Chair of Network Architectures and Services
School of Computation, Information, and Technology
Technical University of Munich

TECHNICAL UNIVERSITY OF MUNICH

SCHOOL OF COMPUTATION, INFORMATION, AND TECHNOLOGY

INFORMATICS

BACHELOR'S THESIS IN ELECTRICAL ENGINEERING AND INFORMATION TECHNOLOGY

Autonomous System Models using BGP Data and GNNs

Iñigo González-Varas Vallejo

Technical University of Munich

School of Computation, Information, and Technology

Informatics

Bachelor's Thesis in Electrical Engineering and Information Technology

Autonomous System Models using BGP Data and GNNs

Modellierung Autonomer Systeme mit BGP Daten und GNNs

Author:	Iñigo González-Varas Vallejo
Supervisor:	Prof. Dr.-Ing. Georg Carle
Advisor:	Max Helm
	Johannes Zirngibl
	Patrick Sattler
	Benedikt Jaeger
Date:	August 15, 2023

I confirm that this Bachelor's Thesis is my own work and I have documented all sources and material used.

Garching, August 15, 2023	
Location, Date	Signature

Abstract

In the realm of global networking, the Border Gateway Protocol (BGP) serves as the backbone of Internet routing, enabling communication between Autonomous Systems (ASes). ASes possess diverse attributes, often requiring inference or estimation for undisclosed properties. Simultaneously, Graph Neural Networks (GNNs) have emerged as a tailored machine learning approach for analyzing graph-structured data, spanning domains like social networks and molecular structures.

This study bridges network communication and machine learning, applying GNNs to BGP graphs to uncover behavioral insights. By leveraging GNNs, we decode intricate AS properties and reveal latent dynamics within BGP graphs. Our work contributes to understanding BGP networks through a novel analytical perspective.

Concluding our study, we emphasize the significant strides made in geographic clustering, where our models demonstrate impressive proficiency. A remarkable accuracy of 93.8% in continent classification was reached, demonstrating the effectiveness of our approach in accurately categorizing regions. Additionally, in the domain of business type classification, we surpassed existing benchmarks, achieving an accuracy rate of 81.7%. These outcomes validate our approach and highlight its practicality in handling diverse and complex classification tasks, contributing to a deeper understanding of BGP networks.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The Border Gateway Protocol (BGP) serves as the established norm for inter-domain routing within the Internet. Functioning as a path-vector protocol, BGP facilitates interactions among Autonomous Systems (ASes), holding responsibility for the global Internet's reachability. An Autonomous System denotes a network or a collection of networks overseen by one or more administrative entities, each presenting a standard and clearly defined routing policy towards the Internet.

ASes have several inherent properties that can be modeled as node attributes, some of which may not be publicly available. These properties include size, geographic coverage, connectivity, and performance metrics, providing a deeper understanding of ASes and their role in the network ecosystem. Inferring their unavailable properties provides valuable insights for network understanding, performance optimization, security, policy making, and market analysis.

Inherent within ASes lie various attributes, some of which might remain undisclosed. Among them are aspects such as size, geographic expanse, connectivity, and performance metrics, which collectively yield a comprehensive perspective on ASes and their network roles.

Graph Neural Networks (GNNs) have been shown to be effective in learning node representations in graph-structured data. In this work, we use GNNs to infer properties of ASes, including their geographical location (continent and country level), Regional Internet Registry (RIR), business type, and type of relationships with other ASes.

The knowledge gained from categorizing these attributes holds significant value across a range of fields. When we deeply understand networks, it helps us make them better by improving their structure and how they work, leading to improved overall performance. Additionally, this understanding could serve as a vital foundation for enhancing network security, inferring properties of malicious ASes, and identifying potential threats.

Beyond operational considerations, the insights we gather also have a crucial role in shaping policies. Being able to identify subtle characteristics of Autonomous Systems empowers effective allocation of resources and strategic networking approaches.

Furthermore, the attributes we extract offer opportunities for comprehensive market analysis. By breaking down the features of Autonomous Systems, we can uncover trends, patterns, and potential future directions in the market.

In summary, exploring these attributes goes beyond just understanding them. It guides networks towards optimization, security, policy refinement, and a more profound grasp of the market landscape.

Chapter 2

Background

This chapter provides background information on the Border Gateway Protocol (BGP) and Graph Neural Networks (GNNs). This information is necessary to understand the context of this thesis and the problem it aims to solve.

2.1 The Border Gateway Protocol

Border Gateway Protocol (BGP) [1] is the core routing protocol used on the Internet to exchange routing information between nodes called Autonomous Systems (ASes). An Autonomous System [2] refers to a network or a set of networks managed by one or more administrative entities; or domains that present a standard and clearly defined routing policy to the Internet.

BGP operates by facilitating the exchange of routing information between ASes. It selects the best path for routing traffic using a path-vector algorithm that considers factors like AS path length, local preference, and other attributes. ASes advertise the IP prefixes they own, and this information is propagated through neighboring ASes until it reaches all the ASes in the network. [1]

BGP is fundamental to the Internet routing infrastructure, facilitating interdomain routing between ASes. It works in conjunction with interior gateway protocols (IGPs) like OSPF or IS-IS, which handle intra-domain routing within an AS. BGP enables ASes to have fine-grained control over their routing policies, optimizing decisions based on cost, performance, and security requirements. It also establishes and manages peering relationships, essential for Internet connectivity, allowing ASes to exchange traffic directly and enhance performance. [3]

BGP networks are inherently graph-structured due to the interconnection of Autonomous Systems (ASes) and their routing policies. Therefore, BGP graphs can be created, with nodes representing ASes, and links representing the interconnections between them. This BGP graph would capture the relationships, dependencies, and connectivity patterns among ASes, enabling the analysis and understanding of the ASes properties and their relationships.

ASes have several inherent properties that can be modeled as node attributes, some of which may not be publicly available. These properties include size, geographic coverage, connectivity, and performance metrics, providing a deeper understanding of ASes and their role in the network ecosystem. Inferring their unavailable properties provides valuable insights for network understanding, performance optimization, security, policy making, and market analysis.

ASes are assigned unique identifiers known as Autonomous System Numbers (ASNs) by RIRs, which are regional organizations responsible for allocating and registering Internet number resources. There are five main RIRs: ARIN, RIPE NCC, APNIC, LACNIC, and AfriNIC. The region where an AS primarily operates determines to which RIR it belongs. [4]

Business type is another property of the ASes, it provides insights into the nature of the entity managing the network and its operational industry or sector. These include Internet and Network Service Providers (ISPs), telecommunications companies, content delivery networks (CDNs), research institutions, educational institutions, government entities, private corporations, and more.

Geographical data is also included in ASes. They have assigned a country and continent headquarters that indicate the geographical base from which the AS is managed and controlled. ASes can have headquarters assigned in several countries, but most ASes are purely local [5].

ASes establish relationships with other ASes, which can be categorized into different link types. Common link relationships include customer-provider, peer-to-peer (peering), and sibling relationships. Customer-provider relationships involve one AS providing network services (transit) to another AS. Peer-to-peer relationships denote a mutual traffic exchange between two ASes without monetary agreements. Sibling relationships typically occur between ASes under the same administrative entity. Link relationships provide insights into the connectivity and business partnerships among ASes. [6]

2.2 Graph Neural Networks

Graph Neural Networks (GNNs) are a type of machine learning model that analyze and extract information from graph-structured data. Graphs consist of nodes and links representing relationships or connections between nodes. [7]

GNNs leverage the inherent connectivity patterns within graphs to learn and propagate information between nodes and links, capturing local and global dependencies. By utilizing innate connectivity patterns, GNNs can acquire knowledge from the neighboring nodes within the graph and obtain information derived from them. GNNs can also propagate information across the entire graph, considering the global structure and dependencies present in the network. Through iterative message-passing mechanisms, GNNs exchange and aggregate information across nodes, allowing them to capture broader patterns and dependencies that span the entire graph. [7]

GNNs have gained significant attention due to their ability to model complex relationships and patterns in various domains. They excel at tasks like node classification, link prediction, graph classification, and graph generation. [7]

GNNs can classify nodes based on their features and connectivity patterns within a graph. For example, in a social network, GNNs can predict the interests or attributes of individuals based on their relationships with other individuals. GNNs can also predict the existence, likelihood, or type of connections (links) between two nodes in a graph.

The fundamental idea behind GNNs is to learn node representations by iteratively aggregating and updating information from neighboring nodes. Message passing is a core concept in GNNs, which involves the exchange of information (messages) between connected nodes in a graph. Each node collects information from its neighbors and updates its own representation based on these messages. The messages typically carry information about the neighboring nodes' features, which helps to enrich the target node's representation. [7]

The aggregation step is where the information from neighboring nodes is combined to form a meaningful message for each node in the graph. There are different aggregation methods used in GNNs, including:

- Sum: the features of neighbor nodes are summed up to form the message.
- Mean: the mean of the features of the neighboring nodes is computed and used as the message.

- Max: the maximum value of the features of the neighboring nodes is taken as the message.
- Attention-based: an attention mechanism is used to weight the importance of each neighbor's information before aggregation.

After the aggregation step, each node uses the aggregated message to update its own representation. The update function is typically a neural network layer that takes the aggregated message and the node's current representation as input and produces an updated node representation. This updated representation is then used in the next iteration of message passing, if multiple iterations are employed. [8]

Some popular GNN architectures are GraphSAGE, Graph Convolutional Networks (GCN), Graph Attention Networks (GAT), and Graph Isomorphism Networks (GIN).

GCN utilizes a simple aggregation strategy, computing the mean of the neighboring nodes' feature vectors. It uses localized convolution operations to ensure smoothness of node embeddings across the graph. [9]

GraphSAGE is characterized by its neighborhood sampling strategy, where for each node, a fixed-size neighborhood is sampled, and information from these neighbors is aggregated using an aggregator. This neighborhood sampling strategy makes GraphSAGE scalable to large graphs. [10]

GAT introduces an attention mechanism to GNNs, which allows the model to learn the importance of each neighbor's information, assigning each of them a weight parameter that is learned during training, and used during the aggregation process. [11]

GIN is a graph neural network architecture with the original purpose of learning graph isomorphism. Graph isomorphism is a graph property that determines whether two graphs are structurally identical. GIN can also be used for other tasks such as node classification and graph classification. [12]

Chapter 3

Related Work

In this chapter, related work in the field of AS property estimation and classification is presented. The common points and differences with the work carried out in this thesis are discussed.

3.1 BGP2Vec

BGP2Vec [5] is a project that aims to infer AS properties using a neural-network-based embedding process based on the state-of-the-art techniques used in the field of Natural Language Processing. It uses a dataset of BGP routing tables from the RouteViews project [13] and IRR [14]. The classification tasks performed in this project are ASN business-type classification and AS Type of Relationships (ToRs) inference.

ASN business-type classification: The authors grouped the ASes into three categories: Transit Access (Cable/DLP/ISP, NSP), Enterprise(Enterprise, Education/Research), and Content. The achieved accuracy for this classification task was 79.2%.

AS ToR inference: The authors used two different datasets for this task. Both of them included labels for "Peer to Peer," "Provider to Customer," and "Customer to Provider" relationships. The second dataset additionally included the "Sibling to Sibling" relationship. The accuracy for the first dataset was 95.8%, and for the second dataset, 95.%.

In this thesis, business type and link relationship classification tasks are carried out. The results obtained by the BGP2Vec project can be used to compare the performance of the models trained in this thesis to other approaches that have been used in the past to solve the same tasks.

3.2 Adaboost algorithm for AS business type classification

Similarly to the BGP2Vec project, this work [15] also aims to classify ASes into business types. The classification is performed using the Adaboost algorithm [16], a machine learning algorithm that combines multiple weak classifiers to create a robust classifier. The authors divided the dataset into six classes: Large ISP, Small ISP, Customer ASes, Universities, Internet Exchange Points (IXPs), and Network Information Centers (NICS). The achieved accuracy for this classification task was 78.1%.

3.3 Benchmarking GNNs for Internet Routing Data

In this work [17], Graph Neural Network models and other machine learning-based classifiers are benchmarked and compared for AS property and AS link relationship inference. The tested GNNs are Graph Convolutional Networks (GCNs) [9], Graph Attention Networks (GATs) [11], and GraphSAGE [10]. Node2Vec [18] is also tested, a node embedding algorithm based on random walks. BGP2Vec [5] is also included in the comparison. Regarding the link classification task, some promising results are obtained with the GNNs, with precision and accuracy over 90%. The AS properties that are classified include Traffic Ratio (if the traffic through the AS is balanced, outbounds, or inbounds), Scope (geographical parameter), Network Type (similar to business type), and Peering Policy (open, selective, or other). However, the results achieved by the project for node classification could be more promising, with accuracies below 55% for all tasks.

3.4 Summary

The results obtained by the other projects can be used as reference figures for the results obtained in this thesis. Some of the classification tasks carried out in this thesis have already been performed by other projects (business type and link relationship classification) with different approaches and results. However, other classification tasks, like the Country, RIR, and Continent classification tasks, are novel. In the following table, we can see a summary of the tasks carried out in this thesis and the related work presented.

Table 3.1: Comparison of Approaches

	BGP2Vec	Adaboost	Benchmarking	This Thesis
Business Classification	✓	✓	✓	✓
Link Relationship Classification	✓	✗	✓	✓
Continent Classification	✗	✗	✗	✓
Country Classification	✗	✗	✗	✓
RIR Classification	✗	✗	✗	✓
GNN Approach	✗	✗	✓	✓

Chapter 4

Methodology

This chapter describes the approach taken to solve the questions identified in Chapter 1.

The first part of the project will consist of building a dataset that can be used to train Graph Neural Network models. For that purpose, data sources with BGP information must be gathered and processed. There are several sources available that contain BGP information and can be used to build BGP graphs, such as RouteViews [13] and RIPE RIS [19] route collectors, the AS to organization mapping from CAIDA [20], or the information contained in PeeringDB [21]. In our project, we will use the work done by Henschke [22] to build an initial dataset. His work can gather information from the sources mentioned above and build a graph with the Autonomous Systems and the relationships between them. The project not only gathers information and shapes it into a graph, it also enhances the Autonomous Systems by adding calculated topological features. The generated graph can be used as a starting point to build the dataset for the project. First, the graph will have to be enhanced by adding relevant properties for our purposes, such as the continent where the AS is headquartered or the business type of the AS. Moreover, the data must be modified and processed for machine-learning purposes. Some of these processes include the normalization of the numerical features or the one-hot encoding of the features we will be classifying. Additionally, the graph's structure may have to be modified to fit the requirements of the GNN models. Some GNN frameworks require that the features are encoded in a specific way, so that will be a problem we will have to solve.

Once the dataset is ready, the next step is to train the GNN models. For that purpose, we will use the work done in the Benchmarking project [17], which provides a good starting point as some modules, GNN architectures, training processes, and other functions have already been implemented. The first step will be to adapt the dataset graph to the project. Once the original project runs, new features will be implemented to improve performance and better understand the training process of the GNNs. These new features include adding a validation set that will be used to compute the validation loss in every iteration (epoch) of the training process. This will help understand when the model has stopped learning and prevent overfitting. New metrics (precision and recall) and visualization tools (training loss curves and confusion matrixes) will be implemented into the project to better understand the training process and the performance of the models. As a final step, the models will be trained, and the results will be analyzed. The initial training process will consist of testing the performance of GraphSAGE, GCN, GIN, and GAT for all the classification tasks. The values for the parameters like learning rate, learning rate decay, or patience will either be the ones from the original project or randomly picked from a broad range of values. Once the first training round is done, hyperparameter optimization will be implemented into the project to find the best parameters for each model and classification task. The hyperparameter optimization will be done using the Neural Network Intelligence framework [23], a hyperparameter optimization framework for machine learning. The training process results will be analyzed to infer conclusions from them.

Chapter 5

Dataset Creation

This chapter describes the process of creating the datasets used in this thesis, the different data sources used, and the graph and data processing steps performed.

5.1 Data sources

As a starting point for the creation of the datasets, the work carried out by Henschke [22] is used. The project consists of the creation of a pipeline for the construction of BGP graphs. The pipeline combines the BGP information from different sources, including RouteViews [13] and RIPE RIS [19] route collectors, the AS to organization mapping from CAIDA [20], and additional information from PeeringDB [21]. The graph is enriched with additional node and link attributes such as calculated topological features or inferred link relationships using the ASRank [24] and Toposcope [25] algorithms. The files that were used to create the graph were downloaded on April 9th, 2023, as it was the latest version available at the time of the creation of the datasets. The resulting graph contains 74878 ASes and 226529 bidirectional links, with 23 features for each AS and 6 features for each link. Two of the features that were not included in the original graph are the AS business type and the AS continent headquarters. As we will be classifying these properties, some modifications were made to the code to include them in the graph. Business type information is obtained from PeeringDB [21], and the continent information is obtained by mapping the country information to the continent using the pycountry [26] library. With these modifications, each AS has 25 features.

5.2 Graph processing

Once we have a BGP graph to work with, the next step is to process it to obtain a dataset that is valid for the classification tasks. The features contained in the nodes and links of the graph can be classified into two different categories: categorical and numerical.

5.2.1 Numerical features

As it is usually done in machine learning, the numerical features need to be normalized. For that purpose, two normalization functions were implemented: MinMax and Z-Normalization. MinMax is a commonly used normalization function that keeps the original distribution of the features. Its downside is that it is affected by outliers as it scales using the minimum and maximum values of the feature. Z-Normalization is not affected by outliers, as it uses the mean and standard deviation of the feature. With the implementation of both functions, the user can choose which one to use, depending on the presence of outliers in our data.

The MinMax normalization function scales the values of the feature to the range $[0, 1]$, using the following formula:

$$new\ value = \frac{value - min}{max - min} \tag{5.1}$$

The Z-Normalization function scales the values to a normal distribution. It calculates the mean and standard deviation of the feature and uses the following formula:

$$new\ value = \frac{value - mean}{std} \tag{5.2}$$

During the creation process of the dataset, either of the two functions can be used. In our case, the choice was made after analyzing the distribution of the features. All numerical features in the dataset were analyzed in terms of mean, median, standard deviation, minimum, and maximum values. We ensured that the numerical features were not affected by outliers, so the MinMax function was used.

5.2.2 Categorical features

The categorical features need to be encoded to be used in the classification tasks. For this purpose, the One-Hot-Encoding technique is used. One-Hot-Encoding is a representation of categorical data where each category is converted into a vector with all elements as 0 except for the corresponding category index set to 1. As an example, if we have a link between two ASes labeled as "Peer to Peer", which is represented as "p2p", the relationship feature of the link will be encoded as:

$$\textit{Labels vector} = [p2c, p2p, c2p] \tag{5.3}$$

$$\textit{Encoding vector} = [0, 1, 0] \tag{5.4}$$

5.3 Distribution of the categorical features

Once all the data in the graph has been processed, the distribution of the categorical features can be analyzed. This process will provide insights into how the different features are distributed and if the dataset is balanced. We can decide whether to perform additional processing steps to balance the dataset by obtaining this information. The imbalance of the features may affect the performance of the machine learning models, so it is essential to analyze this aspect.

5.3.1 Link relationships

The distribution of the different types of link relationships is shown in Figure 5.1. 55% of the links belong to relationships between Providers and Customers, while 44.9 % of the links are Peer to Peer relationships. The remaining 190 links are not classified into any relationship. Even if our project supports relationships between Sibling ASes, none are found in the graph.

5.3.2 Business types

The distribution of nodes for each business type is shown in Figure 5.2. From this plot, we can observe that 77.65% of the nodes are not labeled. The most common business type is Cable/DSP/ISP, with 12.25% of the nodes, followed by NSP, Content, and Enterprise, representing 4.2%, 2.3%, and 1.6% of the nodes, respectively. The remaining business types are present in less than 1% of the nodes.

5.3.3 Continent and country headquarters

In terms of the distribution of nodes per continent, Figure 5.3 shows that 33.7% of the nodes are located in Europe, followed by North America with 27.8%. The most underrepresented continents are Oceania and Africa, with 2.93% and 2.43% of the nodes, respectively. Only 0.5% of the dataset does not contain known continent headquarters information. There is a total of 235 different countries in the graph. This is the most unbalanced of all of the distributions of the categorical features, as countries like the United States, Brazil, or Russia are represented by more than 5000 nodes, while

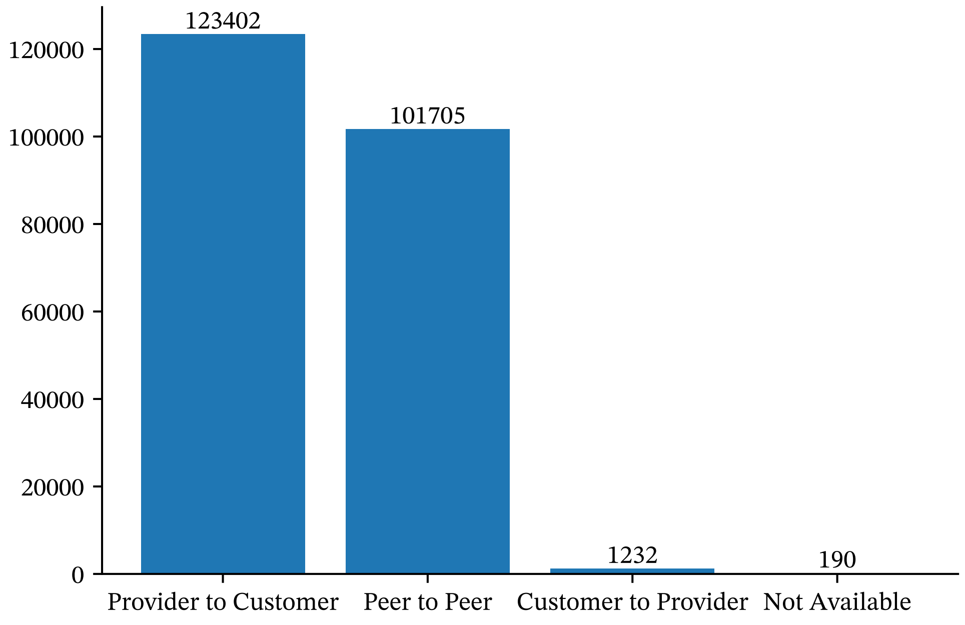


Figure 5.1: Link Relationship Distribution

79 countries are represented by less than 10 nodes. In Figure 5.4, the top ten most represented countries are shown.

5.3.4 Regional Internet Registry

Regarding the RIR, the distribution in Figure 5.5 has similarities to the continent distribution as RIPE and ARIN, the RIRs for Europe and North America, are the most represented ones, with 38.2% and 26.05% of the nodes, respectively. They are followed by APNIC, the RIR for Asia Pacific, with 17.37% of the nodes, LACNIC with 14.84%, and AFRINIC with 2.34%. Lastly, we can observe there is a label for a NIR (National Internet Registry) in our dataset. In the work by Henschke [22], some ASes are labeled as belonging to the NIRs of Korea and Japan, instead of labeled as APNIC. In our graph, no ASes are labeled as the NIR of Korea, but 0.5% of the nodes are labeled as belonging to JPNIC, the NIR of Japan. Therefore, the classification we will be performing will not strictly be for RIRs, but for the labels that are present in the dataset (RIRs and JPNIC). Finally, only 0.5% of the dataset does not contain known RIR information.

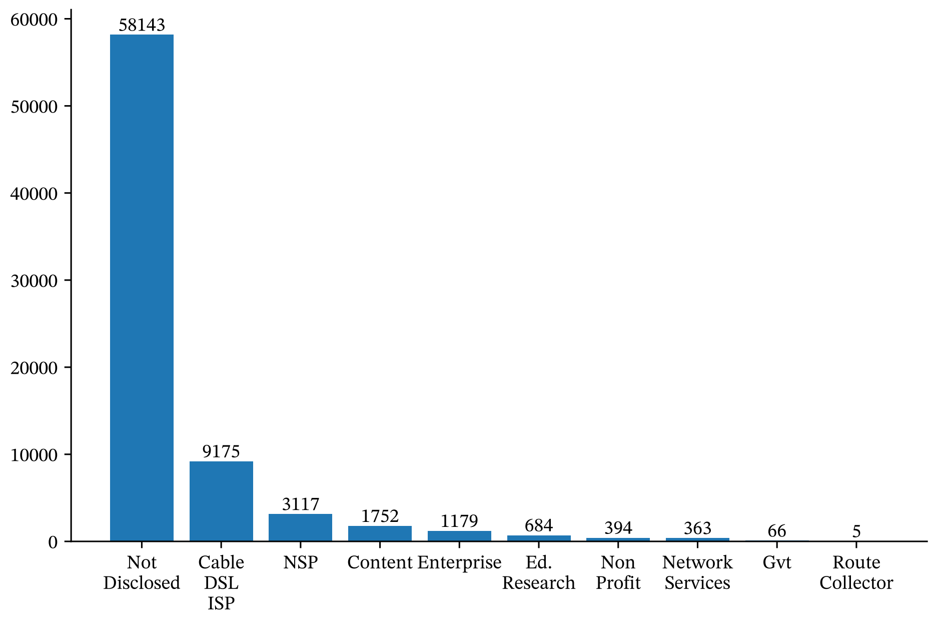


Figure 5.2: Business Type Distribution

5.4 Graph structure variations

In the above sections, we get all necessary information from the BGP graph, process it, and obtain the final information that will go into the dataset. At this point, we faced a challenge: the graph structure. When developing the final dataset, we need to have a clear idea of which kind of graph structure would be compatible with the machine learning models. Some machine learning frameworks can only perform node classification, and others can also perform link classification. As node classification is needed for all the AS properties classification tasks, we decided to create a dataset compatible with node classification. For that purpose, we defined two types of nodes in the graph: the AS and the link nodes. The AS nodes contain all the information about the ASes, and the link nodes contain all the information about the links. The AS nodes are connected to the link nodes through the links that they are part of. Figure 5.6a represents an example of the "Default" graph structure, containing only two AS nodes and one link node.

However, an additional challenge with this graph structure was found. Some of the link relationships are directional, this means, in the case of a link between AS1 and AS2, the relationship of the link may be "Provider to Customer", but the relationship of the link between AS2 and AS1 may be "Customer to Provider". As links are undirected in our

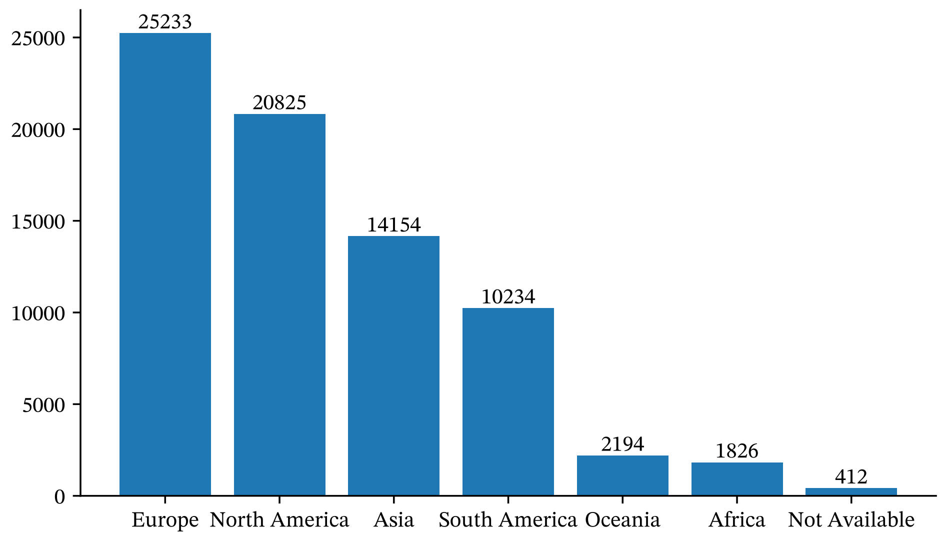


FIGURE 5.3: Continent Distribution

graph, it is not known which AS represents which role of the relationship. Therefore, a new type of node was added to the graph, the "Role" node, representing the role of each AS in the relationship. In Figure 5.6b an example of the "Role" graph structure is depicted, containing two AS nodes, one link node, and two role nodes. If the relationship between AS 0 and AS 1 was "Provider to Customer", the role of AS 0 would be "Provider", and the role of AS 1 would be "Customer". That information would be stored in the role nodes 3 and 4, respectively. This way, the directional information of the link is preserved.

Lastly, the "Classic" graph structure was created. Once the machine learning framework was chosen, it was crucial to be sure the graph structure would be compatible with it. The "Classic" graph structure is the one used by the work done in Benchmarking [17], which will be the starting point for the machine learning part of the project. In this graph structure, only one type of node is included, the AS node. Figure 5.6c represents an example of the "Classic" graph structure, containing two AS nodes and the link between them.

All of the graph structures were adapted to be compatible with the machine learning framework chosen for the project.

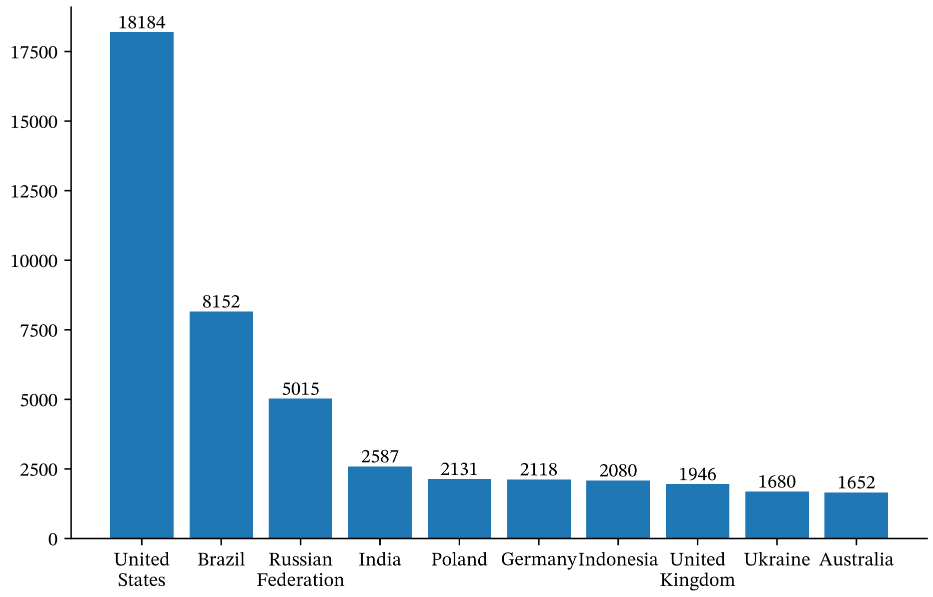


Figure 5.4: Country Distribution: Top 10

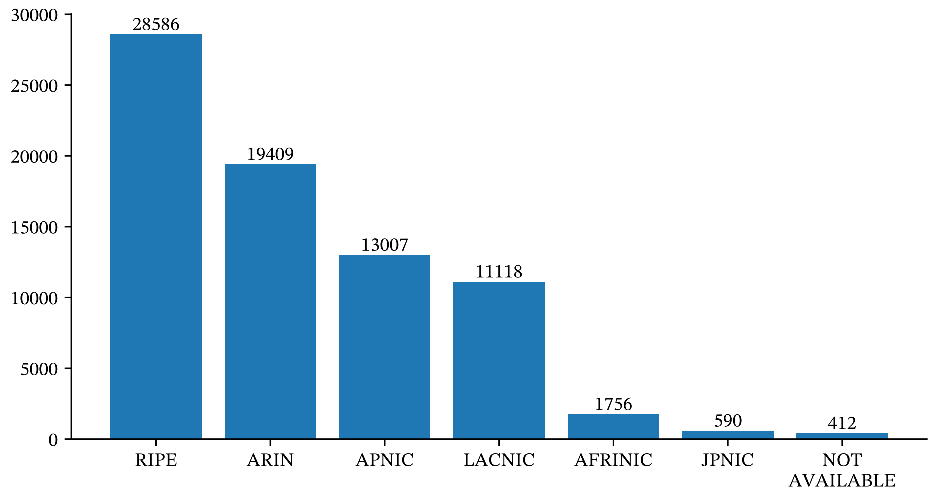
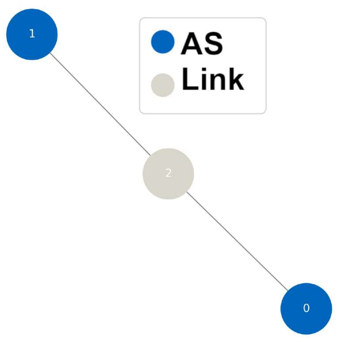
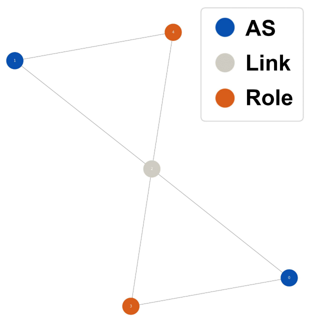


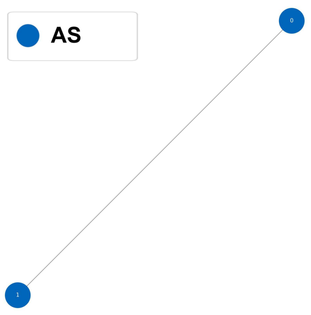
Figure 5.5: RIR Distribution



(a) Default Graph Structure



(b) Role Graph Structure



(c) Classic Graph Structure

Figure 5.6: Graph Structures

Chapter 6

Graph Neural Network Training

In this chapter, the methodology used for the training of the Neural Network models is discussed.

6.1 Choosing a GNN framework

The first step in training the GNNs was to choose a framework to use. We already had a BGP graph that contained all the necessary information, so we had to find a compatible framework that could use it or one that could be easily adapted to use it. As we have seen in Chapter 3, the Benchmarking project [17] performed similar node classification tasks to the ones we wanted to perform, so we decided to use it as a starting point. Using it had both advantages and disadvantages. On the one hand, we would save time because we would not have to implement some modules from scratch. However, it also meant that we would have to deeply understand the implementation and adapt the code to our needs.

6.2 Adaptation of the Graph

The first compatibility problem we faced was that the Benchmarking project used a different graph library than ours. We had been using NetworkX [27] for the creation and manipulation of the graphs, but the Benchmarking project used Deep Graph Library (DGL) [28]. Consequently, we had to convert our graphs to the DGL format. Moreover, the graph structure was different. We encoded the properties of the nodes as different node attributes in each node. However, the nodes in the graphs of the benchmarking project only had two attributes: the node label and the node features. The node label

Table 6.1: Link Classification Results

Node Attributes (NetworkX)	Features vector (DGL)
business type: 5	5
rir: 3	3
transit degree: 0.005	0.005
pfxs originating: 0.0008	0.0008
...	...

is a vector in which the one-hot encoded representation of the node label is stored. The node features is a vector in which the node attributes are stored. Each position represents a different attribute. Therefore, in the adaptation process, we had to transform our graph from NetworkX to DGL and change the node attribute structure to fit the one used in the Benchmarking project. As we performed several different classification tasks, we had to build a different graph for each task, as the node labels differed in each case.

In Table 6.1 a comparison between a node in the NetworkX graph and the feature vector of the same node in the DGL graph can be seen. If the graph is used for a classification task in which the node label is the continent, the node label would be a vector of size 6, where the position of the continent would be 1 and the rest would be 0. This is shown in Equation 6.1.

$$Continent\ Label\ Vector\ (DGL) = \Big[0, 0, 0, 0, 1, 0\Big] \tag{6.1}$$

6.3 Training the GNNs

Once all of the graphs for the different classification tasks have been built and we have checked that the project was working correctly, we analyzed how the training process is being done. The way the training had been implemented consisted of three phases:

1. Splitting the dataset into training and test sets
2. Training the model for a given number of iterations (epochs) using the training set. The epoch number is chosen by the user.
3. Computing performance metrics (accuracy and f1-score) using the trained model on the test set

This approach had some issues, such as the inability of evaluating the model's performance during the training process, the techniques used to deal with unbalanced datasets, the metrics used to evaluate the models or the lack of features like the visualization of metrics or the hyperparameter tuning. We will go over them and explain in depth how we solved them in the following sections.

6.3.1 Validation test

The test set was created by selecting the first 10% of the nodes in the graph and removing them from the graph. The 90% of the nodes that remained were used as the training set. This approach was not optimal, as the test was done once a given number of training iterations (epochs) had been completed, and we wanted to evaluate the model at any point during the training process. In machine learning approaches, it is common to use the validation loss to determine when to stop the training process, as it is a good indicator of when the model has stopped learning and has started overfitting (memorizing the training set). Moreover, the test set was created by removing the first 10% of the nodes in the graph, which could lead to a biased test set as the nodes were not randomly selected.

Therefore, we implemented a split into training, validation, and test sets. The percentage of nodes in each set could be configured, but in the beginning, we decided to use 60% for training, 20% for validation, and 20% for testing. These proportions were chosen because it is a common split in machine learning approaches. The nodes were randomly selected for each set to avoid any bias. The validation set was used to compute the validation loss, which could be used to determine when to stop the training process.

6.3.2 Unbalanced datasets

Another common issue with machine learning approaches is having an unbalanced dataset. If there are more samples of one class than another, the model will predict the most common class more often, leading to a biased model. In the Benchmarking project, a technique to avoid this problem was used. It consisted of using an undersampler that would make all classes have the number of samples of the least populated class. The undersampler is a good approach, but we implemented another additional technique to mitigate the effects of having unbalanced datasets in the training process.

The training loss quantifies the difference between the model's predictions and target values for a given training data set. The training loss will be biased towards the most common class in the unbalanced dataset scenarios. To address this issue, we can introduce weights to the loss function, which modify the impact of each training example on the overall loss.

We implemented the weights calculation by using the inverse of the frequency of each class in the training set. Moreover, we introduced an exponent parameter to the weight vector calculation to control the impact of the weights on the loss function. The formula used to calculate the weights is shown in Equation 6.2.

$$class\ weight = \left(1 - \frac{samples\ in\ class}{total\ number\ of\ samples}\right)^{exponent} \tag{6.2}$$

6.3.3 Metrics

In the original project, only two metrics had been implemented to evaluate the performance of the models: accuracy and f1-score. The accuracy can be defined as the number of correct predictions divided by the total number of predictions.

$$accuracy = \frac{number\ of\ correct\ predictios}{total\ number\ of\ predictions} \tag{6.3}$$

Initially, we used it as the primary metric to evaluate the models, but we soon realized that it could be misleading due to the unbalanced datasets. For example, if we have a dataset with 90% of the samples belonging to class A and 10% belonging to class B, a model that always predicts class A will have an accuracy of 90%, but it could be a better model. Something similar happened in our training process, and we had to find a better metric to evaluate the models.

The f1-score is a metric that considers the model's precision and recall. The precision is calculated as the number of true positives divided by the number of true positives plus the number of false positives. In our case, we first calculated the precision of each class and then computed the average of all classes to obtain the final precision of the model. In the example we explained above, the precision of the model would be 50%, since the precision for class A would be 100% (no false positives) and the precision for class B would be 0% (no true positives).

$$precision = \frac{true\ positives}{(true\ positives\ +\ false\ positives)} \tag{6.4}$$

The recall metric is calculated as the number of true positives divided by the number of true positives plus the number of false negatives. As we did for the precision, the recall is also first calculated for every class and then averaged to obtain the final recall of the model. Recall measures the fraction of positive instances the model correctly identified out of all the positive instances in the data.

$$recall = \frac{true\ positives}{(true\ positives\ +\ false\ negatives)} \tag{6.5}$$

F1-score is the weighted harmonic mean of precision and recall, reaching its optimal value at 1 and its worst value at 0.

$$f1\ \ score = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{6.6}$$

In our project, we modified the scikit-learn library [29] that was being used to calculate the f1-score to obtain the metrics for precision and recall. This way, we could have a deeper understanding of the performance of the models.

6.3.4 Visualizing the training process

Another improvement we made to the project was plotting some graphs to visualize the training process. We plotted the curves for training loss and validation loss to visually represent the learning process and to detect when the model was not improving anymore, overfitting or underfitting. One example of this plot can be seen in Figure 6.1.

Additionally, we plotted the confusion matrix for the predicted values on the test set to better understand the model's final performance. The confusion matrix is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known. It provides a comprehensive view of the model's predictions across all classes and their alignment with the true labels. Figure 6.2 shows one example of this plot. In this case, a RIR classification task was carried out. This example matrix depicts some misclassifications between the RIRs, you can see 58% of the nodes in JPNIC were misclassified as APNIC nodes. However, for the other labels the model is having a solid performance, with at least 83% of the nodes being correctly classified.

6.3.5 Hyperparameter tuning

Once all of these features were implemented, a first round of training was done. The parameters for the training had been chosen arbitrarily, and we wanted to find the best parameters for each model. For this purpose, the Neural Network Intelligence (NNI) library [23] was used. NNI is a toolkit to help users run automated machine learning (AutoML) experiments. It provides the ability to search for optimal hyperparameters and neural architectures. It also supports tuning and pruning deep neural networks

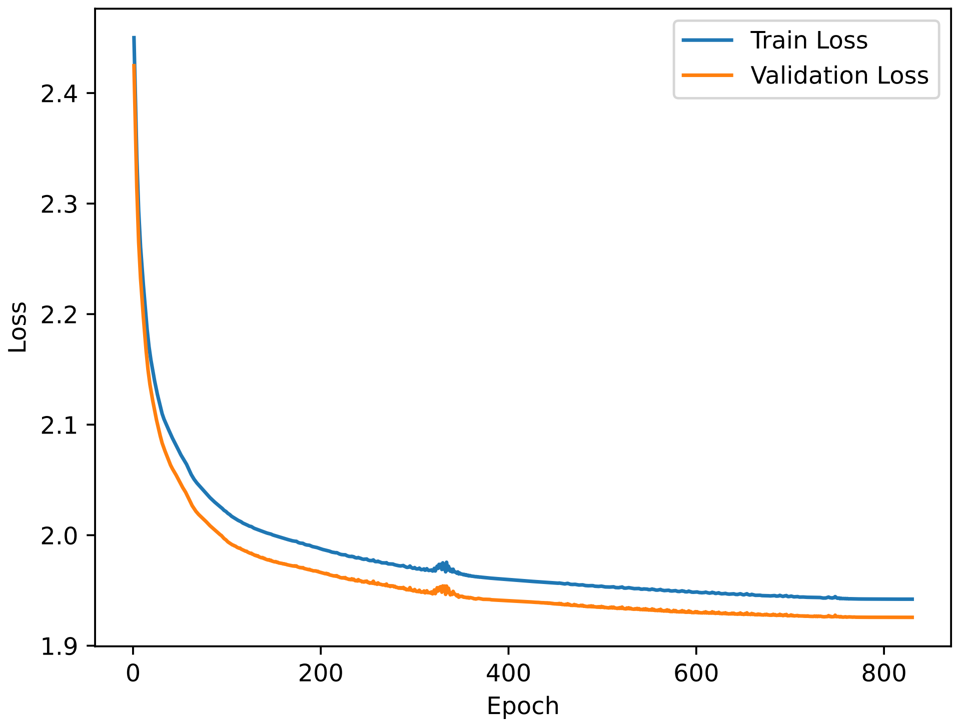


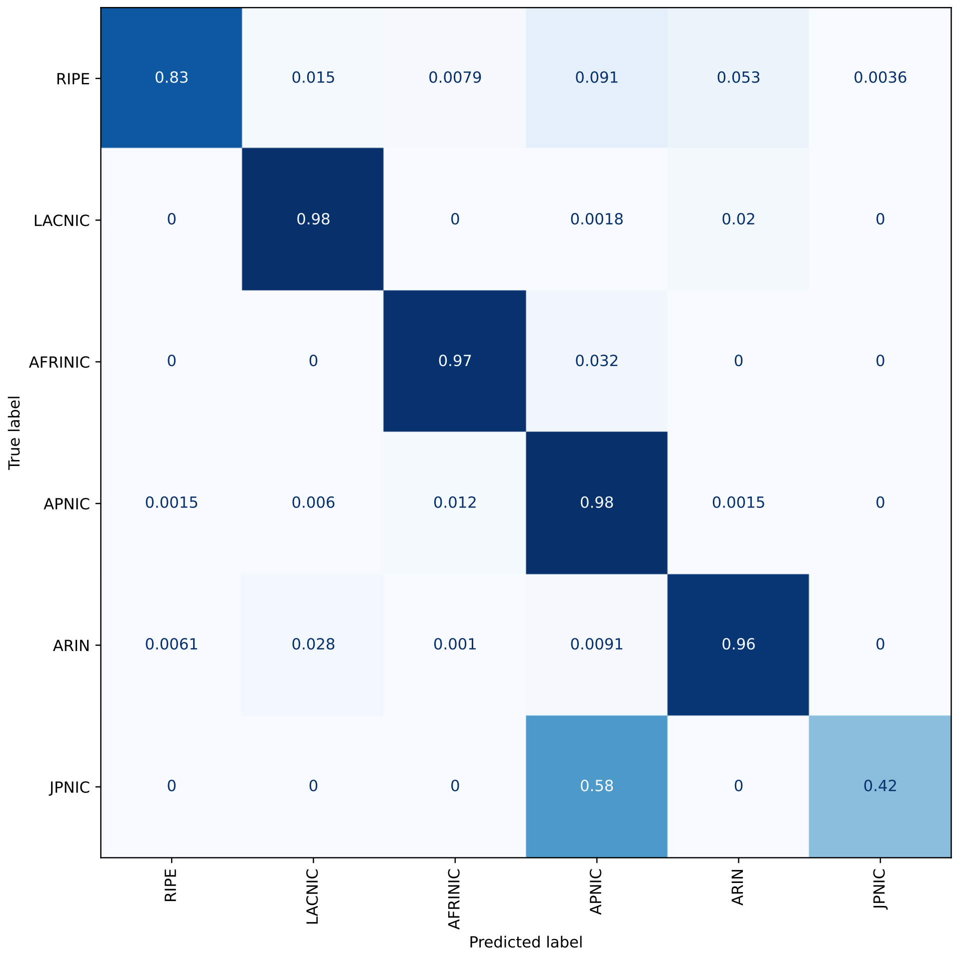
Figure 6.1: Example loss curves for a RIR classification task. The blue curve represents the training loss and the orange curve represents the validation loss. The x-axis represents the number of epochs and the y-axis represents the loss value.

developed by multiple deep learning platforms, including PyTorch. In our case, we used it to find the best hyperparameters for the models. The parameters, type of choice, and search space used for the hyperparameter tuning are the following:

- Learning rate: uniform, (0.0001, 0.01)
- Learning rate decay: choice, (0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9)
- Patience: choice, (5, 10, 15, 20, 25, 30, 35, 40, 45, 50)
- Graph split percentage: choice, (0.05, 0.1, 0.15, 0.2, 0.25, 0.3)
- Usage of the undersampler: choice, (True, False)
- Class weights exponent: uniform, (1, 8)

The choice for the search spaces was based on the first round of experiments we had done before using NNI. For example, the learning rates that had performed best in that round had always been in the range of 0.0001 to 0.01, so we decided to use that range for the search space. The same reasoning was applied to the rest of the parameters. Additionally, some parameters could have been chosen using a uniform distribution instead of choosing from a list. However, as we serialized the combinations of datasets to improve the speed of the training process, we decided to use a list of values to reduce the number of combinations.

Regarding NNI, it employs a reference metric to evaluate the models' performance and choose the best hyperparameters. In our case, we chose that parameter to be the precision. The reason for this is that we wanted to deal with dataset imbalance. Optimizing for accuracy or recall could lead to a bias towards the majority classes, causing the model to perform poorly on minority classes. Therefore, optimizing precision can help ensure the model makes confident and accurate predictions for the classes with fewer instances.



True label \ Predicted label	RIPE	LACNIC	AFRINIC	APNIC	ARIN	JPNIC
RIPE	0.83	0.015	0.0079	0.091	0.053	0.0036
LACNIC	0	0.98	0	0.0018	0.02	0
AFRINIC	0	0	0.97	0.032	0	0
APNIC	0.0015	0.006	0.012	0.98	0.0015	0
ARIN	0.0061	0.028	0.001	0.0091	0.96	0
JPNIC	0	0	0	0.58	0	0.42

Figure 6.2: Example confusion matrix depicting the performance of the GraphSAGE model in a RIR classification task. The rows represent the true labels and the columns represent the predicted labels. The values in the matrix represent the proportion of nodes that were classified as the predicted label. Results are above 83% of the nodes being correctly classified for all labels except JPNIC.

Chapter 7

Results and Discussion

In this chapter, the obtained results are presented and discussed. The results are grouped by classification task, and the performance for each GNN model and dataset variation is compared. Confusion Matrixes that show the distribution of the predicted labels are presented for the best-performing models in each classification task.

7.1 Continent Classification

The continent classification task was the most successful, with the best-performing model achieving an accuracy of 93.8% and a precision of 89.3%. Table 7.1 presents the results for the continent classification task.

In terms of comparison between GNN models, the best performance was achieved by GraphSAGE. It reached a maximum accuracy of 93.8% and a precision of 89.3%. Its performance was also consistent across the three dataset variations, with the lowest accuracy being 84.6% and the lowest precision of 84.4%. GCN and GAT were also able to achieve good results in terms of precision, with the lowest precision being 70.7% for the GAT model with the roles dataset. The results for accuracy could have been more promising with the roles dataset with this model, reaching only up to 34.8% with GAT. GIN could not achieve good results for this task, with the best accuracy being 18.9% and the best precision being 34.8%.

The dataset variation significantly impacts the models' performance, with the classic dataset variation achieving the best results for all models. The impact on the accuracy is less severe than on F1-score, precision, and recall. This could mean it is easier to reduce the bias towards the majority classes in the classic dataset, therefore generalizing better

Table 7.1: Continent Classification Results

GNN Model	Dataset	Accuracy	F1-score	Precision	Recall
GraphSAGE	Classic	0.938	0.898	0.893	0.907
GraphSAGE	Default	0.877	0.679	0.844	0.661
GraphSAGE	Roles	0.846	0.793	0.885	0.766
GCN	Classic	0.908	0.870	0.887	0.867
GCN	Default	0.847	0.687	0.729	0.674
GCN	Roles	0.302	0.402	0.766	0.396
GAT	Classic	0.867	0.851	0.829	0.879
GAT	Default	0.864	0.729	0.750	0.718
GAT	Roles	0.348	0.358	0.707	0.423
GIN	Classic	0.189	0.055	0.348	0.168
GIN	Default	0.181	0.058	0.419	0.169
GIN	Roles	0.345	0.085	0.057	0.167

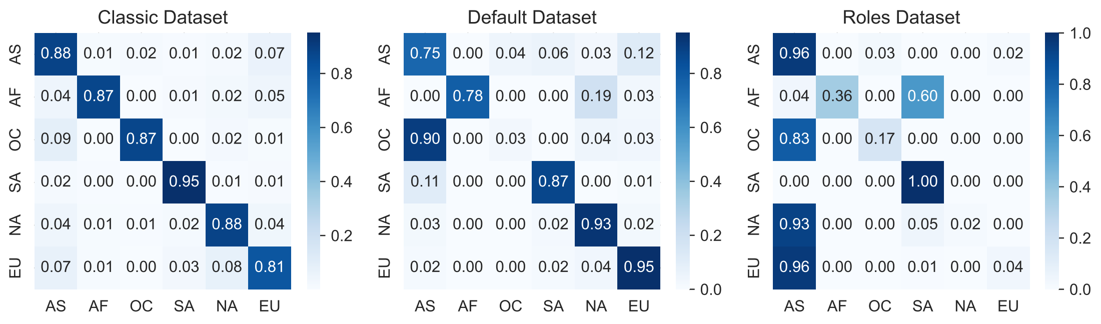


Figure 7.1: Comparison between the normalized confusion matrixes for the continent classification task for the GAT model. Accuracy for the classic dataset is over 81% for all classes. For the default dataset, the classes Oceania, Africa and Asia have misclassification rates up to 90%, 19% and 12% respectively. The roles dataset misclassifies most of the nodes as Asia nodes.

towards the more underrepresented classes. We look deeper at this in the confusion matrixes for the GAT model depicted in Figure 7.1, as it is the model that captured this phenomenon the most. In the case of the GAT model, all classes have at least an accuracy of 81% for the classic dataset. In contrast, in the default dataset, the class for Oceania has a misclassification of 90% of the nodes as Asia nodes. The roles dataset depicts an even stronger bias, as most nodes are classified as Asia nodes. In this case, the model penalizes too much the most populated classes (Europe and North America) using dataset imbalance mitigation techniques. However, it is not being able to learn the patterns of the minority classes either. Therefore, it only classifies the nodes for Asia and South America correctly.

Table 7.2: RIR Classification Results

GNN Model	Dataset	Accuracy	F1-score	Precision	Recall
GraphSAGE	Classic	0.909	0.850	0.874	0.855
GraphSAGE	Default	0.752	0.613	0.774	0.687
GraphSAGE	Roles	0.480	0.431	0.71	0.570
GCN	Classic	0.884	0.697	0.905	0.680
GCN	Default	0.594	0.564	0.801	0.645
GCN	Roles	0.343	0.441	0.352	0.659
GAT	Classic	0.897	0.695	0.899	0.693
GAT	Default	0.811	0.721	0.684	0.850
GAT	Roles	0.602	0.621	0.783	0.709
GIN	Classic	0.175	0.051	0.314	0.167
GIN	Default	0.265	0.071	0.377	0.167
GIN	Roles	0.389	0.093	0.065	0.167

The fact that the classic dataset is achieving the best results is interesting, as this dataset has the simplest topology, containing the least amount of nodes and edges. In terms of training time, it was also the most scalable dataset. On average, each training instance of the classic dataset needed 1450 MB of GPU memory, while the roles dataset needed 6450 MB per training instance. The GPU memory is the limiting factor for the number of training instances that can run in parallel. The classic dataset could train five times more instances in parallel than the roles dataset in an 8 GB GPU. The default dataset was in between, with each training instance needing 3500 MB of GPU memory.

7.2 RIR Classification

Regarding the RIR classification, most of the conclusions that can be drawn are similar to the continent classification task. The results are shown in Table 7.2. The best-performing model was GraphSAGE with the classic dataset, with an accuracy of 90.9% and a precision of 87.4%. In this case, the choice of dataset has a bigger impact in terms of the metrics, with the classic dataset achieving the best precision results for all models except for GIN. However, GIN could not achieve results above 40% for any of the metrics, so we can conclude that the classic dataset is the best for this task as well.

Regarding the confusion matrix, we will analyze the difference in the results between the classic and default datasets, as the roles dataset is not achieving good results for any of the models. We will use the results of GraphSAGE as an example, as it is the best-performing model for this task. In Figure 7.2, we can see that the classic

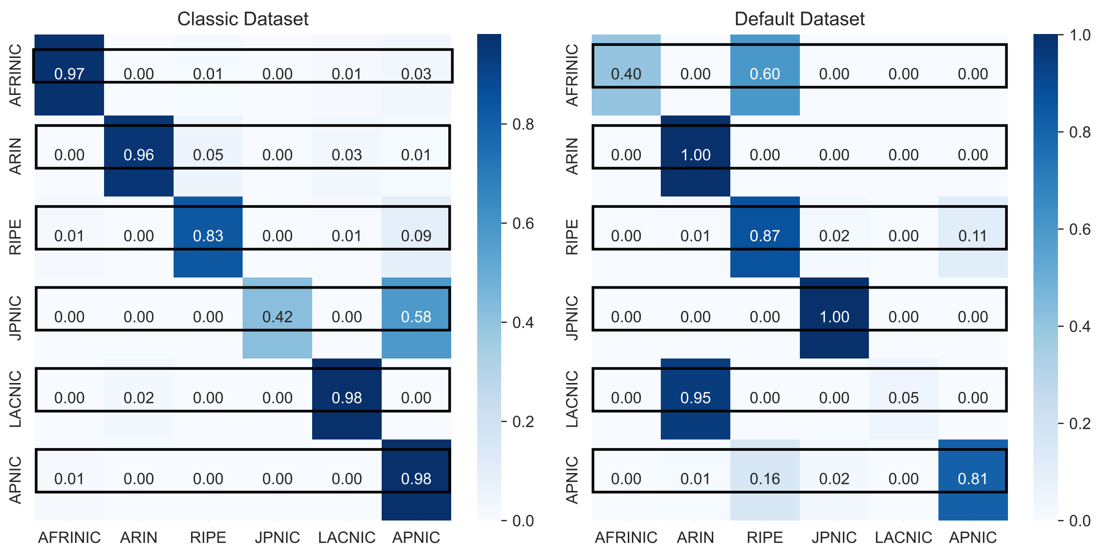


Figure 7.2: Comparison between the normalized confusion matrixes for the RIR classification task for the GraphSAGE model. Accuracy for the classic dataset is over 83% for all classes except for JPNIC. For the default dataset, the classes AFRINIC and LACNIC have misclassification rates up to 60% and 95% respectively.

dataset has only one significant misclassification between the RIRs, with 58% of the JPNIC nodes being classified as APNIC nodes. This is a reasonable misclassification, as JPNIC is the RIR for Japan, and APNIC is the RIR for the Asia-Pacific region, the interconnection between these two regions is strong due to geographical reasons. However, GraphSAGE could not reduce the bias toward the most populated classes in the default dataset, with 60% of the AFRINIC nodes being classified as RIPE nodes and 95% of the LACNIC nodes being classified as ARIN nodes. A similar phenomenon to what we saw in the continent classification task can be seen, with the model being able to generalize better towards the minority classes in the classic dataset but not being able to learn the patterns of the minority classes in the default dataset.

7.3 Link Classification

The results for link classification are shown in Table 7.3. This task achieved good results for the default dataset for all models except GIN. The best performance was achieved by GraphSAGE, with an accuracy of 85.7% and a precision of 64.6%. However, GAT achieved the best results for the roles dataset, with an accuracy of 76.4% and a precision of 61.7%. It was the only model that could achieve both accuracy and precision above 60% for the roles dataset. A better performance was expected on this task, especially considering it had the least amount of labels. This could be because we have two labels

Table 7.3: Link Classification Results

GNN Model	Dataset	Accuracy	F1-score	Precision	Recall
GraphSAGE	Default	0.857	0.658	0.646	0.871
GraphSAGE	Roles	0.462	0.292	0.624	0.553
GCN	Default	0.819	0.602	0.621	0.807
GCN	Roles	0.302	0.064	0.090	0.088
GAT	Default	0.798	0.587	0.628	0.747
GAT	Roles	0.764	0.562	0.617	0.726
GIN	Default	0.446	0.206	0.482	0.333
GIN	Roles	0.547	0.235	0.182	0.333

for the relationship between providers and customers, and it is impossible to distinguish between them with the data we have, making the learning task harder.

In comparison to the results achieved in BGP2Vec [5], we can see that the results are slightly worse, as BGP2Vec achieved 95.8% accuracy for the dataset that contained the same labels as our dataset. However, the results are still promising, as the models are able to learn the patterns of the relationships between ASes. Further improvements could be made if the relationship between providers and customers could be better distinguished, as it was done in BGP2Vec.

7.4 Country Classification

The country classification task was one of the hardest for the models to achieve good results. This is logical, as none of the other tasks had a dataset with such a high number of labels.

In the beginning, more than 250 countries were present in the dataset, but most had a very low number of nodes, so they were grouped into a "rest of the continent" label. The threshold value we chose to group the countries was 250 nodes, so the countries with less than 250 nodes were grouped into the "rest of the continent" label. This value was chosen thinking about the undersampler we used. If the threshold value was too low, the undersampler would remove most of the nodes in the dataset and would not be able to train the models. By choosing 250 nodes, the number of labels was reduced to 50. The number of nodes available for training after undersampling can be calculated as shown in Equation 7.1. In the original case (250 labels, least populated class having one node), the number of nodes available for training would be 250 nodes. We found that 50 classes were a reasonable amount to perform classification.

Table 7.4: Country Classification Results

GNN Model	Dataset	Accuracy	F1-score	Precision	Recall
GraphSAGE	Classic	0.561	0.377	0.439	0.388
GraphSAGE	Default	0.200	0.204	0.295	0.262
GraphSAGE	Roles	0.243	0.090	0.124	0.154
GCN	Classic	0.289	0.149	0.208	0.186
GCN	Default	0.202	0.084	0.174	0.118
GCN	Roles	0.302	0.064	0.090	0.088
GAT	Classic	0.144	0.098	0.2166	0.129
GAT	Default	0.474	0.201	0.2463	0.217
GAT	Roles	0.105	0.088	0.141	0.126
GIN	Classic	0.164	0.091	0.174	0.139
GIN	Default	0.214	0.087	0.195	0.127
GIN	Roles	0.343	0.441	0.352	0.659

$$N_{training\ nodes} = N_{country\ labels} * N_{nodes\ in\ least\ populated\ country} \tag{7.1}$$

For this classification task with 50 classes, the best-performing model was GraphSAGE with the classic dataset, with an accuracy of 56.1% and a precision of 43.9%. The results are shown in Table 7.4. In this task, it needs to be clarified which is the best dataset variation, as the difference between the results is less significant than in the other tasks. The classic dataset achieved the best results for GraphSAGE, whereas GAT had its best results with the default dataset and GIN with the roles dataset. GCN had its best accuracy with the roles dataset but its best precision with the classic dataset. This could mean that the best dataset variation depends on the model used and that no clear winner exists.

In terms of the confusion matrix, we can see the best-performing model and dataset results in Figure 7.3. The diagonal of the confusion matrix is more prominent than what the results in Table 7.4 would suggest. This is because the diagonal represents the correct classification of the nodes, and the results in Table 7.4 are the average of all the classes. The individual countries are ordered by the number of nodes, so the least populated countries are in the left part of the matrix. After the country with the most nodes (US), the countries are grouped in a "rest of the continent" label. We can see that the diagonal is less prominent in the left part of the matrix, as the least populated countries are being heavily misclassified. If we look at the right part of the matrix, we can see that some of the most populated and the "rest of the continent" labels

Table 7.5: Business Classification: First Results

GNN Model	Dataset	Accuracy	F1-score	Precision	Recall
GraphSAGE	Classic	0.330	0.104	0.147	0.184
GraphSAGE	Default	0.130	0.077	0.150	0.182
GraphSAGE	Roles	0.159	0.072	0.204	0.176
GCN	Classic	0.890	0.158	0.197	0.153
GCN	Default	0.246	0.080	0.256	0.112
GCN	Roles	0.097	0.066	0.254	0.109
GAT	Classic	0.064	0.044	0.138	0.126
GAT	Default	0.029	0.034	0.154	0.104
GAT	Roles	0.004	0.010	0.005	0.125
GIN	Classic	0.007	0.005	0.134	0.127
GIN	Default	0.124	0.028	0.140	0.125
GIN	Roles	0.008	0.026	0.049	0.129

have prominent columns, meaning that other countries are being misclassified as these countries. For example, if we look at Germany (DE) column, we will see that countries like Norway, Netherlands, or France are being misclassified as Germany. It would be interesting to analyze the misclassified nodes to see if they are mainly classified into neighboring countries or the average distance between them and the correct country. Given the continent classification results, I would assume that the misclassifications are mostly performed on neighboring countries, but it would be interesting to confirm this assumption.

7.5 Business Classification

Regarding the business classification task, two experiments were performed. In the first experiment, the dataset containing all the labels was used. The results for the first experiment are shown in Table 7.5. Given the results, it was clear that the dataset was too unbalanced for the models to achieve high accuracy and precision. The highest precision was reached by GCN with the default dataset, going up to 89% accuracy. However, this experiment could only reach a precision of 19.7%. If we look at its confusion matrix in Figure 7.4, the model is only classifying the nodes as the most populated classes (Cable/DSP/ISP, NSP, Content), and it is not being able to learn the patterns of the minority classes.

As the results were not promising, a second experiment was conducted. This experiment aimed to see if the results could be improved by grouping the labels in a similar way

Table 7.6: Business Classification: Second Results

Accuracy	F1-score	Precision	Recall
0.817	0.405	0.485	0.396

to the grouping in BGP2Vec [5]. In that work, the labels were grouped into three categories: Transit Access, Enterprise, and Content. We used the same categories, but more classes were present in our dataset. Then, each category contained more classes: Transit Access (Cable/DLP/ISP, NSP, Network Services, and Route Collectors), Enterprise (Enterprise, Government, Non-Profit, Educational/Research), and Content (Content). Regarding models and dataset variation, we decided to use GraphSAGE with the classic dataset, as it had been proven to be the most consistent combination for all the other tasks. The results for this experiment are shown in Table 7.6. In comparison to the first experiment, the results improved significantly, with the accuracy increasing to 81.7% and the precision to 48.5%. In terms of the confusion matrix, in Figure 7.5, we can see that the model is able to classify over the baseline of 33%. However, the classification could be further improved, especially regarding the Enterprise label, where only 50% of the nodes are classified correctly.

These results are better than the ones achieved by BGP2Vec, which achieved an accuracy of 79.2%, and no precision was reported. From this comparison, we can infer that GNNs can reach similar results to BGP2Vec. However, it is difficult to compare the results directly, as the precision in BGP2Vec was not reported. Our results could be further improved by using a cleaner dataset as it was done in BGP2Vec, where the nodes belonging to minority classes are removed instead of placed in a grouping class as we did.

This task, however, is complicated for the models, as the dataset is heavily unbalanced, and most of the nodes in the graph have not disclosed their business type. Further analysis could be made if we enhance the dataset with more labeled nodes by using new sources of information.

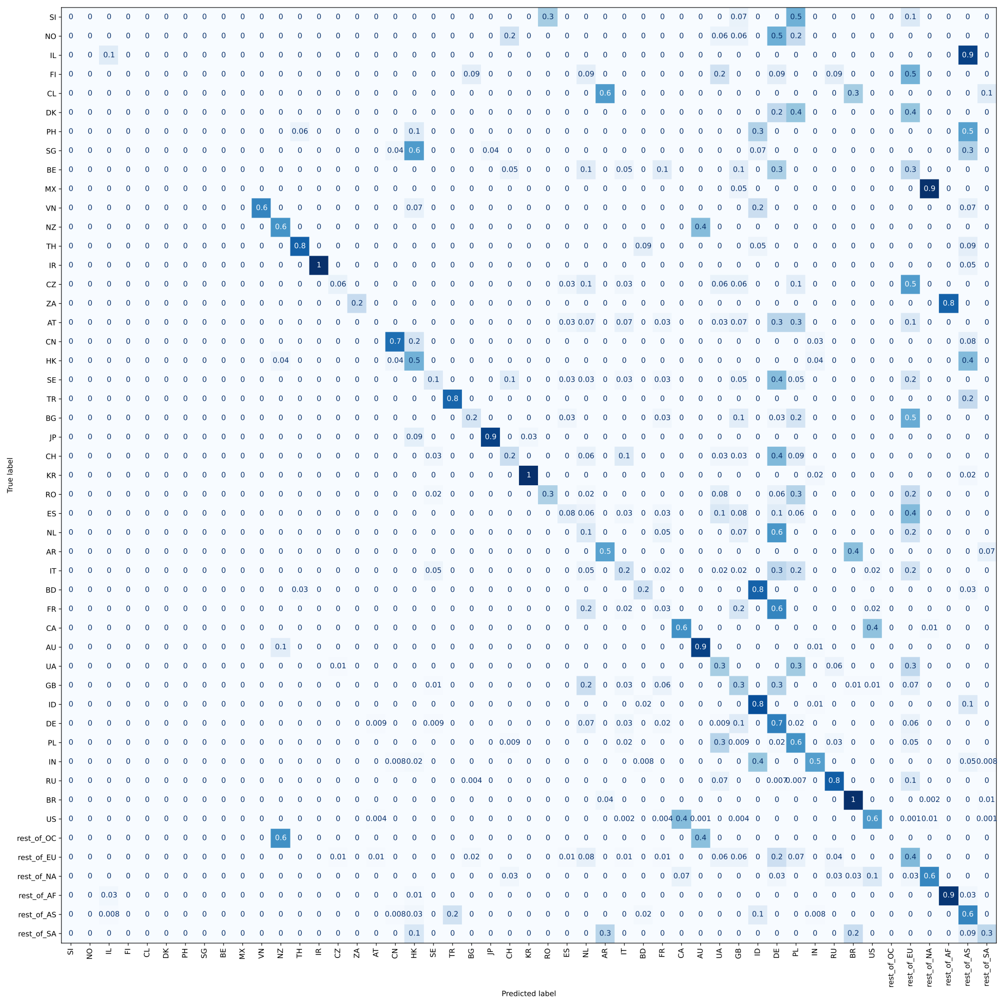


FIGURE 7.3: Normalized confusion matrix for the country classification task for the GraphSAGE model with the classic dataset. The diagonal is more prominent than what the results in Table 7.4 would suggest. This is due to the fact that the diagonal represents the correct classification of the nodes, and the results in Table 7.4 are the average of all the classes. Less populated classes are being misclassified as more populated classes.

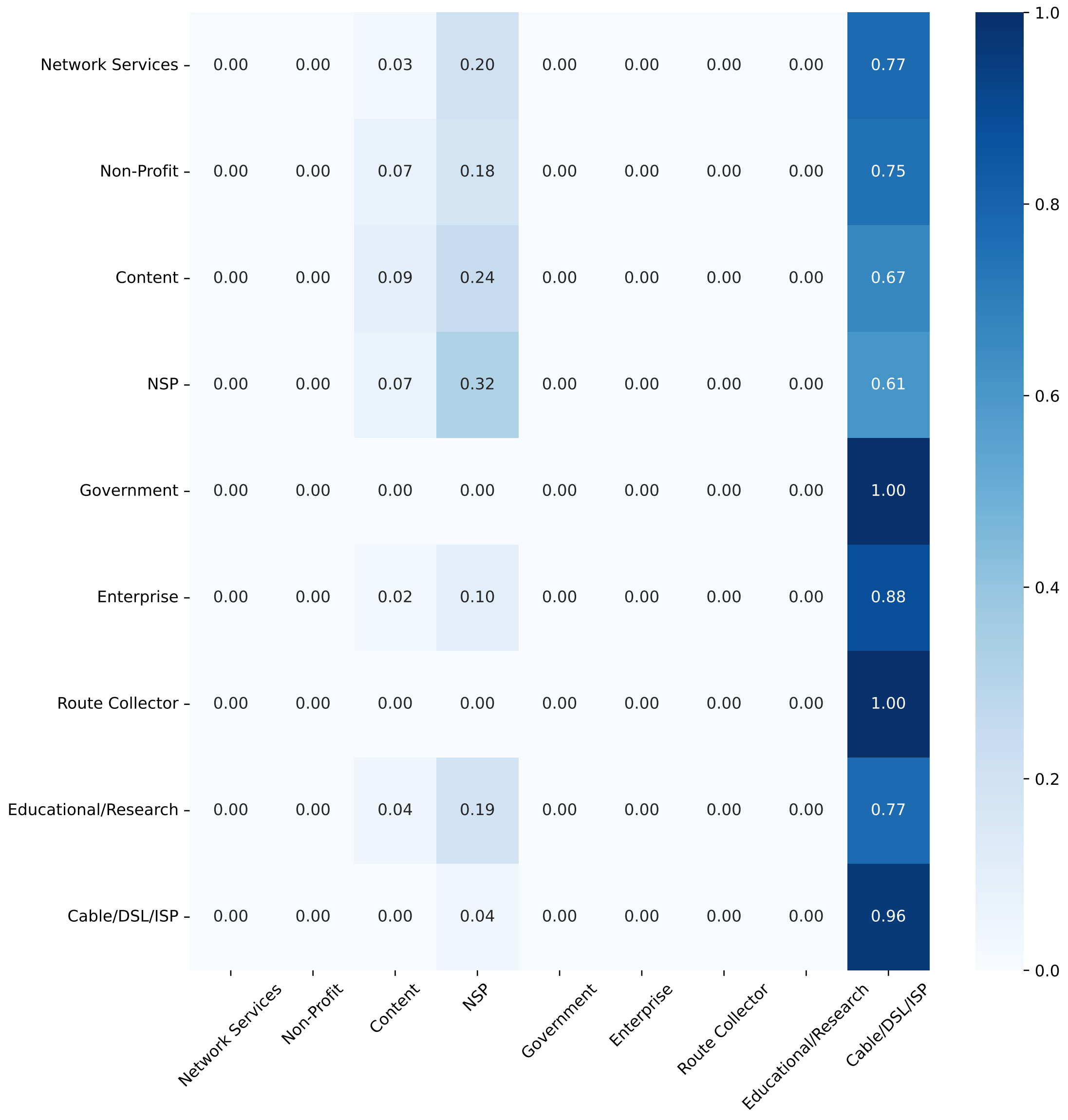


Figure 7.4: Normalized confusion matrix for the business classification task for the GCN model with the classic dataset. Even if accuracy was of 89%, the model is only classifying the nodes as the most populated classes (Cable/DSP/ISP, NSP, Content), and it is not being able to learn the patterns of the minority classes.

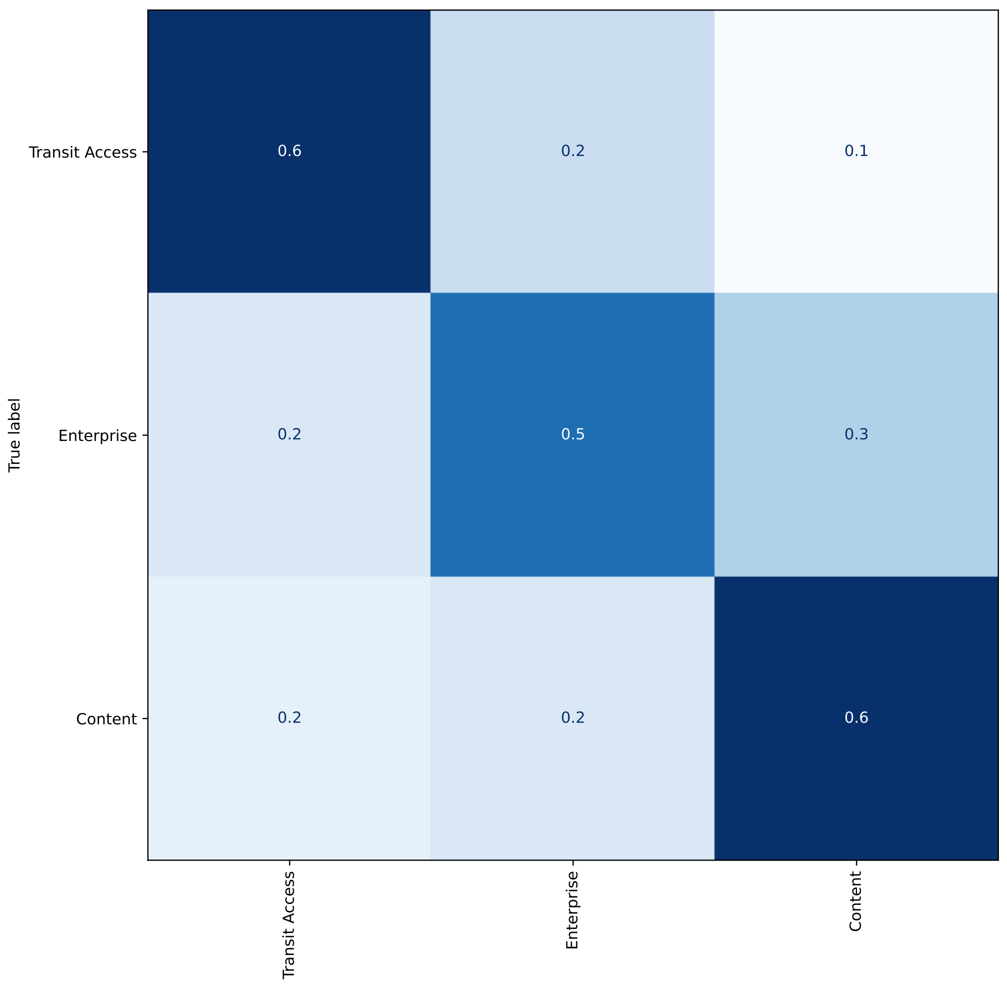


Figure 7.5: Normalized confusion matrix for the business classification task for the GraphSAGE model with the classic dataset. The model is able to classify the classes over the 33% baseline for all classes. However, the classification could be further improved, especially regarding the Enterprise label, where only 50% of the nodes are classified correctly.

Chapter 8

Conclusions and Future Work

After analyzing the results for all tasks, we can obtain some general conclusions about the performance of the different GNN models and datasets.

Overall, the most consistent GNN model was GraphSAGE, with the best results in terms of accuracy, F1-score, precision, and recall across all classification tasks. There are several reasons for this, but the main reason is that the sampling strategy used by GraphSAGE allows it to generalize better to unseen nodes and work with larger graphs.

GCN and GAT also performed well in some tasks but were not as consistent as GraphSAGE. For the default and roles datasets, they performed worse than GraphSAGE.

GIN was the worst-performing model, with the worst results in all tasks except for the business classification task, where it was not the worst but still performed poorly. This could be due to several factors, such as the hyperparameter search space needing to be broadened for GIN to find a good set of hyperparameters or the GIN implementation used in this work not being suited for the classification tasks we were trying to solve.

Even if we have not talked about it in depth because it was not one of the main goals of this work, it is worth mentioning that the GAT model took the longest to train, taking, on average, four times longer than the other models. This makes GCN a better option for our classification tasks, as it usually reaches similar results to GAT but in a shorter time. It is out of the scope of this work, but further research could be done in this area, as it would be interesting to see if the results and training time obtained by GAT could be improved by using a different implementation or a different hyperparameter search space.

Regarding the datasets, the best-performing dataset in most cases was the classic dataset, followed by the default dataset, and finally, the roles dataset. This is most likely because simple topologies are easier to learn than more complex ones, and the classic dataset has the simplest topology. This is an interesting finding, as the classic dataset is the one that takes the least amount of computing resources to train the GNN models, as it has the least amount of nodes and edges.

From all the experiments conducted, we can conclude that the best GNN model and dataset combination is GraphSAGE with the classic dataset. In case future work is done on this topic, this combination should be used as a baseline. However, it could also be the case that the hyperparameter search space used in this work needed to be broadened and that there are better hyperparameter combinations for the other GNN models and datasets that were not found in this work.

Regarding the classification tasks, given the obtained results, we can infer some conclusions about them.

In terms of geographical classification, the models are capable of classifying the nodes with high accuracy at a continental level. However, it is more difficult to classify the nodes at a country level. This is probably due to the fact that the dataset is more unbalanced in terms of countries than continents and that the country classification task has around ten times more labels than the continent classification task. Additionally, a number of ASes are assigned to one country but operate in several countries, making it harder for the models to learn the specific patterns of each country. Another metric that could be calculated for this purpose would be checking if the true label of the country is among the top N predicted labels or if there is a geographical correlation between the true label and the predicted label.

The RIR classification was a similar task to the continental classification. However, as the dataset is more balanced in terms of RIRs, the models achieved better results in the continent classification task than in the RIR classification task.

The business classification task had a poor performance in the first experiment due to the unbalanced nature of the dataset, with the majority class being three orders of magnitude above the minority class. However, after grouping the labels according to the grouping done in the BGP2Vec paper [5], the results were much better. By grouping the labels, the dataset became more balanced, which allowed the models to learn the patterns of each label more easily.

In general, we can state that the main factor affecting the models' performance is the level of imbalance of the datasets. The more balanced the dataset is, the better the models will perform. We were able to mitigate this by using the undersampler and the weighted loss, but more is needed to solve the problem completely. Using a different formula for the weighted loss that would not penalize so much the majority classes could improve the results even further.

To conclude, we can say that the work done in this thesis has been successful in achieving the goals set at the beginning of the work. Further work could be done in this area, such as using different GNN models, datasets, or hyperparameter tuning. The datasets could be modified in order to carry out an analysis of the impact of masking different features on the classification tasks. Additional classification tasks could also be explored, such as classifying the nodes according to other properties, including some of the numerical features.

Bibliography

[1] Y. Rekhter, S. Hares, and T. Li, *A Border Gateway Protocol 4 (BGP-4)*, RFC 4271, Jan. 2006. DOI: `10.17487/RFC4271`. [Online]. Available: `https://www.rfc-editor.org/info/rfc4271`.

[2] J. A. Hawkinson and T. J. Bates, *Guidelines for creation, selection, and registration of an Autonomous System (AS)*, RFC 1930, Mar. 1996. DOI: `10.17487/RFC1930`. [Online]. Available: `https://www.rfc-editor.org/info/rfc1930`.

[3] J. Moy, *OSPF Version 2*, RFC 2328, Apr. 1998. DOI: `10.17487/RFC2328`. [Online]. Available: `https://www.rfc-editor.org/info/rfc2328`.

[4] N. R. Organization, *Regional Internet Registries*, 2023. [Online]. Available: `https://www.nro.net/about/rirs/`.

[5] T. Shapira and Y. Shavitt, "BGP2Vec: Unveiling the Latent Characteristics of Autonomous Systems", *IEEE Transactions on Network and Service Management*, vol. PP, pp. 1–1, 2022. DOI: `10.1109/TNSM.2022.3169638`.

[6] L. Gao, "On inferring autonomous system relationships in the internet", *Networking, IEEE/ACM Transactions on*, vol. 9, no. 4, pp. 431–447, 2001.

[7] B. Sanchez-Lengeling, E. Reif, A. Pearce, and A. B. Wiltschko, "A gentle introduction to graph neural networks", *Distill*, 2021, https://distill.pub/2021/gnn-intro. DOI: `10.23915/distill.00033`.

[8] J. Zhou, G. Cui, S. Hu, *et al.*, *Graph Neural Networks: A Review of Methods and Applications*, 2021. arXiv: `1812.08434 [cs.LG]`.

[9] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks", *arXiv preprint arXiv:1609.02907*, 2016.

[10] W. L. Hamilton, R. Ying, and J. Leskovec, *Inductive Representation Learning on Large Graphs*, 2017. arXiv: `1706.02216 [cs.SI]`.

[11] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph Attention Networks", *arXiv preprint arXiv:1710.10903*, 2017.

[12] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, “How Powerful are Graph Neural Networks?”, *CoRR*, vol. abs/1810.00826, 2018. arXiv: `1810.00826`. [Online]. Available: `http://arxiv.org/abs/1810.00826`.

[13] University of Oregon Advanced Network Technology Center, *Route views project*, `http://www.routeviews.org/`, 2018.

[14] I. R. Registry, *I. R. Registry*, `http://www.irr.net/`, 2018.

[15] X. Dimitropoulos, D. Krioukov, G. Riley, and K. Claffy, “Revealing the Autonomous System Taxonomy: The Machine Learning Approach”, arXiv, Technical Report, 2006, Assumed arXiv.org perpetual, non-exclusive license to distribute this article for submissions made before January 2004. DOI: `10.48550/ARXIV.CS/0604015`. [Online]. Available: `https://arxiv.org/abs/cs/0604015`.

[16] Y. Freund and R. E. Schapire, “A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting”, *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997, ISSN: 0022-0000. DOI: `https://doi.org/10.1006/jcss.1997.1504`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S002200009791504X`.

[17] D. P. Giakatos, S. Kostoglou, P. Sermpezis, and A. Vakali, “Benchmarking Graph Neural Networks for Internet Routing Data”, arXiv, Technical Report, 2022, Creative Commons Attribution Non Commercial Share Alike 4.0 International. DOI: `10.48550/ARXIV.2210.14189`. [Online]. Available: `https://arxiv.org/abs/2210.14189`.

[18] A. Grover and J. Leskovec, “node2vec: Scalable Feature Learning for Networks”, in *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2016.

[19] *RIPE NCC Routing Information Service*, `https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris`, [Online].

[20] *The CAIDA UCSD AS to Organization Mapping Dataset*, `https://www.caida.org/data/as_organizations`, [Online].

[21] *PeeringDB*, `https://www.peeringdb.com/`. [Online]. Available: `https://www.peeringdb.com/`.

[22] P. Henschke, “BGP-based Internet Graph Analysis”, Department of Informatics, M.S. thesis, Technical University of Munich, 2023.

[23] *Neural Network Intelligence*, `https://github.com/microsoft/nni/`, Accessed: August 1, 2023.

[24] M. Luckie, B. Huffaker, A. Dhamdhere, V. Giotsas, and k. claffy, “AS Relationships, Customer Cones, and Validation”, in *Proceedings of the 2013 Conference on Internet Measurement Conference*, ser. IMC ’13, Barcelona, Spain: Association for Computing Machinery, 2013, pp. 243–256, ISBN: 9781450319539. DOI:

10.1145/2504730.2504735. [Online]. Available: https://doi.org/10.1145/2504730.2504735.

[25] Z. Jin, X. Shi, Y. Yang, X. Yin, Z. Wang, and J. Wu, "TopoScope: Recover AS Relationships From Fragmentary Observations", in *Proceedings of the ACM Internet Measurement Conference*, ser. IMC '20, Virtual Event, USA: Association for Computing Machinery, 2020, pp. 266–280, ISBN: 9781450381383. DOI: 10.1145/3419394.3423627. [Online]. Available: https://doi.org/10.1145/3419394.3423627.

[26] *pycountry*, 2023. [Online]. Available: https://pypi.org/project/pycountry/.

[27] *NetworkX*, https://networkx.org/, Accessed: August 1, 2023.

[28] *Deep Graph Library*, https://www.dgl.ai/, Accessed: August 1, 2023.

[29] *scikit-learn*, https://scikit-learn.org/stable/, Accessed: August 1, 2023.