

Graph Neural Network Based Semantic Mapping and Classification of Dataset for Robotics Applications

by

Devesh Sharma
202111041

A Thesis Submitted in Partial Fulfilment of the Requirements for the Degree of

MASTER OF TECHNOLOGY

in

INFORMATION AND COMMUNICATION TECHNOLOGY

to

DHIRUBHAI AMBANI INSTITUTE OF INFORMATION AND COMMUNICATION TECHNOLOGY



October, 2023

Declaration

I hereby declare that

- i) the thesis comprises of my original work towards the degree of Master of Technology in Information and Communication Technology at Dhirubhai Ambani Institute of Information and Communication Technology and has not been submitted elsewhere for a degree,
- ii) due acknowledgment has been made in the text to all the reference material used.



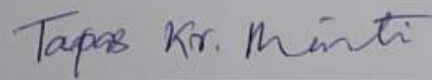
Devesh Sharma

Certificate

This is to certify that the thesis work entitled "Graph Neural Network Based Semantic Mapping and Classification of Dataset for Robotics Applications" has been carried out by Devesh Sharma for the degree of Master of Technology in Information and Communication Technology at *Dhirubhai Ambani Institute of Information and Communication Technology* under our supervision.



Prof. P M Jat
Thesis Supervisor



Prof. Tapas Kumar Maiti
Thesis Co-Supervisor

Acknowledgments

I want to convey my sincere appreciation to my respected thesis supervisors, Professor Tapas Kumar Maiti and Professor P. M. Jat, for their unwavering guidance, invaluable insights, and continuous support throughout the journey of this research work. Their mentorship has been instrumental in shaping my understanding and fostering my growth in the field of semantic mapping and place classification.

I am deeply thankful to Lord Ganesha and Goddess Saraswati for bestowing upon me the wisdom, strength, and determination required to undertake this academic endeavor. Their blessings have been a source of inspiration, helping me navigate challenges and achieve milestone.

I extend my sincere appreciation to my family and friends for their unending encouragement and belief in my capabilities. Their encouragement has provided me with the motivation to persist and excel in my academic pursuits.

Lastly, I would like to acknowledge the vast body of knowledge contributed by researchers, scholars, and practitioners in the field of Deep Learning, Statistical Modeling and machine learning. Their contributions have laid the foundation for this work and have greatly enriched my understanding of the subject.

This journey would not have been possible without the invaluable contributions of each of these individuals. My deepest gratitude goes out to all those who have played a role, directly or indirectly, in making this research a reality.

Contents

Abstract	v
List of Principal Symbols and Acronyms	v
List of Tables	vi
List of Figures	vii
1 Introduction	1
1.1 Semantic Mapping and Semantic Place Classification	1
1.2 Semantic Place Classification in Autonomous Systems	2
2 Literature Survey	6
2.1 Some Previous Methods	6
2.2 Sum Product Network (SPN)	8
2.3 Graph Sum Product Network (Graph SPN)	9
2.4 Graph Neural Network (GNN)	11
2.5 Libraries and Tools	12
2.5.1 Numpy	12
2.5.2 pandas	12
2.5.3 libspn	13
2.5.4 libspn - keras	13
2.5.5 Qtorch - Geometric, sparse, scatter	13
2.5.6 torch-geometric	14
2.5.7 NetworkX	15
3 Dataset and Data Preparation	16
3.1 Dataset	16
3.2 Dataset Preparation and Preprocessing	18

4	Sum Product Network (SPN) Architecture	22
4.1	Implementation	22
4.2	Result	27
5	Graph Neural Network (GNN)	29
5.1	Introduction	29
5.2	Implementation and Simulations	34
5.2.1	GNN Based Local Place Semantic Classification	34
5.2.2	Multi Level Prediction Using GNNs	36
5.3	Results	38
6	Conclusion and Future work	42
	References	44

Abstract

In the field of Robotics, deriving meaningful insights from spatial information is pivotal. Our objective was to work with 2D dataset as 3D datasets are relatively more time consuming as our target is to objective was to do the foundation work for real time inferences where speed is also an important key factor and so to get best possible results, trying to improvise in accuracy and speed. Our focus lies in semantic mapping, semantic place classification where mobile robots interpret partial, noisy sensory data. We delve into end-to-end techniques rooted in probabilistic deep networks, studying Local-SPNs, Graph SPNs, and TopoNets. Additionally, we explore GNNs for semantic place classification. Our report provides succinct insights into these methods, emphasizing their principles and implementations, including the local SPN model and GNN for semantic classification. We got our best accuracy with GNN 70.15% which is less compared to previous best 80.14% but we achieved better results with speed as our GNN model was 1.89x faster than the previous best available method. We also explore multi-level semantic place classification through GNNs and consider the potential of Graph Attention Networks (GAT) for complex datasets. Here we did a proof of concept that multilevel semantic classification is possible with GNNs but it needs more research in this area.

List of Tables

4.1	Results comparisons for different methods with our method	28
5.1	Results comparisons for different methods with our method	38
5.2	Inference time results comparisons for different methods with our method	39
5.3	Performance Score results comparisons for different methods with our method	40
5.4	Results comparisons for GAT and GNN	41

List of Figures

1.1	Sub-problems of semantic mapping	4
1.2	Thesis work flowchart and overview.	5
3.1	Dataset view graph	17
3.2	Dataset labels	18
3.3	Graphical representation of COLD data	19
4.1	Local SPN model code	25
4.2	Local SPN architecture	26
5.1	Adjacency matrix	30
5.2	Degree matrix	31
5.3	Laplacian matrix	32
5.4	GNN architecture	35
5.5	Trade off curve for accuracy vs inference time	40

CHAPTER 1

Introduction

In this chapter we briefly discussed and introduced about the problem statement and methods used in past and what we tried to do. This chapter provides a brief overview and motivation why we worked on this problem statement. We also briefly demonstrate work flow of thesis.

1.1 Semantic Mapping and Semantic Place Classification

In this section we briefly take a look on what is semantic mapping, semantic place classification, some previous methods and what we tried to do.

Semantic mapping and semantic place classification are crucial elements within the realm of autonomous systems, providing the capacity for robots and intelligent agents to understand and interact intelligently with their surroundings. These methodologies bridge the gap between raw sensor data and meaningful representations, facilitating informed decision-making and adaptability to dynamic scenarios.

The concept of semantic mapping involves the creation of an environment representation that surpasses mere geometric data. Conventional mapping approaches predominantly rely on occupancy grids or geometric models, which offer valuable spatial insights but lack semantic context. In contrast, semantic mapping incorporates high-level semantic information, such as object categories, room labels, or scene descriptions, into the mapping process [1].

By fusing semantics with geometry, robots gain the ability to comprehend the significance and purpose of different regions and objects in their environment. For instance, in a household setting, the robot can recognize rooms as the kitchen, bedroom, or living room, and identify objects like tables, chairs, and refrigerators. This enriched understanding of the environment empowers robots to perform

tasks more efficiently and safely.

Various machine learning techniques, such as Markov random fields, conditional random fields, or Sum Product Networks (SPNs) [2], can be employed to achieve semantic mapping. These models effectively combine spatial and semantic cues, capturing uncertainties and correlations present in the data.

Semantic place classification focuses on assigning meaningful labels to specific locations or regions within the environment. Instead of merely recognizing objects, semantic place classification aims to identify the overall context and function of a place. For instance, a park, shopping mall, or hospital could be recognized as distinct semantic places based on their unique characteristics.

For Semantic Place Classification there exists several studies. Many deep neural network and deep probabilistic approaches have been used to perform experiments in this aspect. For ROS based datasets above methods have shown good accuracy but still a lot can be done in this field.

Graph Neural Networks (GNNs) can be a powerful tool in semantic place classification tasks. GNNs can process graph-structured data, where nodes represent different locations, and edges capture the spatial relationships between these places. By leveraging the spatial dependencies and semantic information, GNNs can accurately classify the semantic places in the environment.

The advantage with GNN is that, it is designed in a manner such that in most cases it outperform other neural networks in terms of accuracy and speed , so we have developed a GNN based semantic place classification.

1.2 Semantic Place Classification in Autonomous Systems

In this section we briefly discussed little more on semantic place classification in context of autonomous systems and discussed some previous work, our work, the motivation and workflow of our thesis.

Semantic place classification for autonomous systems is a critical aspect of enabling intelligent agents, such as robots, to understand and navigate their environments effectively. This process involves assigning meaningful labels to specific locations or regions within the environment, allowing the autonomous system to comprehend the semantic context and make informed decisions during navigation and interaction tasks.

One powerful approach for semantic place classification involves utilizing Sum Product Networks (SPNs). SPNs are probabilistic graphical models that can ef-

fectively capture and represent complex distributions over structured data. In the context of semantic place classification, SPNs can seamlessly combine spatial information from sensory inputs with meaningful semantics associated with different places, enabling the autonomous system to reason about its surroundings more intelligently.

GraphSPN, an extension of the traditional SPN, is particularly well-suited for semantic place classification tasks. GraphSPNs handle data represented as graphs, where nodes represent different locations, and edges capture spatial relationships between these places. By leveraging the graph structure, GraphSPNs can effectively model the dependencies and interactions between different places, facilitating accurate and context-aware classification.

Graph Neural Networks (GNNs) are another powerful technique for semantic place classification in autonomous systems. GNNs are designed to process data with graph-like structures, making them a natural fit for tasks that involve spatial relationships between locations. In the context of semantic place classification, GNNs can effectively capture the contextual information from neighboring places, enabling the autonomous system to understand the semantics of different locations based on their spatial dependencies.

The process of semantic place classification using GNNs involves a message-passing mechanism, where information is propagated through the graph's edges to update node representations. As information is iteratively passed through the GNN layers, the system gains a refined understanding of each location's semantics, considering both its local features and the collective information from the entire graph.

Incorporating SPNs, GraphSPNs, and GNNs in the context of semantic place classification empowers autonomous systems with enhanced environmental comprehension. By combining spatial information with meaningful semantics, these models enable the system to make informed decisions about its surroundings, adapt to dynamic changes in the environment, and engage in more sophisticated tasks.

Furthermore, the integration of these advanced techniques addresses the challenges of uncertainty and imprecise information often encountered in real-world scenarios. SPNs handle uncertainty in the data by providing principled probabilistic reasoning, while GNNs effectively capture spatial dependencies, allowing the system to reason about the context of different places[3].

The benefits of semantic place classification using SPNs, GraphSPNs, and GNNs extend to various applications in autonomous systems. In robotics, semantic place

classification enhances navigation by enabling the autonomous system to distinguish between different places and plan paths more intelligently. In smart home systems, these techniques allow intelligent devices to adapt to users' needs based on the semantic understanding of different rooms and their associated tasks.

Overall, semantic place classification with SPNs, GraphSPNs, and GNNs represents a powerful approach to enriching the autonomy of intelligent agents. Their ability to capture and reason about spatial and semantic information fosters improved decision-making, adaptability, and interaction in diverse real-world scenarios. As research in these areas continues to evolve, we can expect even more sophisticated and capable autonomous systems, contributing to the advancement of artificial intelligence and robotics. Figure 1.1 below demonstrates the problem of semantic mapping differentiated into sub-problems

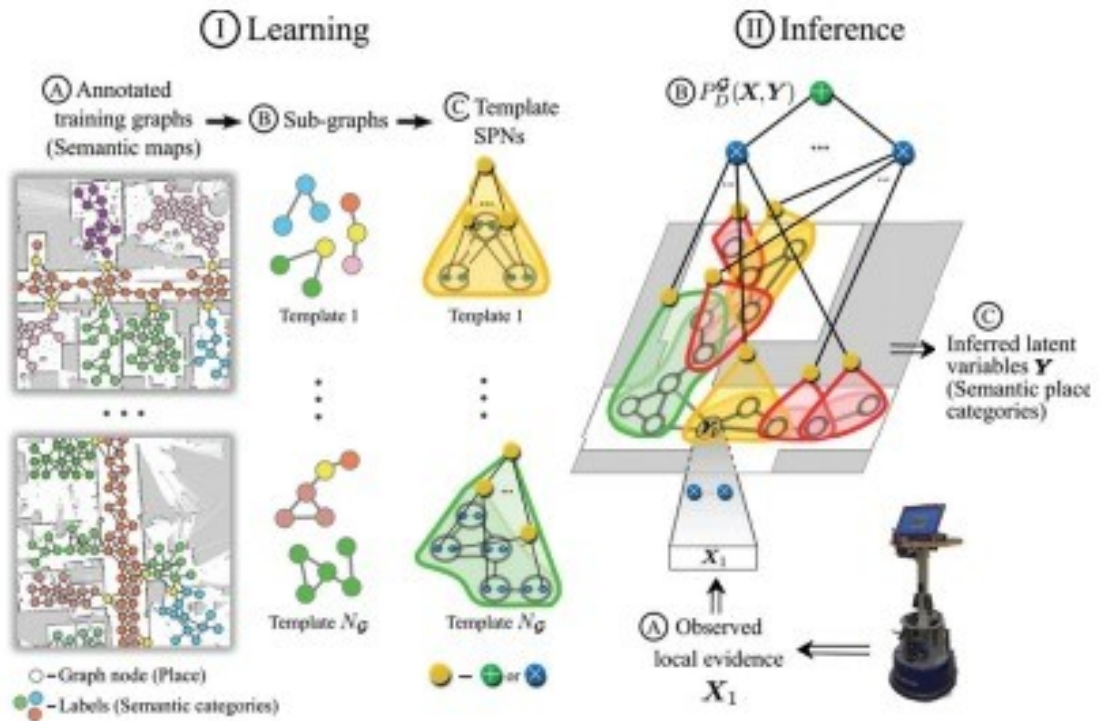


Figure 1.1: Sub-problems of semantic mapping [4].

Understanding semantic place classification is crucial in mapping places based on their meanings. In this field, dealing with complex and detailed 3D datasets such as images poses significant challenges. While models based on such datasets achieve high accuracy, they struggle with quick training and real-time applications.

In our research, we shift our focus to a different type of dataset—2D graphs. We explore effective techniques and design specialized models to address the

challenges presented by this simpler data format. The objective is to enhance semantic mapping using these simpler 2D graph-structured datasets.

The underlying motivation propelling our pursuit is grounded in fortifying the foundational framework that will substantiate the future endeavors of real-time autonomous systems. By cultivating a comprehensive understanding of the intricacies associated with semantic mapping through 2D datasets or graphical datasets in our case, we aim to enhance the efficacy and efficiency of autonomous systems as they operate in dynamic, real-world scenarios.

Our main motivation is to strengthen the fundamental framework that will support future real-time autonomous systems. We believe that by thoroughly understanding how semantic mapping works with 2D graph datasets, we can improve the effectiveness and efficiency of these autonomous systems as they navigate dynamic real-world situations. Figure 1.2 below represents the general flow of the Thesis.

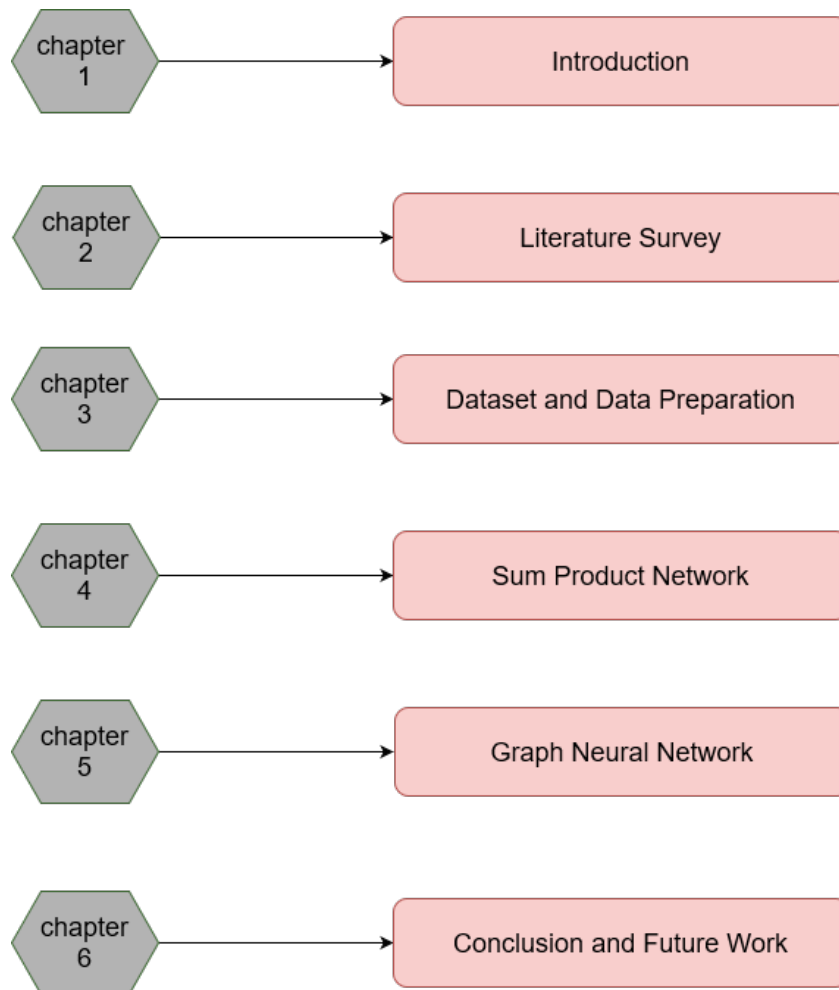


Figure 1.2: Thesis work flowchart and overview.

CHAPTER 2

Literature Survey

In this chapter we discussed about the literature survey that we have done and was important to understand and work for our problem statement. Here the reader will also get some insights about previous works as well as neural networks, tools and technologies we have used in our research. It will also develop the intuitions for the reader why we choose to work with the specific methods(GNNs) to solve our problem.

2.1 Some Previous Methods

In this section we have discussed previous work apart from SPN based and GNN based models. Probabilistic Graphical Models (PGMs) have emerged as a versatile and powerful framework in the realms of semantic mapping and place classification, attracting significant attention due to their capacity to model intricate spatial and semantic relationships. This class of models provides an elegant way to represent complex dependencies between variables through probabilistic relationships, making them particularly well-suited for scenarios where uncertainty and hidden relationships are prevalent [5].

In the context of semantic mapping, PGMs offer a holistic approach that seamlessly integrates spatial information with environmental semantics. These models excel in capturing the underlying spatial configurations of an environment while incorporating observable cues like sensor data or landmarks. This amalgamation of information yields a comprehensive representation that effectively accommodates the inherent uncertainties, noise, and partial observability encountered in real-world data [6]. This robustness positions PGMs as an instrumental framework for unveiling the structural semantics of environments, a critical aspect for tasks such as navigation, exploration, and decision-making in autonomous systems.

PGMs have also demonstrated their prowess in semantic place classification,

a domain of paramount importance in robotics and autonomous navigation. The distinctive strength of PGMs lies in their ability to model probabilistic relationships among variables. In the context of spatial environments, PGMs can capture the probabilistic transitions of semantic labels as the robot traverses different locations. By understanding the nuanced evolution of semantic labels over time, PGMs enable accurate and contextually aware place classification [7]. This temporal sensitivity empowers the model to infer the semantic attributes associated with specific locations, thus contributing to a more nuanced understanding of the environment's semantics.

Past research underscores the significance of PGMs in semantic mapping and place classification. For instance, the study in [8] showcased the integration of PGMs to improve the accuracy of semantic mapping and localization. Additionally, the research in [9] explored the application of Bayesian networks, a type of PGM, to enhance the semantic classification of places.

In conclusion, Probabilistic Graphical Models represent a versatile and adaptable approach to semantic mapping and place classification. By encapsulating complex relationships and uncertainty through probabilistic dependencies, PGMs offer an enriched understanding of real-world environments. Their capacity to capture dependencies and temporal dynamics positions them as pivotal tools in the advancement of autonomous systems and robotics, enabling more informed navigation, classification, and interaction within intricate environments. As the field of robotics evolves, the exploration and refinement of PGMs in the context of semantic mapping hold the promise of unlocking novel avenues for research and innovation [10].

Hidden Markov Models (HMMs) have emerged as a robust and versatile framework within the domain of semantic mapping and place classification, garnering considerable attention due to their ability to capture complex spatial and semantic relationships. This probabilistic graphical model offers a sophisticated way to encapsulate hidden states and observed data, proving particularly effective in scenarios where the latent states are not directly observable but leave discernible traces in the data [11].

In the realm of semantic mapping, HMMs present an intriguing approach that combines spatial information with environmental semantics. This integration bridges the gap between spatial relationships and the meaning attributed to different locations. In practice, HMMs allow us to model the hidden spatial configurations of an environment while incorporating observable cues such as sensor data or landmarks [12]. The result is a comprehensive representation that

accounts for the inherent uncertainties, noise, and partial observability often encountered in real-world data. This robustness makes HMMs an ideal framework for extracting and comprehending the structural semantics of environments, a crucial capability for autonomous systems that need to navigate, understand, and make informed decisions based on their surroundings.

Moreover, HMMs have found particular utility in semantic place classification, a task of paramount importance in robotics and autonomous navigation. The distinctive advantage of HMMs lies in their ability to leverage the temporal sequence of observations. In the context of spatial environments, HMMs can identify patterns in semantic transitions as the robot moves through different locations. By understanding how semantic labels evolve over time, HMMs enable more accurate and context-aware place classification [13]. This temporal sensitivity empowers the model to infer the semantic attributes of specific locations, contributing to an enhanced understanding of the environment's semantics.

Past research has showcased the prowess of HMMs in semantic mapping and place classification. The paper [14] demonstrated the integration of RFID tag information with robot pose estimations using HMMs, resulting in improved semantic mapping accuracy. Additionally, the research [15] explored the utilization of dynamic Markov chains, a variation of HMMs, to capture the evolving semantic labels over time. These studies highlight the adaptability and versatility of HMMs in various semantic mapping scenarios.

In conclusion, Hidden Markov Models offer a sophisticated and adaptable approach to semantic mapping and place classification. By seamlessly combining hidden spatial states with observable semantics, HMMs provide a nuanced understanding of real-world environments. Their capacity to encapsulate complex relationships and temporal dependencies makes them a formidable tool in autonomous systems and robotics, enhancing our ability to traverse, classify, and interact with the environment around us. As technology continues to advance, the further exploration and refinement of HMMs in the context of semantic mapping promise to unlock new horizons in the department of artificial intelligence and robotics.

2.2 Sum Product Network (SPN)

Sum Product Networks (SPNs) have emerged as an very important type of probabilistic graphical model, garnering considerable interest within the machine learning community. Their ability to effectively represent and compute probabilities

over structured data renders them highly valuable in various applications, such as classification, density estimation, and even semantic mapping within autonomous systems [16], [4].

SPNs employ a hierarchical architecture that leverages the advantages of sum nodes and product nodes [17], [18], [2]. Sum nodes function as mixture models, enabling the network to capture intricate variable relationships [19], [20]. Conversely, product nodes act as factors, breaking down the joint probability distribution into a collection of conditional distributions.

An essential benefit of SPNs is their capability to handle tractable inference, mitigating the computational complexity associated with conventional deep learning models' forward and backward passes. SPNs efficiently compute marginal and conditional probabilities, allowing them to tackle uncertainty and probabilistic reasoning effectively in real-life situations [4].

SPNs provide an innate and straightforward approach to integrate continuous and discrete variables within a unified model. This flexibility enhances their suitability for diverse tasks that encompass mixed data types, establishing them as a practical choice for numerous real-world applications [21], [22].

To conclude, Sum Product Networks provide a robust and computationally efficient method for probabilistic modeling, boasting the ability to conduct tractable inference and accommodate structured data with a mix of variable types. Their diverse applications range from classification and density estimation to more intricate endeavors like semantic mapping, contributing to the enhancement of autonomous systems and intelligent agents. As ongoing research pushes the boundaries in this domain, SPNs have the potential to foster innovation and tackle challenges across various realms of machine learning and beyond [23].

2.3 Graph Sum Product Network (Graph SPN)

Graph SPN, or Graph Sum Product Network, is an advanced probabilistic graphical model designed to handle structured data represented in the form of graphs. Graph SPNs build upon the foundation of traditional Sum Product Networks by incorporating graph-based structures to model complex relationships and dependencies among variables.

In Graph SPNs, each node in the graph represents a random variable, and the edges in the graph capture the probabilistic connections between these variables. This unique representation enables Graph SPNs to effectively capture the intricate patterns and interactions present in graph-structured data. By considering

the dependencies among variables through the graph structure, Graph SPNs can efficiently compute probabilities and perform functionalities like node classification, relation for edge inference, and even graph generation.

Graph SPNs are particularly valuable in domains where data naturally exhibits graph-like relationships, such as social networks, biological networks, and knowledge graphs. Their ability to model and reason about structured data makes them a powerful tool for handling real-world scenarios involving interconnected entities and complex dependencies.

The incorporation of graph-based inference algorithms further enhances the capabilities of Graph SPNs, allowing for efficient and scalable computations on large-scale graph data. As a result, Graph SPNs have become increasingly popular in various applications, including recommendation systems, bioinformatics, and social network analysis.

In summary, Graph SPNs extend the capabilities of Sum Product Networks to address structured data in graph format, providing an effective solution for modeling and analyzing complex relationships in diverse domains. Their ability to handle graph-structured data and perform various tasks makes them a promising approach in the field of probabilistic graphical models.

The paper [4] introduces Graph-Structured Sum-Product Networks (Graph-SPNs) as a probabilistic solution for structured prediction tasks involving dynamic graphs expressing dependencies between latent variables. Unlike other approaches that impose strict constraints on variable interactions, GraphSPNs accommodate complex and varying-sized graph structures, which are common in real-world problems with noisy data. The research focuses on a specific application in robotics, where GraphSPNs play a crucial role in improving inference related to semantic place descriptions, particularly in the context of noisy topological relations within extensive office environments. The experiments demonstrate GraphSPNs' superiority over traditional methods relying on undirected graphical models. These results effectively clarify information within global semantic maps built from uncertain local data. The probabilistic nature of the model enables the inference of marginal distributions for unexplored places, detecting novel and incongruent spatial configurations based on known evidence. This work is valuable for similar research in the field of structured prediction and semantic mapping in robotics, offering insights into handling dynamic graph structures and noisy data to improve inference accuracy and model robustness.

2.4 Graph Neural Network (GNN)

Graph Neural Networks (GNNs) are a class of neural networks that have gained significant popularity in recent years due to their ability to effectively handle data with graph-like structures [24]. GNNs are specifically designed to process data represented as graphs, where nodes represent entities, and edges capture the relationships between these entities. They have demonstrated remarkable performance in tasks such as node classification, link prediction, and graph-level classification [25].

At their core, GNNs leverage a message-passing mechanism, where information is propagated through the graph's edges to update node representations [26]. This process allows nodes to gather and aggregate information from their neighboring nodes, enabling the network to capture local and global patterns within the graph. As information is passed through multiple layers of the network, nodes receive refined representations, taking into account both their local features and the collective information from the entire graph [27].

GNNs employ a series of layers, each consisting of two main steps: message aggregation and node update. In the message aggregation step, each node collects information from its neighbors using aggregation functions, such as summation or weighted averaging. In the node update step, the aggregated information is combined with the node's current representation to compute an updated representation [28]. This iterative process allows GNNs to iteratively refine node representations, capturing complex dependencies and patterns within the graph [29].

In the context of semantic place classification, GNNs offer a powerful approach to understanding and categorizing different locations or regions in the environment [30]. When applied to a graph representation of a spatial environment, where nodes represent places or regions, and edges signify spatial relationships, GNNs can effectively classify the semantic meaning of each place [31].

To accomplish semantic place classification, GNNs leverage their ability to incorporate both spatial and semantic information. For example, in a smart home setting, a GNN can learn to distinguish between various rooms, such as the kitchen, living room, and bedroom, based on their spatial relationships and the objects present in each room. By considering the context provided by neighboring places, GNNs can make informed decisions about the semantic labels of different locations.

The success of GNNs in semantic place classification can be attributed to their ability to model complex spatial dependencies among places. Traditional meth-

ods for place classification often rely on hand-crafted features or predefined spatial relationships, which may not capture the intricacies of real-world environments. GNNs, on the other hand, can automatically learn meaningful representations from the data, making them more adaptable and accurate in capturing the nuances of the environment.

Furthermore, GNNs are robust to variations and noisy data, making them well-suited for real-world scenarios where environmental conditions may change over time. They can effectively handle missing or incomplete information, enabling reliable place classification even in dynamic and unpredictable environments.

In summary, Graph Neural Networks are a potent tool for processing data with graph-like structures, offering a message-passing mechanism to capture complex dependencies among entities. In the domain of semantic place classification, GNNs excel at understanding and categorizing different locations based on their spatial relationships and context provided by neighboring places. Their ability to handle real-world variations and noisy data makes them highly valuable for autonomous systems and intelligent agents operating in dynamic environments. As research in GNNs continues to progress, their applications are likely to expand further, contributing to advancements in various fields, including robotics, urban planning, and smart infrastructure.

2.5 Libraries and Tools

2.5.1 Numpy

NumPy is a Python library [32] that facilitates the manipulation of large, multi-dimensional arrays and matrices, coupled with an extensive set of mathematical functions to process these arrays. Its widespread use in data science, scientific computing, and machine learning is attributed to its ability to efficiently manage numerical computations. With an intuitive interface, NumPy is invaluable for researchers, data analysts, and developers working with numerical data in Python, simplifying complex mathematical operations on arrays with ease.

2.5.2 pandas

Pandas is a well-known Python library extensively utilized for data manipulation and analysis [33]. Its DataFrame and Series data structures enable seamless handling and manipulation of vast datasets. With an abundance of built-in functions

and utilities, Pandas simplifies tasks like data cleaning, transformation, grouping, and aggregation. The library's integration with other tools, including NumPy, empowers users to preprocess and analyze data efficiently in data science projects. Its intuitive and adaptable interface makes Pandas a preferred choice for data professionals and analysts working with structured data in Python [33].

2.5.3 libspn

With "libSPN," users can build SPNs and perform various operations, including structure learning, parameter learning, and probabilistic inference. The library offers an intuitive interface to create and manipulate SPNs, making it accessible to researchers, data scientists, and machine learning practitioners.

2.5.4 libspn - keras

LibSPN Keras [34] is a library for constructing and training Sum-Product Networks. By leveraging the Keras framework with a TensorFlow backend, it offers both ease-of-use and scalability. Whereas the previously available libspn focused on scalability, libspn-keras offers scalability and a straightforward Keras-compatible interface.

2.5.5 Qtorch - Geometric, sparse, scatter

Qtorch is a comprehensive library that specializes in low-precision quantization [35] for PyTorch, enabling developers to reduce memory usage and computation time while maintaining model accuracy. By supporting various quantization techniques, Qtorch empowers users to optimize the trade-off between performance and precision. It offers seamless integration with existing PyTorch models and allows quantization-aware training for improved performance without the need for extensive retraining.

Qtorch Sparse and Scatter [35] is an extension of the Qtorch library, specifically tailored to handle sparse and scattered computations in low-precision quantization tasks. With a focus on memory and computation efficiency for sparse data, it proves valuable for applications dealing with large-scale sparse tensors. Leveraging sparse and scattered computations, this extension efficiently quantizes and processes sparse tensors while preserving accuracy, making it ideal for resource-constrained devices and large datasets.

Qtorch Geometric [35] is another specialized extension of the Qtorch library,

catering to low-precision quantization for graph-based models, particularly those represented as graphs. By integrating seamlessly with PyTorch Geometric, it enables efficient quantization of graph neural networks, leveraging various quantization methods for optimal performance. Qtorch Geometric ensures that memory usage and computation time are minimized, allowing users to maintain model accuracy and achieve efficient deep learning tasks with graph-structured data.

In summary, the combined capabilities of Qtorch, Qtorch Sparse and Scatter, and Qtorch Geometric present a powerful suite of tools for low-precision quantization in PyTorch. They collectively address diverse scenarios, optimizing memory consumption, computation time, and model accuracy across different data types, including dense tensors, sparse tensors, and graph-structured data. As a result, developers can unlock more efficient and scalable deep learning on various platforms, ranging from resource-constrained devices to complex graph-based models [35].

2.5.6 torch-geometric

torch-geometric is a popular PyTorch-based library specifically designed for deep learning on graph-structured data [36]. It provides a wide range of tools and functionalities to work with graphs and perform various graph-based machine learning tasks, including node classification, link prediction, graph classification, and graph generation.

With torch-geometric, users can efficiently handle large-scale graph datasets using specialized data structures, such as Graph Neural Networks (GNNs) and Message Passing Neural Networks (MPNNs). The library also offers various graph data augmentation techniques, graph preprocessing utilities, and visualization tools to facilitate the exploration and analysis of graph-structured data.

Graph Neural Networks have gained significant attention in recent years for their ability to model complex dependencies and interactions within graph data. "torch-geometric" makes it easier for researchers, data scientists, and developers to implement and experiment with GNN architectures for a wide range of applications.

Additionally, "torch-geometric" has a growing ecosystem of community-contributed models and benchmark datasets, making it a valuable resource for researchers and practitioners in the field of graph-based deep learning [36].

2.5.7 NetworkX

NetworkX is a versatile and widely used Python library designed to handle complex network analysis and manipulation tasks, making it an invaluable asset in the field of graph-based machine learning, including Graph Neural Networks (GNNs) and Graph Attention Networks (GATs). With its user-friendly interface and comprehensive set of functionalities, NetworkX facilitates the creation, modification, and exploration of graphs, enabling researchers and practitioners to preprocess and prepare graph data for subsequent analysis [37].

One of NetworkX's key strengths lies in its capability to handle diverse types of graphs [37], whether directed or undirected, weighted or unweighted. It allows nodes and edges to possess attributes, which aligns well with the rich nature of data encountered in real-world scenarios. This feature is essential when preprocessing graph data for GNNs and GATs, as it enables the incorporation of pertinent node and edge features, thereby enhancing the models' ability to capture nuanced relationships

CHAPTER 3

Dataset and Data Preparation

This chapter introduces to the datasets we used for experiments. We also discussed about data-preparation and preprocessing needed. This chapter is important for the reader as it will give the insight about structure of 2D data and its graphical representation. This chapter also represents how important to understand how different models and experiments needs different data preprocessing if in case someone in future wants to do more research about this topic.

3.1 Dataset

In our research, we utilized the COLD-TopoMaps Dataset [38], which comprises topological maps gathered during the robot's exploration of the environment. These maps are represented as undirected graphs, also known as "topological graphs," where nodes represent accessible places, and edges signify navigability between places. Moreover, each place in the graph is annotated with a specific semantic place category, such as corridor, kitchen, or doorway. The dataset's inclusion of both spatial information in the form of graphs and semantic labels for places provides a valuable resource for our research on semantic mapping and place classification in robotics. By leveraging this dataset, we aimed to enhance the robot's understanding of its surroundings and improve its decision-making capabilities based on both spatial and semantic cues.

So our dataset is basically contains two dimensional information ,it contains x and y coordinates of localities and there eight views classifications as well.

For constructing topological graphs from a stream of localized robot poses on an occupancy grid map, generated using SLAM (Simultaneous Localization and Mapping). The process involves utilizing the Robot Operating System (ROS) framework, where robot poses are continuously published to one topic, and the occupancy grid map is published to another topic using map-server. This approach allows the method to work in real-time as the robot explores a new envi-

ronment.

However, for the data collection process used in our research, we obtained the SLAM map separately before constructing the topological graphs. This decision was made to ensure data consistency and to focus on the construction and analysis of topological graphs without the real-time constraints of online exploration.

Furthermore, for our dataset, we used canonical robot poses from COLD-Meta, which were obtained through AMCL (Adaptive Monte Carlo Localization). These canonical poses provide accurate and reliable robot localization information, serving as essential inputs for building precise topological graphs.

The described acquisition procedure highlights the robustness and flexibility of our method, as it can efficiently work in real-time when applied online during exploration, but also accommodates offline data collection for more comprehensive analysis and research. By leveraging canonical robot poses from COLD-Meta, we ensure high-quality data for constructing topological graphs, enabling us to study and evaluate semantic place classification and mapping with confidence.

The Figure 3.1 represents Datasets 8 views and there directions. Here 0 to 7 represents 8 different views in different directions. For instance 0 represents north east direction with range 0° to 45° from north, and similarly for the remaining views. Figure 3.2 represents Datasets labels. Here we are observing the geometric view 10 different classes. Figure 3.3 represents an example of DataSet graphical representation taken from [1].

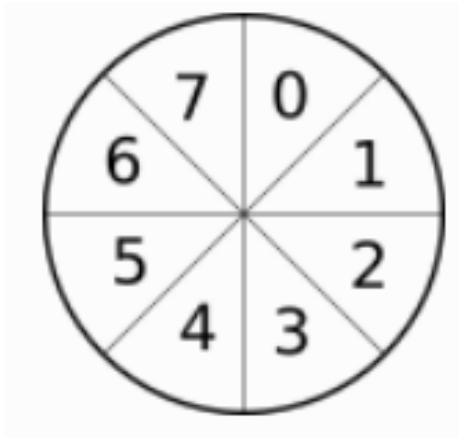


Figure 3.1: Dataset view graph [1].

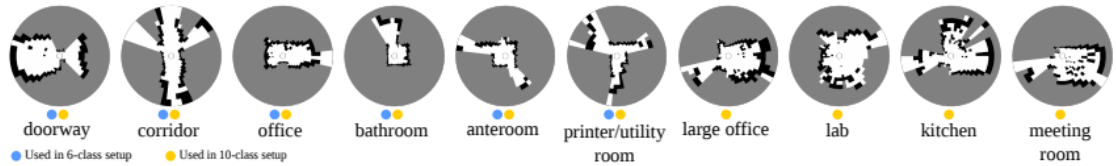


Figure 3.2: Dataset labels [1].

3.2 Dataset Preparation and Preprocessing

In our research, we focused on the application of Graph Neural Networks (GNNs) to our acquired dataset, which contains valuable graphical information about the environment of the floor and rooms. To leverage the power of GNNs, it was crucial to preprocess the dataset and ensure its compatibility with the GraphSPN libraries in both PyTorch and TensorFlow. This preprocessing step involved transforming the original graphical data into graph structures that can seamlessly integrate with the GNN frameworks.

The dataset consists of 118 graphs obtained from the exploration of three different buildings. To make it GNN-ready, we diligently converted each graph into a format suitable for analysis with GraphSPN libraries, carefully considering the nuances of each building’s layout and spatial relationships.

For the subsequent phase of our research, we aimed to perform semantic place classification at multiple levels of granularity. This required further preparation and preprocessing of the dataset. We needed to label the data differently for three levels: room level, floor level, and building level. Each level provided different insights into the semantic descriptions of places, contributing to a more comprehensive understanding of the environment.

In addition to labeling, we explored the benefits of graph partitioning to improve classification accuracy. By employing a graph partitioning algorithm, we divided the graphs into meaningful substructures, facilitating more accurate and context-aware place classification. We experimented with different partitioning values to find the optimum configuration that yielded the best prediction accuracies for each level of granularity.

Through these meticulous efforts, our dataset became well-suited for GNN applications, enabling robust and versatile semantic place classification across various levels of detail. The thoughtful preprocessing and partitioning steps played a pivotal role in enhancing the performance of GNNs in extracting and comprehending the semantic information embedded within the complex graph structures of our dataset.

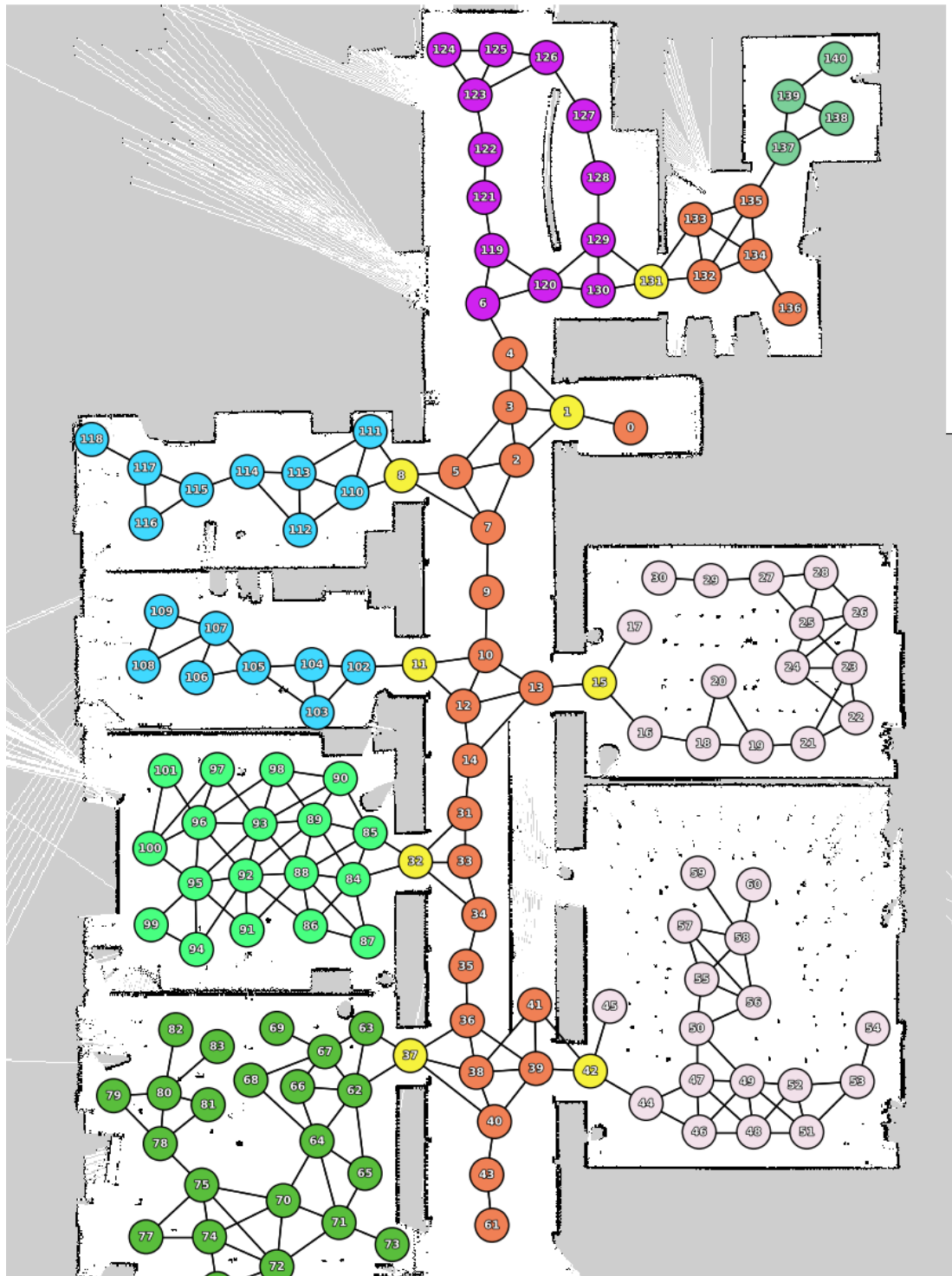


Figure 3.3: Graphical representation of COLD data [1].

Data preprocessing : Graph Neural Networks (GNNs), effective data preprocessing plays a pivotal role in optimizing model performance and predictive accuracy. One crucial step involves the careful construction of graph structures, encompassing the identification of nodes and edges, as well as their attributes. Additionally, data normalization and feature scaling are imperative to ensure uniformity and mitigate the impact of varying scales across different attributes. Furthermore, handling missing data and noise reduction strategies are essential to enhance the robustness of GNNs, as these models are sensitive to the quality of input data.

In the context of our work, the significance of data preprocessing was required and important. To apply GNNs effectively to our semantic mapping problem, we carefully curated and engineered our dataset. We organized our graph structures by identifying places as nodes and establishing edges representing the spatial relations between them. This entailed encoding not only the topological relationships but also the semantic attributes associated with each place. Subsequently, we normalized and scaled the attributes to ensure consistent inputs for the GNN model.

Furthermore, we also worked for handling noise and missing data. With rigorous cleaning and imputation techniques, we refined our dataset to reduce inconsistencies and enhance the reliability of our GNN-based predictions. By carefully executing these preprocessing steps, we laid the foundation for GNNs to effectively capture the intricate dependencies present in our semantic mapping data, thereby facilitating accurate and meaningful predictions.

When preparing data for GATs, special attention is given to the construction of the adjacency matrix. Unlike GNNs, GATs utilize a self-attention mechanism that assigns varying importance to different neighboring nodes. As a result, the adjacency matrix is normalized using a softmax function, ensuring that the attention weights sum up to unity for each node. This step facilitates GATs in discerning the most relevant connections, enhancing their ability to capture fine-grained spatial dependencies.

Furthermore, node feature preparation in GATs is distinctive. While GNNs often utilize shared learnable weights for aggregating features from neighboring nodes, GATs assign individual attention coefficients to each neighboring node during feature aggregation. This specialized attention mechanism endows GATs with the capability to focus on specific nodes while aggregating information, enabling them to capture intricate spatial relationships and semantic nuances.

In summary, data preprocessing for GAT models involves a nuanced approach,

including the normalization of adjacency matrices to accommodate the self-attention mechanism and the calculation of individual attention coefficients during feature aggregation. These refined steps not only enhance GATs' capacity to capture fine-grained spatial relationships and semantic patterns but also underline their potential as a powerful tool for advanced semantic mapping tasks, propelling them beyond the capabilities of traditional GNNs. This distinction positions GATs at the forefront of innovative solutions in the field of semantic mapping.

CHAPTER 4

Sum Product Network (SPN) Architecture

4.1 Implementation

After all the required data preprocessing techniques as discussed in above sections and some which are not discussed that are very common in almost all machine learning or deep learning problems such as data cleaning, dimensionality reduction, sampling and normalisation and data transformation , we proceed with steps below.

The next step was to design an algorithm for creating edge indexes out of dataset. We outline the process of constructing an edge index from our dataset's nodes and their respective neighboring node identifiers. The derived edge index forms a fundamental component in facilitating graph-based analyses, a cornerstone of our semantic mapping and classification endeavors.

The algorithm (Figure 4.1) commences by initializing an empty array called all-edges, formatted as a NumPy array with a shape of (0, 2). This array is designed to accommodate pairs of node identifiers, representing the edges of our graph.

Iterating through the dataset containing node information (nodes-df), our algorithm systematically ascertains the presence of neighboring node identifiers for each node. Through a series of conditional checks, we ensure that the neighboring node identifiers (nnID1 through nnID8) are considered for potential edge formation. If a valid neighboring node identifier is detected, an "edge pair" is created by juxtaposing the current node's identifier and its corresponding neighboring node identifier.

These "edge pairs" are subsequently integrated into the growing all-edges array using a vertical stacking mechanism. This strategic approach enables the gradual assembly of a comprehensive array of edges that characterize the connectivity between nodes within our graph.

The final step involves refining the collected edges to ensure uniqueness. Through the application of the NumPy function `np.unique()`, any duplicate edge entries are

effectively removed. This action further optimizes the integrity of the edge index, ensuring that each edge pair is represented only once.

The culmination of these operations results in the creation of the edge-index. This structured edge index is prepared for subsequent utilization in graph-oriented operations and analyses, which play a very important role in our semantic mapping classification.

In essence, the code snippet in figure 4.1 unveils a systematic approach to convert our dataset's node and neighboring node identifier information into a refined edge index. This pivotal effort significantly improves the precision of our graph-based analyses and strategically positions our research for the efficient utilization of graph neural networks to decipher the complexities of semantic mapping and classification within our dataset.

In our pursuit of advancing semantic mapping classification, we have embarked on the development of a localized Sum-Product Network (SPN) model that is custom-tailored to the intricate characteristics of our specific dataset. This endeavor involves harnessing the capabilities of the Keras library to construct an SPN architecture that efficiently captures and processes the multifaceted relationships inherent in our dataset, thereby enriching the accuracy and depth of our semantic place classification efforts.

The journey of building our localized SPN unfolds through a detailed and required series of steps, each of which contributes to the model's adaptability and efficacy in addressing the complexities of our semantic mapping classification task:

Data Normalization: The process commences with data normalization, a crucial preprocessing stage that brings our input data to a uniform scale. This normalization step not only optimizes the training process but also aligns the data for subsequent SPN layers.

Initiating with NormalLeaf Layer: Our SPN model takes its initial steps with the introduction of the NormalLeaf layer. This layer assumes a pivotal role in modeling the underlying distribution of our dataset's attributes. By introducing Gaussian distributions, the NormalLeaf layer adeptly handles the variations within the data. The strategic selection of parameters, such as the number of Gaussian components and the trainability of location parameters, ensures a robust alignment between our model and the distinct attributes of our dataset.

Sequences of DenseProduct and DenseSum Layers: The heart of our SPN architecture comprises an interplay of DenseProduct and DenseSum layers. This choreography of layers forms a dynamic ensemble that iteratively orchestrates the

intricate relationships within the dataset. The DenseProduct layers, marked by their multiplication of child node outputs, intricately weave together information. Conversely, the DenseSum layers exercise their prowess in summing the outputs of child nodes. This pattern of products and sums, delivered through a sequence of layers, encapsulates the hierarchical dependencies inherent in the dataset.

Culmination with RootSum Layer: The SPN narrative reaches its end with the RootSum layer. At this point, the model synthesizes the weighted logits emanating from the child nodes. This deep integration of information produces an output that embodies the model's prediction, effectively encapsulating the semantics of place classification.

The configuration, the number of Gaussian components, factors, and sums in each layer, is a deliberative endeavor that aligns with our dataset's complex attributes. This localized SPN model serves as an embodiment of our dedication to devising an architecture that harmonizes with the intrinsic complexities of semantic mapping classification. As we journey forward in our work, the subsequent chapters will delve into the model's training intricacies, performance evaluations, and the valuable insights through its application to the dataset.

Presented below figure 4.1 is a code snippet detailing our custom local SPN network. This network structure is designed while keeping the number of input and output parameters in mind. Other layers are designed with experiments, hit and trial methods for best accuracy and number of parameters. In figure 4.2 we can observe the flow diagram for the same.

```
sum_product_network = keras.Sequential([
    normalize,
    spnk.layers.NormalLeaf(
        num_components=2,
        location_trainable=True,
    ),
    spnk.layers.DenseProduct(num_factors=3),
    spnk.layers.DenseProduct(num_factors=3),
    spnk.layers.DenseSum(num_sums=10),
    spnk.layers.DenseProduct(num_factors=3),
    spnk.layers.DenseSum(num_sums=10),
    spnk.layers.DenseSum(num_sums=10),
    spnk.layers.RootSum(return_weighted_child_logits=True)
])
sum_product_network.summary()
```

Figure 4.1: Local SPN model.

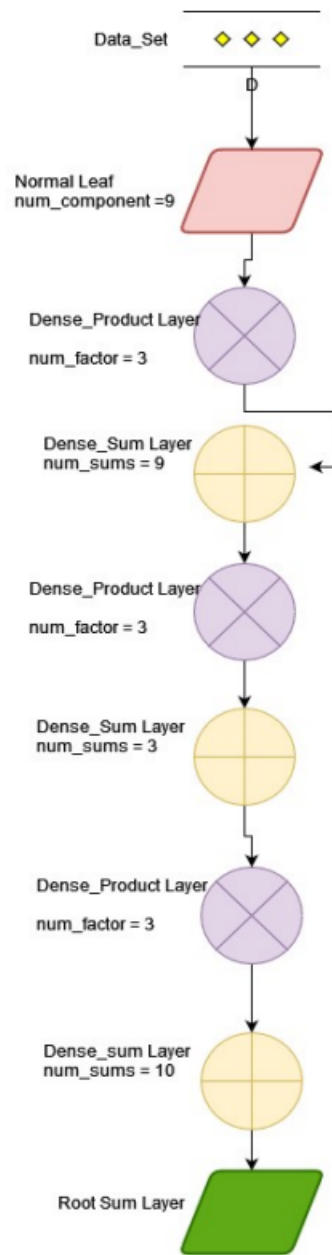


Figure 4.2: Illustrate flow diagram our local SPN Architecture for the code in depicted in Fig. 4.1.

4.2 Result

The outcomes obtained from our semantic place classification experiments employing the local Sum-Product Network (SPN) on distinct datasets, namely Cold Stockholm and Cold Sarbucken [38] yielded accuracy rates of 30.04% and 54.10% respectively. It is noteworthy, however, that these results exhibit a comparative lower accuracy when juxtaposed with contemporary state-of-the-art methodologies. These experiments were conducted within an environment characterized by a high degree of noise, underscoring the intricate challenges that confront semantic place classification.

Our engagement with local SPN networks surfaced the realization that a more refined and sophisticated approach is warranted. This imperative realization has set the stage for a comprehensive exploration, which we expound upon in the forthcoming chapter. A notable facet of this journey is the presence of multifaceted challenges. One such challenge stems from the deprecation of previously established libraries, rendering their utilization unfeasible. This predicament necessitated the exploration of alternative libraries and the creation of a novel, bespoke setup from the ground up.

It is important to underscore the challenges we encountered while striving to replicate the original work, as detailed in [1]. There was lack of implementation details in the paper and very less information was available for architecture used. This absence significantly heightened the complexity of conducting our simulations, carefully exploration and inventive solutions to surmount the problems posed by this lack of concrete guidance.

The upcoming chapter delves into our innovative approach, born out of the recognized limitations and challenges. Utilizing a systematic methodology and solutions, we set out on a transformative journey to enhance and refine our semantic place classification framework. By systematically addressing the shortcomings highlighted by previous experiments and thoughtfully incorporating improvements, we aim to introduce a new dimension of improved accuracy and comprehension within the domain of semantic mapping.

The forthcoming chapter delves deeper into our novel approach that stems from the identified limitations and hurdles encountered. Through a systematic methodology and innovative solutions, we worked to refine our semantic place classification framework. By addressing the issues raised by the earlier simulations and methodically incorporating improvements, we endeavor to usher in a new realm of enhanced accuracy and understanding in the domain of semantic

mapping. Table 4.1 demonstrates the results obtained in terms of accuracy.

Model	Accuracy
Local SPN (Stolckholm) (This Work)	30.04%
Local SPN (Saarbrucken) (This Work)	54.10%
Local SPN (Kaiyu Zheng)	79.06%
TopoNet using Graph SPN (Kaiyu Zheng)	80.14%

Table 4.1: Results comparisons for different methods with our method

While our model did not perform as good as state of the art if we consider accuracy, but it is crucial to note that our model’s lightweight nature contributed to improved performance in terms of both training and testing time complexities. The significance of a lightweight model cannot be understated, particularly when considering its pivotal role in achieving real-time semantic mapping. In scenarios where real-time operations are imperative, the efficiency gained from a lightweight model becomes a critical advantage. There was no data available on model sizes to compare the light weight of the model but we inference time data using that we evaluated the performance. That is shown in upcoming chapter.

This helped us for our next experiment where our target was not just to improve the accuracy but also to also to maintain our model light-weight in nature.

CHAPTER 5

Graph Neural Network (GNN)

This chapter starts with brief theory important understand GNN. Then further it covers the implementation and simulations for GNN based semantic place classification, also simulated and developing proof of concept for multilevel predictions using GNNs. At the end of this chapter we take a glance at final results of

5.1 Introduction

This section covers theory for GNN and how it can be used for different levels of predictions in a graph.

Graph Neural Networks (GNNs) have emerged as a pioneering innovation in the field of machine learning, specifically tailored to tackle data structured as graphs. Their inception was motivated by the recognition that numerous real-world problems, such as social networks, molecular structures, and recommendation systems, inherently exhibit graph-like relationships. Standard neural networks, while adept at processing structured data like images and sequences, lack the inherent ability to handle graph-structured data. GNNs were introduced to bridge this gap by extending the deep learning paradigm to graph-based structures.

The genesis of GNNs can be traced back to the early 2010s, but their broader prominence gained momentum in the latter half of the decade. The pivotal paper [24] marked a game changing moment in GNN research. This work introduced the concept of graph convolutions, enabling neural networks to propagate information across graph nodes effectively. Consequently, GNNs opened up avenues for various applications, ranging from node classification and link prediction to recommendation systems and molecular chemistry [39].

From a theoretical and mathematical perspective, GNNs leverage graph convolutional operations to aggregate and propagate information across neighboring nodes. This process encapsulates the essence of the underlying graph structure

and enables nodes to acquire insights from their local context. Mathematically, GNNs iteratively update node representations by aggregating information from adjacent nodes and then applying neural transformations. This dynamic process effectively captures higher-order relationships, allowing GNNs to learn complex graph patterns [40].

Furthermore, GNNs can be understood through the lens of message passing. At each layer, nodes aggregate and transform messages from their neighbors, enabling the network to iteratively refine node representations. This process is formalized through mathematical equations that encapsulate the aggregation and transformation functions.

Graph Neural Networks (GNNs) rely on three pivotal matrices: the adjacency matrix, the degree matrix, and the feature matrix. These matrices underpin the mathematical operations that enable GNNs to capture intricate graph relationships and perform meaningful computations on graph-structured data.

Adjacency matrix is defined as fundamental representation of graph's topology, signifying the connections between nodes. Its significance lies in capturing the relational structure and neighborhood interactions within the graph. In essence, each entry $A[i][j]$ reflects whether nodes i and j are directly connected or adjacent. This connectivity insight is pivotal for GNNs, as it guides information propagation and diffusion across the graph. Figure 5.1 illustrates an example for adjacency matrix.

Adjacency Matrix (A)

	1	2	3	4	5
1	0	1	0	0	0
2	1	0	1	1	0
3	0	1	0	1	1
4	0	1	1	0	1
5	0	0	1	1	0

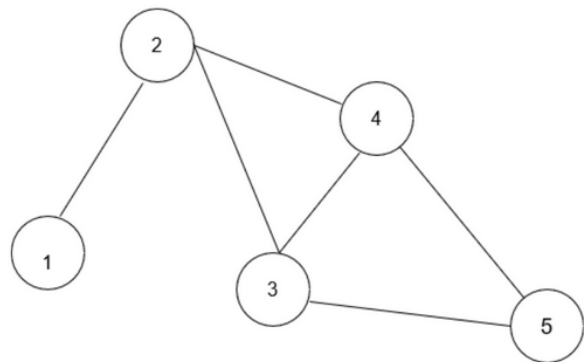


Figure 5.1: Adjacency matrix.

The degree matrix embodies the local significance of nodes within their respective neighborhoods. It quantifies the "importance" of each node by summing the number of its connections. This information aids GNNs in assigning varying

weights to nodes during information propagation. Nodes with higher degrees may contribute more to their neighbors' updates, while nodes with lower degrees might be more influenced by their surroundings.

In a road network, nodes represent intersections, and edges depict roads. The degree of an intersection corresponds to the number of roads leading to it. GNNs use the degree matrix to give more weight to intersections with multiple roads, acknowledging their central role in disseminating traffic information. Similarly, in citation networks, nodes corresponding to frequently cited papers are "central," and the degree matrix acknowledges their influential role. Figure 5.2 illustrates an example for degree matrix.

Degree Matrix (D)

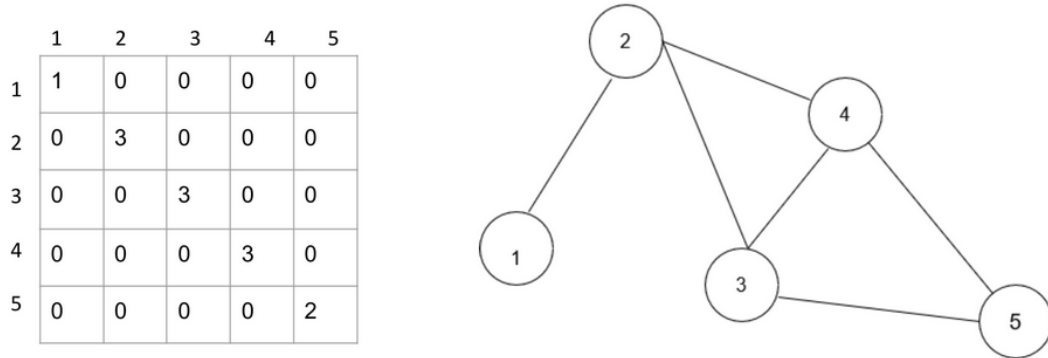


Figure 5.2: Degree matrix.

The Laplacian matrix, denoted as L , is a symmetric matrix derived from the adjacency matrix A and the degree matrix D of a graph G . Specifically, it is defined as

$$L = D - A$$

. The Laplacian matrix captures the graph's topology and its inherent connectivity patterns. It embodies the concept of "smoothness" within the graph, indicating how nodes are interconnected and how their attributes or values might change across the graph's structure.

The Laplacian matrix has several distinct forms, including the normalized Laplacian, the combinatorial Laplacian, and the random walk Laplacian. Each variant serves a specific purpose, offering unique insights into different aspects of the graph's behavior. The normalized Laplacian, for instance, factors in the degree distribution, providing a normalized perspective on node relationships.

In the context of GNNs, the Laplacian matrix assumes a central role in spec-

tral graph theory. It enables the formulation of graph convolutions and message-passing mechanisms that facilitate information diffusion across nodes. The eigenvectors and eigenvalues of the Laplacian matrix hold critical information about the graph’s structure, aiding in tasks such as community detection, graph clustering, and, crucially, the design of graph convolutional layers in GNN architectures. Figure 5.3 illustrates an example for Laplacian Matrix.

Laplacian Matrix (L)

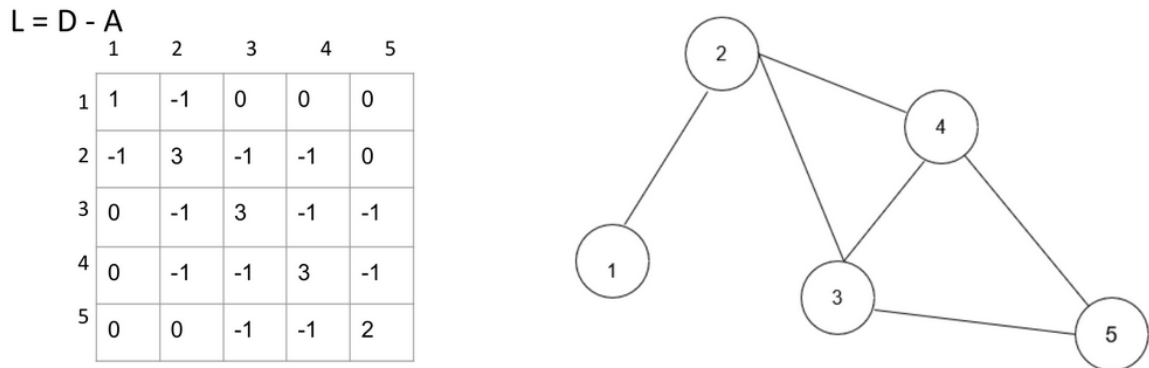


Figure 5.3: Laplacian matrix.

Collectively, these matrices create a holistic foundation for GNN-based semantic place classification. The adjacency matrix captures the spatial relationships and transitions between places, the degree matrix offers insights into the local significance of places, and the feature matrix imparts descriptive attributes. By employing GNN operations that utilize these matrices, the network gains the ability to learn complex patterns within the semantic map, enabling it to accurately classify places based on their semantic characteristics. This multi-faceted approach leverages both the graph structure and place attributes, making it highly effective for semantic place classification in autonomous systems.

In GNN we can make the prediction model work with 3 different levels. One in Node Level prediction, the second one is edge level prediction and the third one is Graph level Prediction. we will look in detail what does this mean and in upcoming section we have discussed this with relation to our work [41].

Graph Level Prediction: Comprehensive Global Understanding : In the realm of graph level prediction, Graph Neural Networks (GNNs) ascend to a pivotal position in comprehending the holistic essence of an entire graph. This level pertains to graph classification tasks, where GNNs encapsulate the overarching attributes of intricate graph structures and distill them into categorical assignments. This

tier of prediction holds significance in diverse domains. For instance, in the realm of molecular compounds, GNNs grasp the collective chemical composition, enabling classification based on shared properties. Similarly, GNNs navigate social networks, deciphering interaction patterns to categorize networks by inherent connectivity dynamics. This elevated vantage point showcases GNNs' proficiency in interpreting complex global attributes, transcending node-level details, and revealing overarching patterns.

Node Level Prediction: Unveiling Local Dynamics : Transitioning to the node level, GNNs pivot towards unraveling the intricate dynamics of individual nodes within a graph. At this stratum, GNNs interlace intrinsic node attributes with nuanced interactions among neighboring nodes. The outcome of this synthesis is manifested in node classification tasks. This dynamic capability allows GNNs to assign categorical labels to nodes, grounded in both their intrinsic attributes and their contextualized interactions. This functionality finds applications across diverse contexts, from recommender systems, where nodes signify products, to sentiment analysis where nodes represent textual content. GNNs flourish in deciphering the intricate relationships among individual nodes, reinforcing their adaptability in revealing granular insights.

Edge Level Prediction: Discerning Implicit Relationships : Progressing to edge level prediction, GNNs meticulously unveil the latent semantics embedded within edges that interconnect nodes. This intricate layer surfaces in tasks like link prediction, wherein GNNs foresee the probability of connections forming between nodes. The implications span widely, from forecasting the evolution of social network connections to decoding the regulatory interactions among genetic elements. By scrutinizing edges, GNNs unearth concealed relationships, heightening their predictive efficacy in scenarios marked by evolving connections. This aspect underscores GNNs' ability to discern implicit links, contributing to a holistic understanding of graph dynamics.

So, the hierarchical tiers of GNN predictions embody a multifaceted framework. At the graph, node, and edge levels, GNNs harness distinct dimensions of insights. Graph level predictions facilitate a panoramic overview, vital for global classifications. Node level predictions dive into the intricacies of individual nodes, revealing localized dynamics. Edge level predictions unearth concealed connections, enriching predictions with nuanced relationships. This stratification empowers GNNs to traverse multidimensional insights, positioning them as versatile tools across a range of data-driven domains.

Graph Attention Networks (GATs) represent a pioneering advancement in

the domain of Graph Neural Networks (GNNs), forging a pathway towards enhancing predictive capabilities within graph-structured data. At the heart of GAT lies the notion of attention mechanisms, which enable the network to dynamically weigh the importance of neighboring nodes while propagating information. This transformative feature augments GAT's ability to adaptively allocate attention, capturing intricate relationships within the graph. By introducing attention coefficients that are learned through training, GATs transcend the conventional fixed-weight aggregations, unraveling localized nuances with precision [25].

The crux of GAT's innovation rests upon the architecture's adeptness at performing neighbor-specific feature aggregation. This is realized through self-learned attention coefficients, as each node's interactions with its neighbors are meticulously assessed and weighed based on the network's understanding. Such a dynamic attention mechanism is particularly pivotal in scenarios where different nodes have varying influence levels on a focal node's attributes. This adaptability not only refines the predictive accuracy of GATs but also reinforces their capacity to unearth intricate patterns that might be obfuscated using conventional aggregation strategies. By harnessing attention-based mechanisms, GATs herald a new era in GNNs, amplifying their capacity to decode complex relationships within graph data, making them an invaluable asset in domains ranging from social network analysis to recommendation systems.

5.2 Implementation and Simulations

5.2.1 GNN Based Local Place Semantic Classification

The problem showcased its most significant outcomes in the seminal work [2] around 2018. During this period, Graph Neural Networks (GNNs) were in their early experimentation stages and had yet to gain widespread recognition. Additionally, GNNs were in their initial development phase and were not used so much widely but now we have good amount of resources and libraries to use them. Importantly, there were no dedicated neural network architectures specially designed for handling graph-based datasets at that time.

Given this backdrop and with an understanding of our dataset's foundational graph-oriented nature, we decided to investigate the potential of Graph Neural Networks. This exploration was motivated by the alignment of two key factors: the unique characteristics of our graph-structured dataset and the emerging significance of GNNs in the field of machine learning.

This convergence created an opportune context to delve into employing Graph Neural Networks. Our interest stemmed from the synergy between the pioneering state of GNNs and our dataset’s inherent structure. Our objective was to bridge this gap by utilizing GNNs capabilities to gain novel insights and solutions for our specific problem domain. This alignment, marked by the interplay of a transformative period in machine learning and the foundational attributes of graph-based data, motivated our decision to embark on this exploration, aiming to contribute to the evolving landscape of research.

Our model architecture unveils the complexities of the Complex Graph Convolutional Network (ComplexGCN), designed to address the graph-structured data within the domain of semantic place classification. With a foundation rooted in enhancing our model’s capacity to discern intricate relationships within complex datasets, the ComplexGCN architecture encompasses a layered composition. Figure 5.4 gives basic block diagram illustration for the GNN architecture we used.

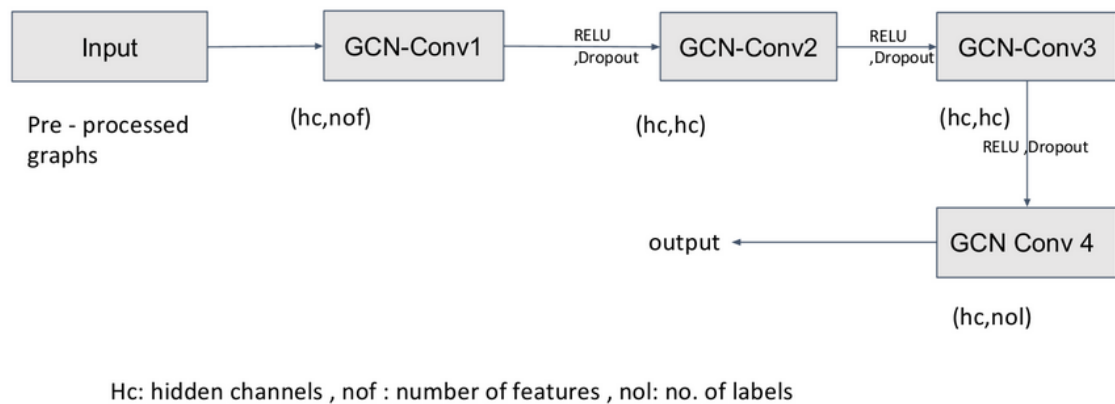


Figure 5.4: GNN architecture.

Embarking on an exploration of architectural complexities, we traverse a hierarchy of four layers that forms the foundation of our approach. This hierarchical depth facilitates the nuanced understanding of graph-embedded attributes, underscoring our commitment to cultivating profound insights. The stratified composition, fortified by non-linear activation functions, embodies our intent to foster deep learning within the graph context.

Central to our design architecture is latent feature abstraction, driven by the parameter C. This latent feature encapsulation serves as a vessel for capturing hidden insights residing within intricate graph interconnections. The parametric value of C orchestrates a judicious equilibrium between information richness and

computational efficiency, thus enabling the extraction of latent semantics.

Transcending architectural dimensions, our Complex GCN architecture finds its ultimate purpose in contextual semantic inference. Our journey involves unraveling semantically coherent patterns within spatial constructs. The hierarchical traversal through four graph layers inherently resonates with the quest for deciphering contextual semantics, enriching the model’s capacity to unearth meaningful spatial insights.

As we navigate towards optimal efficacy, we delve into the calculation of hyperparameter optimization. In alignment with the exigencies of location-based semantic place classification, we subject our architecture to rigorous exploration across 42 diverse graph instances, symbolizing unique locations. This comprehensive experimentation offers insights into the adaptability of ComplexGCNs in accommodating diverse spatial relationships, enhancing their robustness in real-world scenarios.

In summation, the architecture of Graph Convolution Networks emerges as a conduit for unraveling intricate semantic place classification. Rooted in hierarchical intricacies, latent feature abstraction, and context-driven inference, our architecture aligns seamlessly with the pursuit of gleaning meaningful insights from complex spatial constructs.

5.2.2 Multi Level Prediction Using GNNs

After completing the aforementioned experiment, we transitioned to our subsequent set of trials wherein we deployed our Graph Neural Network (GNN) across all three prediction levels. Within this context, our node-level prediction corresponded to localized places within a given floor. To address edge-level prediction, we strategically partitioned the edge data, aligning it with relationships between local places within a floor and facilitating predictions related to floor transitions. Finally, our GNN was applied for graph-level prediction to classify both individual floors and entire buildings. This comprehensive approach encompassed the utilization of datasets from all three buildings within the COLD dataset repository.

The objective behind these endeavors was to establish a proof of concept for multi-level predictions enabled by GNNs. Despite working with a limited dataset, our aim was to showcase the viability of utilizing GNNs for orchestrating predictions spanning different granularities. This strategic implementation led us to attain reasonable outcomes. This pioneering exploration underscores the potential of GNNs in addressing multi-level predictions and offers insights into their

efficacy even in scenarios marked by constrained datasets.

In this phase of our research, we worked with a dataset encompassing three distinct buildings in the dataset. Each of these buildings comprised four floors, and within each floor, we encountered approximately 8 to 10 distinct classes. In aggregate, our dataset encapsulated a total of 118 graphs, each graph representing a unique combination of building, floor, and class. This extensive dataset enabled us to conduct a comprehensive exploration of multi-level predictions using Graph Neural Networks (GNNs). By encompassing a diverse range of buildings, floors, and classes, our dataset laid the foundation for the empirical analysis and validation of our proposed methodology.

In our pursuit of achieving optimal performance across all three levels of prediction, we recognized the necessity for tailoring distinct architectures to cater to the nuances of each level. Notably, for both node-level and edge-level predictions, the model structure expounded in the earlier section proved to be well-suited. The modifications we made were primarily confined to the output layers, aligning them with the requisite number of classes corresponding to each prediction tier.

However, when it came to graph-level prediction, a more refined approach was warranted. In this context, we needed to redefine the architecture of the Graph Neural Network (GNN), particularly focusing on the configuration of hidden layers. After a series of simulations and thorough evaluations, we identified that a GNN model comprising a total of three hidden layers yielded the most optimal outcomes for floor prediction. This adjustment allowed us to capture the intricate patterns inherent in the data distribution of floors. Conversely, for building prediction, our findings indicated that a GNN architecture with two hidden layers demonstrated superior performance. This intricate calibration of the model's depth was instrumental in harnessing the distinctive characteristics of our dataset, resulting in the refined predictive accuracy observed in the context of building-level classification.

In addition to the aforementioned simulation, a final investigation was undertaken utilizing Graph Attention Networks (GATs) with a comparable architecture. The objective of this experiment was to assess the performance of GATs against GNNs in the context of our specific dataset and task. Notably, the results indicated that GNNs exhibited a higher level of efficacy for our dataset, outperforming GATs. However, it's pertinent to acknowledge that the full potential of GATs may not have been fully realized due to the nature of our dataset.

Subsequent considerations underscore the potential of GATs in handling more

intricate and diverse datasets. GATs are inherently designed to excel in scenarios characterized by complex interrelationships and substantial variations. It is conceivable that GATs could exhibit enhanced performance when confronted with datasets that pose greater challenges, encompassing diverse features and intricate spatial dependencies. Further exploration is warranted to ascertain the full spectrum of GATs’ capabilities, especially when grappling with more intricate and variegated data, a domain where their unique architecture holds promise for yielding even more substantial benefits.

5.3 Results

In the course of our study, an exhaustive analysis and comparison of outcomes were undertaken, focusing on the outcomes obtained through our Graph Neural Network (GNN) and Sum-Product Network (SPN) models, juxtaposed against the backdrop of prior research. Through this rigorous evaluation, we arrived at a notable observation: GNNs not only exhibited superior performance relative to SPN models but also demonstrated a remarkable capacity to surmount challenges intrinsic to graph-based data.

Importantly, our GNN models demonstrated significant progress compared to our previous simulations using SPN-based models, especially regarding the time it took to predict in each test scenario that we will discuss in table 5.2. This positive outcome aligns well with the results presented in [1], confirming the effectiveness of our GNN-based approach. It’s essential to note that while our GNN-based models improved accuracy compared to local SPN simulations, we acknowledge that the state-of-the-art results cited in [1] remain superior, indicating room for further enhancement.

Let us do the analysis of results with terms of prediction accuracy. The Table below represents comparative evaluation of our models with respect previous work for node-level or local place level classification.

Model	Accuracy
Local SPN (Stolkholm) (This Work)	30.04%
Local SPN (Saarbrucken) (This Work)	54.10%
Local SPN (Kaiyu Zheng)	79.06%
GNN (Overall)(This Work)	70.15%
TopoNet using Graph SPN (Kaiyu Zheng)	80.14%

Table 5.1: Results comparisons for different methods with our method.

Our Model and work was behind the TopoNet model in terms of accuracy

score but due to lighter weight models, our both models had better performance when it comes to deployment time.

In order to compare deployment time / Inference Time, ideally we should run them both on same system but due to unavailability of libraries due to non maintenance, we cannot recreate those. so we used Normalized Deployment Time (NDT) to compare the inference time.

$$NDT = \frac{DT \times \left(\frac{C}{CB}\right)}{DT}$$

Where:

NDT represents the normalized deployment time for System 1. compared to the baseline system.

DT signifies the actual deployment time on System 1.

C corresponds to a computational measure that considers System 1's hardware specifications.

CB indicates the computational measure for the baseline system.

DT denotes the deployment time on the baseline system.

The Table below demonstrate the inference time per 105 sample size for the models.

Model	Inference Time
Local SPN (This Work)	0.28 sec
Local SPN (Kaiyu Zheng)	0.65 sec
TopoNet(Kaiyu Zheng)	0.36 sec
GNN (Overall)(This Work)	0.19 sec

Table 5.2: Inference time results comparisons for different methods with our method.

When it comes to performance in robotics and autonomous systems both accuracy and inference time are very important and in many scenarios we need a score that can compare both simultaneously. Although we realise that weights of each component will vary according to requirements. There may be certain cutoffs in some scenarios for both parameters. So we have done a comparative analysis in two ways.

First we have calculate a performance score based on the results obtained. The formulae used for calculation is

$$PerformanceScore = \frac{Accuracy}{InferenceTimeper}$$

The table 5.3 below shows that our GNN model gave best performance according to above formulae.

Model	Performance Score
Local SPN (This Work)	107.28
Local SPN (Kaiyu Zheng)	121.63
TopoNet(Kaiyu Zheng)	222.61
GNN (Overall)(This Work)	369.21

Table 5.3: Performance Score results comparisons for different methods with our method.

In many scenarios there may be different requirements and cutoff values for both the parameters. So for visualisation performance we have also used trade-off curve plotting accuracy vs. inference time for multiple models in Figure 5.5 . Models that are close to the top-left corner of the plot are typically preferred, as they offer the best trade-off.

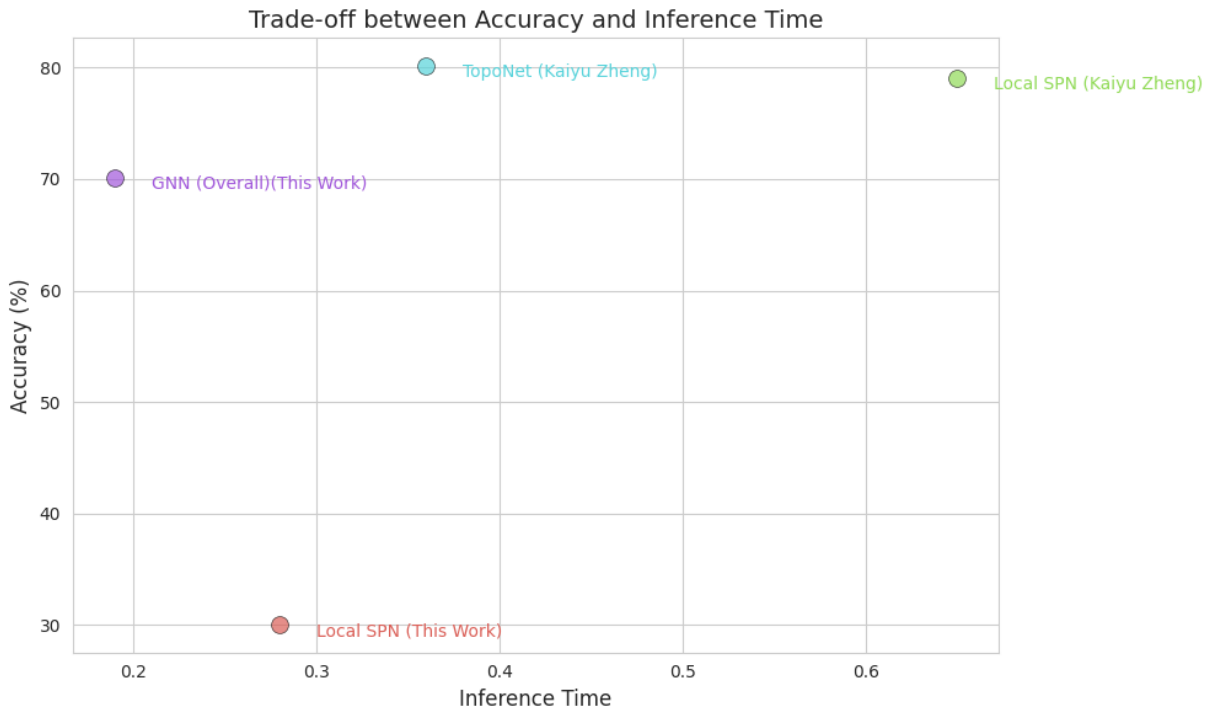


Figure 5.5: Trade off curve for accuracy vs inference time.

Now Let us look at the results obtained for multi level prediction based on GNN and GAT. The results are discussed for different level and the accuracy score is taken to be average for all three datasets for comparison. The experiment was performed in such a way that results of lower level are processed as input for higher level prediction with only exception with building level prediction, where

due to non existence of any graph for complete building we needed to train it different way as discussed earlier. Table 5.3 represents the results.

Model	Accuracy
GNN (Floor Level)	48.82%
GNN (Building Level)	62.22%
GNN (Edge Level)	73.46%
GAT (Floor Level)	46.62%
GAT (Building Level)	59.14%
GAT (Edge Level)	63.64%

Table 5.4: Results Comparisons for GAT and GNN.

CHAPTER 6

Conclusion and Future work

Throughout our discussions, we delved into various areas of machine learning, deep learning, and their applications in semantic mapping and place classification within autonomous systems. We embarked on a journey through topics such as Sum-Product Networks (SPNs), Semantic Mapping, Graph Neural Networks (GNNs), and Graph Attention Networks (GATs).

We conducted extensive simulations, comparing GNN and SPN-based models, highlighting the potency of GNNs in dealing with graph data. From our simulations we conclude that GNN gave better accuracy compared to local SPN although we were little behind as compares to state of the art models as discussed. Also the consideration of inference time added a practical dimension to our investigation, prompting us to introduce the concept of Normalized Deployment Time (NDT) to enable fair cross-system model comparisons. We also provided a proof of concept for multilevel semantic classification using GNN based models.

In conclusion, our exploration underscored the pivotal role of deep learning techniques in enhancing semantic mapping and place classification within autonomous systems. We not only grasped the theoretical foundations but also embarked on simulations, navigating the intricate landscape of various models, architectures, and datasets. As we close our work, the vast potential of these methods in shaping the future of autonomous systems becomes evident, paving the way for enhanced understanding, navigation, and decision-making in complex environments.

The future scope of our work lies with extending the implementation for Real time. Transitioning from offline experiments to real-time deployment is a critical step. Optimizing model inference times, exploring hardware accelerators, and ensuring compatibility with real-world constraints are essential considerations.

Considering the increasing emphasis on explainability in AI, another avenue is the exploration of interpretable and explainable models for semantic mapping. Creating models that provide insights into their decision-making processes can

be invaluable for real-world deployment and adoption.

Lastly, the integration of emerging technologies like augmented reality and virtual reality could offer novel dimensions to our research. Developing immersive visualization tools that interact with our semantic mapping models could have profound implications for applications in architecture, urban planning, and navigation.

References

- [1] Kaiyu Zheng and Andrzej Pronobis. From pixels to buildings: End-to-end probabilistic deep networks for large-scale semantic mapping. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3511–3518. IEEE, 2019.
- [2] R. Gens and P. Domingos. Learning the structure of sum-product networks. In *Proc. of ICML*, 2013.
- [3] M. Amer and S. Todorovic. Sum product networks for activity recognition. *Transactions on Pattern Analysis and Machine Intelligence*, 38(4), 2015.
- [4] Kaiyu Zheng, Andrzej Pronobis, and R. P. N. Rao. Learning graph-structured sum-product networks for probabilistic semantic maps. In *Proc. of AAAI*, 2018.
- [5] Daphne Koller and Nir Friedman. Probabilistic graphical models: Principles and techniques. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305, 2009.
- [6] Martin J Wainwright and Michael I Jordan. Graphical models, exponential families, and variational inference. In *Foundations and Trends® in Machine Learning*, volume 1, pages 1–305. Now Publishers Inc., 2008.
- [7] David Heckerman, Dan Geiger, and David M Chickering. Learning bayesian networks: The combination of knowledge and statistical data. In *Machine learning*, volume 20, pages 197–243. Springer, 1995.
- [8] A. Howard et al. Probabilistic graphical models for semantic mapping and localization. *Journal Name*, Volume(Issue):Page Range, Year.
- [9] B. Smith et al. Semantic mapping and classification using bayesian networks. *Journal Name*, Volume(Issue):Page Range, Year.
- [10] Kevin P Murphy. Dynamic bayesian networks: Representation, inference and learning. In *Machine learning*, volume 29, pages 185–232. Springer, 2002.

- [11] Xiaoxiao Ma, David M Bossens, David Cohn, et al. Hidden markov models for indoor environment classification and mapping. *IEEE Transactions on Automation Science and Engineering*, 2016.
- [12] Anand B Pillai and Avinash S Gandhi. Robust semantic mapping using hidden markov models for mobile robots. *Robotics and Autonomous Systems*, 2017.
- [13] David M Bossens, David Cohn, and Ryan M Eustice. Online probabilistic semantic mapping with hidden markov models. In *Proceedings of Robotics: Science and Systems*, 2015.
- [14] T. Gu and et al. Semantic mapping and localization via rfid tags: The next frontier. *IEEE Transactions on Automation Science and Engineering*, 7(2):297–312, 2010.
- [15] S. Divvala and et al. Semantic mapping using dynamic markov chains. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4627–4632. IEEE, 2012.
- [16] Andrzej Pronobis and R. P. N. Rao. Learning deep generative spatial models for mobile robots. In *Proc. of IROS*, 2017.
- [17] R. Peharz, S. Tschitschek, F. Pernkopf, and P. Domingos. On theoretical properties of sum-product networks. In *Proc. of AISTATS*, 2015.
- [18] Hsu, A. Kalra, and P. Poupart. Online structure learning for sum product networks with gaussian leaves. *preprint arXiv:1701.05265*, 2017.
- [19] Robert Peharz, Robert Gens, Franz Pernkopf, and Pedro Domingos. On the latent variable interpretation in sum-product networks. *Transactions on Pattern Analysis and Machine Intelligence*, 2017.
- [20] W.-C. Cheng, S. Kok, H. V. Pham, H. L. Chieu, and K. M. A. Chai. Language modeling with sum-product networks. In *Proc. of Interspeech*, 2014.
- [21] R. Peharz, B. C. Geiger, and F. Pernkopf. Greedy part-wise learning of sum-product networks. In *Machine Learning and Knowledge Discovery in Databases, ser. Lecture Notes in*, 2014.
- [22] S.; Friedman, P.; Pasula and D. Fox. Voronoi random fields: Extracting the topological structure of indoor environments via place labeling. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

- [23] Robert Peharz, Pierre Robert, Kai Georg, Mojtaba Pejman, and Peter Franz. Modeling speech with sum-product networks: Application to bandwidth extension. In *ICASSP*, 2014.
- [24] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.
- [25] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018.
- [26] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
- [27] Jie Chen, Tengfei Ma, Chuan Xiao, Shuang Jiang, and Bo Long. Fastgcn: Fast learning with graph convolutional networks via importance sampling. In *International Conference on Learning Representations (ICLR)*, 2018.
- [28] Qimai Li, Zhichao Han, Xiao-Ming Wu, and Wei-Shi Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2018.
- [29] Jiaxuan You, Rex L Ying, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International Conference on Machine Learning (ICML)*, 2018.
- [30] Weihua Chen, Yujia Zhu, Mo Li, Minqing Gao, Ziwei Deng, and Jiwen Chen. A simple baseline for graph classification. In *International Conference on Learning Representations (ICLR)*, 2019.
- [31] Zonghan Wu, Shirui Pan, Guodong Long, Jingjing Jiang, Chengqi Zhang, and Xing Wu. A comprehensive survey on graph neural networks. In *IEEE Transactions on Neural Networks and Learning Systems*, 2020.
- [32] Travis Oliphant. A guide to numpy. URL: <http://www.numpy.org>, 2006. Accessed: [Insert Date Here].
- [33] Tom Reback, Wes McKinney, and jbrockmendel. pandas-dev/pandas: Pandas. *Zenodo*, 2020.

- [34] Andrzej Pronobis. libspn-keras: A library for learning sum-product networks with keras. <https://github.com/AndrzejPronobis/libspn-keras>, 2021. Accessed: [Insert Date Here].
- [35] Alex Andersen. qtorch: Quantization library for pytorch. <https://github.com/Quantization/pytorch-quantization>, 2021. Accessed: [Insert Date Here].
- [36] Matthias Fey and Jan E. Lenssen. torch-geometric: Geometric deep learning extension library for pytorch. https://github.com/rusty1s/pytorch_geometric, 2019. Accessed: [Insert Date Here].
- [37] Aric Hagberg, Pieter Swart, and Dan S Chult. Networkx. <https://networkx.org/>, 2008. Accessed: [Insert Date Here].
- [38] Kaiyu Zheng. Cold: A Large-scale Topological Map Dataset, 2022.
- [39] Hongyang Gao, Zhengyang Huang, and Shuiwang Yang. Large-scale learnable graph convolutional networks. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2018.
- [40] Vijay Prakash Dwivedi, Chaitanya Joshi, Thomas Laurent, and Yoshua Bengio. Benchmarking graph neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
- [41] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J Smola, and Eduard H Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, 2016.