Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Developing interpretable graph neural networks for high dimensional feature spaces

Bachelor Thesis

Arvid Ban

arvban@student.ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Karolis Martinkus, Lukas Faber
Prof. Dr. Roger Wattenhofer

January 4, 2023

Acknowledgements

I would like to thank Karolis Martinkus and Lukas Faber for their invaluable expertise and their support throughout this thesis. Without their help, this project would not have been possible. I would also like to extend my appreciation to Prof. Dr. Roger Wattenhofer for accepting me into his group and for providing me with the computing resources necessary for the execution of the practical aspects of this Thesis.

Abstract

Graph Neural Networks (GNNs) are a powerful variations of classical neural network models that are suited for learning on graph structured data. However, as is the case for other neural networks, GNNs are not transparent and their internal logic cannot be readily interpreted by humans.

In this thesis we built on top of an existing interpretable Decision Tree GNN architecture (DT-GNN) that acts as a surrogate for the underlying GNN. This architecture however struggles with high dimensional datasets.

The aim of the thesis is twofold: First we facilitated the transfer from GNN to DT-GNN by using dimensionality reduction as well as L1-orthogonal regularization. Second, we provided an extension to the DT-GNN approach, which allows us to keep the generically trained GNN architecture in place, without requiring training with Gumbel Softmax. This extension could lead to down-the-line benefits when working with high dimensional datasets and facilitates the implementation of an interpretable alternative in existing models.

Contents

Chapter 1

Introduction

Graph Neural Networks (GNNs) are a class of powerful artificial neural network structures that, similar to other neural networks, learn on input data-sets by optimizing model parameters for downstream predictions [1, 2]. In contrast to other neural networks however, Graph Neural Networks make use of the inherent graph structures of their input data-sets. In particular, the Graph Neural Network uses the data's underlying graph to create a framework where nodes are embedded into a feature space based on the data of surrounding nodes. This framework is known as the message passing framework which aggregates the node-states of surrounding nodes along the edges of its underlying graph and then adjusts model parameters to update node embeddings. This allows GNNs to make decisions that are closely related to the graph structure of the input data and makes GNNs uniquely suited for data that has a complex interdependent structure. Unfortunately however, as it is the case for many other neural networks, GNNs are not transparent and their internal logic cannot be readily interpreted by humans.

The starting point of this thesis was the Master thesis of Peter Müller [3] in which an important step was made towards making the decision process of graph neural networks fully transparent so that the decision sequence could be followed in a human readable format. This is a very challenging feat since the decision process of Graph Neural Networks, like many other Neural Network decision processes, remain inside the black box model. Techniques have been used and implemented to create post hoc explanations of GNN models [4, 5], however, none of them can truly follow the decision process. Peter Müller together with his supervisors, Karolis Martinkus and Lukas Faber, developed a process in which a Graph neural Network was built that allowed a transfer from the GNN to a Decision Tree model called DT-GNN that, on a set of benchmark datasets, could be understood by humans and where it was possible to follow the decisions at a node level while the model retained accuracy.

The developments described in this thesis aim to generalize GNN interpretability to a wider range of problems compared to the existing DT-GNN model. One aspect where this model fell short is in maintaining the accuracy level for data sets with higher dimensionality per node embedding such as citation datasets that

often have a large amount of feature dimensions. We can see in the figure bellow how for most Citation datasets the accuracy significantly drops as we move from the GIN to the interpretable alternative, DT-GNN 1.1. An additional limitation is the fact that that GNN needs to be trained using Gumble-Softmax at the end of each GNN layer. This makes it difficult to implement and, in some instances, can limit the models ability to be expressive.

			DT+GNN		
Dataset	Features	GIN	Diff.	No pruning	Lossless pruning
CORA	1433	0.87	0.82	0.69	0.68
CiteSeer	3703	0.77	0.70	0.61	0.61
PubMed	500	0.88	0.87	0.85	0.85
OBGN-Arxiv	128	0.68*	0.68	0.28	-

Table 1.1: DT+GNN results for citation datasets with high dimensionality. *Since the dataset has 40 classes, we use a state-size of 50 for DT+GNN variants and 128 wide embeddings for GIN.

We address these limitations by exploring the following strategies:

One approach we pursued is to reduce dimensionality on the original dataset while preserving essential feature information that is discriminative during the prediction process. Additionally, this dimensionality reduction itself must be understandable in order to keep every step in the GNN to DT-GNN model fully interpretable. This approach would spare the DT model from not having to work with overly large feature spaces, which should facilitate the transfer from GNN to DT-GNN.

Additionally, we explored to use of regularizers to constrain the GNNs weights in a favorable way to promote a better GNN to DT-GNN transfer. An essential part in the GNN to DT-GNN conversion process is approximating the Multi Layer Perceptron (MLP) layers as a decision tree equivalent. Considering the importance for Neural Networks to be understandable when applied to real world tasks, it was reported that Decision Trees can be used as a surrogate model for some MLP structures as a interpretable alternative [6]. Often however the accuracies and fidelities of these surrogate models are unsatisfactory compared to that of the original Neural Network. To overcome this problem Scheff et al. suggested the use of L1-Orthogonal regularization of the MLP weight which is favorable for the generation of decision trees [7]. It allows for the axis-parallel decision boundaries to be more expressive and lets the Decision Tree to fit the data within a smaller tree.

Lastly, we explored clustering as an alternate approach to discretize GNNs for their ultimate transfer to the DT-GNN model. Our goal was to keep the original GNNs untouched allowing us to retain the generically trained GNN architecture and avoid having to train using Gumble-Softmax and simply using clustering as

a way to discretize inputs and outputs. This process also removes some of there hurdles of implementation on different GNNs and might lead to some improvements when dealing with high dimensional datasets since a cluster considers all feature components at once while a decision tree is forced to perform k steps to analyze k features.

We tested the performance of all of these developed computational approaches using benchmark datasets Cora [8] and CiteSeer [9], which consists of one-hot encodings of words in paper abstracts as binary feature values. By default, each abstract constitutes a node with 1433 (Cora) and respectively 3703 (CiteSeer) feature dimensions. In the this thesis most of the experiments are done on these data sets since they serves as excellent examples to test the limitation of the GNN to DT-GNN transfer process when applied to high dimensional datasets 1.1.

Chapter 2

Background

2.1 Architecture of Graph Neural Networks

The Graph Neural Network updates node embeddings at each of its layers. In contrast to some other Neural Networks, the Graph Neural Network builds its Neural Network Architecture in a way that resembles the underlying graph [2]. Layers are defined in a unique way by degree of removal from the node that is to be updated. Simultaneously every node is updated in an iteration step across the graph with each node having a unique computational graph based on the surrounding graph structure. This process can be visually represented as a message passing step on the input graph with nodes that are to be updated $\mathbf{h}_i^l$ being offered with the node embeddings of its surrounding neighbors $\mathcal{N}(h_i^l)$ as seen in figure 2.1. The messages are passed to the central node, where they are aggregated in a preferable way. The central node state, becomes updated by an activation function, which as an input takes the aggregated node states of its surroundings as well as its own node state. In the next iterative step this updated embedding $\mathbf{h}_i^{l+1}$ can be used by its own neighborhood $\mathcal{N}(h_i^{l+1})$ to further update the node embeddings in the graph. Eventually each graph node becomes embedded in a learned way and can perform graph and node level predictions on the input graph.

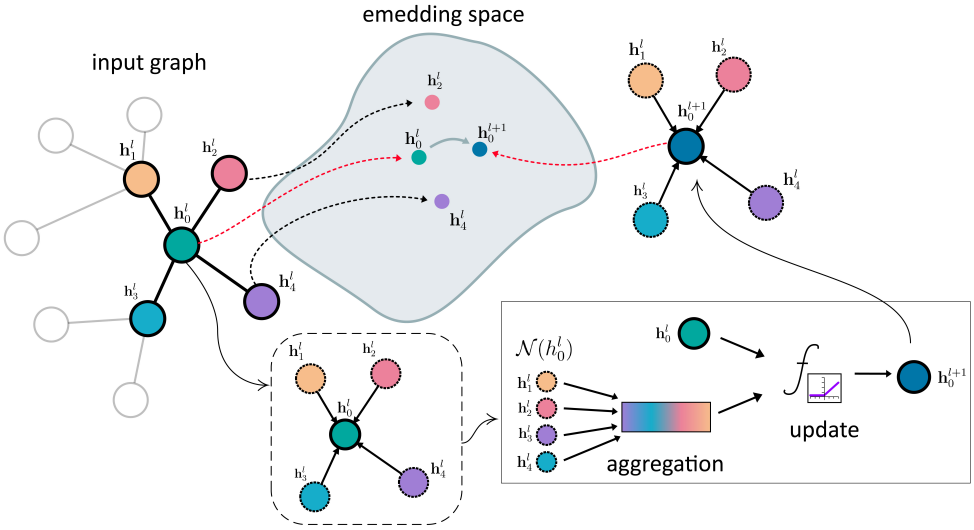

Figure 2.1: Updating a singular node during one iteration of the embedding framework in Graph Neural Networks

$$\mathbf{h}_i^{l+1} = \text{update}(\mathbf{h}_i^l, \text{aggregation}(\mathcal{N}(h_i^l))) \tag{2.1}$$

2.2 The path from GNN to a Decision Tree-GNN

To translate the GNN to a fully interpretable model as proposed by P. Müller et al. [3], the GNN needs to be designed in a unique way. This is achieved by implementing a Graph Neural Network structure to accurately perform predictions on evaluation data sets, and then use the outputs and inputs of the GNNs layers to fit decision trees (DT) that emulate the GNNs internal decisions. To make this process interpretable, the Graph Neural Networks layer outputs need to be categorical, so that explicit node state decisions can be made. This is achieved by designing the GNN with Gumble-Softmax function at the end of each layer to force categorical states between layers that can then be used by the DT model to fit the categorical inputs and outputs. This allows humans to follow the decision process throughout the DT.

The GNN to DT-GNN model performed comparably to the Graph Isomorphism Network(GIN) [10] on most of GINs commonly used benchmarks. Using pruning, the decision trees become humanly readable and offer straight forward interpretation and insight into many GNN decisions [3]. This was a great step forward in advancing interpretability while keeping performance accuracy.

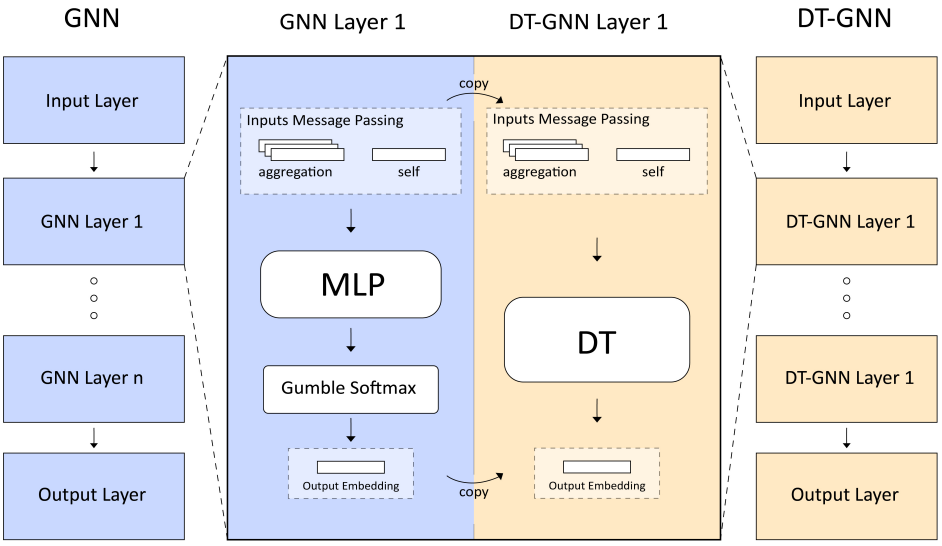

Figure 2.2: transfer from the discrete version of GNN to the DT-GNN for interpretability

Chapter 3

Dimensionality reduction and regularization approaches to improve GNN to DT-GNN transfer

3.1 Principle Component Analysis dimensionality reduction

Principal Component Analysis (PCA) is a widely used linear dimensionality reduction technique that allows identification of the most expressive features in the dataset. It can mathematically be understood as linear combination of the feature components in the original data, that retains maximum variance [11]. In that sense PCA can be seen as linear superposition which is considered interpretable. Naturally, this lends itself well to reducing the number of components in datasets with a lot of features, such as Cora, while retaining the overall interpretability of the GNN to DT-GNN model.

We implemented the PCA dimensionality reduction and tested the effects of this process on the GNN to DT-GNN conversion when applied to Cora as well as CiteSeer datasets. The dimensionality reduction was applied as the a dataset gets loaded initially. Different extents of reductions were evaluated across a 10-fold validation and compared with the control where no dimensionality reduction was applied. The results for Cora as well as CiteSeer are listed in the tables 3.2 and 3.2. For readability purposes the data is reduced to the most essential parameters used for out evaluations.

For CiteSeer the results show there are visible improvements of the relative performance in the discrete GNN to DT-GNN transfer, contrary to Cora where we have no clear improvements in the performance of the DT-GNN. Both GNNs seem to improve in accuracy with dimensionality reduction. This can be attributed to fact that the PCA contributes to a more regularized space for the weight vectors. Eventually the dimensionality reduction weakens the signal in the datasets too much and both the GNN accuracy and the DT accuracy are reduced.

Table 3.1: 10 fold cross-validation of accuracies for the Cora dataset across different dimensions following PCA dimensionality reduction

	Test Acc		
Dimensions after PCA	**GNN**	**DT-GNN**	**DT-GNN pruned**
2	0.631	0.506	0.489
5	0.734	0.632	0.624
16	0.811	0.680	0.645
48	0.840	0.661	0.630
128	0.850	0.673	0.644
no dim reduction	0.814	0.694	0.675

Table 3.2: 10 fold cross-validation of accuracies for the CiteSeer dataset across different dimensions following PCA dimensionality reduction

	Test Acc		
Dimensions after PCA	**GNN**	**DT-GNN**	**DT-GNN pruned**
2	0.507	0.507	0.494
5	0.685	0.650	0.632
16	0.698	0.642	0.633
48	0.724	0.652	0.637
128	0.718	0.621	0.614
no dim reduction	0.700	0.610	0.610

3.2 Orthogonal weight regularization in MLP layers

L1-orthogonal regularization has two main objectives; inducing sparsity in the weight vectors of MLP layers, as well as aligning weight vectors orthogonally in every MLP layer. Schaaf et al. [7] showed that L1-orthogonal regularization helps prime MLP layers leading to a better DT conversion. This is the case because it allows the DTs Axis-parallel Decision Boundaries to be more expressive, and allows fitting of the data within a smaller DT structure.

L1 norm is defined as the sum of absolute values of vectors components. Combinations of vector components that reduce this norm have fewer components. Due to that often the norm is often used as a regularizer of model weights given by $\|w\|_1 = \sum_{j=1}^{p} |w_j|$ in order induce sparsity in the weight vectors.

In order to orthogonally align the weight vectors it is important to realize that if two weight vectors are ortho-normal, their scalar product is zeros and their individual L2 norms are close to one. To promote orthogonal aligning, we compute the Gram matrix expressed as $G_{i,j} = w_i^T w_j$. If the weight vectors are close to being ortho-normal the gram matrix is close to the identity matrix. With

these considerations, we created a regularizing term as 3.1.

$$\mathcal{L}_{\text{ortho-L1}} = \sum_{i=1}^{l} \|G_l - I\|_1 \tag{3.1}$$

We extracted the weights from the MLP layers in the discrete GNN and calculated the regularizing factor according to 3.1 multiplied with a constant lambda. Based on the accuracy results of calculations carried out when some MLP layers were kept in place and transferring others to the DT- GNN, we hypothesized that the bottleneck in the transfer between discrete GNN and the DT-GNN might be the input layer. Considering the given hypothesis the implementation was done in two different variants. In one case every layer was regularized, whereas in the other only the input layer was regularized. For both cases lambda was varied according to logarithmic increments from 0.05 to 0.000005 and ran across the two different implementations. The evaluation was done on both the datasets Cora 3.3 and CiteSeer 3.4. The results of the regularization of the input layer are shown side by side with the results from the implementation that covers all layers in table.

For both datasets, the results of the all-layer implementation shows that depending on how strong the regularizer is applied, there is a tradeoff between how much the regularizer promotes a better transfer from GNN to DT-GNN and how much the regularizer act as a limiting factor in the GNNs ability to train. This is evident from the observation, that the performance of the DT-GNN approaches the performance of discrete GNN at higher lambda values. The optimal DT-GNN accuracy for Cora is 0.766 at lambda=0.0005 and the optimal DT-GNN accuracy for CiteSeer is 0.639 at lambda=0.005. Interestingly the accuracy of the discrete GNN seems to improve as well with the right choice of regularizer. This is most likely due to the fact that the regularization stabilizes the behavior of MLP layers in the model similar to the way PCA stabilized the optimization of GNN in the previous section.

In line with our hypothesis, the results for the input-layer implementation show a very similar distribution of datapoints when the regularizer was applied across all layers. We also observe that the same tradeoff between the efficiency of model transfer to the DT layer and the GNNs learning ability. As for the all-layer implementation, the highest DT-GNN accuracy of 0.769 for Cora is also observed at lambda 0.0005, whereas the highest DT-GNN accuracy for CiteSeer is 0.649 respectively observed at lambda 0.005.

Table 3.3: 10 fold cross-validation test accuracies on Cora across different lambda values for the L1-orthogonal regularizer applied only to the input layer and for the L1-orthogonal regularizer applied to all layers

	Input Layer Reg		**All Layer Reg**	
Lambda	**GNN**	**DT-GNN**	**GNN**	**DT-GNN**
0	0.812	0.683	0.812	0.683
0.000005	0.793	0.676	0.806	0.699
0.00005	0.825	0.704	0.837	0.700
0.0005	0.858	0.766	0.861	0.769
0.005	0.788	0.709	0.782	0.737
0.05	0.724	0.666	0.696	0.644

Table 3.4: 10 fold cross-validation test accuracies on CiteSeer across different lambda values for the L1-orthogonal regularizer applied only to the input layer and for the L1-orthogonal regularizer applied to all layers

	Input Layer Reg		**All Layer Reg**	
Lambda	**GNN**	**DT-GNN**	**GNN**	**DT-GNN**
0	0.700	0.610	0.700	0.610
0.000005	0.704	0.619	0.697	0.613
0.00005	0.703	0.617	0.700	0.610
0.0005	0.703	0.629	0.705	0.612
0.005	0.731	0.649	0.716	0.639
0.05	0.385	0.376	0.332	0.328

Chapter 4

Developing a GNN to DT-GNN Architecture that works with natural GIN node embeddings

4.1 Extension of DT-GNN architecture with Clustering process in GIN layers

To discretize the GNN model, while preserving the standard GIN architecture, we applied a clustering approach. In every GIN layer, all the data points were embedded in a unique way in the embedding space. To find discrete embeddings, we clustered the datapoints according to different clustering algorithms. When the data points are needed in a discrete way, based on their cluster association the data points become one-hot encoded as visible in the figure bellow 4.1. This process should serve as an alternative to creating a discretized GNN version of our model to be interpreted, while keeping the GNN to DT-GNN transfer accuracy high. This clustering process provides a lot of freedom to the embeddings while eventually still generalizing to the Decision Tree. This approach could also help with more complex high-dimensional data sets such as Cora since a cluster is able to considers all feature components at once whereas a decision tree is forced to perform k steps to analyze k features.

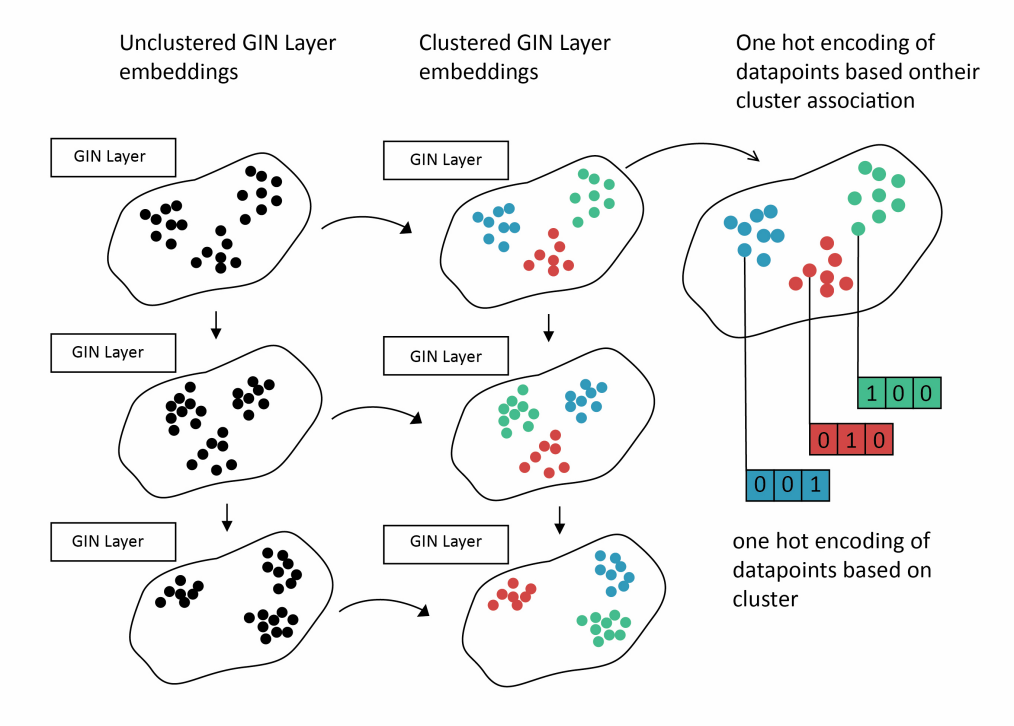

Figure 4.1: Clustering process between GNN layers

The new architecture with GIN node embeddings requires a model transfer from GNN to DT-GNN as shown schematically for one layer in figure 4.2. In this regard, the new architecture is the same as in the previous architecture that did not include clustering, however, in the new implementation, the GNNs discrectizing architecture using Gumbel-Softmax is removed and the nodes are free to embed without restrictions 4.2). In a second step after the model has been trained, the clustering process is performed on every data point in every layer (step 2 in the figure 4.2). As a result, this gives a unique cluster association to each datapoint in a particular layer. In the transfer step from GNN to DT-GNN, instead of directly copying the node embeddings from the inputs and outputs of the GNN layers, they are one-hot encoded based on their cluster association (step 3 in the figure 4.2). These one-hot encoded vectors make up the categorical states that are passed to the DT layer as inputs and outputs (step 4 in the figure 4.2).

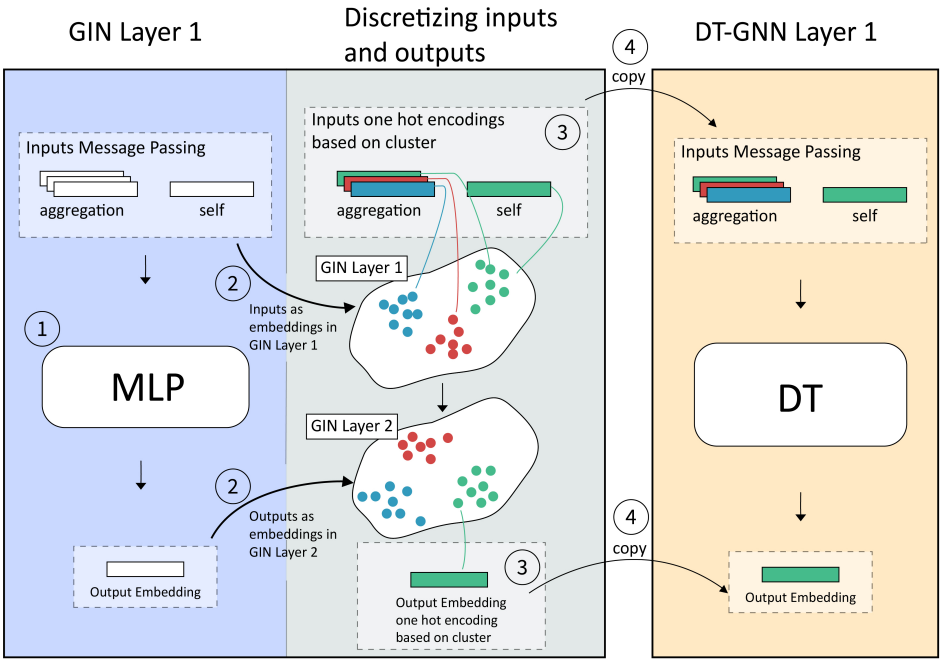

Figure 4.2: Novel architecture uses clustering to discretize GNN for DT-GNN transfer

Next, we compared the performances of three different clustering algorithms 4.1. Spectral Clustering and k-means clustering outperformed DBSCAN. The k-means and spectral clustering algorithms featured very similar accuracies that are comparable to the discrete GNN to DT-GNN architecture. To analyze the performance of the clustering algorithms we used the t-distributed stochastic neighbor embedding (t-SNE) method [12] on all data points in every layer to reduce the dimensionality. This allowed plotting and coloring the points based on their cluster association both for the k-means clustering 4.3, Spectral clustering 4.4 as well as DBSCAN clustering 4.5. The analysis shows that, in general, the points are distributed cohesively. Prior to clustering, in the input layer, the distribution of points appears to be more isometric compared to the later layers where the distribution of points seems elongated. The Spectral algorithm visually appears to be the most accurate at identifying clusters in the embedding space as can be seen in the t-SNE plots in Figures 4.4with k-means being close second in Figure 4.3. However, this should be taken with caution since the 2-dimensional representation can be misleading.

Table 4.1: 10 fold cross-validation across different clustering alternatives

	Test Acc		
Clustering Type	**GNN**	**DT-GNN**	**DT-GNN pruned**
Spectral clustering	0.792	0.666	0.659
k-means clustering	0.794	0.640	0.641
DBSCAN clustering*	0.786	0.573	0.560
original GNN to DT-GNN	0.814	0.694	0.675

*Since the input layer did not cluster well with DBSCAN. it was clustered by k-Means instead of DBSCAN. All other layers are clustered with DBSCAN

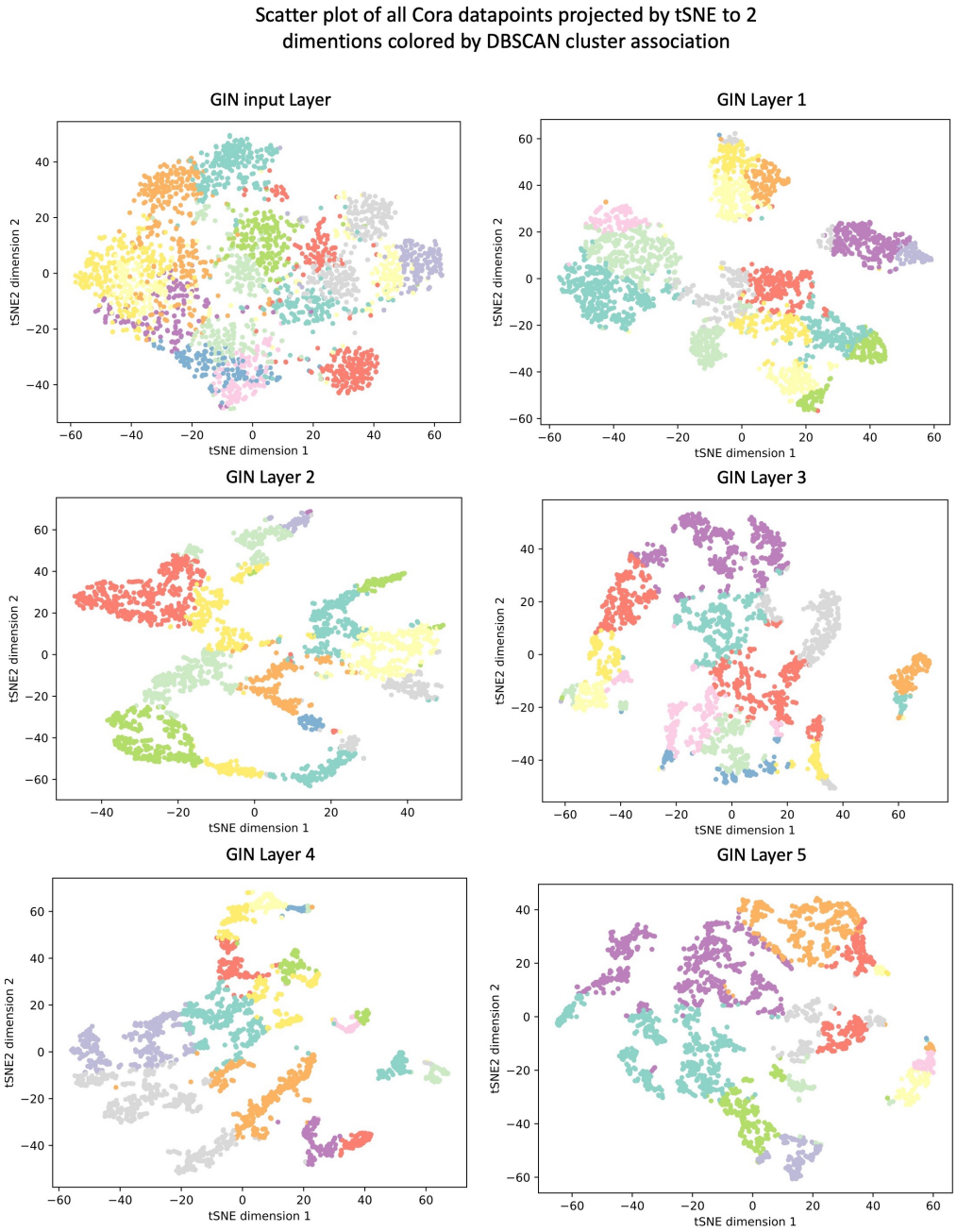

Figure 4.3: Scatter plot of all Cora datapoints projected by t-SNE to 2 dimensions colored according to k-means-cluster association

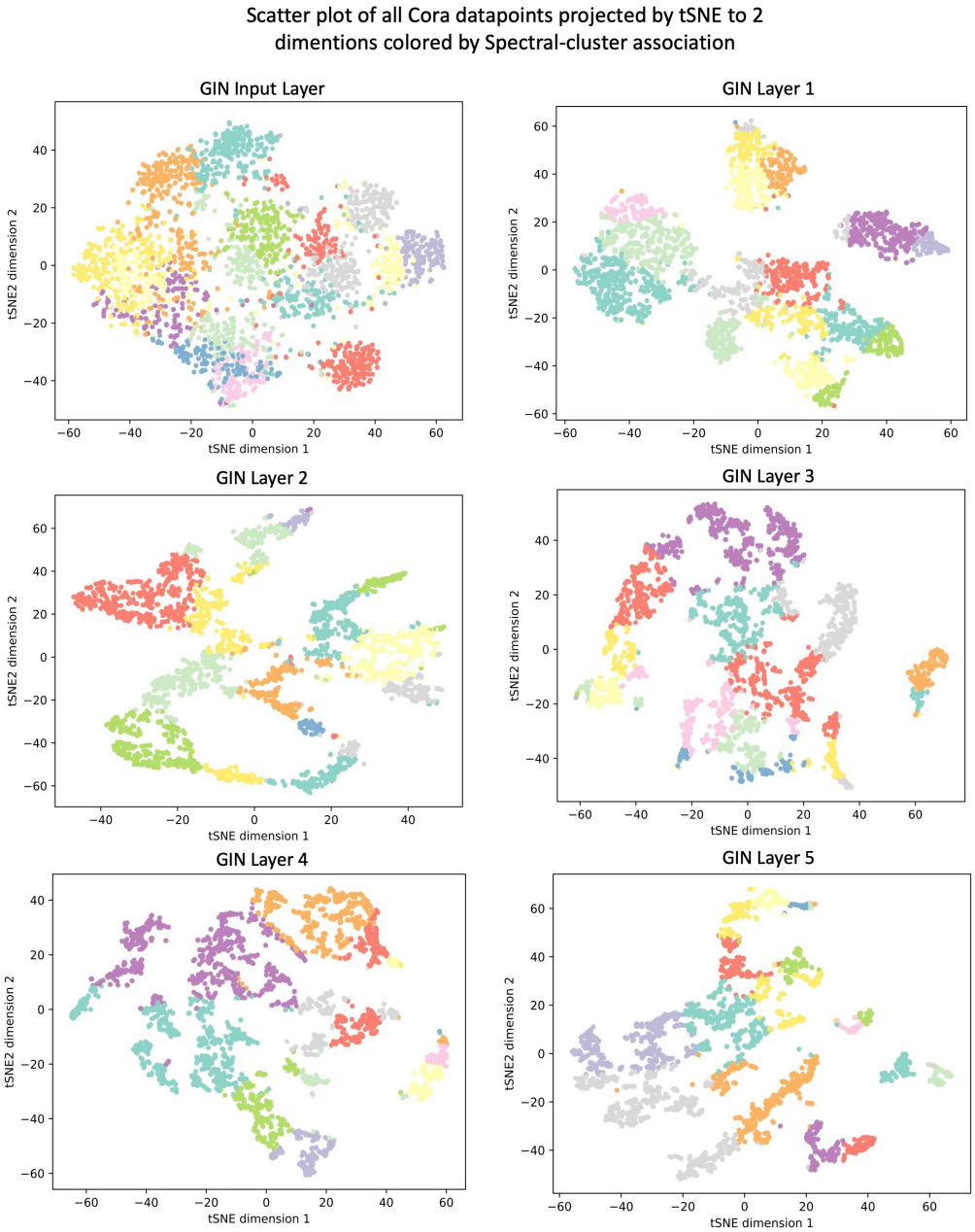

Figure 4.4: Scatter plot of all Cora datapoints projected by t-SNE to 2 dimensions colored according to Spectral-cluster association

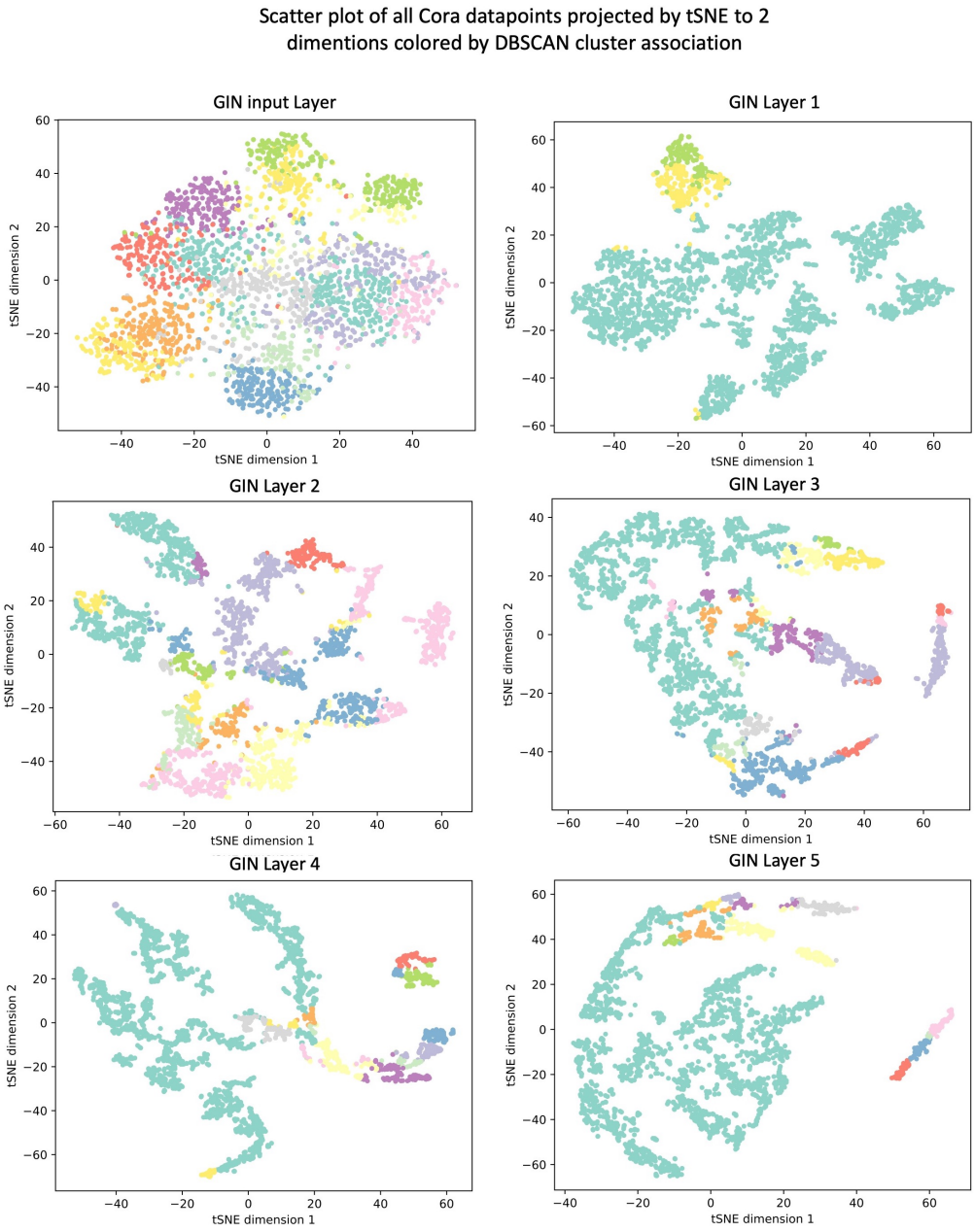

Figure 4.5: Scatter plot of all Cora datapoints projected by t-SNE to 2 dimensions colored according to DBSCAN-cluster association

Chapter 5

Discussion

5.1 Discussion and Conclusions

In this thesis I explored different strategies to improve the conversion of a GNN architecture to make it interpretable when applied to a broader range of machine learning tasks. To this account I have introduced modifications to the procedures to make the discrete GNN to DT-GNN model transfer better for datasets that are high in feature dimensions.

The strategy to reduce the dimensionality of the data using principal component analysis improved the performance of the DT-GNN such that it approached the performance of the GNN when applied to the CiteSeer dataset, however no apparent improvements could be observed when the Cora dataset was processed. This is most likely because the tradeoff between the extent of used features and the success of GNN to DT-GNN transfer is too steep for the Cora dataset. As a consequence, too much signal is lost when the features are reduced to a small enough number that it could have a positive impact on the GNN to DT-GNN transfer. It appears that the PCA dimensionality reduction is a process that should always be tested, as it has the potential to yield good results depending on the given dataset.

Orthogonal-L1 regularization proved to be a great way to improve the accuracy of DT-GNN towards that of discrete GNN. The results indicated substantial improvements for both datasets that the procedure was applied to. Orthogonal-L1 regularization appears to be the more consistent approach in dealing with high dimensional datasets compared to PCA. We have discovered that the MLP structure in the input layer is difficult to emulate with the DT and, consequently, the regularizer significantly helps the transfer in the input layer for both datasets since it allows for better axis-parallel decisions.

In the last approach, we implemented a novel GNN to DT-GNN conversion architecture as an alternative to the existing implementation with the aim to make it more generally applicable to different types of problems and data complexities. Although the results did not outperform the existing GNN model for

the Cora dataset, the procedure remains to be tested on other datasets with high dimensionality such as ogbn-arxiv [13], CiteSeer or PubMed. The current implementation provides an excellent starting point for future exploration of strategies aimed at improving the DT-GNN so that it could provide similarly accurate predictions on high dimensional data sets as GIN does.

5.2 Future research

In the presented novel clustering GNN to DT-GNN architecture, the original neural network is not trained to work with categorical embeddings. Because of this, error propagation likely occurs when changing to the categorical cluster states in the DT. In a possible continuation of the work presented this thesis, one could add an intermediate training step where the GNN is trained in a categorical way following the initial training phase using Slot Attention [14], which would bring the data points to categorical states in a probabilistic way. This would allow for the model to maintain freedom in its embedding structure while adding a step in which it is forced to learn to fully rely on categorical inputs. This could reduce the error propagation between the models resulting in a more accurate transfer from GNN to DT-GNN.

Bibliography

[1] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666651021000012

[2] W. L. Hamilton, "Graph representation learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 14, no. 3, pp. 1–159.

[3] P. Müller, L. Faber, K. Martinkus, and R. Wattenhofer, "Dt+gnn: A fully explainable graph neural network using decision trees," 2022. [Online]. Available: https://arxiv.org/abs/2205.13234

[4] R. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "Gnnexplainer: Generating explanations for graph neural networks," 2019. [Online]. Available: https://arxiv.org/abs/1903.03894

[5] D. Luo, W. Cheng, D. Xu, W. Yu, B. Zong, H. Chen, and X. Zhang, "Parameterized explainer for graph neural network," 2020. [Online]. Available: https://arxiv.org/abs/2011.04573

[6] T. Ueno and Q. Zhao, "Interpretation of deep neural networks based on decision trees," in *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)*, 2018, pp. 256–261.

[7] N. Schaaf, M. F. Huber, and J. Maucher, "Enhancing decision tree based interpretation of deep neural networks through l1-orthogonal regularization," 2019. [Online]. Available: https://arxiv.org/abs/1904.05394

[8] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI Magazine*, vol. 29, no. 3, p. 93, Sep. 2008. [Online]. Available: https://ojs.aaai.org/index.php/aimagazine/article/view/2157

[9] R. Rossi and N. Ahmed, "The network data repository with interactive graph analytics and visualization," in *AAAI Conference on Artificial Intelligence*, vol. 29, New York, NY, USA, 2015, pp. 4292–4293.

[10] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" 2018. [Online]. Available: https://arxiv.org/abs/1810.00826

[11] I. Jolliffe and J. Cadima, "Principal component analysis: a review and recent developments," *Philos Trans A Math Phys Eng Sci*, vol. 374, no. 2065, p. 20150202, 2016.

[12] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008. [Online]. Available: http://jmlr.org/papers/v9/vandermaaten08a.html

[13] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: Datasets for machine learning on graphs," *CoRR*, vol. abs/2005.00687, 2020. [Online]. Available: https://arxiv.org/abs/2005.00687

[14] F. Locatello, D. Weissenborn, T. Unterthiner, A. Mahendran, G. Heigold, J. Uszkoreit, A. Dosovitskiy, and T. Kipf, "Object-centric learning with slot attention," *CoRR*, vol. abs/2006.15055, 2020. [Online]. Available: https://arxiv.org/abs/2006.15055