 | -24086.04 | -23733.45 |
| -59552.39 | -59273.20 | -57510.31 | -57264.67 | -59514.27 | -59293.82 | -36535.35 | -35945.79 | -31634.88 | -31988.93 |
| -57963.76 | -58240.11 | -56619.80 | -56364.64 | -56876.04 | -57095.56 | -34268.08 | -34880.25 | -30247.54 | -30589.82 |
| -58499.27 | -58219.53 | -57000.89 | -56774.69 | -59898.86 | -60082.94 | -35345.93 | -34747.59 | -30932.22 | -31296.90 |

Table 5.2: Raw Data for Synthetic Graphs with 1000 Sampled Data Points

| Dataset | Size | BIC (true) | SP-GNN (predicted) |
|---|---|---|---|
| Asia | 8 | -2.35e+03 | -2.33e+03 |
| Alarm (25) | 37 | -1.21e+04 | -1.04e+04 |
| Alarm (50) | 37 | -1.21e+04 | -1.56e+04 |
| Pathfinder | 109 | -3.84e+04 | -3.70e+04 |

Table 5.3: BIC Scores vs SP-GNN Scores on Benchmark Networks

The difference between true and predicted scores is greatest with the Alarm network. The Alarm network was specifically chosen due to its size in comparison with the training data of the 25-node and 50-node models. This presented an opportunity to see how well the SP-GNN performs on networks with architecture that is much bigger/smaller. Predicted scores of the respective models mimic that of its training data, with the 50-node model over-scoring the network due to its training data having higher scores, and the 25-node model under-scoring due to lower score in training.

It is encouraging that the proposed SP-GNN can handle minor differences in network architecture, though more work will have to be put into attaining higher accuracy approximations for networks with more considerable architecture changes.

**Awareness Scores**

Examining the awareness scores for the structural and positional encoders respectively will provide more insight into which encoder plays a larger role in the scoring ability of the GNN and in which instance is most reliant.

| | Size | Uniform | Normal | Dirichlet | Binomial | Gumbel |
|---|---|---|---|---|---|---|
| Position | 10 | 0.5934 | 0.6372 | 0.6740 | 0.5040 | 0.5318 |
| | 25 | 0.5306 | 0.6024 | 0.6466 | 0.4923 | 0.5309 |
| | 50 | 0.5778 | 0.5598 | 0.6275 | 0.4624 | 0.5287 |
| | 100 | 0.5192 | 0.5351 | 0.6108 | 0.4109 | 0.5133 |
| Structural | 10 | 0.8017 | 0.7942 | 0.8497 | 0.7338 | 0.8603 |
| | 25 | 0.8133 | 0.8116 | 0.8520 | 0.7457 | 0.8733 |
| | 50 | 0.8234 | 0.8165 | 0.8708 | 0.7491 | 0.8794 |
| | 100 | 0.8373 | 0.8330 | 0.9236 | 0.7501 | 0.8805 |

Table 5.4: Awareness Scores based on Respective Embeddings in SP-GNN

As can be seen in Table 5.4, structural awareness has a much stronger influence on the graph scoring task, accounting for most of the awareness. Positional awareness on the whole has a weaker influence and, comparatively, plays a smaller role in assisting the model. A key difference observed is the difference in awareness trends as dimensionality increases. Positional awareness shows to decrease with a larger number of nodes within the graph whereas structural awareness shows to increase, with the same distributions showing the same trends in awareness. Overall performance can therefore be attributed to the structural embeddings and the role they play in graph classification tasks.

Structural information is shown to play the more important role, doing much of the heavy lifting for graph-level tasks such as graph scoring compared to positional information provided. Potentially increasing multi-hop sequences could lead to improvement in overall performance, but care must be taken to avoid over-smoothing of internal parameters. Positional encoders merely examine node distances within graphs, and this may prove useful in future works wherein pairwise link prediction becomes a topic of interest.

An important note is the use of hyperparameters to ensure efficient differential in data samples so as to avoid overlay and patterns found within the data.

### 5.1.3 Scalability

A crucial aspect of this framework for graph scoring is its hopeful indifference toward graph size to maintain predictive accuracy. Intuitively, one would expect the GNN to be less capable of accurately scoring a graph of larger sizes should the sample space remain the same size. This would essentially be depriving the GNN of sufficient amounts of data to produce trustworthy predictions.

Looking at Figure 5.2, we can see this trend reoccurring with MAE increasing as the dimensionality increases across all distributions. However, before discussing it, a non-trivial consideration to keep in mind when contextualizing the results is the range of true BIC scores within which the graphs exist. Certain distributions produce graphs that, when scored against their respective data samples, produce score ranges wider than others. Section 5.1.3 provides the approximate range of true scores to better understand the results.

| | ∼Score Range | | | |
|---|---|---|---|---|
| Distribution | 10-Nodes | 25-Nodes | 50-Nodes | 100-Nodes |
| Uniform | 1000 | 3000 | 3600 | 5000 |
| Normal | 1400 | 1750 | 2500 | 4000 |
| Dirichlet | 1000 | 1750 | 2000 | 3000 |
| Binomial | 3000 | 5000 | 5000 | 8000 |
| Gumbel | 2000 | 3000 | 4000 | 8000 |

Table 5.5: Approximate range of true BIC score for each distribution per graph size.

Examining the plot in Figure 5.2, it is evident that every model is capable of predicting the score of a graph with increasing dimensionality. However, each distribution varies differently with increasing graph sizes, with some depreciating at a rate worse than others. Taking these results in conjunction with the awareness scores from the previous section, it can be deduced that the GNN becomes more dependent on structural information to make accurate predictions should no more data be provided.

The binomial distribution shows the steepest increase in MAE, followed by Gumbel. This is explained by the non-conformity to which the linear SEM model can sample noise to those distributions, making it more difficult for the autoencoders to accurately encode/decode the respective datasets. This is evident when compared to the remaining distributions which have lower increases in MAE accommodating the suitability thereof in the SEM model. In the case of uniform samples, the corresponding positional awareness scores are higher compared to others, indicating the assistance of positional embeddings in maintaining suitable accuracy.

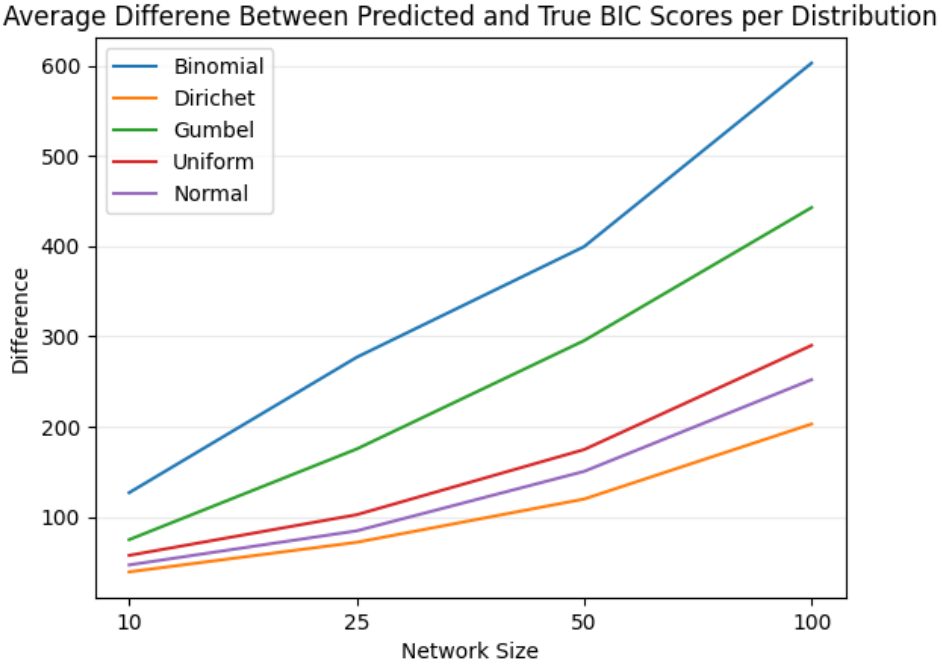



Figure 5.2: Difference in scores per distribution scaling with graph size.

## 5.2 Effect of Data Sample Sizes

It goes without saying that deep learning models will perform better when provided with more data. With the advent of models capable of capturing more beneficial information and utilizing it to improve performance, it is always worthwhile to investigate how much data is required to obtain adequate performance. This includes ensuring that the quality of the data is good and that the mechanics being applied to it are capable of processing it.

### 5.2.1 Hyperparameter Tuning

A crucial aspect of examining the effectiveness of the model is adjusting the hyperparameters and seeing how its performance is affected. This includes changing learning rates, episodes being run, and the values used to sample from each distribution.

With regards to the other hyperparameters such as learning rate, epochs, optimizers, weight initializer, the RandomWalksCV approach was chosen for its respective tuning. As implied by its name, this method selects values at random for each hyperparameter and tests said combination for some predefined number of iterations. The combination chosen is the one that ultimately provides the best performance for the model. This method has been shown to converge on an optimal combination faster than traditional GridSearchCV by traversing the grid in a random fashion.

The following parameters were found to be the most optimal and provided the best performance:

- **Epoch Number:** 10 000
- **Learning Rate:** 0.003
- **Layers:** 3
- **Nodes Per Layer:** 64
- **Optimizer:** Adam
- **Weight Initializer:** Xavier Uniform
- **Activation Function:** ReLU

### 5.2.2 Encoder/Decoder

Much of the work surrounding neural causal discovery employs some adaptation of variational autoencoders using encoders and decoders for model learning. Although the task of inferring a causal graph from data samples is not the goal of this framework, the same approach to data analysis can be taken to ensure correct association of data and graphs with its respective score as a global attribute.

**Encoder:** Supported by the use of the linear SEM for data generation, the encoder assumes the role of an inference model encoding a sample $X^k$ into a latent variable $Z$ with density $q(Z|X^k)$ equal to the log-evidence using variational posterior $q(Z|X)$. This results in the encoder:

$$Z = f_2((I - A^T)f_1(X)), \tag{5.2}$$

where $f_1$ is a multilayer perceptron and $f_2$ an identity mapping. The activation function is chosen to be ReLU, a necessity given the nature of variables being discrete. This will require a modification to the likelihood in the decoder.

**Decoder:** The decoder plays the role of a generative model, attempting to reconstruct the sample $X^k$ from $Z$ by performing potentially nonlinear transforms on $X$ and $Z$ respectively:

$$\hat{X} = f_4((I - A^T)^{-}1f_3(Z)), \tag{5.3}$$

where $f_3$ and $f_4$ are parameterized functions that play the inverse roles of the encoder. To avoid notational ambiguity, $\hat{X}$ represents the output of the encoder. For effective employment on discrete variables, the likelihood $p(X|Z)$ transforms into a probability matrix $P_X$ where each row denotes the probability of each variable (rows in $P_X$) taking on a categorical value (columns in $P_X$). The assumption that there are a finite number of values any one variable can assume, is made. This restructures the architecture of $f_4$ to a row-wise softmax network:

$$P_X = \text{softmax}(\text{MLP}((I - A^T)^{-}1Z, W^3, W^4)), \tag{5.4}$$

where $W^3, W^4$ are the parameter matrices of the respective MLP.

It can be seen that this encoder/decoder model is developed directly from linear SEM, pairing the sampling procedure as an encoder with the generative model as the decoder, resulting in tractable learning as proposed by [Yu *et al.* 2019].

### 5.2.3 Results

This experiment examines the effect that training sample size has on the ability of the GNN to score graphs, specifically how it affects the accuracy of the predictions being made. An unspoken standard sample size amongst literature pertaining to continuous optimization is 1000 data samples. This paper examines the use of larger datasets as described in Chapter 4.

Examining the figures in Figure 5.3 on the next page showing the results for graphs of 10 nodes, a majority of the distributions display an increase in accuracy with the addition of more samples, bar one. The uniform distribution shows an irregular pattern with the increase in sample size, initially displaying an improved accuracy with a shift from 1000 to 2000 samples, but then showing a decrease in accuracy when sample size increase further to 3500. In some instances, such as 50-node or 100-node graphs, a maximum size of 5000 samples proves to be, at best, just as good as 1000 if not worse.

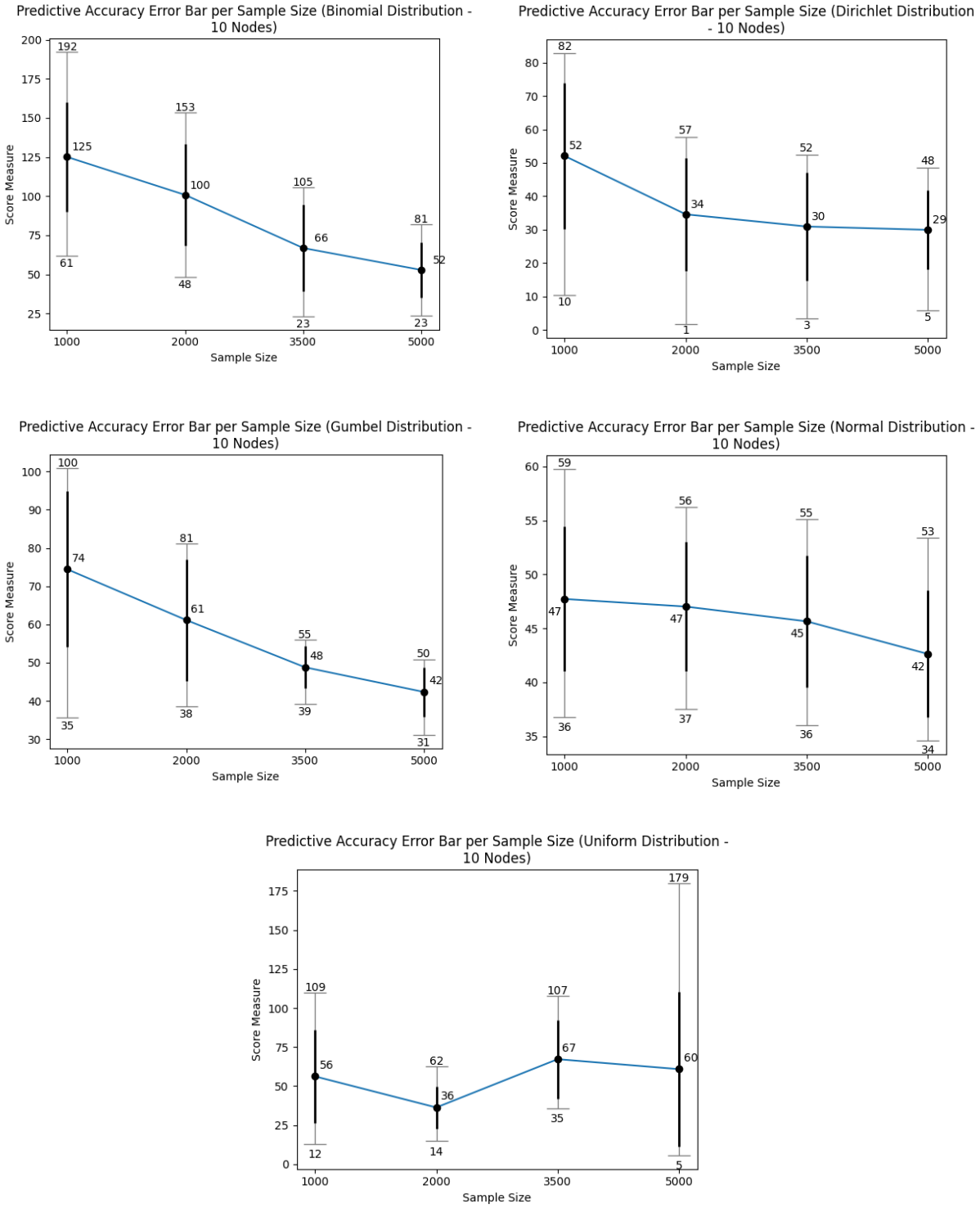



Figure 5.3: Mean Absolute Error between graphs scored using BIC score function and graphs scored using GNN. Plots indicate, the mean, maximum and minimum error, as well as standard deviation.

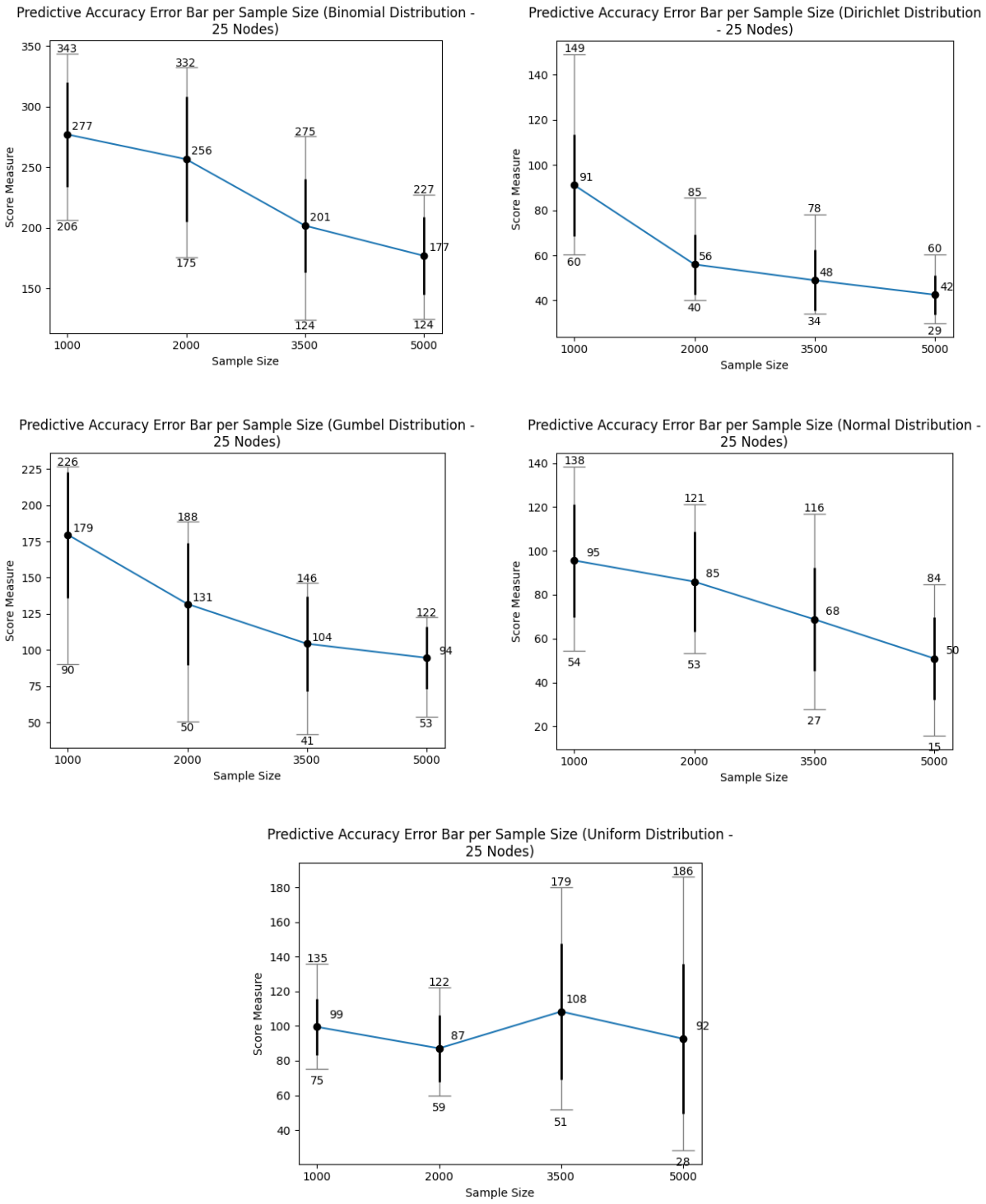



Figure 5.4: 25 Nodes

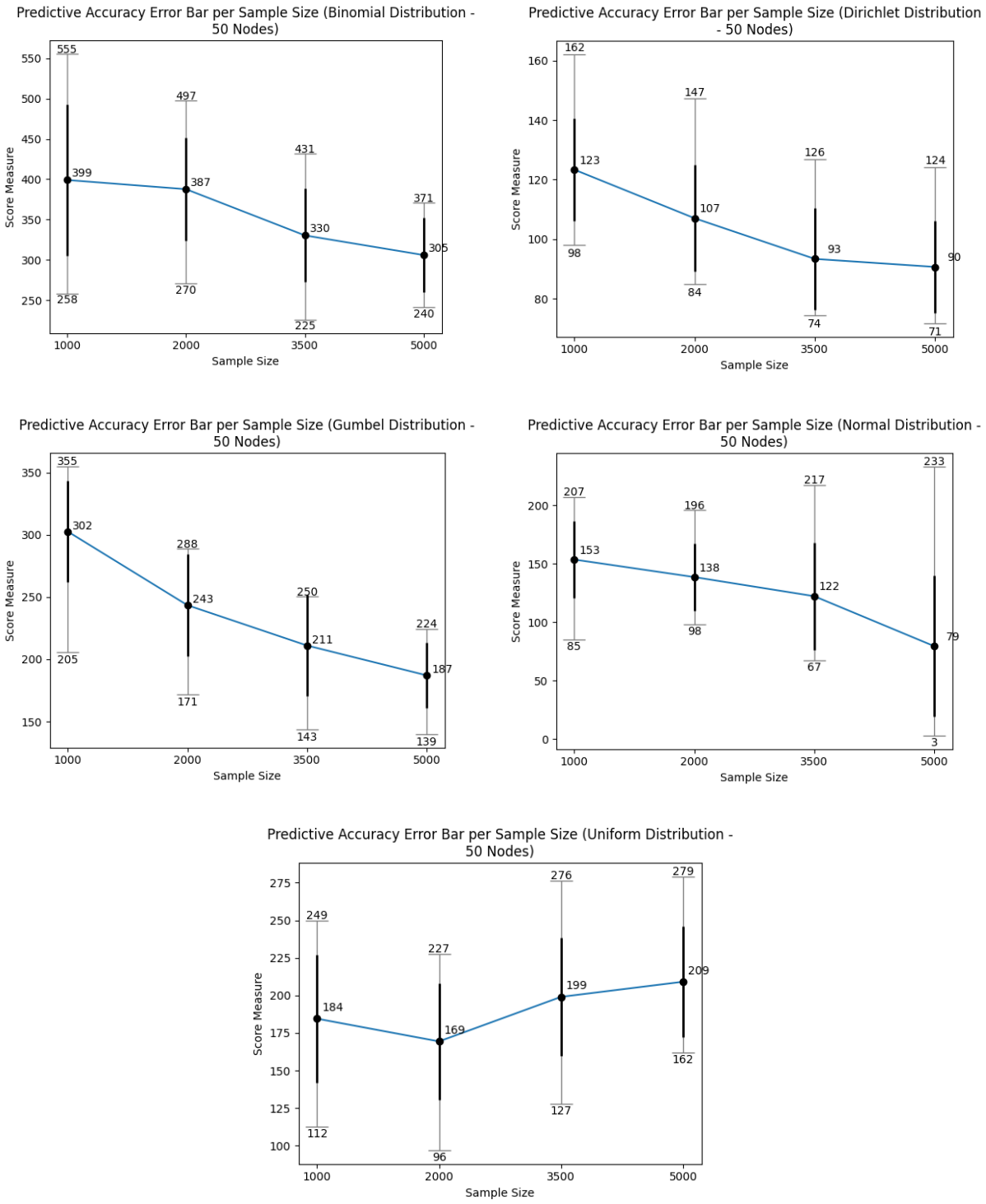



Figure 5.5: 50 Nodes

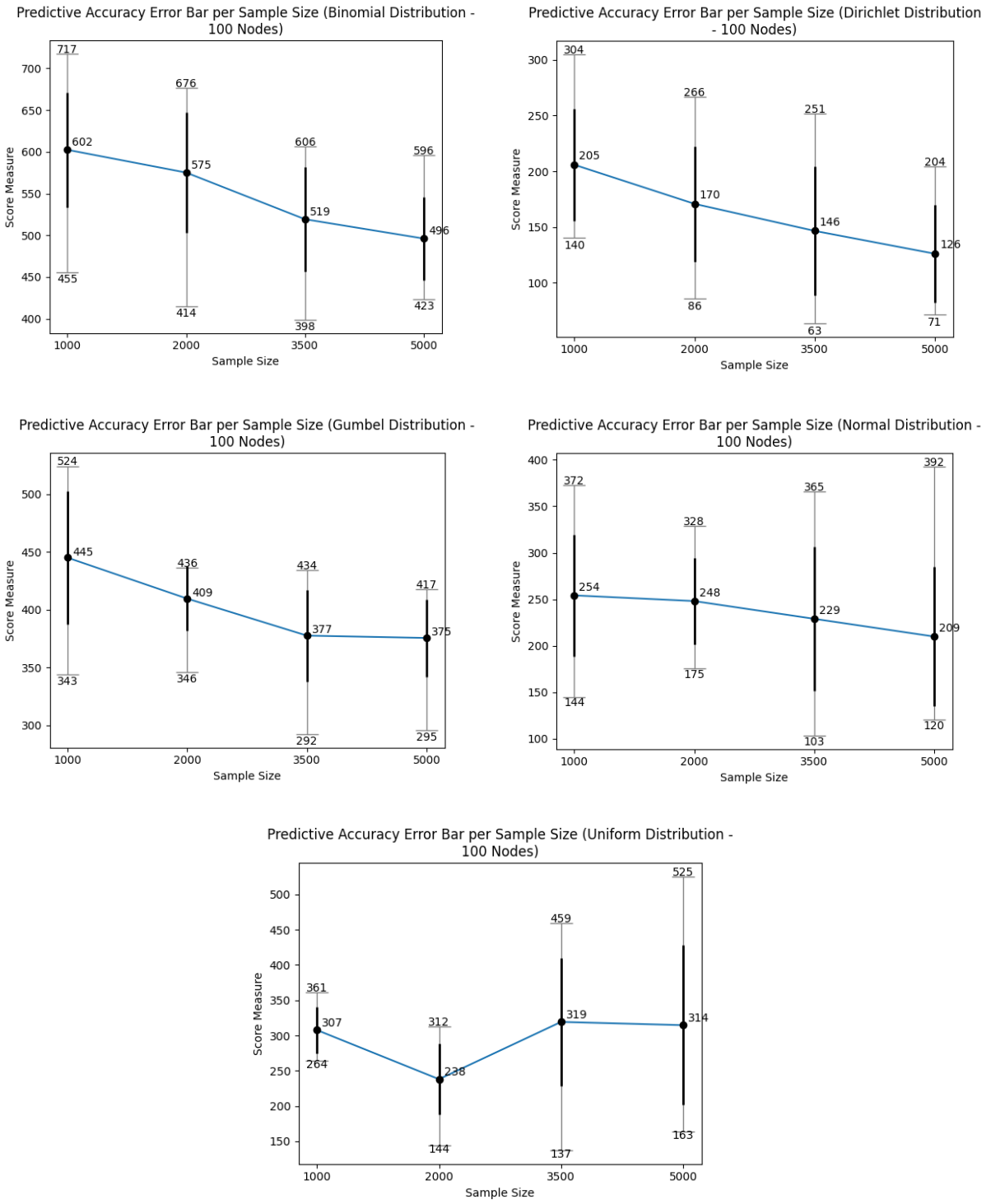



Figure 5.6: 100 Nodes

The remaining distributions all demonstrate a similar increase in accuracy to varying degrees, with the binomial distribution showing the most interesting. It can be observed that an initial improvement of the model takes place as the sample size increases, from 1000 samples to 3500 samples. However, a steep drop-off in improvement takes place when further increasing the sample size to 5000 samples. This is the smallest decrease in MAE across the board. Furthermore, it can be seen that while this increase in sample size does not improve accuracy as much, there is a reduction in standard deviation, more so than any other.

Both the Dirichlet and Gumbel distribution possess a similar improvement with increasing sample sizes, both experiencing improvement not dissimilar to exponential decay. In the case of Dirichlet, this drop off takes place rather substantially after 2000 samples, indicating the need to focus on other areas of the framework for further improvement. The standard deviation shows to be rather constant with a slight improvement in 5000 samples, however it is skewed towards the maximum MAE. This can be interpreted as the models tendency to predict further from the MAE.

The Gumbel distribution shows to have a linear improvement up to 3500 samples indicating the need for more data. Its standard deviation shows the same trends as Dirichlet skewed towards the maximum MAE. This can be attributed to a much broader range of trained true BIC scores, making it more challenging to successfully pinpoint a true score.

The normal distribution shows to be the only one that consistently improves performance as sample size increases, though it is generally lower than any other. We also see a consistently large standard deviation despite increased data sample size with a slight skewness towards the lower end of the range.

Across the board, the range of approximations decreases with more data provided, allowing the GNN to create much richer and deeper embeddings for more informed predictions of graph scores. Performance may vary depending on the scale of each distribution chosen during the sampling process, with greater scales potentially resulting in a less successful model.

Another factor to consider the toll larger sample sets takes on both the awareness and sampling autoencoders. Regarding the awareness autoencoders, their effectiveness remains unchanged regardless of how much data each input graph is given, and scores for both structural and positional remain the same as those shown in Table 5.4. This is based on the embeddings utilizing node specific information within graphs as opposed to data that may be sampled from them, especially in the domain of graph classification tasks.

The SEM based autoencoders paint a different picture. Figure 5.3 shows how performance increases, but it does not speak to the increased training times that comes with these larger datasets. This is symptomatic of the method used whereby the model attempts to reverse engineer data samples provided by reconstruction through posterior approximation. It is particularly well suited for autoencoders given a close relation to the evidence lower bound (ELBO) [Kingma *et al.* 2013].

This is based on the structure equation model as a method of data generation and does not account for more complex nonlinear cases. This includes data generated using the model:

$$x = A^T h(x) + z, \tag{5.5}$$

where $h$ is some chosen nonlinear function (those used by Yu *et al.* [2019] includes $sin$ and $cos$). Cases where data is sampled using these methods will resultant in longer training times wherein nonlinearity occurs post linear combination of variables. However, these may result in more accurate results of the GNN.

Another commonly used data type within this paradigm not accommodated at this stage is vector-valued data. Vector-valued samples can rather easily be generated by selecting a magnitude of each variable greater than 1 and generating the $k$-th dimension of each sample based on standard normal vectors (or any other multivariate distribution of the persons choosing). Samples will then be found in the form $X = [x^1, x^2, \ldots, x^d]$ where $d$ is the magnitude of the respective random variable. Despite noise matrices being found within the latent space of the autoencoders, one may assume it would result in longer training times as well, should it be included.

## 5.3 Model Selection Guidance

Assessing the ability of the GNN to guide the search procedure was not a focus of this paper, however it is a non-trivial aspect of the larger picture. While the GNN has proven effective and relatively accurate at scoring graphs, it must not be an oversight that the predictions being made are estimates of the true graph score. As such, a method of mitigating the error between true and predicted scores is a necessity for the convergence of the search procedure on a single model.

### 5.3.1 Learning Process

The process of learning a DAG is simple: first an empty graph is taken containing all random variables present within the network. An edge is placed between any 2 randomly selected nodes, and the resulting network is scored using both the BIC score and the SP-GNN. This process is repeated until there is no improvement in score between successive networks, or the improvement is smaller than some chosen interval $\epsilon$.

As can be seen, this presents a significant issue for the GNN: it is producing approximations of the networks score rather than true scores. No matter how close it may be to the true score, it is near impossible to obtain successive scores within the interval required to terminate the learning. As such, the thresholding mechanism explained in Section 4.5.3 is put in place to try mitigate this issue.

### 5.3.2 Results

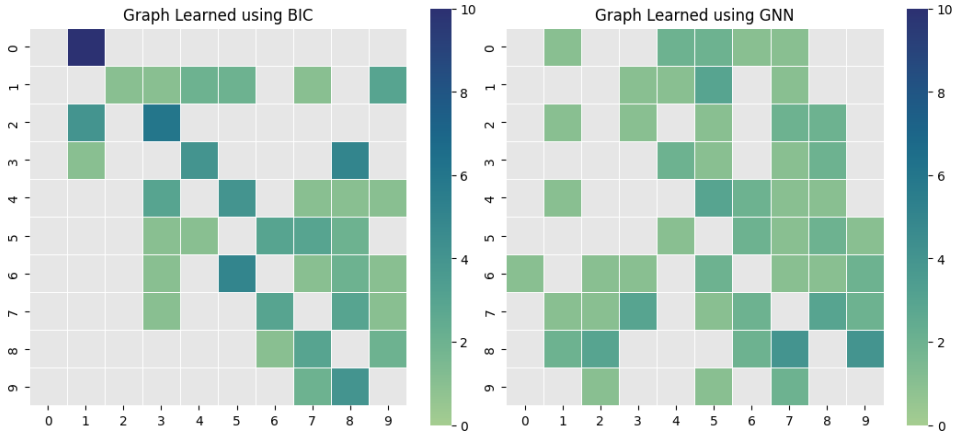



Figure 5.7: Comparison of graphs learned using BIC score and GNN scoring respectively

For ease of interpretation, the graphs are represented using a heatmap indicating the presence (or lack) of edges between random variables. When looking at Figure 5.7, it is evident that the SP-GNN learns graphs that are still relatively inaccurate in comparison to their true graph counterparts. Despite being relatively inaccurate, it can be seen that there is a subtle skewness towards the concentration of edges in the upper triangular matrix, consistent with the Erdõs-Rényi algorithm for graph generation.

| Method | SHD | # Predicted Edges |
|---|---|---|
| BIC | 2 | 16 |
| SP-GNN | 7 | 8 |
| SP-GNN with thresholding | 5 | 11 |

Table 5.6: Comparison of learned graphs using BIC and SP-GNN with respect to ground truth graphs summed over 10 graphs.

Table 5.6 shows the average SHD and number of predicted edges of the learned graphs with respect to their ground truth models, comparing the BIC score function with SP-GNN predicted scores. For graphs of 10 nodes, it is evident that the graphs learned using SP-GNN are worse than that of the BIC, predicting far fewer edges on average than graphs learned using the BIC score function. This can be attributed to the difficulty at which convergence is reached during the greedy hill-climbing search procedure. Despite its impressive accuracy, the SP-GNN struggles to consistently obtain similar scores for graphs that closer resemble the ground truths.

A naive approach implemented in this paper is based on a simple thresholding mechanism, scoring the same graph multiple times and averaging the score for a more accurate representation of the true score. For the intention of choosing a model with the lowest error, the Dirichlet model with 10-Node graphs trained on a sample size of 5000 samples was used.

The graph containing the operation with the lowest average difference in score was chosen as the next state of the graph. Search was terminated when graphs all had a similar average difference within the standard deviation for this trained model (according to Figure 5.3 on page 42, standard deviation is 29), a soft epsilon value. While this increased the occurrence of graphs closer to ground truths this was still relatively rare, though the average SHD decreased from the results without thresholding.

# Chapter 6

# Impact and Implications

Causal discovery in structure learning is central to understanding underlying relationships and influences that govern complex systems. Identifying these relationships rather than simple correlations between observations enables researchers to extract much richer insights. As outlined in the related work section, many of the continuous optimization approaches aimed at doing so, battle with the ability to efficiently score graphs that are produced, either due to their complex nature which makes them difficult to integrate within the proposed frameworks. The framework proposed within this dissertation alleviate the issues complex score functions presents within those frameworks, built with a native deep learning methodology that can be directly integrated into those existing works.

In all the works from which this research took inspiration, a common issue faced by most is the lack of a scoring function, or framework, that can efficiently score the candidate structures that are formed by the models used for causal discovery. Not only does this affect the quality of the final graph with respect to the data and conditional dependencies it is supposed to model, it negatively alters the learning trajectory. In other instances, such as RL-BIC, scoring graphs takes a longer time than it does for the model to produce candidate structures and reinforce patterns based on the scores obtained. These issues were the motivating reasons behind pursuing a more efficient manner of scoring graphs, while also taking care in ensuring there is not a loss of effectiveness in whatever method is developed.

Neural Graph Scoring furthers the assimilation between probabilistic graphical models and deep learning, combining the expressive representational power of GNNs with respect to graphical data and retaining the interpretability of these models. It offers improved scalability with high-dimensional problems of up to 100 nodes (and potentially more), subverting the common limitations due combinatorial explosion while offering a robust framework that could reduce false edges and improve reliability during learning. And since it is a data-driven framework, it is generalizable which can be adapted to different domains. This hybrid approach strengthens both fields through their respective employment for the benefit of the other.

This work also offers a number of practical applications in numerous other fields:

- Genetics: a robust and scalable framework for graph scoring can accelerate the discovery of regulatory networks or signalling pathways, aiding in identifying theraputic targets and disease mitigation.
- Neuroscience: understanding brain connectivity requires the inference of relationships between regions within the brain from what is often high-dimensional, time-series data. The proposed framework could help in improving the accuracy of these inferences and support advances in cognitive research.
- Economics and Social Sciences: policy-making involves understanding the relationship between numerous civil services such as education, healthcare, taxation, development, maintenance and many others. Improved causal discovery can help support decision-making for policies that will best serve communities.

Beyond specific practical applications, incorporating neural networks into causal discovery as part of structure learning helps shift the focus from mere prediction toward explanation and reasoning. This helps mitigate a common criticism of black-box models, and paves the way to more ethical and fair uses of AI in sensitive areas such as risk modelling and credit scoring. It can also be used to enhance causal analysis, making way for automating trained frameworks to learn evaluation metrics as opposed to having them being hand-crafted and fine-tuned manually.

Broadly, this work aims at aiding in bridging the gap between statistical rigor and neural expressivity, paving the way for more causal models that can help separate sporadic occurences from genuine correlation and causal effects. It has the potential to make causal discovery more scalable, accurate, efficient and applicable to real-world problems.

# Chapter 7

# Limitations and Future Work

While this paper demonstrates the promising success of this framework, there are some limitations to be discussed. Firstly, the performance of a model differs substantially depending on the distribution to which the data samples belong to. As shown, Binomial sampled data proved to be the least accurate in prediction across the board. Gumbel sampled data also proved to be less accurate. It should be reiterated that while the results were less than ideal, these complex distributions require more parameters for sampling. Those used in this paper are generalized forms, and future work can involve testing on a more stringent set of parameters for improved performance. Alternatively, research into modifying the model to be more agnostic is also an area of interest for future work.

Secondly, all training data was exclusively scalar valued. Many of the deep learning approaches for causal discovery caters for vector-valued data, an accommodation not made by the encoder/decoder modules processing the raw data. In the linear SEM $Z$ is interpreted as noise matrix, where it resides in the latent space of the autoencoders for vector-valued data. Yu *et al.* [2019] describes the use of the linear SEM to randomly sample variable dimensions. Adapting the model to facilitate data of this nature is another facet of this model that can be implemented.

Thirdly, as demonstrated in Section 5.3 on page 48, the model is at its current stage unsuitable for direct use in structure learning algorithms. Future work can involve a link prediction task, generating a set of possible modifications to make to an input graph to closer approximate the global maximum solution. Additionally, this can be tested on real world data as well.

This applicability of the work presented in this paper is notable as well. Motivated by the need for a faster, more efficient way of scoring graphs generated using deep learning and continuous optimization approaches, this framework can be applied to those models in an attempt to see just how much it boosts performance and network discrimination. Investigating the quality of models produced using this framework to assess graphs such as that of [Zhu *et al.* 2019] using reinforcement learning could potentially improve the end results.

Future work need not be limited to the scope of probabilistic models exclusively, but can be seen as a gateway to its assimilation with other spaces in machine learning. A particularly interesting scenario is related to robotics: if we consider an agent in some environment attempting to understand it, through its gradual exposure to elements in the environment (i.e. variables) it will structure its world view in how the various elements in the world interact with each other. Its ability to construct a causal understanding in an efficient and reliable manner on a moment-to-moment basis becomes vital for navigation and task completion.

The framework presented in this paper can greatly assist in assessing potential world views in constructs at any given moment, choosing to adopt the best generated model with the available information it has accumulated over a period of time. The scalability boasted shows its ability to maintain its assessment of graphs as the agent discovers more concepts in the world it is exploring and displays its improving accuracy with more information available.

Another potential avenue of exploration is the adoption of neural network architectures that deal with predictive distributions as opposed to the implemented single point prediction. This would facilitate more robust insights into the predictive uncertainties of the neural network with confidence and prediction intervals and a better quantification of the predictive uncertainty.

# Chapter 8

# Conclusion

This paper proposes a new method for graph scoring during structure learning of probabilistic graphical models. This is a significant departure from the traditional method of scoring graphs using information-theoretic scoring functions or Bayesian scoring functions. Opting for a deep learning-based approach to graph scoring that is capable of producing estimates of graph scores, this framework is designed specifically to counter the existing challenge of inefficient graph scoring functions present in continuous optimization approach to causal discovery. This paper also shows the potentials thereof by demonstrating its structure learning capabilities by incorporating it into existing heuristic search algorithms at a basic level.

This framework shows its ability to accurately score graphs within relatively small margins deviating from the mean, and is capable of doing so with considerably large graphs up to 100 nodes. It is also suitable for a wide range of samples containing noise matrices belonging to a number of distributions. It also shows that, at its current standing with sufficient systems in place, it can guide the model selection process with decent confidence. This is fertile ground for future work to improve the overall process, factoring in link prediction for more robust scoring.

With the continuous emergence of new and innovative solutions to existing problems, this paper has shown the viability of combining deep learning and probabilistic graphical modelling to further the enhancement of each other. By utilizing neural networks, we have shown how model selection could be improved in continuous optimization for structure learning through its employment, broadening the horizons on which graphical models can be applicable.

This paper argues the importance of thinking outside the box when it comes to finding solutions to many of the challenges faced by graphical models for structure learning. The paradigm of deep learning is ever expanding with new technologies being brought to the forefront at a faster rate than ever before thanks to much more powerful hardware being readily available. In a space where efficient solutions are not likely to be found, this paper intends to serve as another reminder of the wealth of possibility that lies in combining multiple paradigms to find new solutions.

# Bibliography


[Akaike 1973] H Akaike. Information theory as an extension of the maximum likelihood principle. á in: Petrov, bn and csaki, f. In *Second International Symposium on Information Theory. Akademiai Kiado, Budapest, pp. 276Á281*, 1973.

[Al-Mohy and Higham 2010] Awad H Al-Mohy and Nicholas J Higham. A new scaling and squaring algorithm for the matrix exponential. *SIAM Journal on Matrix Analysis and Applications*, 31(3):970–989, 2010.

[Anderson Jr and Morley 1985] William N Anderson Jr and Thomas D Morley. Eigenvalues of the laplacian of a graph. *Linear and multilinear algebra*, 18(2):141–145, 1985.

[Andréasson *et al.* 2020] Niclas Andréasson, Anton Evgrafov, and Michael Patriksson. *An introduction to continuous optimization: foundations and fundamental algorithms*. Courier Dover Publications, 2020.

[Bello *et al.* 2022] Kevin Bello, Bryon Aragam, and Pradeep Ravikumar. Dagma: Learning dags via m-matrices and a log-determinant acyclicity characterization. *Advances in Neural Information Processing Systems*, 35:8226–8239, 2022.

[Bengio *et al.* 2017] Yoshua Bengio, Ian Goodfellow, Aaron Courville, et al. *Deep learning*, volume 1. MIT press Cambridge, MA, USA, 2017.

[Bengio *et al.* 2021] Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. *Advances in neural information processing systems*, 34:27381–27394, 2021.

[Berg *et al.* 2017] Rianne van den Berg, Thomas N Kipf, and Max Welling. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263*, 2017.

[Berrar and others 2019] Daniel Berrar et al. *Cross-validation.*, 2019.

[Bondy *et al.* 1976] John Adrian Bondy, Uppaluri Siva Ramachandra Murty, et al. *Graph theory with applications*, volume 290. Macmillan London, 1976.

[Bronstein *et al.* 2017] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[Buchanan and Shortliffe 1984] Bruce G Buchanan and Edward H Shortliffe. *Rule based expert systems: the mycin experiments of the stanford heuristic programming*

*project (the Addison-Wesley series in artificial intelligence)*. Addison-Wesley Longman Publishing Co., Inc., 1984.

[Buntine 1991] Wray Buntine. Theory refinement on bayesian networks. In *Uncertainty in artificial intelligence*, pages 52–60. Elsevier, 1991.

[Byrd *et al.* 1995] Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on scientific computing*, 16(5):1190–1208, 1995.

[Cannings *et al.* 1978] C Cannings, EA Thompson, and MH Skolnick. Probability functions on complex pedigrees. *Advances in Applied Probability*, 10(1):26–61, 1978.

[Chen *et al.* 2016] Eunice Yuh-Jie Chen, Yujia Shen, Arthur Choi, and Adnan Darwiche. Learning bayesian networks with ancestral constraints. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

[Chen *et al.* 2023] Yangrui Chen, Jiaxuan You, Jun He, Yuan Lin, Yanghua Peng, Chuan Wu, and Yibo Zhu. Sp-gnn: Learning structure and position information from graphs. *Neural Networks*, 161:505–514, 2023.

[Chickering *et al.* 2004] Max Chickering, David Heckerman, and Chris Meek. Large-sample learning of Bayesian networks is NP-hard. In *Journal of Machine Learning Research*, volume 5, pages 1287–1330, 2004.

[Chow and Liu 1968] CKCN Chow and Cong Liu. Approximating discrete probability distributions with dependence trees. *IEEE transactions on Information Theory*, 14(3):462–467, 1968.

[Darroch *et al.* 1980] John N Darroch, Steffen L Lauritzen, and Terry P Speed. Markov fields and log-linear interaction models for contingency tables. *The Annals of Statistics*, pages 522–539, 1980.

[Dawid 1979] A Philip Dawid. Conditional independence in statistical theory. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 41(1):1–15, 1979.

[De Campos and Friedman 2006] Luis M De Campos and Nir Friedman. A scoring function for learning bayesian networks based on mutual information and conditional independence tests. *Journal of Machine Learning Research*, 7(10), 2006.

[Deleu *et al.* 2022] Tristan Deleu, António Góis, Chris Emezue, Mansi Rankawat, Simon Lacoste-Julien, Stefan Bauer, and Yoshua Bengio. Bayesian structure learning with generative flow networks. In *Uncertainty in Artificial Intelligence*, pages 518–528. PMLR, 2022.

[Dobra *et al.* 2004] Adrian Dobra, Chris Hans, Beatrix Jones, Joseph R Nevins, Guang Yao, and Mike West. Sparse graphical models for exploring gene expression data. *Journal of Multivariate Analysis*, 90(1):196–212, 2004.

[Drton and Maathuis 2017] Mathias Drton and Marloes H Maathuis. Structure learning in graphical modeling. *Annual Review of Statistics and Its Application*, 4(1):365–393, 2017.

[Fan *et al.* 2019] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The world wide web conference*, pages 417–426, 2019.

[Gao and Wei 2018] Tian Gao and Dennis Wei. Parallel bayesian network structure learning. In *International Conference on Machine Learning*, pages 1685–1694. PMLR, 2018.

[Goodfellow *et al.* 2014] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[Goudet *et al.* 2018] Olivier Goudet, Diviyan Kalainathan, Philippe Caillou, Isabelle Guyon, David Lopez-Paz, and Michele Sebag. Learning functional causal models with generative neural networks. *Explainable and interpretable models in computer vision and machine learning*, pages 39–80, 2018.

[Grover and Leskovec 2016] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[Hamilton *et al.* 2017] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[Hastings 1973] NAJ Hastings. Dynamic probabilistic systems: Vol. 1: Markov models; vol. ii: Semi-markov and decision processes. *Journal of the Operational Research Society*, 24(2):327–329, 1973.

[Heckerman *et al.* 1995] David Heckerman, Dan Geiger, and David M Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Machine learning*, 20(3):197–243, 1995.

[Hinton *et al.* 2012] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

[Hu *et al.* 2020] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

[Ju *et al.* 2024] Wei Ju, Zheng Fang, Yiyang Gu, Zequn Liu, Qingqing Long, Ziyue Qiao, Yifang Qin, Jianhao Shen, Fang Sun, Zhiping Xiao, et al. A comprehensive survey on deep graph representation learning. *Neural Networks*, page 106207, 2024.

[Kaelbling *et al.* 1996] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[Kalisch *et al.* 2010] Markus Kalisch, Bernd AG Fellinghauer, Eva Grill, Marloes H Maathuis, Ulrich Mansmann, Peter Bühlmann, and Gerold Stucki. Understanding human functioning using graphical models. *BMC Medical Research Methodology*, 10(1):14, 2010.

[Kearns and Vazirani 1994] Michael J Kearns and Umesh Vazirani. *An introduction to computational learning theory*. MIT press, 1994.

[Kingma *et al.* 2013] Diederik P Kingma, Max Welling, et al. *Auto-encoding variational bayes*, 2013.

[Kipf and Welling 2016] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[Kipf *et al.* 2018] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In *International conference on machine learning*, pages 2688–2697. PMLR, 2018.

[Krizhevsky *et al.* 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[Liu *et al.* 2012] Zhifa Liu, Brandon Malone, and Changhe Yuan. Empirical evaluation of scoring functions for bayesian network model selection. *BMC bioinformatics*, 13(Suppl 15):S14, 2012.

[Mikolov *et al.* 2011] Tomáš Mikolov, Anoop Deoras, Daniel Povey, Lukáš Burget, and Jan Černockỳ. Strategies for training large scale neural network language models. In *2011 IEEE workshop on automatic speech recognition & understanding*, pages 196–201. IEEE, 2011.

[Miller *et al.* 1985] Randolph A Miller, Harry E Pople Jr, and Jack D Myers. Internist-i, an experimental computer-based diagnostic consultant for general internal medicine. In *Computer-assisted medical decision making*, pages 139–158. Springer, 1985.

[Murphy 2001] Kevin Murphy. An introduction to graphical models. *Rap. tech*, 96:1–19, 2001.

[Nemirovsky 1999] AS Nemirovsky. Optimization ii. numerical methods for nonlinear continuous optimization. 1999.

[Ostrowski 1937] Alexander Ostrowski. Über die determinanten mit überwiegender hauptdiagonale. *Commentarii Mathematici Helvetici*, 10(1):69–96, 1937.

[Pearl 2014] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier, 2014.

[Pearl 2022] Judea Pearl. Reverend bayes on inference engines: A distributed hierarchical approach. In *Probabilistic and causal inference: the works of Judea Pearl*, pages 129–138. 2022.

[Puterman 2014] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[Quinlan 1982] John Ross Quinlan. *Inferno: a cautious approach to uncertain inference*. Technical report, 1982.

[Rényi 2007] Alfréd Rényi. *Probability theory*. Courier Corporation, 2007.

[Salehi and Davulcu 2019] Amin Salehi and Hasan Davulcu. Graph attention auto-encoders. *arXiv preprint arXiv:1905.10715*, 2019.

[Sanchez-Gonzalez *et al.* 2018] Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. Graph networks as learnable physics engines for inference and control. In *International conference on machine learning*, pages 4470–4479. PMLR, 2018.

[Schwarz 1978] Gideon Schwarz. Estimating the dimension of a model. *The annals of statistics*, pages 461–464, 1978.

[Shortliffe and Buchanan 1975] Edward H Shortliffe and Bruce G Buchanan. A model of inexact reasoning in medicine. *Mathematical biosciences*, 23(3-4):351–379, 1975.

[Sperduti and Starita 1997] Alessandro Sperduti and Antonina Starita. Supervised neural networks for the classification of structures. *IEEE transactions on neural networks*, 8(3):714–735, 1997.

[Spirtes *et al.* 2001] Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, prediction, and search*. MIT press, 2001.

[Sutton *et al.* 1999] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[Tompson *et al.* 2014] Jonathan Tompson, Arjun Jain, Yann LeCun, and Christoph Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. *Advances in neural information processing systems*, 27, 2014.

[Tsamardinos *et al.* 2003] Ioannis Tsamardinos, Constantin F Aliferis, and Alexander Statnikov. Time and sample efficient discovery of markov blankets and direct causal relations. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 673–678, 2003.

[Tsamardinos *et al.* 2006] Ioannis Tsamardinos, Laura E Brown, and Constantin F Aliferis. The max-min hill-climbing bayesian network structure learning algorithm. In *Machine learning*, volume 65, pages 31–78. Springer, 2006.

[Uhler *et al.* 2013] Caroline Uhler, Garvesh Raskutti, Peter Bühlmann, and Bin Yu. Geometry of the faithfulness assumption in causal inference. In *The Annals of Statistics*, pages 436–463. JSTOR, 2013.

[Vaswani *et al.* 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[Velickovic *et al.* 2017] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. Graph attention networks. *stat*, 1050(20):10–48550, 2017.

[Vowels *et al.* 2022] Matthew J Vowels, Necati Cihan Camgoz, and Richard Bowden. D'ya like dags? a survey on structure learning and causal discovery. *ACM Computing Surveys*, 55(4):1–36, 2022.

[Wang and Williams 2010] Xun Wang and Mary-Anne Williams. A graphical model for risk analysis and management. In *International Conference on Knowledge Science, Engineering and Management*, pages 256–269. Springer, 2010.

[Wei *et al.* 2020] Dennis Wei, Tian Gao, and Yue Yu. Dags with no fears: A closer look at continuous optimization for learning bayesian networks. *Advances in Neural Information Processing Systems*, 33:3895–3906, 2020.

[Weisfeiler and Leman 1968] Boris Weisfeiler and Andrei Leman. The reduction of a graph to canonical form and the algebra which appears therein. *nti, Series*, 2(9):12–16, 1968.

[Wermuth and Lauritzen 1982] Nanny Wermuth and Steffen L Lauritzen. *R: Graphical and Recursive Models for Contingency Tables*. Eksp. Aalborg Centerboghandel, 1982.

[Xu *et al.* 2018] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *arXiv preprint arXiv:1810.00826*, 2018.

[Yang and Buehler 2022] Zhenze Yang and Markus J Buehler. Linking atomic structural defects to mesoscale properties in crystalline solids using graph neural networks. *Npj Computational Materials*, 8(1):198, 2022.

[You *et al.* 2019] Jiaxuan You, Rex Ying, and Jure Leskovec. Position-aware graph neural networks. In *International conference on machine learning*, pages 7134–7143. PMLR, 2019.

[You *et al.* 2021] Jiaxuan You, Jonathan M Gomes-Selman, Rex Ying, and Jure Leskovec. Identity-aware graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 10737–10745, 2021.

[Yu *et al.* 2019] Yue Yu, Jie Chen, Tian Gao, and Mo Yu. Dag-gnn: Dag structure learning with graph neural networks. In *International conference on machine learning*, pages 7154–7163. PMLR, 2019.

[Zarebavani *et al.* 2019] Behrooz Zarebavani, Foad Jafarinejad, Matin Hashemi, and Saber Salehkaleybar. cupc: Cuda-based parallel pc algorithm for causal structure learning on gpu. *IEEE Transactions on Parallel and Distributed Systems*, 31(3):530–542, 2019.

[Zhang and Chen 2018] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31, 2018.

[Zhang and Ji 2009] Lei Zhang and Qiang Ji. Image segmentation with a unified graphical model. *IEEE transactions on pattern analysis and machine intelligence*, 32(8):1406–1425, 2009.

[Zhang *et al.* 2018] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[Zheng *et al.* 2018] Xun Zheng, Bryon Aragam, Pradeep K Ravikumar, and Eric P Xing. Dags with no tears: Continuous optimization for structure learning. *Advances in neural information processing systems*, 31, 2018.

[Zhu *et al.* 2019] Shengyu Zhu, Ignavier Ng, and Zhitang Chen. Causal discovery with reinforcement learning. *arXiv preprint arXiv:1906.04477*, 2019.

[Zitnik and Leskovec 2017] Marinka Zitnik and Jure Leskovec. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, 33(14):i190–i198, 2017.