POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Enhanced Graph Reconstruction using Graph Neural Networks

Tesi di Laurea Magistrale in
Computer Science Engineering

Author: **Francesco Puddu**

Student ID: 967500
Advisor: Prof. Giacomo Boracchi
Co-advisors: Mauro Sozio
Academic Year: 2021-22

Abstract

Graphs are a widely used data structure to represent information, where each interconnected item typically comes with a description. This representation has recently gained significant relevance in a variety of disciplines, including chemistry (where it is used to describe molecular structures) and the social sciences (where it is used to describe social networks). Different Machine Learning techniques have been proposed for graph analysis, but these typically cannot be used directly in situations where some of the item descriptions are missing. Given how frequently this issue arises in actual databases, finding methods that enable analysis in the presence of partial missing descriptions is a crucial issue. In this thesis, we address this issue by introducing a Deep Learning-based approach to learn how to reconstruct missing data from the observable portion of the graph. The presentation of the solution is preceded by the formalisation of the task, a review of the relevant literature and the necessary technical background. After describing the process to enable unsupervised optimization, i.e. without necessarily having complete graphs, the architecture of the reconstruction model, which specifically belongs to the class of Graph Neural Networks, is described in detail. The proposed approach is mainly motivated by the possibility of efficiently exploiting the observable information contained in the graph. We highlight how this is precisely a critical issue in the solutions proposed in the literature. Finally, The proposed solution is tested in our experiments under various quality metrics and experimental conditions, and its performance is compared to the main methods that make up the state of the art. The gathered data shows the viability of the suggested strategy and its competitiveness compared to the main solutions in the literature.

Keywords: Missing Data Imputation, Graph Neural Networks, Deep Learning, Graphs

Abstract in lingua italiana

I grafi sono una struttura dati ampiamente utilizzata per rappresentare le informazioni, in cui tipicamente ogni item interconnesso ha una descrizione associata. Negli ultimi anni questa rappresentazione è diventata particolarmente rilevante in ambiti come la chimica, per descrivere strutture molecolari, e le scienze sociali, in cui vengono utilizzate per descrivere reti sociali. Diverse tecniche di Machine Learning sono state proposte per l'analisi dei grafi, ma tipicamente queste non possono essere direttamente applicate nel caso in cui parte delle descrizioni degli item siano mancanti. Considerata la frequenza con cui questo problema si verifica nelle basi di dati reali, la ricerca di strategie che permettano l'analisi in condizioni di mancanza parziale delle descrizioni rappresenta un argomento di rilevanza critica. In questa tesi, affrontiamo il problema proponendo un approccio basato su Deep Learning per imparare a ricostruire i dati mancanti sulla base della parte osservabile del grafo. La presentazione della soluzione è anticipata dalla formalizzazione del problema, una revisione della letteratura rilevante e il background tecnico necessario. Dopo aver specificato la procedura per permettere un'ottimizzazione non supervisionata, ovvero senza disporre necessariamente di grafi completi, viene dettagliata l'architettura del modello di ricostruzione, che appartiene in particolare alla classe delle Graph Neural Networks. L'approccio proposto viene motivato principalmente in virtù della possibilità di sfruttare in maniera efficace l'informazione disponibile contenuta nel grafo da ricostruire, mettendo in luce come questo rappresenti appunto una criticità delle soluzioni proposte in letteratura. Infine, i nostri esperimenti mettono alla prova la soluzione proposta secondo diverse metriche di qualità e condizioni sperimentali, confrontandola con le principali strategie che costituiscono lo stato dell'arte. I risultati raccolti dimostrano la validità dell'approccio proposto e la sua competitività rispetto alle principali soluzioni presenti in letteratura.

Parole chiave: Grafi, Imputazione di Dati Mancanti, Graph Neural Network, Deep Learning

Contents

Introduction

Graphs are data structures that model a set of items connected by relationships, represented by nodes and edges respectively. Each node or edge can also be described by a set of properties, called features. The graph structure is typically used to model a large number of systems in various areas, such as natural sciences, social networks, web search, recommendation systems and many others. For instance, a social network is a group of users connected by their relationships whereas a molecule is a set of atoms connected by their chemical bonds.

Due to the high expressive power of graphs and their widespread use, there is growing interest in analysing them with Machine Learning techniques. This area of research has proposed various successful applications, but recently great attention is being paid to the introduction of Graph Neural Network (GNN). Essentially, this family of models learns to recognise and utilise meaningful patterns within the graph for the task at hand, like the categorization of nodes into different groups. The fact that such learning takes into account both the connection structure and features is one of their key characteristics. Due to their effective results, GNNs have quickly established themselves as the benchmark for analysing graphs in the vast majority of tasks.

Models such as GNNs typically assume that the set of features associated with the input graph is completely known. However, this is often not the case in real-world scenarios, where each feature is generally only observed in a subset of nodes. For instance, not all products in a co-purchase network might be accompanied by a complete description. In situations like this, the feature set contains missing values and models cannot be directly applied. The problem of inferring missing data is generally known as "Missing Data Imputation" (MDI). The search for effective MDI strategies for graph-structured data is nowadays a critical problem to enable the widespread adoption of Machine Learning algorithms in this domain. In this thesis, we will informally refer to this research topic with the term "Graph Reconstruction".

The aim of this thesis is to present a novel approach to Graph Reconstruction. The proposed strategy learns to efficiently reconstruct missing features by exploiting the observable information of the graph, both structural (encoded in the relationships between nodes) and semantic (encoded in the descriptive features). An unsupervised training procedure will be described, which at no stage requires the availability of fully observable graphs or external labelling. Thereby, starting with just the incomplete graph, the solution in question will specify a self-contained pipeline for reconstruction. Finally, we evaluate the quality of the graphs obtained, both in terms of reconstruction error and usefulness for various graph analysis tasks.

Outline of the Thesis

Briefly, the thesis is structured as follows:

- In Chapter 1, a formalisation of the Graph Reconstruction problem is presented.
- In Chapter 2, we discuss the literature on MDI, both in the case of graph-structured and tabular data.
- In Chapter 3 we present an overview of Graph Machine Learning techniques and in particular GNNs, to lay the necessary technical foundation for understanding the solution proposed in this thesis.
- In Chapter 4 we present our approach, which we refer to as Graph Reconstruction Network (GRN), describing both its implementation and the rationale behind the adopted technological solutions.
- In Chapter 5, we analyse analysing the performance of GRN in our experimental setting, comparing its performance with that of the main state-of-the-art methods.
- In Chapter 6 we summarise the work done and draw our conclusions.

1 | Problem Formulation

In this chapter, we formalise the Graph Reconstruction problem by modelling it from an input-output perspective, specifying format, assumptions and constraints for both. The criterion for evaluating the effectiveness of the proposed solution will also be anticipated. Critically, the setting we describe is unsupervised: it enables the optimisation of a reconstruction model using only the target graph with partially missing features as an input. In no part of the process are external labels or complete graphs required.

1.1. Inputs

Given as input:

- An attributed graph $\mathcal{G}$ composed of N nodes. The connections are encoded by the adjacency matrix $A \in \{0,1\}^{N \times N}$. The formulation of the problem is independent of the specific format chosen for A, we refer to Section 3.1 for more details.
- A feature matrix $X \in \mathbb{R}^{N \times F}$, where F is the number of features, composed of a description vector for each node. This is the actual target of the imputation, where missing values are filled with a placeholder μ. This matrix can be thought of as deriving from the original complete version $\tilde{X}$.
- A binary mask $M \in \{0,1\}^{N \times F}$ that specifies whether feature values are available (1) or not (0). Intuitively, any formulation of the MDI problem provides a clear source of information as to which values are missing. In this setting, this mapping is fully encoded in M, where each M_{ij} value specifies the availability of the corresponding X_{ij} feature value.

The following expression encapsulates the relationship between X, M, and $\tilde{X}$:

$$x_{ij} = \begin{cases} \mu, & \text{if } m_{ij} = 0 \\ \tilde{x}_{ij} & \text{if } m_{ij} = 1 \end{cases} \tag{1.1}$$

We use artificially masked feature matrices for the experiments detailed in Chapter 5, so that we can measure the quality of the reconstruction against ground truth values. The two assumptions we make about the missingness of node features, and which we then enforce in the creation of the synthetic masks, are as follows:

- The missingness mechanism is "Missing At Random" (MAR). In statistics, what causes the data to be missing is known as the missingness mechanism. There are essentially three mechanisms: In "Missing Completely At Random" (MCAR) data, the missingness in your data is presented in a completely random pattern, whereas in MAR data, the missingness only depends on the observed values and not the missing ones. Finally, the data is "Missing Not At Random" (MNAR) if neither definition applies. Unless there is a valid reason to believe otherwise, it is common to assume that data is MAR. Additionally, the majority of methods for handling missing data depend on the MAR assumption.
- The missingness pattern is non-monotonic. Following the classification made in [29], this is a missing data pattern whereby the missingness of one variable does not affect the missingness of any other variables.

1.2. Outputs

The goal is to produce a new feature matrix X' in which the originally missing features are imputed by GRN. The matrices X' and A thus define a fully observable graph that is ready to be processed with a standard Machine Learning technique. In our experiments, we evaluate the quality of the output from two angles:

- The **reconstruction loss**. By applying synthetic masks on complete graphs (as anticipated in the previous section) we can measure the deviation between the values imputed by the model and the actual values.
- The **performance loss**. We evaluate this metric by measuring the difference in terms of accuracy between inferences of the same model computed from $\tilde{X}$ and X'.

Exhaustive details on the implementation of these metrics will be provided in Chapter 5.

2 | Related Work

This section aims to provide an overview of the main approaches and the context in which MDI techniques operate, with a particular emphasis on the domain of graph-structured data and Graph Reconstruction techniques.
Missing data treatment techniques can be broadly categorized into two groups:

- Ignoring and discarding: in the case of graph-structured data, this entails removing nodes or edges where missing features are discovered. This operation has repercussions that, unlike in the case of tabular data, go beyond the loss any remaining descriptive data related to the particular item (node) that is being eliminated. In fact, we should also take into account the loss of structural information. For example, the discarding of a node implicitly entails the discarding of all the edges attached to it, which compromises the downstream operation of searching for structural patterns.

- Estimation or imputation: techniques belonging to this category (MDI) aim to fill in the missing values with estimated ones. Approaches can generally be statistical, based on parameter estimation (as Maximum Likelihood) or learning based. The aim of the latter is to employ known patterns that can be identified in the observable values of the data set to assist in estimating the missing values.

This thesis presents a technique belonging to the second category, namely an MDI approach on the graph domain. In this chapter, we present an overview of the main MDI techniques in literature. To give a more comprehensive picture of the key approaches in the area, we first review the literature on classical tabular data. Then, we focus on techniques applied specifically to the graph domain, with which the solution proposed in this thesis is directly compared. Finally, we consider a family of methods that addresses the problem of missing data by adapting existing analysis models rather than focusing on the imputation of missing items, thus embedding the Graph Reconstruction problem in the actual downstream analysis task. In Chapter 5, a subset of the most pertinent state-of-the-art solutions will be picked for comparison with GRN.

2.1. Reconstruction of Tabular Data

By the term tabular data we refer to datasets in which no relations are defined between the individual data points, or equivalently where no graph is defined on the data. The three major categories of MDI techniques that apply to this extremely broad domain are probabilistic approaches, Machine Learning-based approaches, and finally deep learning-based approaches. We list the most pertinent examples for each of these categories. In practice, simple methods like mean imputation are still frequently used and should be regarded as baselines.

Multiple Imputation by Chained Equations (MICE), one of the probabilistic approaches, assumes (and consequently models) the joint distribution of features. missing data is imputed using an iterative cycle of predictive models. Each specified variable in the dataset is imputed conditionally to the other variables in the dataset once for each iteration. The process is iterated up until convergence.

One of the most popular Machine Learning methods used for MDI is the well-known "K Nearest Neighbors" (KNN) algorithm [2]. The imputation of a missing value is computed by means of a weighted average of the respective known values in the data-points most similar to the one considered. It logically involves a significant design decision in the definition of the similarity metric. The popular MissForest algorithm [27] is another example of an iterative approach in which predictions are computed by averaging over many unpruned decision trees. Both of these methods are non-parametric and thus do not define an explicit imputation model. This implies that they must be run every time data is imputed, which could be problematic in some production environments.

Among the Deep Learning-based methods, autoencoder models [21] learn a representation of the data from the input layer and attempt to reproduce it in the output layer. This allows the model to learn from incomplete data and produce plausible values for imputation. Other methods using Generative Adversarial Networks (GAN) [31] train the model to produce imputations that cannot be distinguished from real data by an ad-hoc discriminator.

Although these solutions are directly applicable in the imputation of graph-structured data, using them in this domain would mean to completely ignore the available structure of the graph, which intuitively constitute a valuable source of information for the imputation.

Figure 2.1: Positionally and structurally close nodes. Nodes A and B are positionally close, having relatively close positions in the global network. Nodes A and C are structurally close, having relatively similar local neighborhood structures. Courtesy of [6].

2.2. Reconstruction of Graph-structured Data

Imputation techniques for graph-structured data, which include the solution presented in this thesis, rely on edge structure to guide the imputation process. In this section, we first discuss the relationship between the learning process and the feature distributions on the graph, and then we give an overview of the primary reconstruction techniques.

2.2.1. Feature Distributions on Graphs

When compared to tabular data, the use of graph-structured data offers a new level of complexity due to its duality between structural and semantic information. Seminal works on the link between these two forms of information and the use of GNNs offer interesting insights into how the features we aim to impute are used. In [1], the authors examine the efficacy of various feature initialization strategies in the scenario where the graph is entirely devoid of them. The authors specifically offer a useful framework for describing the distribution of features on a graph, primarily dividing them into positional and structural features. Figure 2.1 illustrates the difference between the two categories.

The connection between the idea of *graph homophily* and GNNs has received a lot of attention. This term refers to the theory in network science which states that, based on node attributes, similar nodes may be more likely to attach to each other than dissimilar ones. The hypothesis draws mainly from the phenomenon of homophily in social sciences. This assumption is commonly adopted in practice either explicitly (for instance in [26]) or implicitly in the choice of benchmarks (such as Cora, CiteSeer and PubMed) that exhibit this property in a very prominent way. Whether GNNs are inherently optimised with respect to this assumption is critically analysed in [17]. The study suggests a perspective on how GNNs learn, emphasizing the investigation of patterns in feature distributions on the graph. Homophilic distributions are merely a special case of these.

2.2.2. Reconstruction Models

Graph Reconstruction techniques see a variety of approaches that differ also with respect to how structure is used to improve the quality of imputation.

In [5], a probabilistic approach is used, and nodes are explicitly modelled as tuples with two variables —one pertaining to structural information and the other to semantic information. The joint distribution between the two variables is modelled and ultimately used for the imputation of missing features after the assumption of shared latent space between the two variables. This approach is obviously constrained by the ability to completely and effectively encode the structural data associated with a node in a real variable, making it unsuitable for identifying complex structural patterns.

Works such as [19] and [10] propose reconstruction approaches derived from signal theory, thus treating the graph as signals via its Laplacian matrix. Finding efficient methods of estimating this matrix is an open problem [18] and a critical challenge of these approaches. Moreover such methods are too computationally intensive and cannot scale to graphs with more than ∼1,000 nodes because of the optimisation problems involved.

Finally, drawing explicit inspiration from the well known Label Propagation algorithm, [23] suggests a method based on an iterative process of propagating the known information on the graph. The feature matrix is multiplied by a diffusion matrix before the known features are reset to their initial value at each iteration up until convergence. The work is presented as a special case of the GRAND [4] framework, one of the main contributions that approaches Deep Learning on graphs as a continuous diffusion process. The method, as the authors note, is based on a strong assumption of the graph's homophily and cannot thus be used to capture non-trivial structural patterns.

2.2.3. Learning on Incomplete Graphs

Furthermore, rather than focusing on the imputation of missing items, one family of techniques addresses the issue of missing data by adapting current analysis models. In other words, unlike the techniques used up to this point, the ultimate objective is to directly resolve the graph analysis task (such as classifying nodes) despite partial missing features.

The "Sparsity Normalisation" technique revisits zero imputation, the most straightforward and natural approach in this field of study [30]. The authors examine how zero imputation affects a Machine Learning model's training process, paying particular atten-

tion to the negative effects of training samples with different levels of sparsity. After performing zero imputation, this method addresses this issue by scaling the size of each feature vector according to its level of sparsity (i.e., the number of missing features), making the change in output less sensitive to the input sparsity level. Among the techniques that specifically adapt the GNN framework to the issue at hand, [28] represents the missing data using the Gaussian Mixture Model (GMM) and determines the expected activation of neurons in the first hidden layer of the Graph Neural Network while maintaining the other layers of the network's structure. The GMM model and the network's parameters are trained end-to-end.

The strategy employed by these techniques, in contrast to the solution suggested in this thesis, represents a different trade-off with regard to the goal of enabling Graph Machine Learning techniques to be used in the case of missing features. rather than creating an intermediate learning process with the goal of reconstructing the graph, the reconstruction operation is embedded in the downstream model, effectively combining the Graph Reconstruction problem and the downstream problem (classification, regression, etc.). Although the latter method avoids the need to optimize a particular model for reconstruction, it entails both the obvious necessity of having to perform a reconstruction at every inference, and the impossibility of bootstrapping the reconstruction.

3 | Background

This chapter provides an overview of techniques that can benefit from Graph Reconstruction and frames this task within the spectrum of Graph Machine Learning techniques. We first briefly discuss the formalization of graphs before presenting the main task categories and examples. Finally, we concentrate especially on Graph Neural Networks, which serve as the main technology for GRN.

3.1. Graph Representation

As mentioned in the introduction, we can intuitively represent a graph $\mathcal{G}$ with a tuple $(\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, v_2, ...v_N\}$ is the set of nodes and $\mathcal{E} \subseteq N \times N$ the set of edges. Optionally, nodes and/or edges may be defined with a vector representation, to encode the semantic information specifically associated with them. This mathematical representation is typically translated into more practical matrix forms for processing, one of the most popular of these is the one used in this thesis, and it is composed of:

- An adjacency matrix A: a square matrix of size $N \times N$ that shows which nodes are directly connected to which others, where $A_{ij} = 1$ if n_i and n_j are connected, 0 otherwise. Note that most graphs are not densely connected and therefore have sparse adjacency matrices. For this reason, an alternative sparse representation of A is the list of edges, where each one is represented by the pair of the node indexes.
- Up to two feature matrices: a matrix of size $N \times F$, where F is the number of features, which combines the descriptive vectors associated with each node and/or a matrix of size $E \times F$ for the edge features.

Graphs are very different from typical objects used in Machine Learning, mainly because their topology is more diverse than that of a sequence (such as text and audio) or an ordered grid (such as images). Even if graphs can be represented as lists or matrices, this does not detract from the fact that they are inherently unordered objects. Intuitively, if you have a sentence and rearrange its words, you create a new sentence.

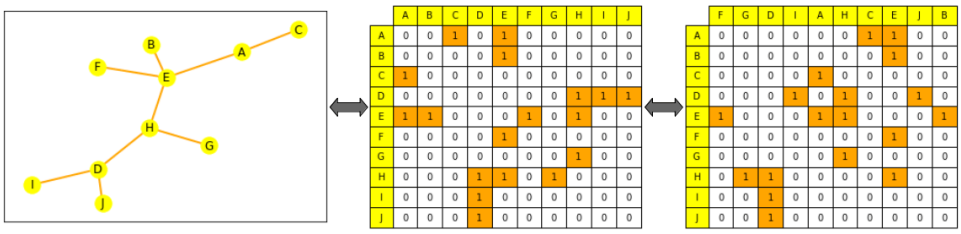

Figure 3.1: On the left, a small graph. In the centre, its adjacency matrix, with columns and rows ordered in alphabetical order: on the row for node A (first row), we can read that it is connected to E and C. On the right, a shuffled adjacency matrix, which is also a valid representation of the graph. Courtesy of Hugging Face (`https://huggingface.co/`).

If you have an image and shuffle its columns, you create a new image. A graph instead retains its identity even after you rearrange its edge list or the columns of its adjacency matrix. This characteristic is formally referred to as permutation invariance.

3.2. Learning Setting

The level of target granularity allows us to distinguish between the tasks we can define on graphs:

- At the **graph level** the main tasks are generation (used for example in drug discovery to generate new plausible molecules), evolution (used in physics to predict the evolution of systems) and of course prediction (categorisation or regression tasks such as predicting the toxicity of molecules).
- At the **sub-graph level**, the majority of studies focuses on property prediction or community detection. Community detection is a technique used by social networks to identify connections between users. Systems for creating itineraries, like Google Maps, use sub-graph property prediction to predict expected arrival times.
- At the **edge level**, property prediction and missing edge prediction are the primary tasks. Drug side effect prediction is aided by edge property prediction in predicting negative side effects when two drugs are used. Systems for making recommendations use missing edge prediction to determine the relationship between two nodes in a graph.
- At the **node level**, it's usually a node property prediction. For example, AlphaFold [14] uses node property prediction to compute the 3D coordinates of atoms given the overall graph of the molecule, and therefore predicts how molecules get folded in 3D space, a hard biochemistry problem.

We can additional distinguish Graph Machine Learning techniques between:

- **Transudctive**: when the task is solely focused on a single graph and all training, validation, and testing are conducted on that graph.
- **Inductive**: when learning occurs through the submission of numerous graphs.

Particularly, the Graph Reconstruction task is here described as transductive and node-level.

3.3. Unsupervised Graph Representation Learning

This field of study is primarily concerned with embedding techniques, i.e. models that learn a low-dimensionality vector representation for each node or graph. At a high level, the operation can be explained as a mapping of the data structure onto a Euclidean space in which the distance reflect the structure of the original graph. For example, in the case of node embedding, we want the distance between two nodes in the Euclidean space to be proportional to their proximity in the graph.

Node2vec [11] is an algorithm to learn low-dimensional embeddings of nodes through the use of random walks on the graph. It follows the intuition that random walks through a graph can be viewed as sentences in a corpus. Therefore, a random walk is treated as a sentence and each node in a graph is treated as a separate word. Finally, a Skip-gram model is used to compute the embedding vectors from the sentences.

The embedding vectors generated by these methods can be interpreted as coordinates in an Euclidean vector space in which we can compute a pairwise distance matrix between nodes. A typical application are graph-based recommendation systems such as [22].

3.4. Graph Neural Networks

Graph Neural Networks (GNNs) were introduced in [9, 25] as a generalization of recursive Neural Networks that can directly deal with a more general class of graphs, e.g. cyclic, directed and undirected graphs. GNNs consist of an iterative process which propagates the node representations until equilibrium; followed by a Neural Network, which produces an output for each node based on its representation.

The feature propagation operation, which is guided by the graph structure, is commonly called *graph convolution*, and can be seen as a generalization of the convolution operation frequently used in computer vision.

Generalising the convolution operator to irregular domains is typically expressed as a *neighbourhood aggregation* or *message passing* scheme. With $h_i^{(l)} \in \mathbb{R}^F$ denoting latent node features of node i in layer l, $e_{ji} \in \mathbb{R}^D$ denoting (optional) edge features from node j to node i, a graph convolution layer can be modeled as:

$$h_i^{(l+1)} = \gamma^{(l+1)} \left(h_i^{(l)}, \bigoplus_{j \in \mathcal{N}(i)} \varphi^{(l+1)} \left(h_i^{(l)}, h_j^{(k-1)}, e_{j,i} \right) \right) \tag{3.1}$$

Where $\bigoplus$ denotes a differentiable, permutation invariant function, e.g., sum, mean or max, and γ and φ denote differentiable functions such as Multi Layer Perceptrons (MLPs). In order to maintain consistency with the nomenclature used in the literature, we use the abbreviations x_i for the original node features and h_i for the subsequent latent representations. In GCN [15], which we take as reference, an instance of the previous graph convolution is used:

$$h_i^{(l+1)} = \sigma \left(\sum_j \frac{1}{c_{ij}} h_j^{(l)} W^{(l)} \right) \tag{3.2}$$

This equation exemplifies the update of the representation vector of the i-th node, i.e. h_i. The combination of the vectors of each of its neighbours j , normalised by the factor c, is passed through a single layer Neural Network, where W are the layer weights and σ the non-linear activation.

Finally, we note that the format chosen to represent the graph (between sparse and dense, as described in Section 3.1) directly influences complexity, both in terms of space and time. As proved by [3], for a model with L convolutional layers of type 3.2 applied to a graph with $N = |\mathcal{V}|$ nodes and $E = |\mathcal{E}|$ edges, the complexity during forward and backward passes is shown in Table 3.1.

		Dense	Sparse
Time	Forward	$\mathcal{O}(LN^2F + LNF^2)$	$\mathcal{O}(LEF + LNF^2)$
	Backward	$\mathcal{O}(LN^2F + LNF^2)$	$\mathcal{O}(LEF + LNF^2)$
Space	Forward	$\mathcal{O}(N^2 + LF^2 + LNF)$	$\mathcal{O}(E + LF^2 + LNF)$
	Backward	$\mathcal{O}(N^2 + LF^2 + LNF)$	$\mathcal{O}(E + LF^2 + LNF)$

Table 3.1: Asymptotic complexity of a GCN model

4 | Proposed Solution

In this chapter we present a novel approach to the feature reconstruction problem for graph-structured data. This will primarily be expressed in two aspects:

- A new learning setting to tackle the Graph Reconstruction problem, discussed in Chapter 1, as an unsupervised node-level task in the Machine Learning framework.
- A reconstruction model to be optimised within this learning setting, which we refer to as Graph Reconstruction Network (GRN). The architecture belongs to the GNN framework and employs graph convolutions aware of the partial lack of node features.

It is important to emphasise that the one presented in this chapter is an approach and not a specific model. Although the technological solutions and engineering principles of the architecture are here explained in detail, different reconstruction tasks may call for a different set of hyperparameters. In light of the experiments (detailed in Chapter 5), we discuss the insights we have into the different degrees of freedom of GRNs.

4.1. Learning Setting

The central idea of this setting is that learning occurs in an unsupervised manner on the input graph, and thus aims to exploit known information to reconstruct missing information on it. For ease of discussion, the presented setting has a feature-specific objective. Experiments performed on a multi-feature extension showed no significant difference in terms of reconstruction quality.

To target the reconstruction of a (partially observable) feature f on G, a subset t_f of its known instances is used to extract ground truth labels for training purposes. The elements of t_f are hidden from the training data, thus defining a new mask matrix M_f. Note that the set complementary to t_f remains available to the model as a source of information, supporting the idea that the distribution of known values of f may also be useful for reconstruction. Figure 4.2 represents a toy example of this setting.

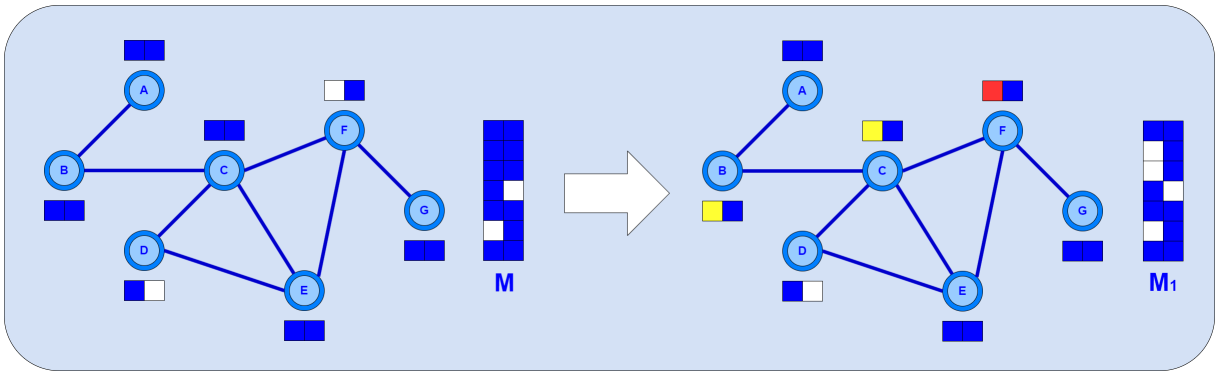

Figure 4.1: To optimize a reconstruction model for the first feature (on the left), a subset of the known (blue) values is hidden (yellow) to extract ground truth labels for training. At inference time, the model imputes actual unknown values (red). We also display the masks, with the original mask on the left and the training-specific mask on the right.

The reader should not be confused by the introduction of labels, typically associated with supervised learning approaches, within the description of an approach declared to be unsupervised. Since the labels are automatically extracted from the input data by masking, they are therefore not the product of external supervision.

4.2. Architecture

Starting with the architectural decisions made, particularly in relation to the GNN framework of which it is a part, we present the GRN model in a top-down manner. As stated in the Introduction, if there are missing features in the input graph, GNN cannot be used directly. As a result, GRN will offer technological solutions developed especially to address that.
The rationale behind selecting GNNs for Graph Reconstruction is to take advantage of the dual nature of the information present in the input data:

- **Semantic**: the feature description of each node.
- **Structural**: the relationships between nodes, encoded by the edges.

We divide the architecture into two phases to clarify how GRN extracts and makes use of this information. The first phase, aimed at feature extraction, consists of a stack of convolutional layers that progressively update the latent representation of each node in the graph. The actual prediction for each target node is computed in the second phase using a single-layer Neural Network, which is shared among the nodes. The two phases are trained end-to-end.

As anticipated, only the first convolutional layer has to process representations from partially missing data and thus requires the mask-aware operator (4.1). A standard graph convolution (3.2) is implemented in subsequent layers.
The number L of stacked convolutional layers, each with a specific set of weights W, is a hyper-parameter that is directly related to the concept of *receptive field*. As mentioned in Chapter 3, each layer computes the aggregation of representations at a distance of 1-hop for each node. It makes intuitive sense that stacking L layers provides latent representations that are, in the end, influenced by nodes located L hops away. Some architectures (such as [20], designed for malware detection) concatenate the activations of all convolutional layers at the end of the feature extraction phase in order to consider multi-scale substructure features more explicitly. Exploratory experiments empirically show that this operation does not lead to any substantial gain in the case of GRN.

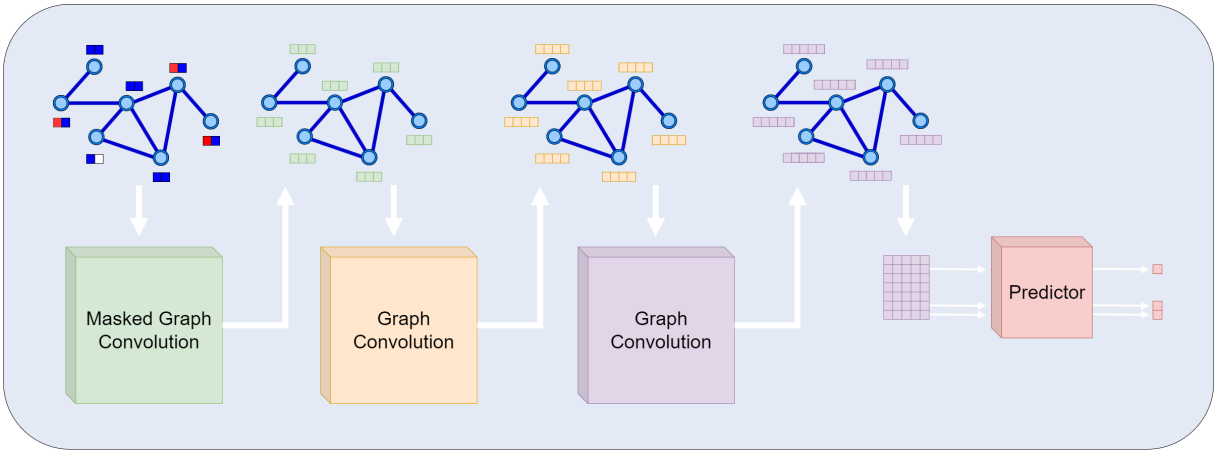

Figure 4.2: A scheme of GRN during inference to impute missing features (red). All feature vectors are progressively updated from the convolutional layers in the first phase, while only the target ones are processed for the actual imputation by the predictor.

Furthermore, we added a dropout layer following each graph convolutional layer for regularization and used ReLU as the activation function to perform non-linear transformations while preventing the vanishing gradient problem. We perform a comprehensive grid search for the best hyper-parameter settings including the learning rate, number of epochs and weight decay. The values used in our experiments are detailed in Chapter 5.
Although a dedicated instance must be trained for each feature to reconstruct, our experiments suggest that using lightweight architectures - ultimately trained on a reasonably sized sub-graph - is sufficient in practice to ensure a competitive trade-off between training time, inference time, and prediction quality.

4.3. Masked Graph Convolution

We have seen how the imputation performed by GRN relies on the recognition of learned patterns on the graph. The network searches for this patterns while updating the node representations with convolutions, whose general form we have expressed in 3.2. However, the feature matrix H is only partially observable in the first convolutional layer, making the use of a mask-aware variant in this layer necessary. We then implement a *masked convolution*, similar to the one proposed in [13]:

$$h_i^{(l+1)} = \sigma\left(\frac{\sum_j \left(m_j \odot h_j^{(l)}\right)}{\sum_j m_j} W^{(l)}\right) \tag{4.1}$$

By allowing the known information on the graph to propagate and subsequently update the vector representations of the nodes, this variant aims to eliminate the contributions of non-observable channels. This is achieved through the hadamard product between each feature vector h_j and the corresponding mask m_j at the numerator, which excludes the contribution of the missing entries from the process. Note that this is fundamentally distinct from, for instance, what we could accomplish by imputing missing values with zero: since neighbourhood aggregation typically involves averaging values, considering null entries would introduce noise into the process. The consequences on the learning process have been fully analyzed in [30]. In practice, the masked convolution operator has been implemented using the matrix form of the previous operator:

$$H' \leftarrow \sigma[(A(M \odot H) \oslash AM)W] \tag{4.2}$$

Where $\odot$ and $\oslash$ denote element-wise multiplication and division respectively. Since both M and A allow the use of sparse matrix multiplication, the overall operation is computationally efficient.

4.4. Subgraph Sampling

We conclude the chapter with a mention to the possibility of performing graph sampling in order to make the training process more efficient. In the field of Graph Neural Networks, scalability is a much-studied characteristic, and many contributions make suggestions for sampling methods on the input graph.

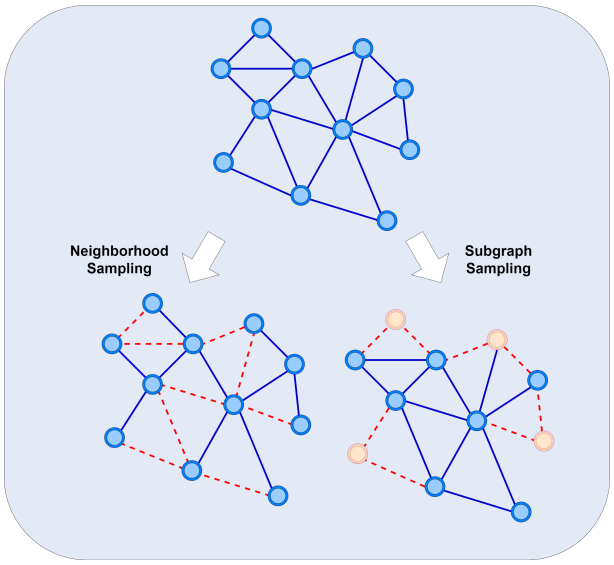

Figure 4.3: A comparison of neighbourhood sampling and subgraph sampling. The edges and nodes highlighted in red are those excluded from the sample.

An important classification among these techniques separates node-wise from subgraph-wise, as shown in Figure 4.3 and thoroughly examined in [16]. The first group is typically the focus of research, where we discover methods to decrease the neighbourhood size of each node by masking edges (e.g. in [12]).

Instead, we propose for the Graph Reconstruction setting the application of a subgraph-wise approach, in which we sample a subset of nodes (and the edges between them). This reduces the dimensionality of the input data considerably more than node-wise techniques, as the size of the result is (potentially) constant. The application of this sampling operation is based on the assumption that a model trained to reconstruct a subgraph will be able to generalise when applied to the general graph. We will analyse this assumption in Section 5.5.3.

As the presentation of new subgraph-sampling techniques is out of the scope of this thesis, we confine ourselves to presenting the exploratory results obtained with a randomised baseline approach. In particular, to select a subgraph from the complete graph, we adopt an heuristic with constant complexity: for T times we randomly select a node and proceed to construct subgraph S from it with a breadth-first approach, until Z nodes are connected. At the end of each iteration, we assign S a score directly related to the amount of missing information within it. Finally, we select the subgraph with the lowest score among those sampled.

5 | Experiments

In this chapter, we outline the experimental setup and the outcomes of a series of tests that we used to evaluate performance of GRN. The characteristics of the datasets (graphs) under consideration as well as the selection criteria will first be discussed. The adopted performance measures will then be explained and motivated. Finally, we will review the findings in relation to a number of state-of-the-art methods that approach the Graph Reconstruction problem in various ways.

5.1. Datasets

The domain of the chosen graphs is social networks, one of the areas of greatest interest for Graph Reconstruction. We use publicly available datasets from the MUSAE project [24] research in particular:

- **GitHub**: nodes (37k) stand for developers, and the edges (289k) represent the connections between them. The professional profile of the developer is encoded in the node features. Labels show whether a node represents a specialist in the field of Machine Learning based on the job title.

- **Facebook**: official Facebook pages are represented by nodes (22k), and site-to-site mutual likes are represented by edges (171k). The site descriptions that the page owners provided to summarize the purpose of the site are the source for node features. Labels such as "company" and "government organization" identify the category to which the page belongs the social platform.

- **Twitch**: nodes (7k) represent users and edges (35k) are mutual follower relationships between them. Vertex features are extracted based on the activity of the specific user on the platform. Labels indicate whether the user has been flagged for the use of explicit language.

Work in the field of Graph Machine Learning typically relies on a specific set of datasets, namely: Cora, CiteSeer and PubMed.

All three of these are citation graphs in which the nodes are scientific publications and the edges are the relationships between citations. For this kind of graph, each node's descriptive attributes typically include a summary of the textual content of the article. This summary typically appears in the form of a *bag of words* vector indicating whether each word from the dictionary is present or not. In this thesis, we chose to consider different datasets mainly for two reasons:

- Citation graphs are typically complete in real-world settings because they are methodically derived from databases of journals, conferences, etc.; as a result, they do not represent an example domain pertinent to the task addressed in this work.
- Citation graphs have an high degree of homophily, as thoroughly discussed in [17], which leads to trivial feature distributions and thus little experimental interest.

To produce masks that are consistent with the data assumptions and that represent various missingness scenarios we use two probabilistic distributions:

- ϕ_1 represents a skew-normal $SN(a)$, with a being the shape parameter.
- ϕ_2 represents a Bernoullian $Bern(p)$.

For each feature, a first sample from ϕ_1 specifies the missingness rate, which serves as parameter p for ϕ_2. Finally, we sample from ϕ_2 whether each value is available (1) or not (0), storing the results in M. As required by the MAR hypothesis, the result is independent of the value of the single features.

5.2. Metrics

We assess our experiments in light of two relevant performance measures:

- **Reconstruction Loss**: similar to most studies, we report the Root-Mean-Square Error (RMSE), with the error being $|\tilde{X} - X'|$. The result is normalised between 0 and 1 with respect to the loss corresponding to zero imputation, for ease of interpretation.
- **Downstream Loss**: since this is a node classification task, we report the difference in terms of average class accuracy score between a model trained on $\tilde{X}$ and one trained on X'.

The first measure describes the potential of the approach in terms of pure reconstruction, while the second helps us understand how much this value translates into the possibility of successfully using the imputed graph for an analysis task.

5.3. Hyper-parameter Optimization

For each dataset, we carry out a tuning process of hyper-parameters such as the number of graph convolutions, latent dimensions, weight decay, and the dropout rate. We employ GRN instances tuned with a dropout rate of 0.5, a weight decay set to 0.0005, and trained for 100 epochs with an initial learning rate of 0.01. Adam serves as our SGD optimiser.

The parameter configurations used vary mainly in terms of the complexity of the feature extraction part of the model. While we use two convolutional layers with hidden dimensions of 128 and 32, respectively, for the GitHub and Twitch datasets, we found that an additional upstream convolutional layer with a hidden dimension of 256 produced the best results for the Facebook dataset. We contend that the reason for this is due to the the increased complexity of the node representation, linked in particular to the presence of textual descriptions (encoded as vectors through embedding techniques).

Although it makes intuitive sense that the complexity of the architecture tends to increase as the complexity of the reconstruction task rises, it is interesting to note that in the proposed experimental setting, the competitiveness of lightweight models emerges empirically. For each of the three benchmarks under consideration, we find the optimal number of parameter to be below 400k.

5.4. Compared Methods

We compare our approach to various state-of-the-art feature imputation solutions. Moreover, as a baseline for each experiment, we report the results obtained with naive statistical methods like as global (GM) and neighbourhood-based (NM) mean imputation. We refer to Chapter 2 for more details on the other techniques chosen for the comparison.

The methods selected from Section 2.1 only take the feature matrix as an input; they do not take the adjacency matrix into account because they are meant to reconstruct tabular data. In particular we choose MICE, MissForest (MF), Variational Autoencoder (VAE) and Generative Adversarial Networks (GAN), following the implementations of the works cited as references.

Since they also take into account the graph structure for the reconstruction task, the methods presented in Section 2.2.2 are those that (GRN) directly compares with. We concentrate on Feature Propagation (FP) due to the limitations of probabilistic or signal theory-based methods mentioned above.

Finally, we select GCNmf [28] from Section 2.2.3. Being an approach aimed directly at the downstream task (and not at the intermediate step of feature imputation) it is not directly comparable with GRN. Consequently, our imputation solution is combined with classification models that complete the downstream pipeline, namely a standard GCN [15] and a more advanced Spline network [8], a state-of-the-art convolutional architecture on the main node classification benchmarks.

5.4.1. KNN in the Embedding Space

The K-Nearest-Neighbors algorithm, which was covered in Section 2.1, is a popular choice for MDI. It is based on reconstructing the missing portion of a given data point using the values observed in the most similar data points, where the similarity function is logically the fundamental component of the strategy. In the Graph Reconstruction problem, defining a similarity function between the data points, or the nodes, is not trivial. To enrich the experimental setting we present a method that extends the KNN approach to the graph domain by defining a similarity function based on the graph structure. The pseudo code of the method is illustrated in Algorithm 5.1, using functions that we detail below.

Algorithm 5.1 K-Nearest-Neighbors Regression in the Embedding Space

Input: X, A, k
C = NodeEmbeddingStrategy(A)
$D = C \cdot C^T$ {compute the pairwise distance matrix}
$X' = X$ {initialize the new feature matrix}
for each node i **do**
  for each feature f **do**
    if f is missing **then**
      neighbors$_{if}$ = KNN(D, i, k)
      prediction$_{if}$ = WeightedMean(neighbors$_{if}$, i, f, X, D)
      X'_{if} = prediction$_{if}$
    end if
  end for
end for
Output: X'

The algorithm performs a KNN regression in the embedding space that encodes the graph structure. Figure 5.1 shows a diagram of the intuition behind the method.

We provide additional details on the functions mentioned in the algorithm:

- *NodeEmbeddingStrategy* is an instance of the node embedding techniques covered in Section 3.3 . Essentially, given as input the adjacency matrix A, it returns a real vector for each node. The outputs can be seen as coordinates in a specific the Euclidean vector space (embedding space). The implementation used in this work employs node2vec [11].
- *KNN* returns the indices of the nodes in the embedding space that are closest to i and contain the feature f.
- *WeightedMean* performs the actual regression by computing a weighted average over the distance of the selected k neighbours.

The complexity of the operation is dominated by the computation of the pairwise distance matrix D, which is generally an $O(n^2)$ operation. A standard evolutionary optimisation algorithm [7] was used to fix the degrees of freedom for this technique, namely the hyper parameter k and the parameters of node2vec. The rationale of this method is to exploit known node embedding techniques to project the graph onto a continuous space in which we can use a Euclidean distance encoding graph proximity, thus creating the conditions to exploit a KNN regression strategy. To the best of our knowledge, this is the first such approach in the context of MDI on graphs.

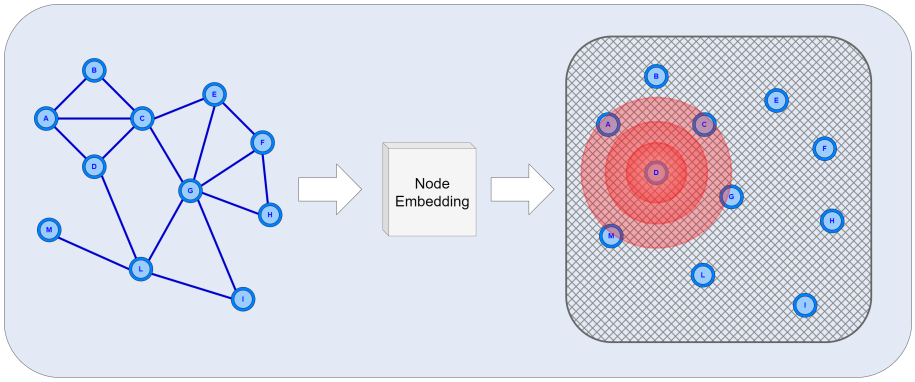

Figure 5.1: The application of a KNN with Euclidean distances between nodes. The original graph's nodes are projected onto a continuous vector space (shown in gray), which allows us to determine the reciprocal distance between them. The nodes' positions in the original graph are related to their coordinates in the new embedding space.

5.5. Results

In this section, we present and discuss the findings of the experiments conducted. The objectives of the experimental process are:

- To assess the performance of GRN according to the metrics of interest and to analyse the main technical characteristics.
- To compare the performance of GRN against relevant solutions of MDI on graphs.
- To compare the performance of GRN against relevant solutions that adapt pre-existing models to enable learning on incomplete graphs.
- To explore the potential of data sampling process to enable a boost in the training process.

While the first objective is across all analyses, the following three objectives each have a unique set of experiments that we present. We repeat the experiments in three different missingness settings, changing the shape of ϕ_1 as depicted in Figure 5.2 to simulate scenarios with high, low, and intermediate missingness.

In all three scenarios, given the score p_f sampled from ϕ_1 for feature f, each value in the f-th column of the feature matrix has exactly probability p_f of being hidden (set to 0) in the mask. Further details about the sampling are provided in Section 5.1. Out of all values in the feature matrix, the distributions were intended to produce on average 75%, 50%, and 25% missing entries, respectively.

5.5.1. Comparing MDI Solutions

The first set of experiments compares imputation solutions for missing features (discussed in Sections 2.1 and 2.2.2). Downstream of the graph reconstruction, we use an inference model (a standard GCN architecture [15]) for assessing the performance loss. The high-level comparison of the results, shown in Table 5.1, clearly shows the intuitive inverse link between reconstruction performance and missingness rate.

Although even the most simplistic methods such as GM and NM succeed in reconstructing some of the missing data, their results are not comparable to the other learning-based approaches. It is significant to note that, in comparison to the other solutions, GRN consistently demonstrates to be relatively less sensitive to the missingness rate, hinting at the fact that it has a higher interpolation capability.

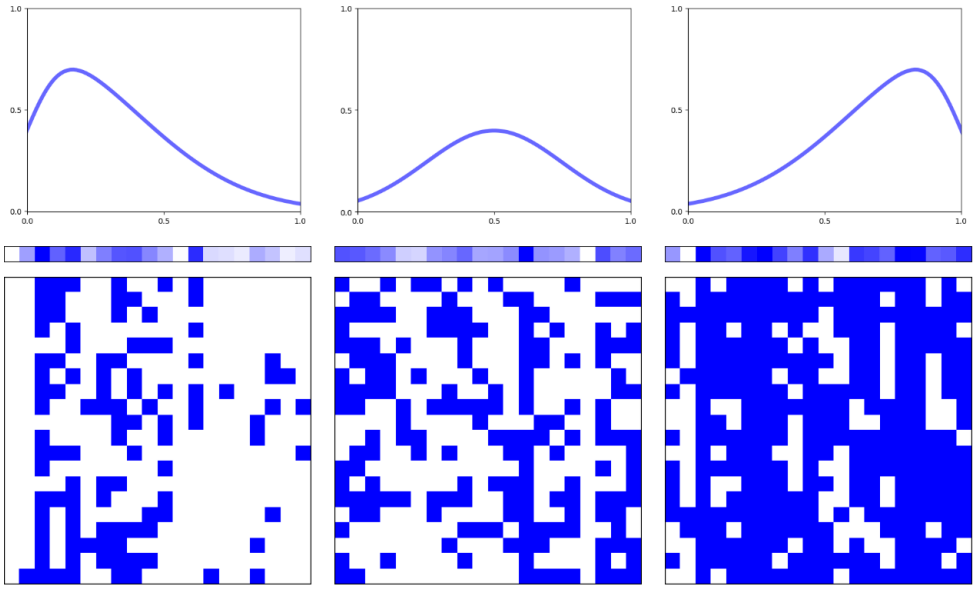

Figure 5.2: The three patterns of missingness used in the experiments: (from the left) high, medium and low. An example of synthetic mask, with each row specifying the known (blue) and missing (white) values for the respective node, is placed beneath the probability density function used to create it. The missingness rate for each feature (column) are indicated by the horizontal vector above, in which the most sparse features have colors tending towards white.

We observe that, in terms of sensitivity to the missingness rate, the results for FP partially conflict with the authors' assertions. We contend that this is because of the different experimental conditions with respect to the ones discussed in Section 5.1.

On both metrics, the results show a competitive performance. It is evident that a significant deficit in reconstruction does not translate into an equivalent gap in downstream loss, especially when comparing the two scenarios with high and low missingness rates. This is due to the fact that part of the information used for classification is only structural and encoded in the adjacency matrix, which remains unchanged in the two scenarios.

We observe that the performance of GRN is generally comparable to, if not slightly superior to, the models taken into consideration, with a stronger advantage in the scenario with a high missingness rate. In scenarios with a moderate or low missingness rate, KNN-based imputation and FP both seem to be competitive as well. In general, we observe a convergence of these two methods, which at their core both employ the local smoothing strategy.

5.5.2. Comparing Complete Pipelines

In order to compare the proposed solution with GCNmf (discussed in Section 2.2.3), we conduct a second set of experiments using two downstream classification models: a basic GCN and SplineCNN [8], which employs a more sophisticated convolution strategy based on B-splines. Comparing the three scenarios, shown in Table 5.2, we find that the margin of improvement is larger in cases of high missingness, indicating that the approach taken by GRN (as well as the other techniques discussed in the previous section) of decoupling the reconstruction task from the classification task has a stronger competitive advantage in the more challenging situation.

The outcomes in terms of performance further show that GRN's reconstruction quality is competitive on the chosen benchmarks. The GitHub and Twitch datasets, in particular, yield the overall best results when tested against GCNmf. Empirically, the downstream classification model chosen has a significant impact on performance: the pipeline using SplineCNN consistently reports the best results.

5.5.3. Graph Sampling Analysis

We conclude by providing an exploratory analysis of the subgraph sampling methodology that we described in Section 4.4. We obtained the outcomes in Table 5.3 by varying the subgraph's order (Z), after limiting the number of iterations (T) to 50 and simply using the number of missing values as score assigned to each sample. The analysis of the results suggests that the sampling procedure represents a potentially advantageous trade-off between reconstruction quality and input dimensionality during training: in several instances, there is a minor difference between the results from the full graph and the fixed-size sampled subgraph.

Although it is reasonable to assume that sampling is advantageous when the patterns the graph's features describe are very local, determining this property a priori is not straightforward. Furthermore, more advanced subgraph sampling techniques might raise the stakes in the trade-off. As was expected, these findings could serve as the foundation for a more in-depth investigation into the significance of sampling methods for the Graph Reconstruction task. The insights discussed in this section are indicative of the potential of this research direction.

Missingness Rate: Low

	Reconstruction Loss			Downstream Loss		
	GitHub	**Facebook**	**Twitch**	**Github**	**Facebook**	**Twitch**
GM	0.210	0.181	0.229	0.116	0.109	0.123
NM	0.157	0.147	0.170	0.098	0.094	0.101
MF	0.101	0.096	0.113	0.073	0.069	0.081
VAE	0.099	0.087	0.105	0.068	0.071	0.083
GAN	0.085	0.073	0.095	0.055	0.059	0.076
FP	**0.041**	0.033	0.071	0.038	0.043	0.055
KNN	0.045	0.035	0.065	**0.032**	0.049	0.061
GRN	0.044	**0.030**	**0.063**	0.035	**0.040**	**0.052**

Missingness Rate: Medium

	Reconstruction Loss			Downstream Loss		
	GitHub	**Facebook**	**Twitch**	**Github**	**Facebook**	**Twitch**
GM	0.488	0.375	0.530	0.354	0.283	0.412
NM	0.429	0.318	0.422	0.292	0.258	0.331
MF	0.313	0.239	0.350	0.225	0.212	0.267
VAE	0.322	0.234	0.344	0.231	0.201	0.266
GAN	0.307	0.240	0.341	0.242	0.210	0.253
FP	0.226	0.201	0.258	0.196	0.185	0.224
KNN	0.231	**0.195**	0.243	0.201	**0.184**	0.236
GRN	**0.172**	0.211	**0.208**	**0.165**	0.186	**0.191**

Missingness Rate: High

	Reconstruction Loss			Downstream Loss		
	GitHub	**Facebook**	**Twitch**	**Github**	**Facebook**	**Twitch**
GM	0.670	0.613	0.662	0.410	0.387	0.471
NM	0.631	0.558	0.544	0.372	0.340	0.421
MF	0.541	0.473	0.437	0.301	0.386	0.360
VAE	0.409	0.351	0.438	0.289	0.320	0.347
GAN	0.388	0.362	0.452	0.257	0.287	0.354
FP	0.445	0.411	0.468	0.326	0.356	0.410
KNN	0.431	0.423	0.454	0.339	0.368	0.419
GRN	**0.341**	**0.334**	**0.390**	**0.299**	**0.315**	**0.334**

Table 5.1: Comparison of feature reconstruction models. Values are averaged over 50 runs for "Reconstruction Loss" metric and 10 training procedures for "Downstream Loss".

Missingness Rate: Low

	Downstream Loss		
	GitHub	**Facebook**	**Twitch**
GCNmf	0.038	**0.037**	0.054
GRN & GCN	0.035	0.040	0.052
GRN & Spline	**0.033**	0.039	**0.048**

Missingness Rate: Medium

	Downstream Loss		
	GitHub	**Facebook**	**Twitch**
GCNmf	0.188	**0.184**	0.211
GRN & GCN	0.165	0.186	0.191
GRN & Spline	**0.159**	0.185	**0.175**

Missingness Rate: High

	Downstream Loss		
	GitHub	**Facebook**	**Twitch**
GCNmf	0.310	0.359	0.468
GRN & GCN	0.299	0.315	0.334
GRN & Spline	**0.270**	**0.296**	**0.413**

Table 5.2: Comparison of complete node classification pipelines. Values are averaged over 10 training procedures.

Missingness Rate: Low

	Reconstruction Loss		
	GitHub	**Facebook**	**Twitch**
Cardinality	37k	22k	7k
Z = 500	0.056	0.038	0.065
Z = 1000	0.052	0.033	0.064
Z = 5000	0.047	0.032	0.064
Full Graph	0.044	0.030	0.063

Missingness Rate: Medium

	Reconstruction Loss		
	GitHub	**Facebook**	**Twitch**
Cardinality	37k	22k	7k
Z = 500	0.214	0.229	0.245
Z = 1000	0.201	0.225	0.222
Z = 5000	0.194	0.224	0.215
Full Graph	0.172	0.211	0.208

Missingness Rate: High

	Reconstruction Loss		
	GitHub	**Facebook**	**Twitch**
Cardinality	37k	22k	7k
Z = 500	0.384	0.349	0.407
Z = 1000	0.372	0.345	0.400
Z = 5000	0.361	0.341	0.396
Full Graph	0.341	0.334	0.390

Table 5.3: Exploratory results on the efficacy of subgraph sampling with constant sized samples. Values are averaged over 50 runs.

6 | Conclusions

This thesis addressed the problem of Missing Data Imputation on Graphs. As a key insight to the problem, we decoupled the semantic descriptive information of nodes and the structural information encoded in the links that connect them. The ability to use these two sources of information for the reconstruction task in a meaningful and collaborative way was identified as a critical issue in the main approaches from literature.

At a high level, our contribution was twofold. First, we designed a learning environment to learn to reconstruct features from the input graph in which they are partially missing, framing the problem as an unsupervised node-level Machine Learning task. Additionally, we introduced a Deep Learning architecture - Graph Reconstruction Network - that, when used in the aforementioned context, learns to impute missing features by exploiting both semantic and structural information. This is accomplished using a multi-layer feature extraction process based on graph convolutions, which learn to recognize patterns in feature distributions on the graph structure that are helpful for reconstruction. To allow the model to process incomplete graphs as input, we also propose a special type of convolutional layer that initially restricts the computation to the available information.

We evaluate the suggested method using a number of performance metrics relevant to the MDI field, contrasting it with state-of-the-art solutions and custom alternatives on a number of real-world benchmarks. We assess the dependence on experimental parameters like the missingness rate and the portion of the graph used to train the reconstruction model as additional analytical dimensions. The obtained results support our initial analysis of the unique characteristics of the problem, and finally they demonstrate competitive performance of GRN terms of imputation quality when measured against current state-of-the-art solutions. We conclude that applying Graph Neural Networks to the problem of imputing missing node features on a graph is a promising research direction to enable higher quality estimations.

Bibliography

[1] R. Abboud, İ. İ. Ceylan, M. Grohe, and T. Lukasiewicz. The surprising power of graph neural networks with random node initialization. *CoRR*, abs/2010.01179, 2020. URL `https://arxiv.org/abs/2010.01179`.

[2] G. Batista and M.-C. Monard. A study of k-nearest neighbour as an imputation method. *Hybrid Intelligent Systems, ser Front Artificial Intelligence Applications*, 30:251–260, 01 2002.

[3] Blakely, Lanchantin, and Qi. Time and space complexity of graph convolutional networks, 2019.

[4] B. P. Chamberlain, J. Rowbottom, M. I. Gorinova, S. Webb, E. Rossi, and M. M. Bronstein. GRAND: graph neural diffusion. *CoRR*, abs/2106.10934, 2021. URL `https://arxiv.org/abs/2106.10934`.

[5] X. Chen, S. Chen, J. Yao, H. Zheng, Y. Zhang, and I. W. Tsang. Learning on attribute-missing graphs. *CoRR*, abs/2011.01623, 2020. URL `https://arxiv.org/abs/2011.01623`.

[6] H. Cui, Z. Lu, P. Li, and C. Yang. On positional and structural node features for graph neural networks on non-attributed graphs. *CoRR*, abs/2107.01495, 2021. URL `https://arxiv.org/abs/2107.01495`.

[7] V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[8] M. Fey, J. E. Lenssen, F. Weichert, and H. Müller. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. *CoRR*, abs/1711.08920, 2017. URL `http://dblp.uni-trier.de/db/journals/corr/corr1711.html#abs-1711-08920`.

[9] M. Gori, G. Monfardini, and F. Scarselli. A new model for earning in graph domains. *Proceedings of the International Joint Conference on Neural Networks*, 2:729 – 734 vol. 2, 01 2005. doi: 10.1109/IJCNN.2005.1555942.

[10] L. Grady and E. Schwartz. Anisotropic interpolation on graphs: The combinatorial dirichlet problem. 08 2003.

[11] Grover and Leskovec. node2vec: Scalable feature learning for networks, 2016.

[12] W. L. Hamilton, R. Ying, and J. Leskovec. Inductive representation learning on large graphs, 2018.

[13] B. Jiang and Z. Zhang. Robustgcns: Robust norm graph convolutional networks in the presence of node missing data and large noises. *CoRR*, abs/2003.10130, 2020. URL `https://arxiv.org/abs/2003.10130`.

[14] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, M. Kohl, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021. doi: 10.1038/s41586-021-03819-2.

[15] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016. URL `http://arxiv.org/abs/1609.02907`.

[16] Y. Ma and J. Tang. *Deep Learning on Graphs*. Cambridge University Press, 2021.

[17] Y. Ma, X. Liu, N. Shah, and J. Tang. Is homophily a necessity for graph neural networks? *CoRR*, abs/2106.06134, 2021. URL `https://arxiv.org/abs/2106.06134`.

[18] Medvedovsky, Treister, and Routtenberg. Efficient graph laplacian estimation by a proximal newton approach, 2023.

[19] S. K. Narang, A. Gadde, and A. Ortega. Signal processing techniques for interpolation in graph structured data. pages 5445–5449, 2013. doi: 10.1109/ICASSP.2013.6638704.

[20] A. S. D. Oliveira and R. J. Sassi. Behavioral malware detection using deep graph convolutional neural networks. *International Journal of Computer Applications*, 174 (29):1–8, Apr 2021. ISSN 0975-8887. doi: 10.5120/ijca2021921218. URL `http://www.ijcaonline.org/archives/volume174/number29/31858-2021921218`.

[21] R. C. Pereira, M. Santos, P. Rodrigues, and P. Henriques Abreu. Reviewing autoencoders for missing data imputation: Technical trends, applications and outcomes. *Journal of Artificial Intelligence Research*, 69:1255–1285, 12 2020. doi: 10.1613/jair.1.12312.

[22] F. Rios, P. Rizzo, F. Puddu, F. Romeo, A. Lentini, G. Asaro, F. Rescalli, C. Bolchini, and P. Cremonesi. Recommending relevant papers to conference participants: A deep learning driven content-based approach. UMAP '22 Adjunct, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392327. doi: 10.1145/3511047.3536413. URL https://doi.org/10.1145/3511047.3536413.

[23] E. Rossi, H. Kenlay, M. I. Gorinova, B. P. Chamberlain, X. Dong, and M. M. Bronstein. On the unreasonable effectiveness of feature propagation in learning on graphs with missing node features. *CoRR*, abs/2111.12128, 2021. URL https://arxiv.org/abs/2111.12128.

[24] Rozemberczki, Allen, and Sarkar. Multi-scale attributed node embedding, 2021.

[25] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009. doi: 10.1109/TNN.2008.2005605.

[26] O. Simşek and D. Jensen. Navigating networks by using homophily and degree. *Proceedings of the National Academy of Sciences of the United States of America*, 105:12758–62, 10 2008. doi: 10.1073/pnas.0800497105.

[27] D. J. Stekhoven and P. Bühlmann. MissForest—non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118, 10 2011. ISSN 1367-4803. doi: 10.1093/bioinformatics/btr597. URL https://doi.org/10.1093/bioinformatics/btr597.

[28] H. Taguchi, X. Liu, and T. Murata. Graph convolutional networks for graphs containing missing features. *CoRR*, abs/2007.04583, 2020. URL https://arxiv.org/abs/2007.04583.

[29] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. A comprehensive survey on graph neural networks. *CoRR*, abs/1901.00596, 2019. URL http://arxiv.org/abs/1901.00596.

[30] J. Yi, J. Lee, S. J. Hwang, and E. Yang. Sparsity normalization: Stabilizing the expected outputs of deep networks. *CoRR*, abs/1906.00150, 2019. URL http://arxiv.org/abs/1906.00150.

[31] J. Yoon, J. Jordon, and M. van der Schaar. GAIN: missing data imputation using generative adversarial nets. *CoRR*, abs/1806.02920, 2018. URL http://arxiv.org/abs/1806.02920.

List of Figures

List of Tables

List of Symbols

Name	Description
A	Adjacency matrix
D	Pairwise distance matrix
E	Number of edges
F	Number of features
$\mathcal{G}$	Graph
H	Latent feature matrix
L	Number of convolutional layers
M	Mask matrix
N	Number of nodes
T	Number of iterations of the sampling procedure
W	Parameter (or weight) matrix
X	Feature matrix with missing values
X'	Imputed feature matrix
$\tilde{X}$	Ground-truth (full) feature matrix
Z	Number of nodes in the subgraph
μ	Placeholder value for missing values
ϕ_1	Skew-normal distribution to generate missing rates
ϕ_2	Bernoullian distribution to fill the binary mask

Ringraziamenti

Per concludere, ci tengo a ringraziare:

Innanzitutto il mio relatore Giacomo, per tutti i progetti a cui mi ha dato la possibilità di partecipare in questi anni. La reciproca considerazione dimostrata lavorando insieme ha significato molto per me sia come studente che, soprattutto, a livello personale.

Il mio correlatore Mauro, per aver riposto una grande fiducia in me fin da subito rendendo possibile l'esperienza di ricerca a Parigi, e non di meno per la preziosa collaborazione a questo lavoro di tesi.