Exploring Architectural Floor Plan Appropriateness in Context of Bangladesh Leveraging Graph Neural Networks in Spatial Context

Mahir Tasin Islam[a], Tanjim Noor[b], Tiham Shafi Islam[c], Mahid Atif Hosain[d], Md Irtiza Anam[e], Md. Tanzim Reza[e] and Dr. Farig Yousuf Sadeque[f]

[a]*Department of Computer Science and Engineering, BRAC University, Dhaka, Bangladesh*

ARTICLE INFO

Keywords:
Architectural interior floor plans
Cross-cultural suitability
Graph Neural Networks
Spatial relationships
Binary classification
Bangladesh
Urban Planning
GraphSAGE
Automated assessment
Annotation framework

ABSTRACT

This paper investigates the use of Graph Neural Networks (GNNs) for classifying architectural floor plans and establishing the applicability of international floor plans with respect to Bangladeshi architectural standards. Flooring plan data is mainly derived from Chinese residential designs, which are converted into graph-based representations where rooms represent the nodes, and the connections through doors form the edges. Node features are prepared that include room area, centroid coordinates of the room, and room type, while door connections form unweighted edges. Three GNN models—GCN, GraphSAGE, and GAT are tested to evaluate their effectiveness in this binary classification task. GraphSAGE yielded the best performance among all the three GNN models tested, showing 87.09% test accuracy and an AUC-ROC score of 0.9512, with good generalization on unseen data. This work illustrates how GNNs can capture spatial relations from architectural data to enable scalable solutions for cross-cultural design evaluation and urban planning. It contributes to the increasingly important intersection of AI and Architecture by going beyond image-based traditional approaches and introducing a framework that automatically assesses the appropriateness of architectural designs concerning different cultural contexts.

1. Introduction

Architectural design varies significantly across regions due to cultural, geographical, and climatic differences. However, most architectural datasets are region-specific, limiting their adaptability across different contexts. For instance, compact urban designs from China may not be suitable for Bangladesh's more varied environment. To address this, our research explores the use of Graph Neural Networks (GNNs)—including GCN, GraphSAGE, and GAT—to assess the adaptability of Chinese residential floor plans for Bangladesh. GNNs are well-suited for this task due to their ability to model spatial relationships in graph-structured data, making them effective for floor plan analysis. This study aims to develop a scalable GNN-based framework to evaluate cross-cultural architectural designs, contributing to the broader application of AI in architecture.

1.1. Problem Statement

Architectural floor plans often reflect local cultural and environmental factors, yet most datasets are region-specific, limiting their global applicability. This research aims to assess whether residential floor plans from China can be adapted for use in Bangladesh, utilizing GNNs like GCN, GraphSAGE, and GAT. By employing these models in a binary classification task, we seek to determine if GNNs can generalize spatial features and identify design patterns relevant to Bangladesh's architectural needs.

mahir.tasin.islam@g.bracu.ac.bd (M.T. Islam); tanjim.noor@g.bracu.ac.bd (T. Noor); tiham.shafi.islam@g.bracu.ac.bd (T.S. Islam); mahid.atif.hosain@g.bracu.ac.bd (M.A. Hosain); md.irtiza.anam@g.bracu.ac.bd (M.I. Anam); tanzim.reza@bracu.ac.bd (Md.T. Reza); farig.sadeque@bracu.ac.bd (Dr.F.Y. Sadeque)

1.2. Research Contribution

This research aims to apply GNNs to test the appropriateness of Chinese floor plans for Bangladesh by analyzing spatial data with GNNs. The goal is to develop an efficient automated method of architectural design evaluation across cultural and geographical contexts by applying GNN to assess spatial data.

- **Cross-cultural relevance**: The research aims to adapt foreign architectural designs to local contexts by using GNNs to automatically assess the appropriateness of floor plans for Bangladesh, thus offering scalable solutions for urban planning.
- **Spatial data analysis**: The study focuses on the coordinates of the rooms and the spatial relationships defining the floor plans, highlighting how GNNs learn complex structural patterns for an exact way to assess the functional suitability of an architectural design.
- **Advancement in Architectural AI**: The research strives to contribute to the ever-evolving AI in architecture by showing the capabilities of GNNs for extending beyond the typical image-based models through spatial context, which could eventually improve design evaluation processes around the world.

2. Literature Review

The paper [1] introduces the Tell2Design-T2D dataset, which provides over 80k paired floor plans with detailed descriptions for natural language instruction-based floor plan generation and they also proposed a Seq2Seq model to translate such instructions into structured layouts, addressing challenges like room semantics, geometry, and topology.

However, dealing with noisy and ambiguous human instructions and keeping exact alignments between text and generated design remains a hard challenge. The paper [2] introduces a data-driven approach to interior floor plan generation for residential buildings using the RPLAN dataset, containing over 80,000 real-world floor plans that effectively predict room locations and walls by applying deep learning techniques, hence offering a new paradigm for the automatic generation of layouts given the boundaries. However, it simplifies room shapes and struggles with incorporating more intricate design features like door and window placement. The [3] paper introduces the Transformer architecture, where traditional recurrent and convolutional layers are replaced with self-attention mechanisms for sequence modeling which allows better parallelization and achieves state-of-the-art performance in translation tasks such as English-to-German and English-to-French. But the model also shows that some challenges still remain regarding computational cost, especially on long sequences, and also in training efficiency on large-scale tasks. The paper [4] introduces Adam, a first-order gradient-based optimization algorithm designed for stochastic objective functions, which is efficient and well-suited for high-dimensional parameter spaces and noisy gradients and it also integrates adaptive learning rates and combines advantages of AdaGrad and RMSProp to provide robust optimization. However, Adam can struggle with convergence in some cases, especially when fine-tuning hyper-parameters like learning rates. The paper [5] develops a methodology through which a designer can feed images to synthesize irregular architectural layouts; herein, the system effectively processes non-rectilinear room shapes and adjacency relationships and incorporates user preferences through visual inputs. However, it faces some complications in processing architectural features that include multi-story designs and does not provide connections with performance metrics including energy efficiency. In this paper [6], the authors have proposed a deep learning-based contour detection approach, which represents object boundary detection with embedded multi-level and multi-scale convolutional feature information. Since it achieved high performance in contour detection and image segmentation tasks by using deep FCNs or holistically-nested networks, in turn, having large training datasets, and facing important scalability issues, such as fine detail detection in complex images and weak annotation problems.The paper [7] surveys Graph Neural Networks in both spatial and non-Euclidean contexts, including the methodology by which GNNs can handle complex graph types such as dynamic, heterogeneous, and large-scale graphs. The paper focuses on a spatial context wherein the structure of graphs, like proximity or relationship between nodes, plays an important role in the node classification tasks. The paper further proposes other contexts of non-structural nature such as text and images where models of graph reasoning extract underlying structures to analyze them. It also points out the scalability issues and the computational challenges of GNNs in larger datasets. The paper [8] introduces a GNN-based recommendation system of generating floor plans based on the spatial relationship of fast and accurate searches of floor plans through user input; it is based on a dataset of spatial relationship graphs and applies SimGNN to compute similarities between the user's input and the existing floor plans. However, this system might be challenged by the size of the dataset and by the time needed for consumption in computing recommendations of large-scale efficiently. In this paper [9], the authors introduce a new Neural Network model, NENN, that provides both node and edge features in Graph Neural Networks through a hierarchical dual-level attention mechanism that performs very strongly for tasks like semi-supervised node classification, graph classification, and regression with specific evidence on molecular and citation networks. However, the model raises scalability and computational efficiency problems when extended to larger, more complex graphs. The proposed paper [10] introduces a methodology for the automated conversion of real estate floor plan images into structured graphs, enables comparison, evaluation, and retrieval of similar floor plans, and uses a deep learning-based semantic segmentation approach with rule-based graph transformation. It achieves 92% similarity with the ground truth data but struggles to accurately segment small features like doors, which would affect the precision of the graph in reflecting the actual layout. This paper [11] has introduced the use of Cross-Entropy loss functions and performed analysis on their theoretical properties, giving H-consistency bounds that upper-bound the zero-one loss concerning surrogate loss estimation errors and it extends these bounds to adversarial settings by proposing smooth adversarial comp-sum losses that improve robustness in machine learning models. The approach does face hurdles during practical implementation, related to the optimization of certain loss functions, such as mean absolute error, when using deep neural networks. The paper [12] introduces DiffPool, a hierarchical graph pooling mechanism, which allows GNNs to capture hierarchical structures in graphs for tasks like graph classification and improve the state-of-the-art performance on multiple benchmarks by creating soft clusters of nodes at every layer of GNN in a differentiable way which extensively enhances the accuracy of graph representations. However, regarding stability during training, the method is not very well-posed and might take more time in convergence compared to other models.

The paper [13] introduces a Graph Neural Network (GNN) model that processes various graph types, including acyclic, cyclic, directed, and undirected graphs, extending neural network methods and it enhances the ability to manage graph-structured data for both node and graph-level tasks. However, the approach involves significant computational demands due to iterative processes required for parameter estimation and state convergence.The paper [14] introduces a scalable Graph Convolutional Network (GCN) model for semi-supervised learning on graph-structured data, demonstrating improved efficiency and accuracy on

node classification tasks which reduces computational complexity by approximating spectral convolutions and propagates features across graph layers. However, the model faces challenges with memory requirements and is limited in handling directed edges and edge features. The paper [15] introduces the Graph Attention Network (GAT), which extends the attention mechanism to the data with the graph structure, enabling each node to assign different importance to its neighbors and it extensively improves, the computational efficiency and flexibility while achieving state-of-the-art results in both transductive and inductive graph classification tasks. A couple of disadvantages regarding this model are the limitation in batch size and the model does not support edge features. This paper [16] systematically compares a wide range of different GNN models for graph classification and it ensures a fair and reproducible experimental setup on several benchmarks by pointing out inconsistencies in previous evaluations and putting forward a more controlled environment for comparison among such models as GraphSAGE, GIN, and DGCNN. However, the work also points out that some models fail to fully leverage structural information on certain datasets, thus limiting performance gains. The paper [17] introduces the TGNs, a framework for learning on continuous-time dynamic graphs using memory modules and graph-based operators and it shows state-of-the-art performance both in transductive and inductive settings on several datasets. However, as far as scalability on large-scale graphs is concerned, this model still suffers from high memory and computational burdens.The paper [18] introduces House-GAN, a graph-constrained generative adversarial network designed to generate diverse and realistic house layouts based on a bubble diagram as an input which outperforms existing methods in realism, diversity, and compatibility with architectural constraints. However, the model simplifies room shapes to rectangles and lacks considerations for detailed room features like size variations and door placements. The paper [19] presents a general benchmarking framework for Graph Neural Networks, comprising a diverse set of medium-scale graph datasets for the model evaluation of GNN under a fixed parameter budget and they also highlight that this is a fair model comparison and allows researchers to explore novel theoretical insights that would improve present GNN architectures. However, it include challenges in scaling to larger datasets and adapting to different graph properties beyond the scope of benchmarked datasets. The paper [20] Graph2Plan, a deep neural network architecture for generating floor plans from layout graphs and performs user-in-the-loop customization of room constraints and layouts by effectively fusing graph neural networks with convolutional processing for handling complex room configurations. However, it generates flexible output per user input on design requirements and limited capability in the handling of intricate design features such as detailed functionalities in rooms and generation process for including doors and windows. The paper [21] introduces the Computation Graph Transformer (CGT), which generates synthetic benchmark graphs that maintain privacy and scalability while effectively preserving the performance of GNNs on real-world task and also, It demonstrates significant improvements in benchmark effectiveness, privacy guarantees, and scalability for large graph datasets. However, it faces challenges in training stability, particularly with differential privacy mechanisms like DP-SGD, which currently offer impractical or coarse privacy guarantees. The paper [22] introduces GraphSAGE, an inductive framework for generating node embeddings that binds sampling and aggregating features from a node's local neighborhood, which performs effectively on unseen nodes and graphs. It significantly improved node classification tasks across citation, Reddit, and protein-protein interaction datasets but was found to be limited when it came to computational scalability at test time compared with transductive methods like DeepWalk.

3. Methodology

The methodology starts with dataset collection, where floor plan data is gathered. Key features like room coordinates, area, and connectivity are extracted to form nodes and edges, representing the floor plan as a graph. An annotation framework sets standards for labeling floor plans, and these labels are paired with graphs. The dataset is augmented by rotating the floor plans to expand the sample size. After selecting models, node features are normalized and encoded, and the dataset is converted into graph objects. Following the train-test split, models are trained, and the results are analyzed to conclude.

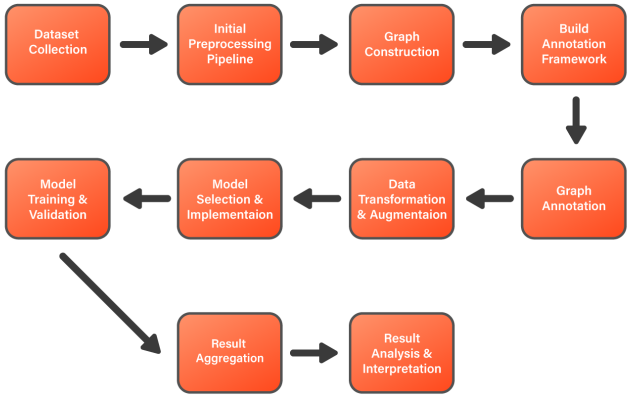


Figure 1: Methodology

3.1. Dataset Overview

The dataset for this study is derived from the RPLAN[2] dataset, containing 80,787 real-world Chinese floor plans. These 256 x 256 pixel pre-processed images capture various room types, layouts, and connections. The floor plan model uses four encoding channels to represent different aspects of the plans.

3.2. Initial Data Preprocessing Pipeline

The dataset is cleaned for graph-based processing by converting raw floor plan images into structured graphs. Binary masks extract contours of key architectural features like walls and doors, simplifying them while retaining essential geometry. These contours are converted into vertices, representing rooms as nodes and doors as edges, capturing spatial relationships. This transformation optimizes the data for further tasks while preserving key architectural details (see Figure 2).

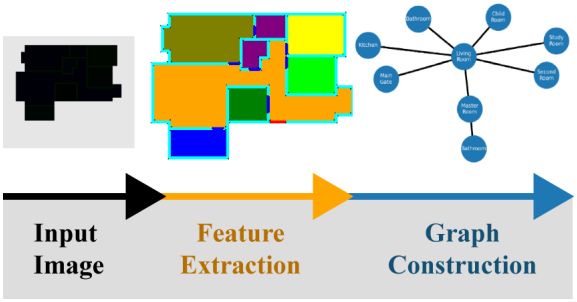


Figure 2: Initial Data Preprocessing Pipeline

3.3. Feature Extraction: Image Encoding and Vertex Data

The feature extraction process uses pre-processed floor plan images from the RPLAN dataset, encoded with a four-channel schema to capture architectural elements. Channel 1 represents exterior walls (127) and the front door (255), while Channel 2 assigns specific integers to various room types, such as 1 for the master room, 2 for the kitchen, up to 17 for the interior door, etc. Channel 3 differentiates rooms with identical labels, and Channel 4 distinguishes between interior (255) and exterior areas (0).

Binary masks are created using Channel 2, isolating architectural features like doors and rooms. From these masks, contours are extracted using the $find\_contours()$ function from the *skimage* library, accurately identifying the boundaries of these elements. The contours are then simplified with $approximate\_polygon()$, reducing the number of vertices while retaining essential geometric features. These vertices are organized into graphs, representing rooms as nodes and doors as edges. The process for one room is shown in Figure 5

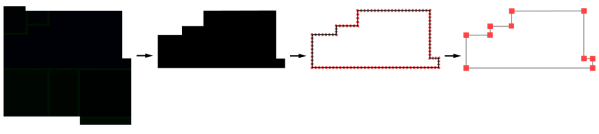


Figure 3: Feature Extraction

3.4. Graph Construction: Nodes and Edges

Nodes are created to represent rooms, with each node assigned features such as room vertices, area, and room type. Edges are established by analyzing spatial relationships and adjacencies between rooms. Doors are connected to the rooms they lead into, forming edges between nodes that represent connected rooms. A custom algorithm determines which doors connect two rooms, and the vertices of the connecting doors are stored as edge features.

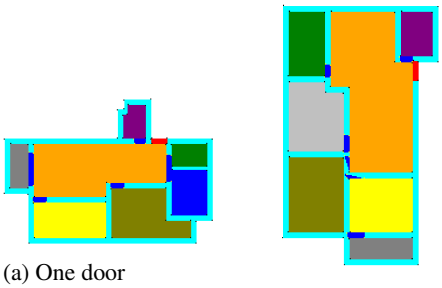
An exception is made for the main gate, which is stored as a dummy node labeled "Main Gate." The main gate's vertex information is stored as an edge feature between the dummy node and the room it connects to.

3.5. Graph Validation and Artifacts

The constructed graph is validated to ensure the data's integrity. During this process, two issues were identified, resulting in a dataset reduction to 51K functional samples.

The first problem is that one door connects multiple rooms, leading to instances where a single interior door connects two rooms. In these cases, the graph only retains one of the connected rooms. If the total number of nodes in the graph does not match the actual number of rooms, that data is excluded from the functional pool. This issue was observed in 11,521 images of the dataset (see Figure 4a).

The second issue involves multiple doors being morphed during the contouring and simplification process, where closely positioned doors are inaccurately combined. If the vertex count for an interior door is not four—since all doors are represented as rectangles—it is not included in the functional data pool. This problem was present in 17,920 images (see Figure 4b), with artifacts within the door outlines highlighting the issue.



(a) One door connecting multiple rooms

(b) Multiple doors morphing

Figure 4: Graph Validation Issues

The initial filtered dataset consists of approximately 51K graphs derived from pre-processed floor plans, highlighting the spatial relationships within residential layouts. From data analysis, it is evident that the Living Room serves as the central hub, showing strong connections to key areas like the Master Room and Kitchen, while rooms such as the Guest Room and Entrance exhibit minimal connectivity. This hierarchical structure of room interactions emphasizes the varying importance of spaces, providing crucial insights

as the dataset moves forward to the annotation and model evaluation stages.

3.6. Annotation Labels in Context of Bangladesh

In Bangladesh, residential architecture must address spatial constraints, efficient zoning, and privacy needs. The labeling system categorizes floor plans into four types: Type A, B, C, and D, reflecting local practices and requirements.

- **Type A:** Type A residences are ideal for proper zoning and optimal space utilization, featuring strong internal connections and clear divisions between public, semi-private, and private areas. Privacy is prioritized. Spatial efficiency minimizes wastage, with room sizes proportionate to their functions, enhancing comfort and usability. In Bangladesh, a north-south orientation is preferred for natural ventilation and airflow, while balconies aid cross-ventilation, making these design elements crucial for comfort and functionality.

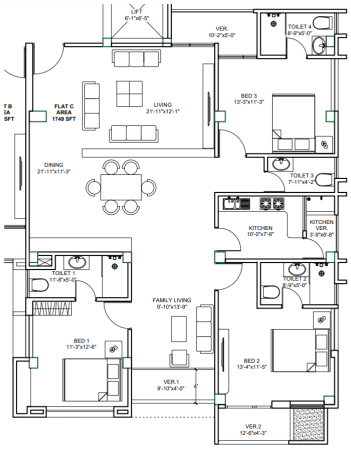


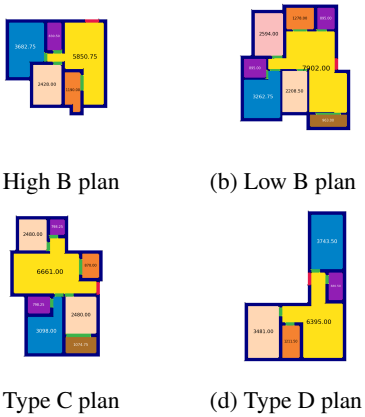
Figure 5: Type A residence interior plan

- **Type B:** Type B residences are functional and meet basic requirements in Bangladesh but lack the design precision and quality of Type A. While zoning and space ratio principles are followed, but the internal relationships between spaces are weaker, leading to less optimal space utilization and privacy concerns.
- **Type C:** Type C residences are not convenient for family residents, because it lacks proper zoning and proper space utilization. Not only that, privacy doesn't come to concern in this type. This type of apartment does exist but is mainly for use as a hostel or mess or for any commercial purpose.
- **Type D:** Type D serves as a benchmark for designs that fail to meet basic residential standards, characterized by disorderly spatial arrangements, improper zoning, and poor space utilization. Architects do not recommend this type of planning, also makes them unsuitable for modern living.

3.7. Data Labeling Workflow and Adjustments

The classification of residential floor plans into Types A, B, C, and D revealed insights into zoning, space utilization, and internal relationships that affect livability and functionality. Through consultations with a professional architect, we refined our classification system to better suit the needs of Bangladeshi architecture. Architects trained exactly 13 architecture students iteratively for the annotation process. Despite reviewing 800 to 1,000 plans, none met Type A criteria, prompting the creation of subcategories High B and Low B to better represent plans needing minor modifications or those meeting basic functional requirements. This adaptive approach ensures the classification system aligns with professional architectural experience in Bangladesh.

- **High B:** The High B category includes floor plans that are close to Type A standards but require slight modifications, such as repositioning entrances or adjusting room orientations.
- **Low B:** The Low B category includes functional family residences that compromise on optimal room relationships and spatial arrangements, making them less suitable for larger families. Similar to common Type B designs in Bangladesh, they remain adequate for smaller households or couples, despite issues with space utilization and circulation flow.



(a) High B plan (b) Low B plan

(c) Type C plan (d) Type D plan

Figure 6: High B, Low B, Type C, and Type D residences interior plan

The website employs a voting mechanism allowing participants to classify floor plan images into High B, Low B, C, or D categories, with the most voted category assigned to each image, provided there are at least five votes from different annotators. Our machine learning model converts floor plans into bubble diagrams, representing rooms as nodes with essential information, though it does not account for ventilation

Table 1
Comparison of Interior Floor Plan Classifications

Types	Zoning	Space Utilization	Privacy	Functionality
High B	Proper zoning	Proper Space Management	Privacy Ensured	Smooth Functionality
Low B	Functional zoning	Functional Space Utilize	Minimal Privacy	May Consist Proper Functionality
Type C	Lack of proper zoning	Decent Space Utilization	No Privacy	Confusing Functionality
Type D	No Zoning	No Space Management	No Privacy	Proper Functionality Absent

and orientation, critical for livability in Bangladesh. After filtering and annotating 9,028 plans from the original dataset of 51,000 due to time constraints, the results included 897 plans classified as High B, 3,165 as Low B, 2,753 as Type C, and 2,213 as Type D.

3.8. Final Data Pre-Processing

The final annotated dataset consists of 9,028 graphs selected chronologically from the original 51K dataset for annotation, revealing significant differences in spatial relationships. While the initial filtered dataset emphasized the Living Room as the central hub of connectivity, this refined analysis reinforces its role as the largest and most connected space. The Master Room and Second Room also show notable connectivity, while smaller rooms like the Dining Room and Child Room exhibit limited presence in many designs. This highlights a hierarchy of room importance, with less frequent interactions observed in the broader dataset being filtered out. Overall, the final dataset offers a concentrated view of room connectivity, reflecting prioritized design elements while potentially missing some patterns present in the larger dataset.

3.9. Dataset Transformation and Augmentation

The annotations collected from multiple annotators are aggregated based on the majority vote, with data obtained from the annotation site in JSON format. This data is then transformed into an HDF5 file, which includes the floor plan information and the corresponding label values. For the binary classification task, labels "A" and "B" are converted to "1" (indicating a positive response), while labels "C" and "D" are transformed into "0" (indicating a negative response). To streamline the dataset, edge features from the filtered data are removed, leading to the loss of valuable edge information. To preserve the crucial front door information captured during annotation, a new room type labeled "front door" is created. This new room is connected to the room where the original dummy node was linked, using the centroid, area, and edge feature details associated with the front door. Subsequently, the dummy node and its edge are dropped from the dataset.

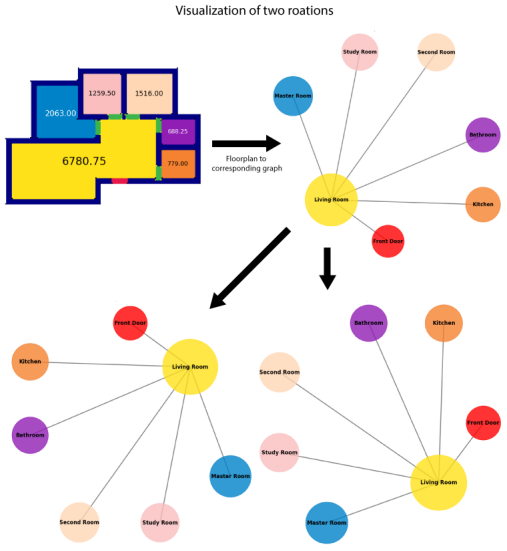


Figure 7: Data augmentation through rotation

To enhance the dataset size and address the class imbalance, data augmentation is performed through rotation. The labeled instances are rotated to create new samples while preserving their classifications; specifically, instances labeled "1" are rotated 6 times at angles of 0° (unchanged), 45°, 90°, 135°, 180°, and 270°, while instances labeled "0" are rotated 7 times at the same angles, plus an additional 225°. This action is demonstrated in Figure 7. This approach results in a balanced total dataset of 58,230 labeled graphs—4,062 initially labeled "1" and 4,966 labeled "0"—helping to combat the class imbalance problem. By reinforcing patterns within the data, the augmentation allows the model to explore various spatial configurations, improving its learning capabilities despite the creative nature of the task.

4. Model Selection

The primary goal of this study is to perform graph-level binary classification to assess whether a given architectural floor plan adheres to Bangladeshi architectural standards. To achieve this, models must effectively handle graph-structured data, capture complex

relationships, and generalize well to unseen samples. The key criteria for model selection were:

- **Ability to Handle Graph-Structured Data**: Models need to process and learn from graph representations of floor plans.
- **Scalability**: Models should accommodate large datasets and complex graphs.
- **Expressiveness**: The ability to capture both local and global patterns within graphs is essential.
- **Computational Efficiency**: Models must offer reasonable training and inference times for practical use.

Based on these criteria, we selected three Graph Neural Network (GNN) models known for their strengths in graph classification tasks:

- **GCN**: Efficient in capturing general graph structures using spectral convolution.
- **GAT**: Leverages attention mechanisms to prioritize important node features.
- **GraphSAGE**: Uses inductive learning with flexible neighborhood aggregation, making it suitable for large graphs.

5. Data loading and Train-test Split

5.1. Feature Representation and Graph Construction

Effective feature representation is key to the performance of GNN models. In this study, both nodes and edges in the graph are associated with specific features that capture geometric and semantic information.

Node Features: Each node, representing a room, has the following features:

- **Normalized Area**: Calculated as:

$$\text{normalized\_area} = \frac{\text{area} - \text{min\_area}}{\text{max\_area} - \text{min\_area}}$$

- **Normalized Centroid Coordinates**: (c_x, c_y), normalized to the range [0, 1] based on the maximum coordinate value:

$$c_x = \frac{x_{\text{centroid}}}{\text{max\_coordinate}}, \quad c_y = \frac{y_{\text{centroid}}}{\text{max\_coordinate}}$$

- **One-Hot Encoded Room Type**: A binary vector representing the room type (13 types).
- **Positional Encodings**: Sine and cosine functions applied to the centroid coordinates using the formula from [3]:

$$\text{PE}_{(p,2i)} = \sin\left(\frac{p}{10000^{2i/d_{\text{model}}}}\right),$$

$$\text{PE}_{(p,2i+1)} = \cos\left(\frac{p}{10000^{2i/d_{\text{model}}}}\right)$$

where p is the normalized centroid position, and d_{model} is the encoding dimensionality (set to 64). This encodes spatial relationships between rooms.

Edge Features: Edges represent room connections.

Graph Construction: The graph construction process begins with data extraction, where node and edge attributes such as area, centroids, and room types are read from HDF5 files. Following this, the features are prepared by normalizing the area and centroid coordinates, one-hot encoding the room types, and computing positional encodings based on the centroid positions. These features are then concatenated to form the node feature matrix X. Edge connections between rooms are formatted into an edge index tensor that is compatible with PyTorch Geometric. Each graph is assigned a binary label for graph-level classification. Finally, the node features, edge indices, and labels are encapsulated into PyTorch Geometric's Data class, which prepares the graph data for training.

5.2. Train-Test Split

From the 58,230 floor plans, a stratified split ensures an even distribution of classes across training, validation, and test sets. After balancing the dataset by selecting an equal number of samples from both classes (28,434 samples per class), the data is split in a 60-20-20 ratio, yielding 34,120 training samples, and 11,374 samples each for validation and test sets. This stratification ensures class proportions are maintained across all split sas signified in the table 2.

6. Model Architecture

6.1. Abstract Model Architecture

The custom models in this study follow a common abstract architecture designed for classifying architectural floor plans. This architecture comprises the following key components:

- **Graph Convolutional Layers:** All models use graph convolutional layers (e.g., GCNConv, GATConv, SAGEConv) to aggregate and transform node features, enabling the model to learn spatial and topological relationships between rooms in the floor plans.
- **Global Pooling:** After the graph convolutional layers, a global pooling mechanism (mean, max, or a combination of both) is applied to condense node-level embeddings into a single graph-level representation. This step is crucial for producing a fixed-size vector that summarizes the entire floor plan.

Table 2
Train-Validation-Test Split with 60-20-20 Ratio

Set	Class 0	Class 1	Total	Split Ratio
Balanced	28,434	28,434	56,868	100%
Training	14,217	14,217	34,120	60%
Validation	5,687	5,687	11,374	20%
Test	5,687	5,687	11,374	20%

- **Fully Connected Network (FCN):** The pooled graph representation is passed through a fully connected network, consisting of one or more linear layers. These layers further refine the graph-level features and produce the final classification output.
- **Binary Classification Output:** The final output layer performs binary classification, determining whether a given floor plan conforms to the specified architectural standards.
- **Loss function and Optimizer:** The binary cross-entropy [11] loss function (via nn.CrossEntropyLoss) is used to optimize the model. The model parameters are updated using the Adam optimizer [4]. Additionally, L2 weight decay is applied to regularize the model.

The details of individual model architecture are highlighted in the table 3.

6.2. Hyperparameter Summary

The hyperparameter tuning is done using Optuna library [23] which uses a form of Bayesian optimization called TPE (Tree-structured Parzen Estimator) for hyperparameter tuning. The best hyperparameters found are used for each model.

7. Result Analysis

The models are evaluated based on statistical metrics, and the final results are analyzed and compared.

7.1. Model Comparison

The figures 8, 9 and 10 are showing the models GAT, GCN and GraphSAGE graphs from left to right respectively. The **GCN** and **GraphSAGE** models exhibit similarly strong performances across various metrics, whereas the **GAT** model struggles in comparison. Both GCN and GraphSAGE achieve comparable validation accuracy, loss patterns, and AUC scores, indicating their effectiveness for the task, while GAT displays lower performance and instability.

GCN and GraphSAGE Performance:

Accuracy and Loss: Both GCN and GraphSAGE show a steady improvement in accuracy and reduction in loss over the course of 300 epochs. They start with higher losses (0.60-0.65) and smoothly decline to a training loss of approximately **0.25** and a validation loss stabilizing between **0.30** and **0.35**. The final validation accuracy for both models reaches around **85%**, demonstrating their ability to generalize well to unseen data. Their training accuracy also stabilizes at around **0.85**, highlighting consistent learning from the data without signs of overfitting or underfitting.

Confusion Matrix: For the GCN model, 4945 instances of Class 0 and 4949 instances of Class 1 are correctly classified, indicating balanced classification. Similarly, the GraphSAGE model correctly classifies 4973 instances of Class 0 and 4933 instances of Class 1. This demonstrates that both models perform comparably well in terms of classifying the two classes.

AUC Scores: Both models achieve high AUC scores on the precision-recall and ROC curves, underscoring their strong discriminative power. GCN records AUC scores of **0.954** for precision-recall and **0.9507** for the ROC curve, while GraphSAGE scores **0.9543** for precision-recall and **0.9512** for the ROC curve. These results confirm that both models are well-suited for distinguishing between the two classes.

GAT Performance:

In contrast to GCN and GraphSAGE, the GAT model exhibits weaker performance and significant instability:

Accuracy and Loss: The GAT model starts with a training loss of around **0.65** but struggles to minimize the loss, plateauing at approximately **0.50**. Its validation loss fluctuates considerably, oscillating between **0.80** and **0.90**, particularly after 100 epochs, indicating issues with generalization. The final validation accuracy is unstable, fluctuating between **0.40** and **0.70**, which further suggests poor generalization compared to GCN and GraphSAGE.

Confusion Matrix: The confusion matrix for GAT reveals a higher number of misclassifications, particularly for Class 1, where 1945 instances are incorrectly classified as Class 0. This imbalance in classification accuracy highlights GAT's inability to accurately distinguish between the two classes.

AUC Scores: The AUC scores for GAT further reflect its lower performance, with the precision-recall curve yielding an AUC score of **0.84** and the ROC curve showing an AUC score of **0.86**. These values are considerably lower than those of GCN and GraphSAGE, underscoring GAT's limited discriminative ability.

Summary:
GCN and GraphSAGE perform similarly and show strong overall performance. Their loss and accuracy

Table 3
Comparison of Custom GNN Model Architectures

Feature	GCN	GraphSAGE	GAN
Number of Convolution Layers	2	2	2
Type of Convolution Layers	GCNConv	SAGEConv	GATConv
Aggregation Function	Spectral Convolution	Mean	Attention
Node Feature Transformation	80 → 256, then 256	80 → 256, then 256	80 → 512
Consistency in Embedding Size	Maintained at 256	Maintained at 256	Maintained at 512
Attention Mechanism	N/A	N/A	Yes (single head)
Pooling Mechanism	Global Mean Pooling	Global Mean Pooling	Global Max and Mean Pooling
Fully Connected Network (FCN)	Two Linear Layers: 64 → 64, 64 → 2	Two Linear Layers: 256 → 256, 256 → 2	Two Linear Layers: 512 → 512, 512 → 2
Dropout Rate	N/A	0.06	0.271
Output Classes	Binary Classification (2 classes)	Binary Classification (2 classes)	Binary Classification (2 classes)
Learning Rate	0.001	0.0005	0.00007
Batch Size	32	16	32
Optimizer	Adam (no weight decay)	Adam (weight decay=1e-6)	Adam (weight decay=4.5e-5)

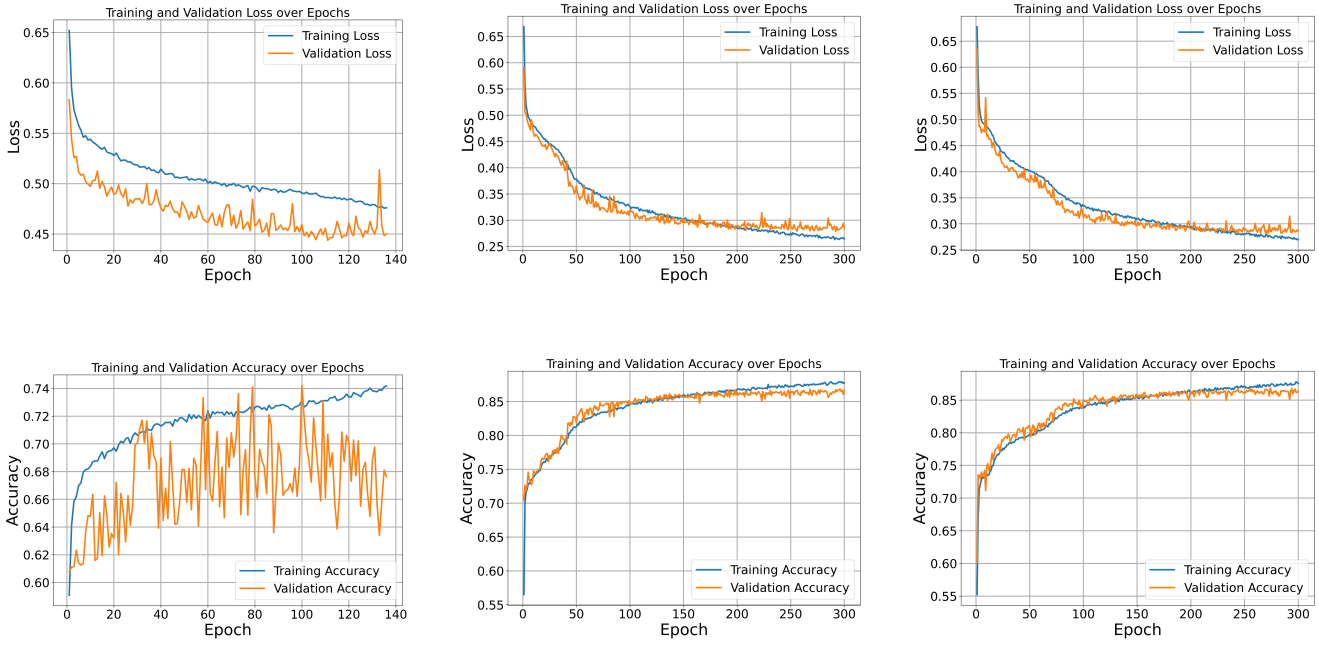


Figure 8: Learning Curves for Different Models

curves indicate efficient learning, good generalization, and stable convergence without overfitting. In contrast, GAT lags behind both models, with slower convergence, higher loss, and unstable validation accuracy.

7.2. Test Results

The models are evaluated across various metrics necessary to determine their competence in capturing the patterns in architectural floor plans and distinguishing between them as shown in table 4. Here, **Accuracy** measures the overall correctness of the model. Although it can be misleading in imbalanced datasets, we trained, validated and tested our with equal number of class present in the dataset. **Precision** indicates how many of the predicted positive cases are actually true positives, it is useful to understand which model has the lowest false positive. **Recall** (or sensitivity) measures the ability to identify all true positives, which

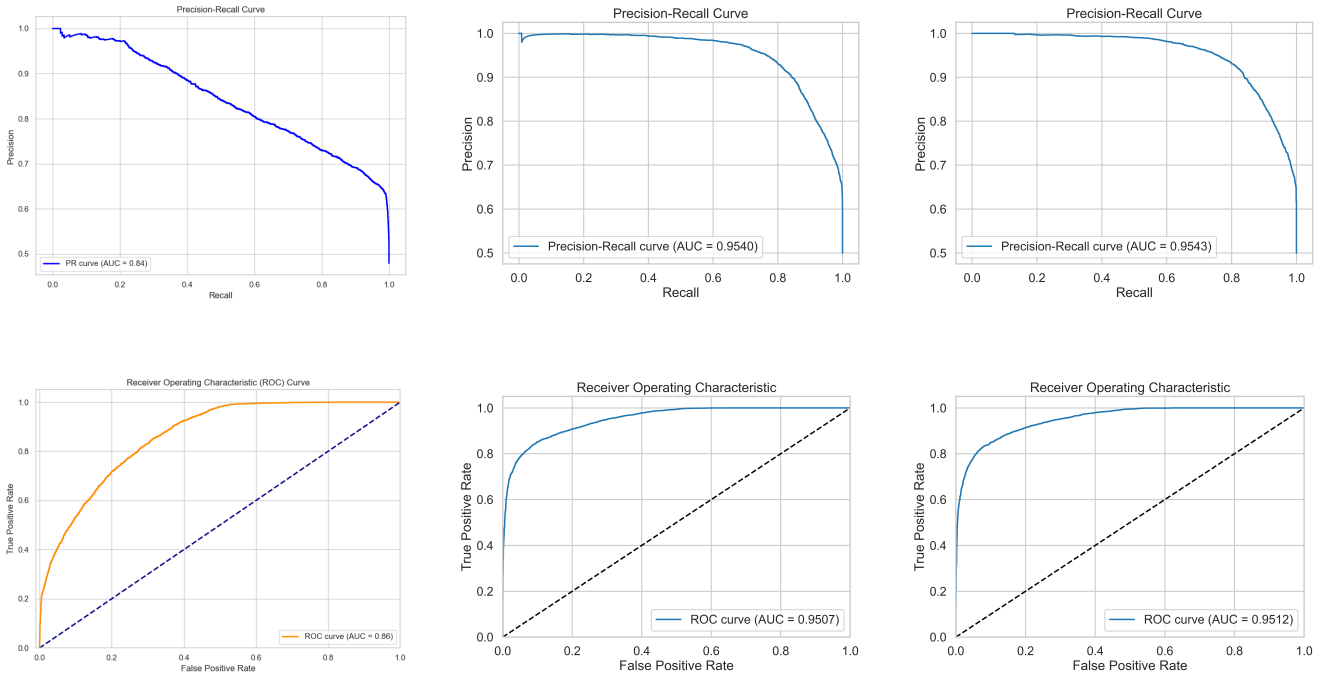


Figure 9: Performance Curves for Different Models

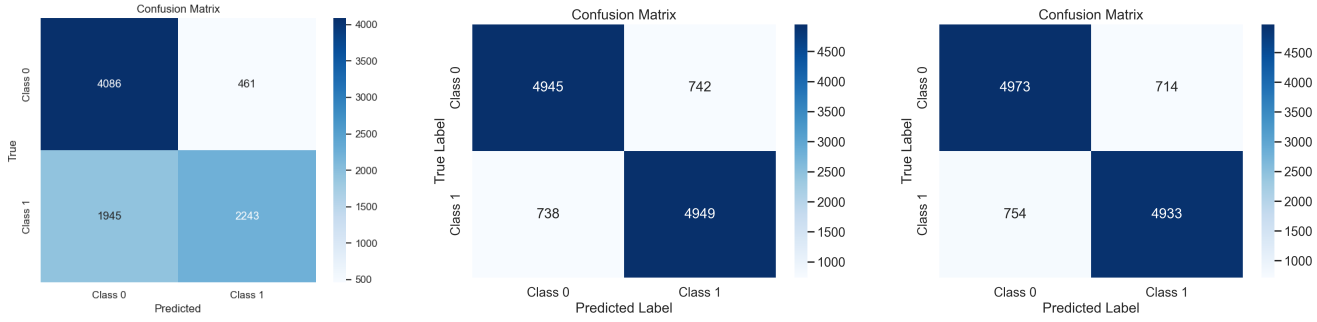


Figure 10: Confusion Matrix of Different Models

in our case is useful to determine which model can the extract the highest number of applicable floor plan from the total dataset. **F1 Score** shows the balance between precision and recall. Lastly, **AUC-ROC** evaluates a model's ability to distinguish between classes. This is very important for our task which is binary classification.

The table clearly shows GCN and GraphSAGE outperform GAT across all the metrics.GraphSAGE shows marginally better performance than GCN in most areas, particularly in precision and AUC-ROC. Although GAT has decent accuracy and precision, it falls short in all other metrics. GAT struggles with recall, leading to a lower F1 score. On the other hand, GCN and GraphSAGE showcase excellent accuracy, precision, and recall. Both of them have very good

Table 4
Test Results for GAT, GCN, and GraphSAGE Models

Metric	GAT	GCN	GraphSAGE
Loss	0.4462	0.2806	**0.2787**
Accuracy	0.7246	0.8699	**0.8709**
Precision	0.8295	0.8696	**0.8736**
Recall	0.5356	**0.8702**	0.8674
F1 Score	0.6509	0.8699	**0.8705**
AUC-ROC	0.8609	0.9507	**0.9512**

AUC-ROC scores which indicates they can distinguish between the classes properly. GraphSAGE is the

best model based on the results. It has better scores in all metrics except recall.

The results suggest that GCN and GraphSAGE models are the most suitable for tasks requiring balanced accuracy, precision, and recall. In contrast, GAT struggles with recall is unsuitable for most cases compared to GraphSAGE. Overall, GraphSAGE is the clear outlier in terms of its performance, but GCN is not too far behind.

8. Future Work and Conclusion

Adapting foreign architectural data for Bangladesh presents challenges due to the lack of critical factors such as orientation and ventilation, essential for residential design in its hot and humid climate. We utilized a dataset of 51,000 Chinese floor plans, enhancing it to better suit Bangladesh's needs. However, the model does not account for crucial aspects like orientation and ventilation, which significantly affect livability. Enhancing the dataset with metadata on orientation, such as north-south facing facades, and incorporating window placement for ventilation, would improve the accuracy of spatial assessments.

In our current research, we annotated 9028 floor plans with 4 labels converted to 2, but plan to expand the dataset and introduce multiclass classification. The NENN model [9], used in this paper, improves on our earlier GraphSAGE model by integrating both node and edge features, including door locations, through a dual-level attention mechanism. This allows the model to better capture spatial relationships, enhancing classification accuracy for Bangladeshi floor plans. We also plan to incorporate DIFFPOOL [12], an advanced pooling technique that enables hierarchical graph representations by clustering nodes, capturing both local and structural patterns. This would improve classification of complex floor plans by effectively utilizing edge features like door locations, essential for distinguishing architectural structures in Bangladesh.

Graph Neural Networks (GNNs) show promise for classifying architectural floor plans, with GCN and GraphSAGE outperforming GAT in accuracy, precision, and generalization. Despite challenges from ambiguities in classifying architectural features, GCN and GraphSAGE demonstrated effectiveness for binary classification, suggesting future work could enhance model complexity and representational learning by incorporating dynamic node and edge features.

References

[1] S. Leng, Y. Zhou, M. H. Dupty, W. S. Lee, S. Joyce, W. Lu, Tell2Design: A dataset for language-guided floor plan generation, in: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Toronto, Canada, 2023, pp. 14680–14697. URL: https://aclanthology.org/2023.acl-long.820. doi:10.18653/v1/2023.acl-long.820.

[2] W. Wu, X.-M. Fu, R. Tang, Y. Wang, Y.-H. Qi, L. Liu, Data-driven interior plan generation for residential buildings, ACM Transactions on Graphics (SIGGRAPH Asia) 38 (2019).

[3] A. Vaswani, Attention is all you need, Advances in Neural Information Processing Systems (2017).

[4] D. P. Kingma, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

[5] H. Hua, Irregular architectural layout synthesis with graphical inputs, Automation in Construction 72 (2016) 388–396.

[6] X.-Y. Gong, H. Su, D. Xu, Z.-T. Zhang, F. Shen, H.-B. Yang, An overview of contour detection approaches, International Journal of Automation and Computing 15 (2018) 656–672.

[7] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, M. Sun, Graph neural networks: A review of methods and applications, AI open 1 (2020) 57–81.

[8] H. Park, H. Suh, J. Kim, S. Choo, Floor plan recommendation system using graph neural network with spatial relationship dataset, Journal of Building Engineering 71 (2023) 106378.

[9] Y. Yang, D. Li, Nenn: Incorporate node and edge features in graph neural networks, in: Asian conference on machine learning, PMLR, 2020, pp. 593–608.

[10] M. Yamada, X. Wang, T. Yamasaki, Graph structure extraction from floor plan images and its application to similar property retrieval, in: 2021 IEEE International Conference on Consumer Electronics (ICCE), 2021, pp. 1–5. doi:10.1109/ICCE50685.2021.9427580.

[11] A. Mao, M. Mohri, Y. Zhong, Cross-entropy loss functions: Theoretical analysis and applications, in: International conference on Machine learning, PMLR, 2023, pp. 23803–23828.

[12] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, J. Leskovec, Hierarchical graph representation learning with differentiable pooling, 2019. URL: https://arxiv.org/abs/1806.08804. arXiv:1806.08804.

[13] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE Transactions on Neural Networks 20 (2009) 61–80.

[14] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, 2017. arXiv:1609.02907.

[15] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, 2018. arXiv:1710.10903.

[16] F. Errica, M. Podda, D. Bacciu, A. Micheli, A fair comparison of graph neural networks for graph classification, arXiv preprint arXiv:1912.09893 (2019).

[17] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, M. Bronstein, Temporal graph networks for deep learning on dynamic graphs, 2020. arXiv:2006.10637.

[18] N. Nauata, K.-H. Chang, C.-Y. Cheng, G. Mori, Y. Furukawa, House-gan: Relational generative adversarial networks for graph-constrained house layout generation, in: European Conference on Computer Vision (ECCV), Springer, 2020, pp. 162–177.

[19] V. P. Dwivedi, C. K. Joshi, A. T. Luu, T. Laurent, Y. Bengio, X. Bresson, Benchmarking graph neural networks, 2022. arXiv:2003.00982.

[20] R. Hu, Z. Huang, Y. Tang, O. van Kaick, H. Zhang, H. Huang, Graph2plan: Learning floorplan generation from layout graphs, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 1–11.

[21] M. Yoon, Y. Wu, J. Palowitch, B. Perozzi, R. Salakhutdinov, Graph generative model for benchmarking graph neural networks, arXiv preprint arXiv:2207.04396 (2022).

[22] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, Advances in neural information processing systems 30 (2017).

[23] T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, Optuna: A next-generation hyperparameter optimization framework, in: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery data mining, 2019, pp. 2623–2631.