Understanding BIM Data

A THESIS

submitted by

M G RAJA SREE

in partial fulfilment of the requirements

for the award of the degree of

BACHELOR OF TECHNOLOGY

&

MASTER OF TECHNOLOGY

in

ELECTRICAL ENGINEERING

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS

May 2022

THESIS CERTIFICATE

This is to certify that the thesis titled **Understanding BIM Data**, submitted by **M G Raja Sree**, to the Indian Institute of Technology, Madras, for the award of the degree of **Bachelor of Technology & Master of Technology in Electrical Engineering**, is a bona fide record of the research work done by her under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. V. Srinivasa Chakravarthy
Project Guide
Professor
Dept. of Biotechnology
IIT-Madras, 600 036

Prof. Mansi Sharma
Project Co-Guide
Inspire Faculty
Dept. of Electrical Engineering
IIT-Madras, 600 036

Place: Chennai

Date: May 2022

ACKNOWLEDGEMENTS

I would like to express my gratitude towards my project guide, Prof. V. Srinivasa Chakravarthy for giving me the opportunity to work on this project. His extensive knowledge and experience has been a great source of inspiration. I would also like to acknowledge Krishna Sridhar, founder and CTO of Conxai Technologies Gmbh for guiding me throughout the project and providing his immense support. I am grateful for the support provided by Mansi Sharma as the co-guide of this project. I would also like to express my deepest gratitude to my project mentor, Vignesh Chandrasekharan for his immense guidance and constant support throughout the course of my project.

I would like to thank my parents and sister for their encouragement and unwavering support throughout my life. Finally, I would also like to thank my friends for extending their help and support at all times.

ABSTRACT

BIM is an acronym for Building Information Model or Building Information Modeling or Building Information Management. In the last 2 decades, the term Building Information Modeling has become ubiquitous in the design and construction fields. Semantic enrichment of Building Information Modeling (BIM) models adds implicit meaning to models, allowing for more applications. The room classification challenge is used in this study to build, test, and illustrate a unique technique to semantic enrichment of BIM models—representing models as graphs and applying graph neural networks (GNNs). In this study, an attempt was made to understand, explore and enrich the data in the BIM model using the following objectives - i) Clash Detection in 3D BIM model ii) Room Classification in 2D BIM model using Graph Neural Networks, (floor plan image) where a room is converted to graph with its rooms as nodes and doors as edges and each room is classified iii) Conversion of 2D BIM Model into a graph structure.

KEYWORDS: BIM; Clash Detection; Floor Plan; GNN.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

BIM	Building Information Modeling
AEC	Architecture, Engineering and Construction
MEP	Mechanilcal, Electrical and Plumbing
HVAC	Heating, Ventilation, and Air Conditioning
CAD	Computer-Aided Design
GNN	Graph Neural Networks
DGL	Deep Graph Library

CHAPTER 1

INTRODUCTION

BIM stands for Building Information Model/Modeling/Management, where BIM is a model, i.e. the container of data and information, which must be read, enriched and modified throughout the life cycle of the work; BIM is a process, i.e. a succession of activities to manage data and information contained within information models; BIM is collaboration, i.e. to ensure that information models are always up-to-date and usable, all operators must collaborate at appropriate times in the process and according to certain rules.

1.1 Introduction to BIM

For many years, builders have used 2D blueprints to create structures of the buildings. However, this becomes problematic when the scale of the buildings grows more prominent, making upkeep a challenging task. BIM is like a digital prototype of a building built by Mechanical, Electrical, Plumbing (MEP) engineers or Architectural, Engineering and Constructional (AEC) professionals, helping coordinate, plan, design and construct the building smoothly and efficiently.

BIM is a 3D model of the building structure that includes all of the components such as pipes, walls, doors, ceilings, stairs, et cetera, as well as a lot of other information that makes it challenging to comprehend. One needs some technical expertise and software knowledge to understand and explore its data. In some cases, even this large amount of data is not enough to obtain some information like 'when is a particular wall going to be built'. So, the initial goal of this project was to transform the BIM data into something less complex with all information in it and semantically enrich it.

1.2 Types of BIM Files

BIM files are of various types based on dimension, file format etc.

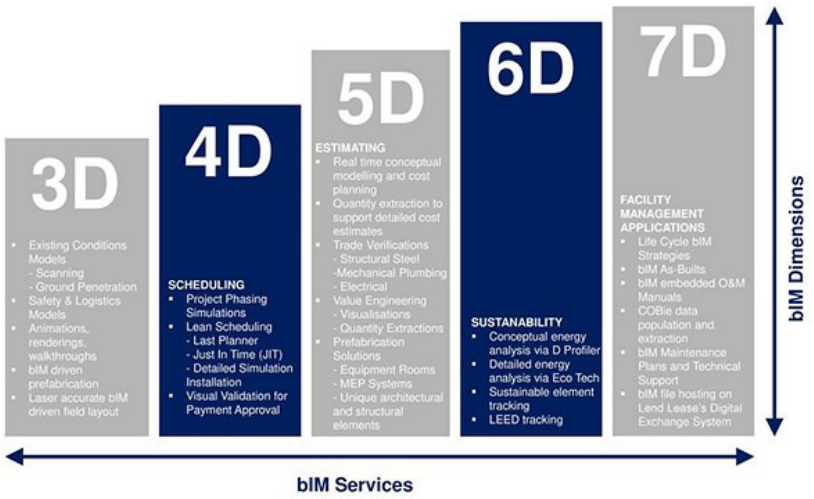


Figure 1.1: BIM files based on dimension

Based on dimensions, BIM files can be 2D, 3D, 4D, 5D, 6D, et cetera. Construction models were initially designed with 2D BIM with X and Y axes. These models are often created by hand or using CAD designs. Nowadays, all construction firms are familiar with 3D BIM, which is likely the most common BIM dimension. A coordinated model is another term for this. The scheduling data or time element in a 4D BIM model is the additional dimensional information apart from the 3D model. 3D BIM + schedule is 4D BIM. 4D BIM + Estimate or Cost is the 5D BIM model. It combines cost, scheduling, and design into a 3D result. Integrated BIM (iBIM) is another name for a 6D BIM model. It entails incorporating other essential data to help the facility's administration and operation in the hopes of achieving a better commercial outcome.

Based on file formats, the top 5 highly used BIM file formats are DWG, DXF, IFC, RVT, and NWD. In this study, we use the NWD file format, which is Autodesk's proprietary format for Navisworks files that can be opened by using Navisworks software.

1.3 BIM File Examples

Figure 1.2: 2D BIM file

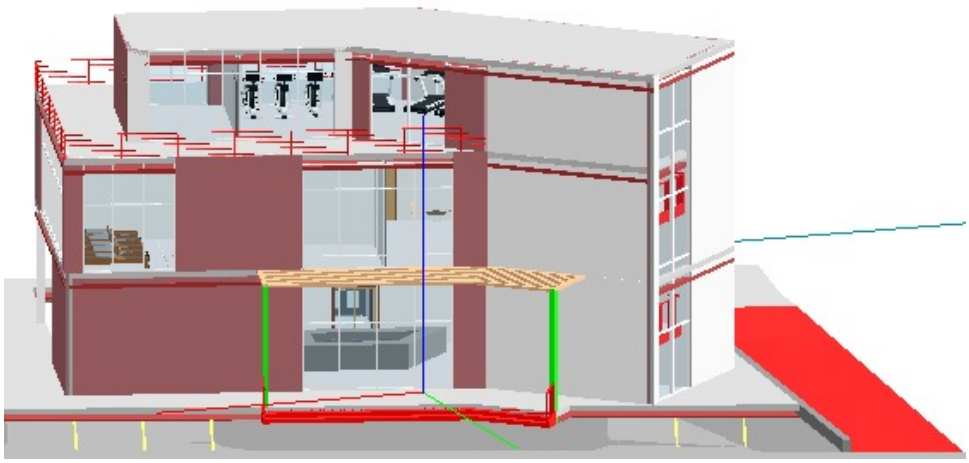


Figure 1.3: Front view of a Wellness Center building

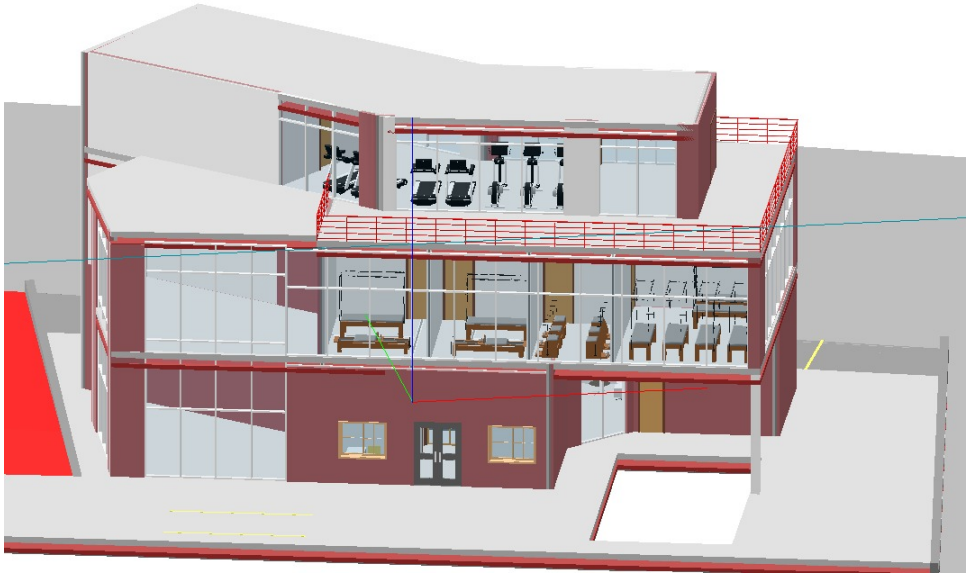


Figure 1.4: Back view of a Wellness Center building

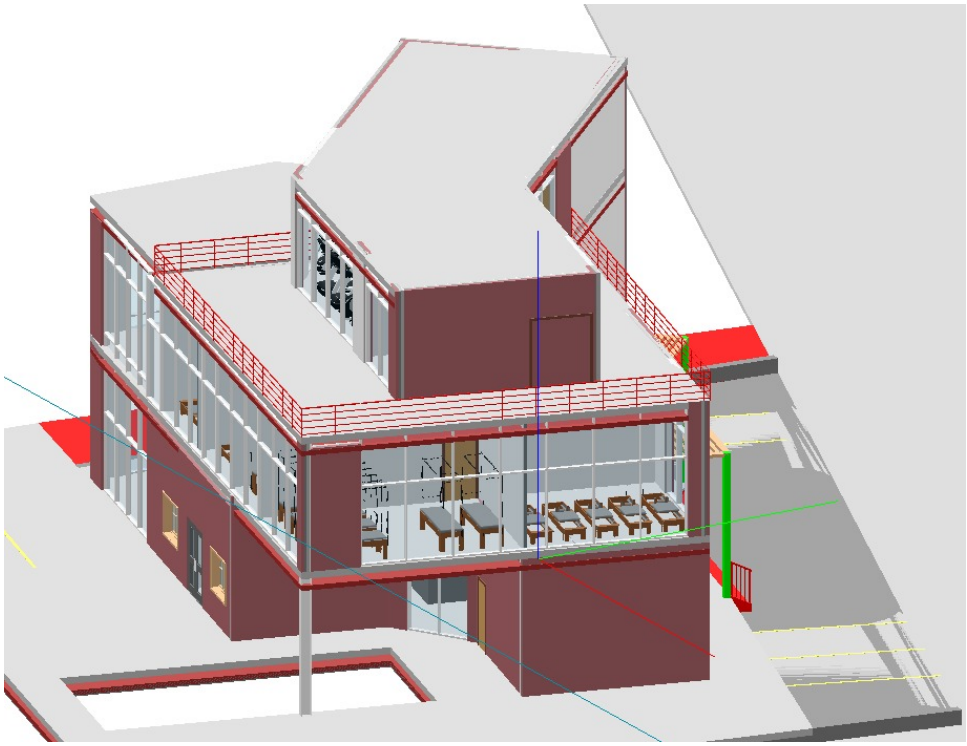


Figure 1.5: Side view of a Wellness Center building

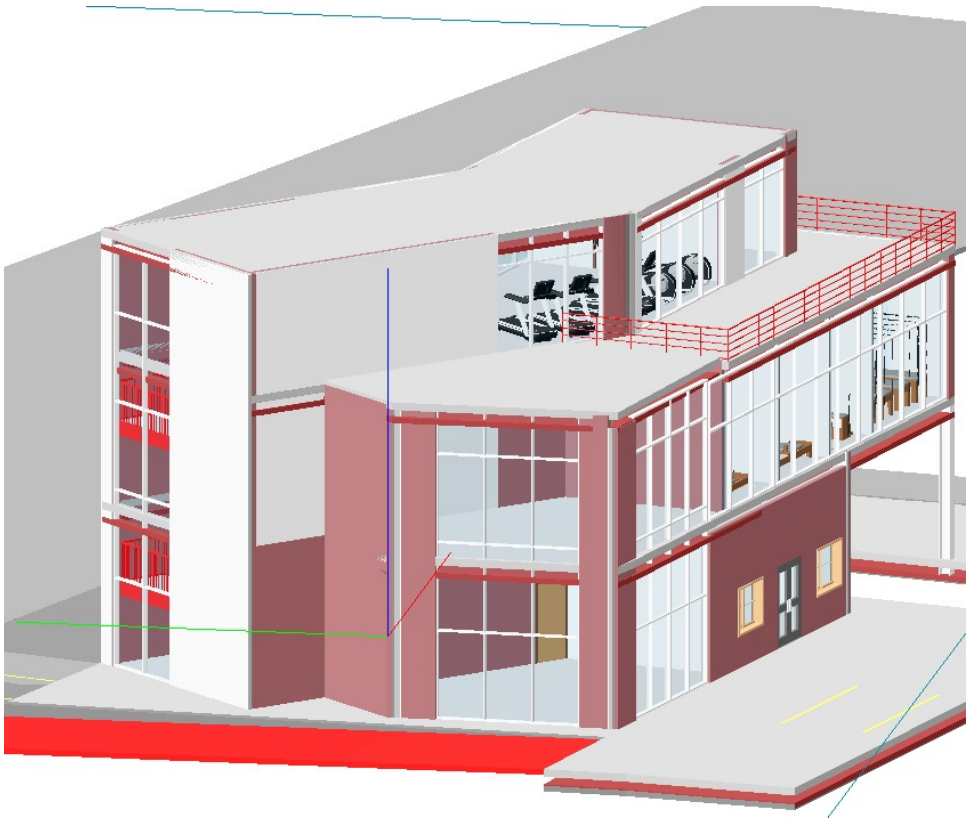


Figure 1.6: Side view of a Wellness Center building

1.4 Organization of thesis

The outline of the thesis is as follows. In the following three chapters, different problem statements are explained and how they have been approached. Chapter 2 introduces Clash Detection, followed by its background review, experiments and conclusion. Chapter 3 discusses the Room Classification problem and how it has been approached. In Chapter 4, we introduce a new problem Layout to Graph and explain how it is done.

Chapter 5 discusses the applications of this study, and Chapter 6 is about the Future Scope of this research. The thesis is concluded by the references used for this study.

CHAPTER 2

CLASH DETECTION

2.1 Introduction

From 1 billion in 1800 to 7.9 billion in 2020, the world's population has increased dramatically. According to the United Nations, the world's population will reach 8.6 billion by mid-2030, 9.8 billion by mid-2050, and 11.2 billion by 2100. With the world's population growing, the global AEC industry must seek more ingenious and more efficient ways to design and build not only to meet global demand but also to help develop smarter, more resilient environments. BIM has grown from a building concept to an industry-defining method and technology in the last two decades. BIM definitely has become the industry standard and is making the AEC business more data-driven.

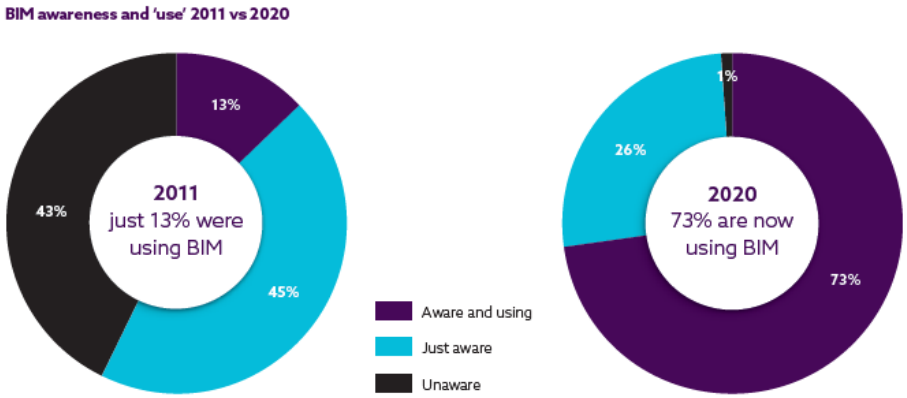


Figure 2.1: BIM adoption over the past few years in the construction field

BIM helps to detect where the clashes are, correct them and also make sure there is a reduction of mistakes on site, et cetera.

2.1.1 Motivation

BIM is a complex field that requires successful collaboration from a variety of stakeholders, i.e., AEC experts, to obtain the desired results. The presence of multiple stakeholders raises the complexity level significantly, but there is also the fact that the project has a deadline. This means that all stakeholders must work together to ensure that the entire construction project is completed accurately and on schedule.

AEC professionals, MEP engineers, HVAC (Heating, Ventilating and Air Conditioning) technicians, Environmental engineers, and other professionals create design models independent of each other for the construction of a building. Once all the models are integrated into the BIM modeling process, clash detection comes into the picture. It is a crucial part of the BIM modeling process because there is not just one model in BIM modeling, but multiple that are eventually combined into a composite master model.

2.1.2 Types of Clash Detection

A 'clash' is the result of two elements in the design taking up the same space. In BIM, clash detection is the technique of identifying if, where, or how two parts of the building (e.g., plumbing, walls, et cetera) interfere with one another as early as the design space.

There are 3 types of clashes:

a) **Hard clash** occurs when two or more components are occupying the same space or are interfering with each other. For instance, plumbing is routed through a wall or ductwork cutting across a steel beam. An example for the same is shown in Fig 2.2.

b) **Soft clash** indicates that an object has not been given sufficient geometric tolerances in the design phase or if its buffer zone has been violated. These can cause safety and maintenance issues, like when a plumbing line is close to a live wire which can easily cause a short-circuit.

c) **Workflow clash** results from inconsistent or conflicting building information. It involves clashes related to contractor scheduling, equipment and material delivery and general workflow timeline conflicts. It is not an object clash. For example, an HVAC maintenance schedule does not align with the scheduled delivery of spare parts.

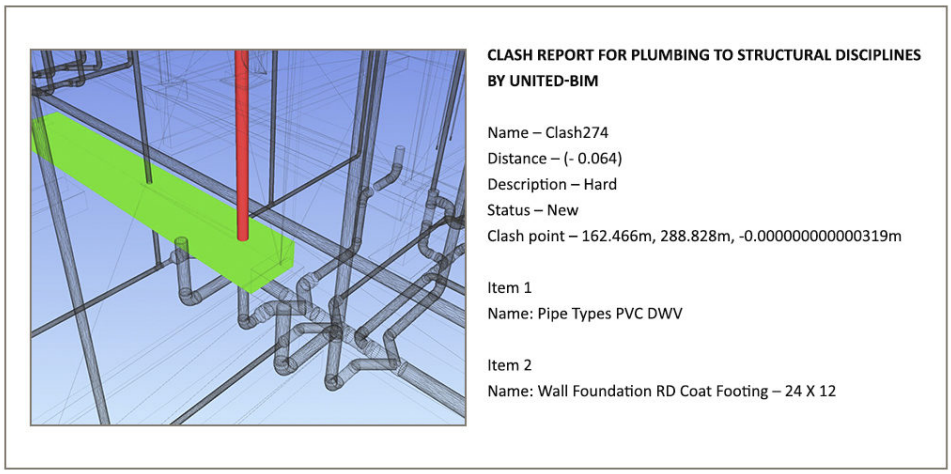


Figure 2.2: Hard Clash Example

2.2 Problem Statement

With the recent advancements in technology, one can find many software tools to identify the clashes in a BIM file at an early stage. However, in a single BIM model, one can have more than a thousand clashes.

When a clash is detected, there must be some changes done to the BIM file to obtain a safe and righteous BIM file which can be constructed later. Of the detected clashes, a few may require changing the entire design of the model to obtain a functioning BIM file; a few may need some small changes to the design, and few can be ignored as the construction engineer can solve these clashes on-site without any change in design. So, one has to identify the clashes to resolve them before constructing the building in real-time.

Based on this, all clashes are divided into two categories - **Relevant Clashes:** where the clash has to be resolved immediately and **Irrelevant Clashes:** which do not harm the building and the construction design process. When clash detection is conducted on a BIM model, the number of clashes detected is enormous. As many studies discovered, 50% or more of the clashes detected from BIM software are found to be irrelevant clashes. Researchers are trying to filter out these irrelevant clashes from the clash detection so that one need not go through all the identified clashes and resolve them as it is a time-consuming job.

This study made an attempt to solve the problem by filtering out the irrelevant

clashes using Machine Learning.

2.3 Background

Reducing the number of clashes can be done in 3 ways - Clash Avoidance, Clash Detection Improvement and Clash Filtering.

Clash avoidance creates a collaborative environment for design teams to operate in, but it inevitably adds to their workload. Clash detection improvement aims to improve the BIM software's clash detection algorithms to improve the detection accuracy and there by reducing the number of irrelevant clashes. However, despite the researches and ongoing improvements, they cannot successfully decrease the number of irrelevant clashes, which are mainly induced by human error.

Clash filtering is popularly done using rule-based knowledge systems. The main idea behind it is to label the clashes, so one can only look into the critical/relevant clashes that need an expert's knowledge to resolve them.

Rule acquisition needs domain knowledge. However, acquiring domain knowledge, which is dispersed and fragmented, necessitates the participation of numerous specialists from other domains. Even with the expertise of specialists, extracting all the rules from data takes time. Also, as this would be done by the involvement of many humans, there might be some conflicts in the rules and all the rules required may not be included.

2.4 Experiment

The Autodesk Navisworks clash detection tool is a construction project evaluation and simulation software that provides a solution by precisely detecting model incompatibilities during the project design stage. Scanning the geometry and time data of the models given by different stakeholders provides clash detection reports for contractors and architects.

2.4.1 Dataset

A BIM source found online was used as the dataset. It contains one floor and a basement. A glimpse of the model is shown below:

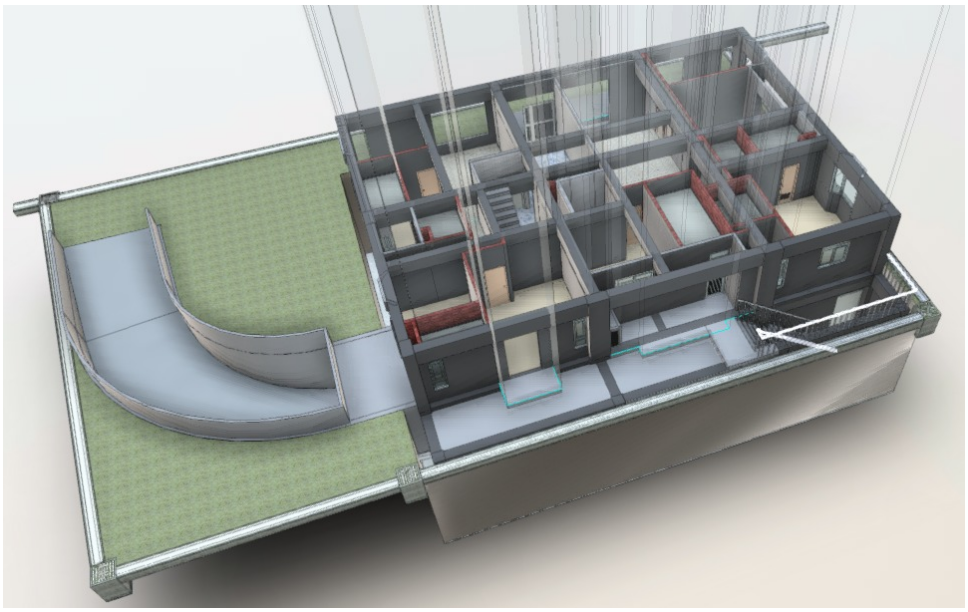
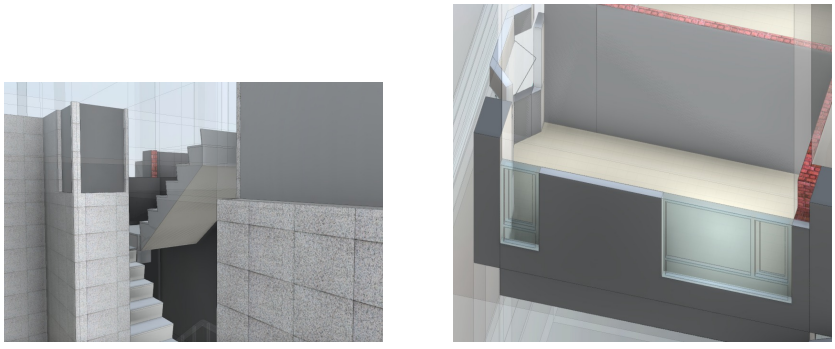


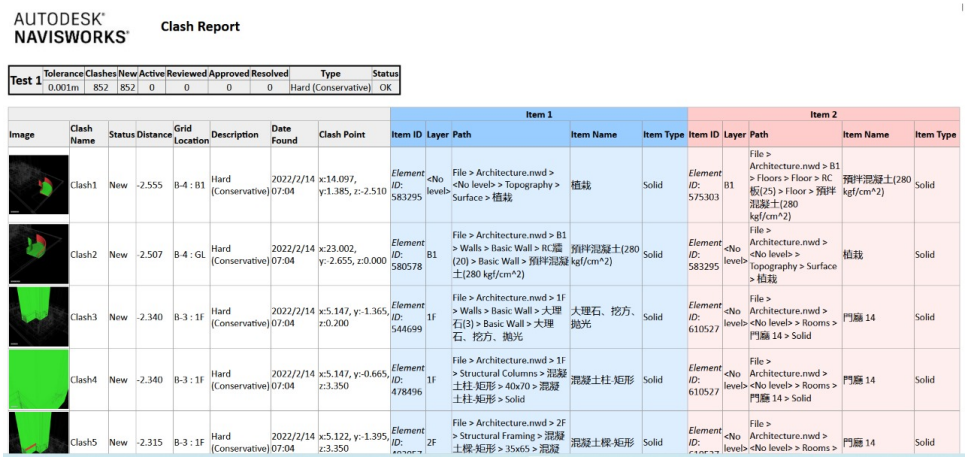
Figure 2.3: Building exterior



(a) Stairs (b) Room

Figure 2.4: Building interiors

The software, Autodesk Navisworks Manage 2017 is used to generate a clash report for this BIM model as shown below:



AUTODESK® NAVISWORKS®

Clash Report

Test 1	Tolerance	Clashes	New	Active	Reviewed	Approved	Resolved	Type	Status
	0.001m	852	852	0	0	0	0	Hard (Conservative)	OK

								Item 1					Item 2				
Image	Clash Name	Status	Distance	Grid Location	Description	Date Found	Clash Point	Item ID	Layer	Path	Item Name	Item Type	Item ID	Layer	Path	Item Name	Item Type
	Clash1	New	-2.555	B-4 : B1	Hard (Conservative)	2022/2/14 07:04	x:14.097, y:1.385, z:-2.510	Element ID: 583295	<No level>	File > Architecture.nwd > <No level> > Topography > Surface > 植栽	植栽	Solid	Element ID: 575303	B1	File > Architecture.nwd > B1 > Floors > Floor > RC板(25) > Floor > 預拌混凝土(280 kgf/cm^2)	預拌混凝土(280 kgf/cm^2)	Solid
	Clash2	New	-2.507	B-4 : GL	Hard (Conservative)	2022/2/14 07:04	x:23.002, y:-2.655, z:0.000	Element ID: 580578	B1	File > Architecture.nwd > B1 > Walls > Basic Wall > RC牆(20) > Basic Wall > 預拌混凝土(280 kgf/cm^2)	預拌混凝土(280 kgf/cm^2)	Solid	Element ID: 583295	<No level>	File > Architecture.nwd > <No level> > Topography > Surface > 植栽	植栽	Solid
	Clash3	New	-2.340	B-3 : 1F	Hard (Conservative)	2022/2/14 07:04	x:5.147, y:-1.365, z:0.200	Element ID: 544699	1F	File > Architecture.nwd > 1F > Walls > Basic Wall > 大理石(3) > Basic Wall > 大理石、挖方、抛光	大理石、挖方、抛光	Solid	Element ID: 610527	<No level>	File > Architecture.nwd > <No level> > Rooms > 門廳 14 > Solid	門廳 14	Solid
	Clash4	New	-2.340	B-3 : 1F	Hard (Conservative)	2022/2/14 07:04	x:5.147, y:-0.665, z:3.350	Element ID: 478496	1F	File > Architecture.nwd > 1F > Structural Columns > 混凝土柱-矩形 > 40x70 > 混凝土柱-矩形 > Solid	混凝土柱-矩形	Solid	Element ID: 610527	<No level>	File > Architecture.nwd > <No level> > Rooms > 門廳 14 > Solid	門廳 14	Solid
	Clash5	New	-2.315	B-3 : 1F	Hard (Conservative)	2022/2/14 07:04	x:5.122, y:-1.395, z:3.350	Element ID:	2F	File > Architecture.nwd > 2F > Structural Framing > 混凝土樑-矩形 > 35x65 > 混凝	混凝土樑-矩形	Solid	Element ID:	<No level>	File > Architecture.nwd > <No level> > Rooms >	門廳 14	Solid

Figure 2.5: Part of the clash detection report

This report should be converted to a trainable format to be able to train models on it. To do that feature processing is done. The numerical features like 'distance', 'floor-1' etc are not changed but the textual feature like 'clash point' which is a coordinate, is classified into 3 independent numerical features - 'Clash Point-x', 'Clash Point-y'and 'Clash Point-z'. Other features whose data types are nominal/text, such as 'ItemType-1'and 'ItemType-2'are transformed to numerical features using one-hot encoding. Now, the original 6 featured report is transformed into a 12 featured dataset and is called as training dataset 1.

Then, 2 experts are given this data to label the clashes as - Errors, Deliberate Clashes, Pseudo Clashes or Unknown Clashes. Five basic thumb rules are formulated by experts using only the theory with which the clashes in the report are classified. Using these rules, clashes are labelled to introduce a new feature called 'Rule-Tag'. These labels are one-hot encoded adding four more features. This dataset is called training dataset 2.

2.4.2 Training and Results

A hybrid method is studied which uses both rule-based system and machine learning to filter out the irrelevant clashes from the clash report. The model architecture is shown below:

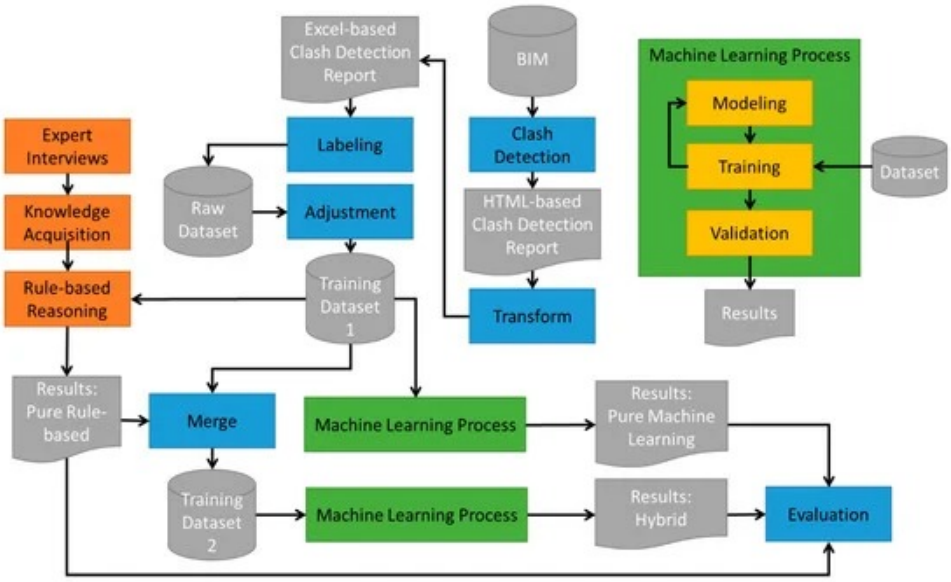


Figure 2.6: Methodology

Training dataset 1 and 2 are given to a support vector machine with ‘linear’ kernel to get "pure-rule based results" and "hybrid based results". To avoid bias, training and testing processes are repeated 30 times, where different training and testing cases were randomly selected for each test. An average F1-score of 0.785 and 0.905 was obtained for training dataset 1 and 2 respectively.

2.5 Conclusion and Future Work

Even though this study gives better results, opting for crowdsourcing to get the clash labels is not a reliable option. As the BIM file containing clashes cannot be used for further research, this study is paused for now and the focus is shifted to 2D BIM file, i.e., the floor plan images.

CHAPTER 3

ROOM CLASSIFICATION USING GNN

This chapter discusses Graph Neural Networks' importance and uses them to classify each room in floor plan/layout images.

3.1 Introduction

Graphs can be found everywhere. It is a simple data structure with only two components: nodes and edges, but it can hold and process any vast amount of data. Researchers have been working on GNNs, which are neural networks that act on graph data, for more than a decade. GNNs can do what CNNs failed to do as they are fundamentally rotation and translation invariant.

3.2 GraphSAGE

The GNN network we use for this study is GraphSAGE. It is a stochastic generalisation of graph convolutions. It is an iterative approach for learning graph embeddings for each node in a graph. In layman's terms, the main idea of GraphSAGE is that 'you are known by the company you keep', i.e., it learns a representation for every node based on some combination of its neighbouring nodes. Unlike the graph based models prior to this, which are transductive, GraphSAGE is inductive.

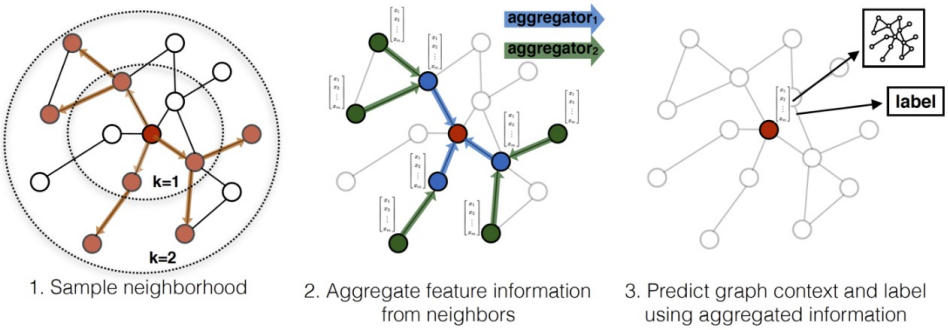


Figure 3.1: Overview of GraphSAGE Algorithm

3.2.1 Notation

All GNNs follow a specific parameter representation as mentioned below:
A graph is defined as G = (V, E), where V is a set of nodes and E is the edges between them. G is represented using an Adjacency matrix, A of dimension N*N, where N is the number of nodes. Feature matrix, X is of dimensions N*F, where F is the number of features of each node. For every node v:

X_v - node feature vector of v

h_v^0 - initial node embedding representation of v

h_v^k - node embedding representation of v at k-th iteration

z_v - final node embedding representation of v after a round of GraphSAGE

3.2.2 Algorithm

Embedding for a node is represented by some combination of its neighbouring nodes. For one round of GraphSAGE, a new representation is obtained for all the nodes in a graph. GraphSAGE algorithm has two steps performed iteratively - Aggregate and Update. During initialisation, the embedding vectors for the nodes are their feature vectors, i.e., $h_v^k - 1 = h_v^0 = = X_v$.

i) **Aggregate:** For a node v, all the embeddings of the nodes u in its neighbourhood are aggregated using some function, represented as $f_{aggregate}$.

$$a_v = f_{aggregate}(\{h_u | u \in N(v)\}) \tag{3.1}$$

where for a_v is the aggregated node representation for node v.

ii) **Update;** For a node v, its embedding is updated by some combination of the aggregated node representation and its previous representation on some function f_{update}.

$$h_v^k = f_{activation}(W^k * f_{update}(a_v, h_v^k - 1)) \tag{3.2}$$

where for W^k is the weight matrix, $f_{activation}$ is the activation function.

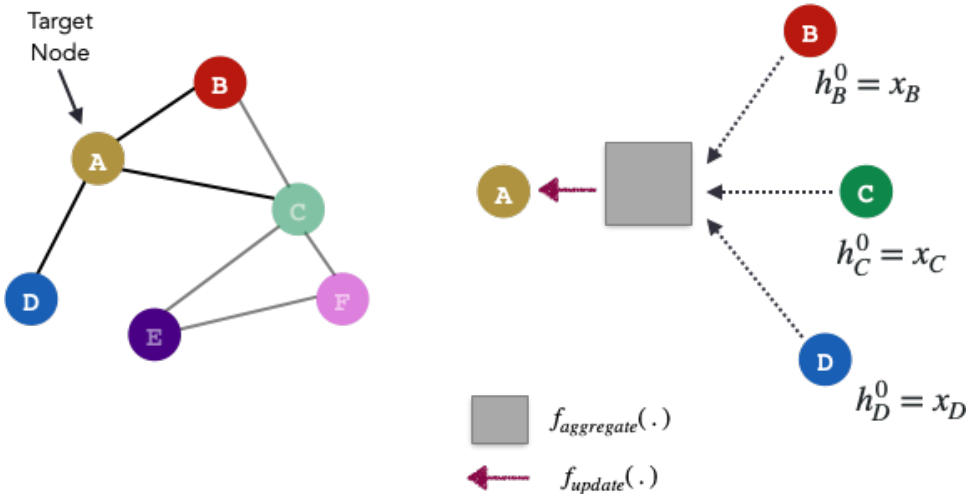


Figure 3.2: Representation of Aggregator and Update functions for the node A

Training: Graph based loss function:

$$J_G(z_u) = -log(f_{activation}(z_u^T z_v)) - Q * E_{v_n \tilde{P}_n(v)} log(f_{activation}(-z_u^T z_{v_n})) \quad (3.3)$$

where, v is a node that co-occurs near node u on a fixed-length random walk, σ is the sigmoid function, P_n is a negative sampling distribution, and Q defines the number of negative samples. This loss function is minimised using Stochastic Gradient Descent.

3.3 Motivation and Problem Statement

Home remodelling firms typically create 3D models from 2D floor plans, and then the computer simulates the decorating of various rooms automatically; thus, determining the precise type of room becomes a vital task for automatic decoration. Although the text in floor plans can be recognised and text recognition technology is well established, the problem is determining the kind of space when the floor plan lacks a textual description of the room.

This study made an attempt to classify the rooms in a floor plan using GNNs.

3.4 Experiments

3.4.1 Dataset

224 different apartment layouts of 3 different countries - US, UK, and China, are taken and are manually converted into a graph with nodes as rooms and edges as the connections - door, wall and virtual wall, where a virtual wall is when there is no actual wall separating two different room spaces. A sample dataset is shown below:

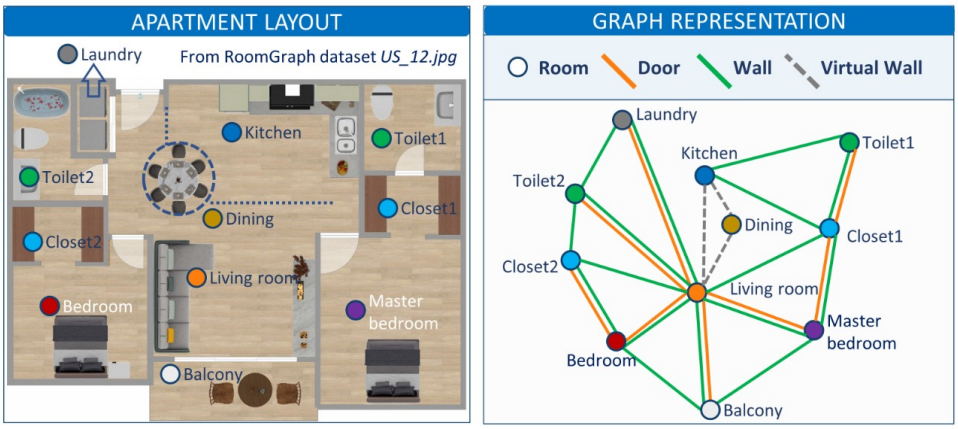


Figure 3.3: Apartment layout image and its graph representation

```
print(train_dataset[10])
print(train_dataset[10].nodes())
print(train_dataset[10].edges())
```

```
Graph(num_nodes=11, num_edges=34,
      ndata_schemes={'label': Scheme(shape=(9,), dtype=torch.int64), 'feat': Scheme(shape=(8,), dtype=torch.float64), 'index': Scheme(shape=(), dtype=torch.int64)}
      edata_schemes={'relation': Scheme(shape=(5,), dtype=torch.int64)})
tensor([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
(tensor([ 0,  0,  0,  1,  1,  1,  2,  2,  2,  2,  2,  2,  2,  2,  3,  3,  3,  3,
          4,  4,  5,  5,  6,  6,  6,  7,  7,  7,  8,  8,  9, 10, 10, 10]), tensor([ 1,  2,  3,  0,  2,  3,  0,  1,  3,  4,  5,  7,  9, 10,  0,  1,  2,  7,
          2,  7,  2,  6,  5,  8, 10,  2,  3,  4,  6, 10,  2,  2,  6,  8]))
```

Figure 3.4: Nodes and edges of the graph of the above layout

These graphs are stored in a NumPy array using the Deep Graph Library (DGL), a python package built for easy implementation of GNNs. All the rooms are classified into nine categories, namely - 0:Kitchen, 1:Dining, 2:Living, 3:Bedroom, 4:Master Bedroom, 5:Toilet, 6:Closet, 7:Balcony, and 8:Laundry.

The node features for each node of the graph are - has a door connection or not, has a wall connection or not, has a virtual wall connection or not, #door connections, #wall connections, is an open space or not, #simple paths for wall connections*0.01, #simple paths for access connections*0.01. The edge features are - is a door connection, is a

wall connection, is a virtual wall connection, has a metal/glass door, has a wooden door.

3.4.2 Model

GraphSAGE model is built only for graphs with node features, but our dataset has both node and edge features. So, the GraphSAGE algorithm is slightly modified to incorporate edge features along with node features. This is called as SAGE-E algorithm. Only the aggregator needs to be modified to use the edge features in the model.

Aggregator function (eqn 3.1) is now:

$$a_v = f_{aggregate}(\{concat(h_u, e_{uv}) | u \in N(v)\}) \tag{3.4}$$

where, e_{uv} is the edge feature for the nodes between u and v.

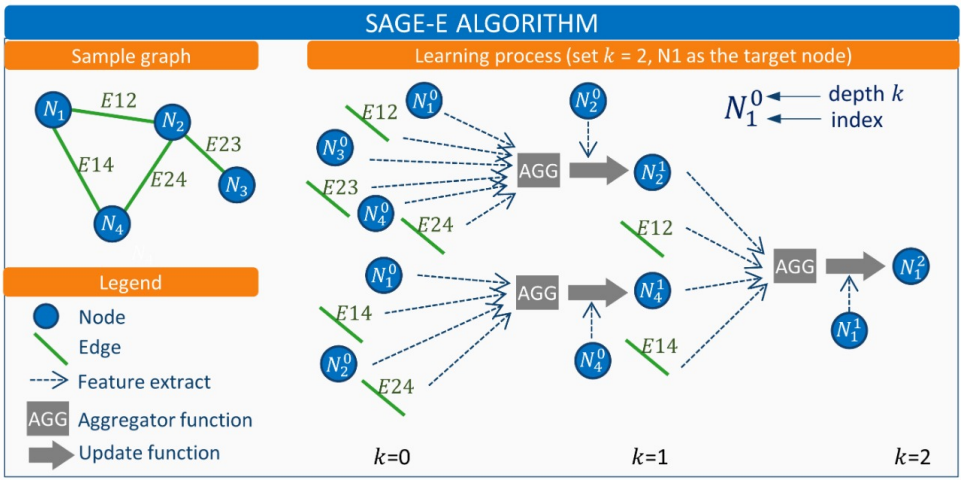


Figure 3.5: Schematic diagram of the SAGE-E algorithm

3.4.3 Training

Data is split in the ratio of 80:20. Four SAGE-E layers are used in the model. Adam optimizer and cross-entropy were adopted for training. Hyperparameters - batch size is 1, the learning rate is set to 0.005, and the training epochs to 200.

For a full comparison, GraphSAGE with four layers is also trained.

3.4.4 Results

To test the SAGE-E performance, it is trained with 3, 4 and 5 layers of SAGE-E. Table below shows the accuracy and F1-score obtained for the same:

Model	Accuracy	F1-score
GraphSAGE with 4 layers	70.84%	0.71
SAGE-E with 5 layers	78.55%	0.77
SAGE-E with 4 layers	79.24%	0.79
SAGE-E with 3 layers	74.22%	0.67

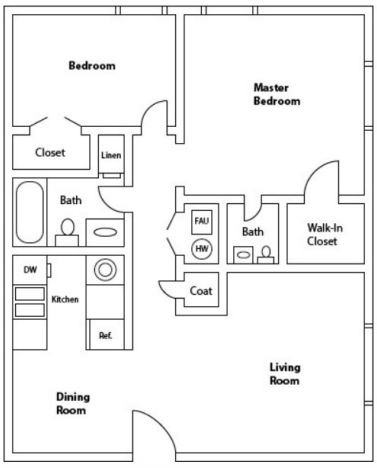
Table 3.1: Accuracy and F1-score for different models

SAGE-E with 4 layers achieved the highest accuracy of 79% and a more balanced prediction with an F1-score of 0.79.

3.4.5 Testing

The trained model is tested on unseen data. It is converted to graph and is given as an input to the model. Results obtained are shown below:

Floor plan:



(a) input floor plan

```
print(acc)
print(one_pre)
print(one_gt)
```

```
0.6666666666666666
tensor([0, 7, 1, 4, 4, 5, 6, 6, 5])
tensor([0, 1, 2, 3, 4, 5, 6, 6, 5])
```

(b) output; acc-accuracy; one_pre-label predictions; one_gt-label ground truth

Figure 3.6: Floor plan and Output room label predictions and ground truth

3.5 Conclusion and Future Work

The conversion from layout to graph for the dataset creation was done manually for all the 224 graphs. To expand the current work, we need to have access to large amount of data and manually creating such data is impractical. So, creating a graph database from the layout/floor plans would be an interesting problem.

CHAPTER 4

FLOOR PLAN TO GRAPH

In this chapter, we study the problem of converting a floor plan image into a graph.

4.1 Problem Statement and Motivation

A floor plan is a visual representation of a building's interior from above. Walls, windows, doors, and stairs, as well as fixed installations like bathroom fixtures, kitchen cabinetry and appliances, are often depicted on floor plans. Floor plans are often scaled drawings that show room kinds, sizes et cetera. To use such a data as an input to GNN, one has to convert it into a graph data.

This study made an attempt to convert a floor plan image to a graph data structure.

4.2 Dataset

LIFULL HOME's dataset is used for this study. The National Institute of Informatics provides it to researchers, which was created by LIFULL Co., Ltd. to promote study in informatics and related subjects. The data in the dataset comes from LIFULL HOME'S, a Japanese Real Estate Information Service.

This data is not publicly available but the vectorized versions of the floor plans in this dataset are available for public use. This data is stored in a form of numpy array. It contains 143184 floor plans, where each floor plan contains more than one room. The data is organized in a list format, where each element represents one floor plan.

A sample from the data is shown below:

```
In [4]:  1 print(data[10])

[[2.0, 4.0, 7.0, 4.0, 4.0, 3.0, 7.0], [array([ 70,  39, 110, 113], dtype=int64), array([113,  39, 131,  54], dtype=int64), arra
y([134,  39, 145,  54], dtype=int64), array([113,  57, 145,  78], dtype=int64), array([113,  81, 145, 125], dtype=int64), array
([ 70, 116, 145, 224], dtype=int64), array([ 70, 227, 145, 251], dtype=int64)], [[146, 37, 146, 55, 0, 7], [132, 37, 146, 37,
0, 7], [111, 37, 132, 37, 0, 4], [68, 37, 111, 37, 0, 2], [132, 37, 132, 55, 7, 4], [111, 37, 111, 55, 4, 2], [68, 37, 68, 114,
2, 0], [146, 55, 146, 79, 0, 4], [146, 79, 146, 126, 0, 4], [111, 114, 111, 126, 4, 3], [146, 126, 146, 225, 0, 3], [146, 225,
146, 252, 0, 7], [132, 55, 146, 55, 7, 4], [111, 55, 132, 55, 4, 4], [111, 79, 146, 79, 4, 4], [68, 114, 111, 114, 2, 3], [68,
225, 146, 225, 3, 7], [111, 126, 146, 126, 4, 3], [68, 252, 146, 252, 7, 0], [111, 55, 111, 79, 4, 2], [111, 79, 111, 114, 4,
2], [68, 114, 68, 225, 3, 0], [68, 225, 68, 252, 7, 0]], [[2], [2], [1], [0], [1, 2], [0, 1], [0], [3], [4], [4, 5], [5], [6],
[2, 3], [1, 3], [3, 4], [0, 5], [5, 6], [4, 5], [6], [0, 3], [0, 4], [5], [6]], [5, 19, 20, 3, 1, 16, 15, 7], 'floorplan_high_r
es/04/cb/582847c5ccc0421288335d7184fb/0001.jpg']
```

Figure 4.1: Vectorized floor plan sample data

As shown above, there are many lists in a vectorized floor plan array. For each floor plan in the dataset we have the following elements (in order):

1) List of room types: mapping between room types and their corresponding class. ROOM-CLASS = {"living": 1, "kitchen": 2, "bedroom": 3, "bathroom": 4, "missing": 5, "closet": 6, "balcony": 7, "corridor": 8, "dining": 9, "laundry": 10}

2) List of room bounding boxes: each bounding box is represented by [x0, y0, x1, y1], where (x0, y0) and (x1, y1) are top-left and bottom-right coordinates, respectively.

3) List of floor plan edges: edges in the floor plan are represented by [x0, y0, x1, y1, *, *], where (x0, y0) and (x1, y1) are the edges endpoints and elements * are not being used.

4) Edge to room mapping: for each edge, we assign up to 2 rooms sharing that edge.

5) Doors to edges list: an element "i" in this list means that the i-th edge contains a door.

6) Vector to RGB mapping: this field contains the name of the original RGB image from the LIFULL HOME dataset.

4.3 Proposed Method

PyTorch-Geometric library is used to build graphs for each floor plan. In each floor plan, the labels are used to form as nodes for the graph. Bounding box data is used to obtain the edges for the graph.

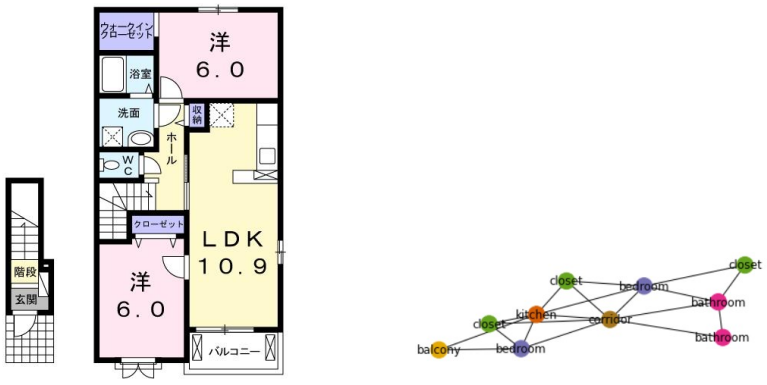
For each floor plan, the bounding boxes data from the dataset is used to obtain the length, breadth and area of the room. It is also used to find out if a room is a parent

room or a child room. This is for nested rooms, where one room is present in an other room - closet in a bedroom, dining inside a kitchen etc. For these rooms, If a room's crossing with another room covers more than 70% of its surface area, we consider that room to be a child room and the other room to be a parent room. Number of doors connecting to a room can be found using the edge to room mapping and the doors to edges data for each floor plan.

This information is used as features for each node - Area of the room, Length of the room, Width of the room, Number of doors in the room, Whether a room is a parent room and Whether a room is a child room.

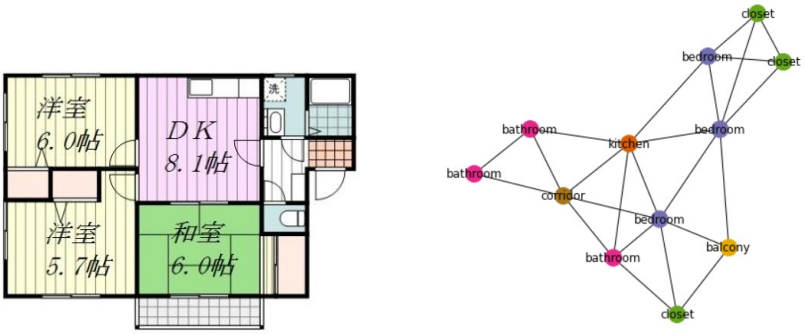
4.4 Results

After obtaining the nodes and node features for each floor plan in the dataset, the graphs are visualised using the matplotlib library. Some graphs for floor plans are shown below:



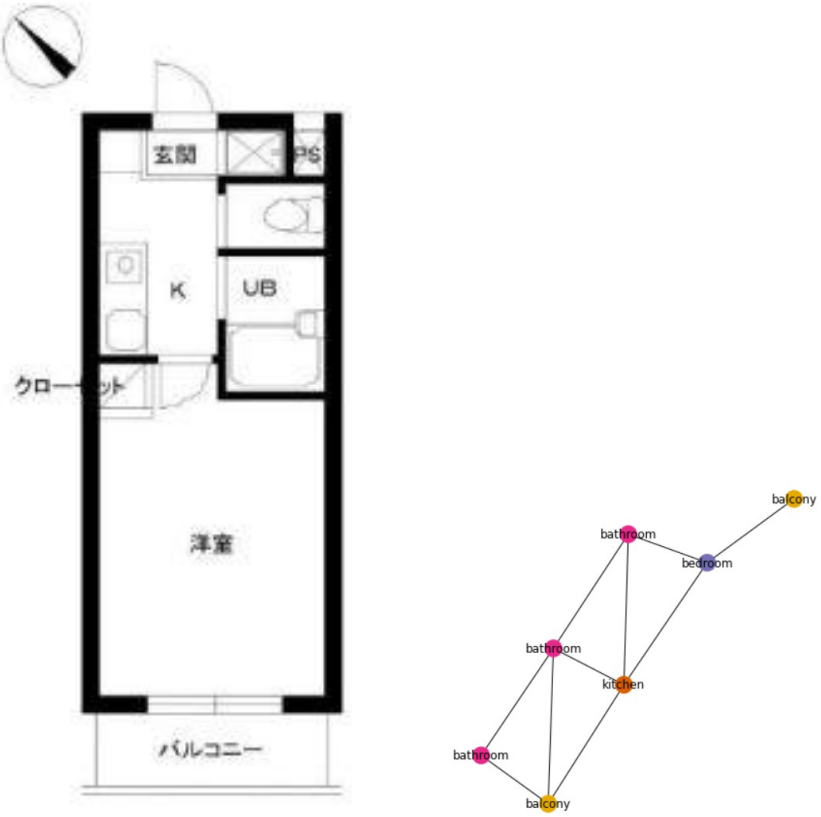
(a) input (b) output

Figure 4.2: Floor plan #1 and its output graphical representation



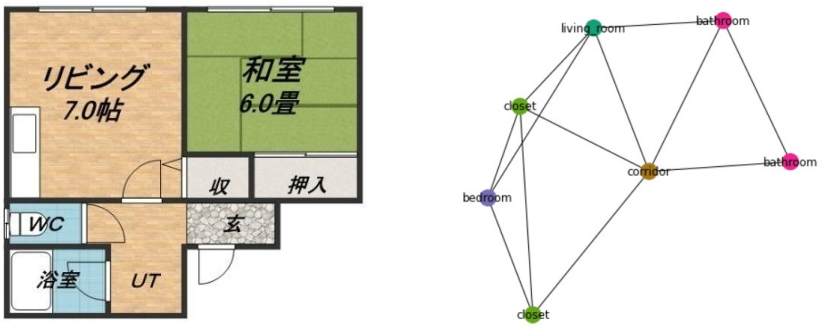
(a) input (b) output

Figure 4.3: Floor plan #2 and its output graphical representation



(a) input (b) output

Figure 4.4: Floor plan #3 and its output graphical representation



(a) input (b) output

Figure 4.5: Floor plan #4 and its output graphical representation

4.5 Extension

This graph dataset is used as an input dataset to perform room classification task. Different models are used to perform the task like - Multi Layer Perceptron (MLP), Graph Convolution Network (GCN), Graph Attention Network (GAT), GraphSAGE, Topology Adaptive Graph Convolutional Network (TAGCN). All these models are present in the PyTorch and the PyTorch-Geometric libraries.

General Model Architecture:

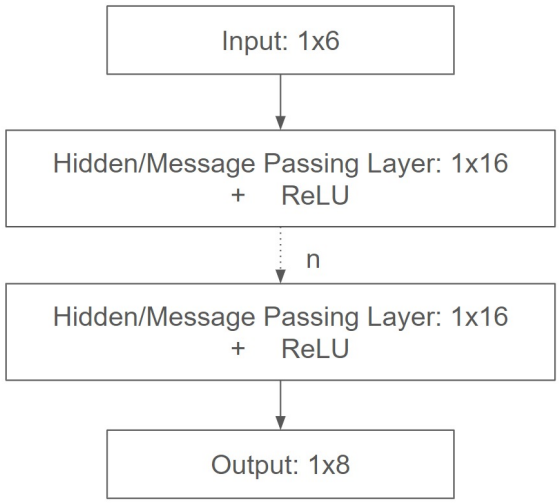


Figure 4.6: General Model Architecture

For training - 100 epochs, cross-entropy loss function, Adam optimizer, a batch size

of 128, 0.004 learning rate are used. Test accuracy results are shown in the below table:

Model	Train	Test
MLP	0.6585	0.6574
GCN	0.5469	0.5485
GAT	0.5269	0.5250
GraphSAGE	0.8068	0.8026
TAGCN	0.8140	0.8107

Table 4.1: Train and test accuracies for different models

We observe that, TAGCN and GraphSAGE gives much better results compared to MLP and other GNN models.

CHAPTER 5

APPLICATIONS AND FUTURE WORK

APPLICATIONS

The generated labelled graph can be used as an input to any GNN network. This labelled graph can also be identified as a semantically enriched BIM model. Semantic enrichment of BIM is essential to achieve seamless interoperability in BIM models.

FUTURE WORK

We can add more node features to the layout graph like, presence of doors in a room, what kind of walls are in the room et cetera. Introduce edge features to the layout graph like, what kind of connection is the edge - door, wall et cetera. Generating a model which converts the floor plan images to the numpy array data format is interesting. We can use this semantically enriched BIM model to use for different applications like cost estimation of a building.

REFERENCES

1. BIM FILE example:
   https://gitlab.com/shaktiproject/cores/i-class.git

2. Yuqing Hu and Daniel Castro-Lacaoture: *Clash Relevance Prediction Based on Machine Learning*, 2021.

3. Will Y. Lin and Ying-Hua Huang *Filtering of Irrelevant Clashes Detected by BIM Software Using a Hybrid Method of Rule-Based Reasoning and Supervised Machine Learning*, 2019.

4. LIFULL HOME'S Dataset:
   https://www.nii.ac.jp/dsc/idr/en/lifull/

5. William L. Hamilton, Rex Ying, Jure Leskovec, *Inductive Representation Learning on Large Graphs* , 2017.

6. Zijian Wang, Timson Yeung, Rafael Sacks and Zhiya Su *Room Type Classification for Semantic Enrichment of Building Information Modeling Using Graph Neural Networks*, October 2021.

7. Zijian Wang, Rafael Sacks and Timson Yeung *Exploring Graph Neural Networks for Semantic Enrichment: Room Type Classification*, December 2021.

8. Abhishek Paudel, Roshan Dhakal and Sakshat Bhattarai *Room Classification on Floor Plan Graphs using Graph Neural Networks*, December 2021.

9. Navisworks Clash Detection Tool
   https://www.autodesk.com/products/navisworks/free-trial