Room Classification from Connected Floor Plan Graphs

Nicholas DeGroot
DSC 180A: Data Science Capstone
University of California San Diego
La Jolla, CA 92122
ndegroot@ucsd.edu

Abstract

Recent work in computer vision has explored using machine learning to digitize the floor plan: the architectural diagrams home builders use when constructing a new home. Most work has been focused in "raster-to-vector" approaches, which attempt to turn raster images of the floor plan into a set of vectors describing a wall, window, or other 2D object. While post-processing has shown some success in associating detected rooms with their purpose, these approaches largely rely on supplementary information such as text descriptions and icon detection. We explore whether the spatial information of detected rooms is sufficient to determine its purpose. In particular, we show that a graphical encoding of room adjacencies contains enough information alone to achieve results far better then chance.

1 Introduction

Home builders and architects use floor plans everyday to coordinate construction efforts on complex projects. These documents detail exactly where every wall, window, and door is located, as well as the purpose of each room. Historically, they are shared through raster formats in printed documents or their modern PDF equivalent. While these formats are convenient for sharing, they are not easily machine-readable. In recent years, computer vision has been used to digitize these plans, turning them into a set of vectors describing the location of each wall, window, and door. This process is known as "raster-to-vector" conversion, and was first popularized for use with floor plans by Liu et al. [7]. These digitized floor plans can then be used for a variety of downstream applications, such as higher fidelity collaboration, construction cost estimation, and 3D model generation.

While raster-to-vector approaches have shown great success, they are not without their limitations. Most approaches rely on supplementary information such as text descriptions and icon detection to associate detected rooms with their purpose - information that isn't available on other types of floor plans. For example, LiDAR sensors in modern iPhones have been able to generate a 3D floor plan by detecting room dimensions and locations through RoomPlan, but don't embed any textual information about the room purpose [1].

Our approach echos that of Paudel et al. [8], which relied only on the spatial information of detected rooms to determine their purpose. Specifically, we encode the spatial information of each room in an adjacency graph, with nodes describing rooms and edges describing adjacent rooms. We examine the performance of a variety of graph neural networks (GNNs) on this task, as well as how they compare to their traditional multi-layer perceptron counterparts. We differ from Paudel et al. [8] in that we use a more detailed but smaller dataset in Cubicasa5k [5]. This allows us to explore whether the models can generalize to more specific floor plans, allowing us to classify up to 35 "top-level" room categories and 69 "sub-level" room categories. Our experiments show that a purely graphical approach to floor plan classification is capable of promising results.

2 Methods

2.1 Data Cleaning & Preprocessing

We start with the SVG-labeled floor plans from Cubicasa5k [5]. Each SVG describes a particular home, with each room labeled and bounded by a polygon. This is easily extracted via BeautifulSoup4 with the `lxml` parser. From there, we extract the following metadata:

1. **Room Name**: The name of the room, e.g. "Bedroom", "Attic", etc.
2. **Polygon**: The polygon describing the room's boundary. This is modeled using Shapely.

We further calculate the following attributes for each room:

1. **Room Category**: The generic category of the room. Included categories:
    - Corridor
    - Storage
    - Dining
    - Room
    - Kitchen
    - Bedroom
    - LivingRoom
    - Bath
    - Outdoor
    - Garage
2. **Width**: The maximum width of the room polygon, in pixels.
3. **Height**: The maximum height of the room polygon, in pixels.
4. **Area**: The area of the room polygon, in pixels.

The distribution of room category can be found in Figure 1. The distribution of room width/height can be found in Figure 2

For our last room-level attribute, we parse out all doors in the document using a similar method to rooms. We then buffer or extend each room's polygon by a small percentage, and check if the door polygon is contained within the room. If so, we add the door to the room's list of doors, and add the total number to each room.

For our graph-level attributes, we calculate all room adjacencies as edges. We do so much the same as we did for doors, but instead of checking if a door is contained within a room, we check if two rooms are adjacent. We define two rooms as adjacent if their buffered polygons intersect, or if one polygon is contained within the other.

Finally, we format the graph into a PyTorch Geometric Data object, which is the common language that all of our models speak. We use the width, height, area, and number of doors as our node attributes, and the adjacency as our edge attributes. Since this whole process can take awhile, we preprocess the entire dataset and save it to disk. This allows us to quickly load the data when training. The end result is approximately 40MB of compressed tensor data.

2.2 Modeling

We model the data using a variety of neural networks. We start with a simple baseline model: a simple feed-forward neural network. We then explore a variety of graph neural networks, including GraphSAGE [4], Graph Convolutional Networks (GCN) [6], Graph Attention Networks (GAT) [9], and Topology Adaptive Graph Convolutional Networks (TAGCN) [2].

For each model, we use the Pytorch Geometric library [3] to implement a node-level classifier model that attempts to predict the room category. A hyperparameter determines how many layers of the associated model are used. After each layer, the outputs are ReLU activated and passed through. The model ends with a final linear classifier layer, which outputs the probability of a node belonging to a particular room category.

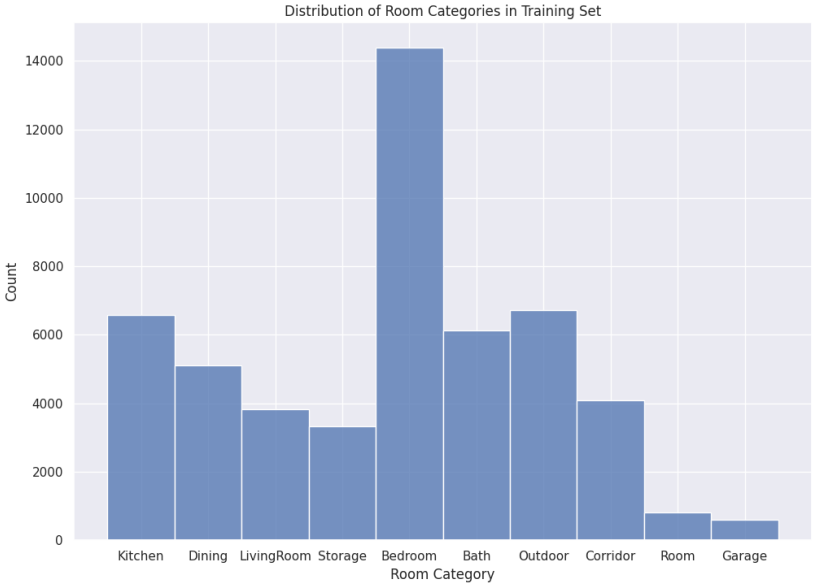

Figure 1: Distribution of room categories in the training dataset.

Figure 2: Distribution of width/height in the training dataset.

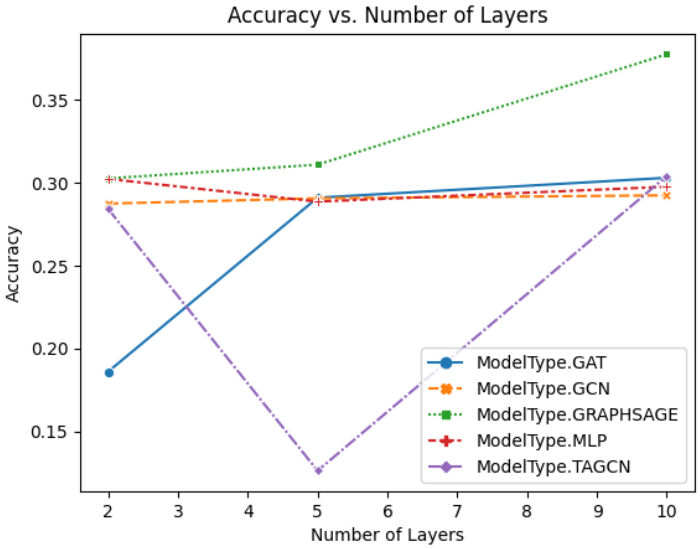

Figure 3: Accuracy of each model on the test set.

Model Type	Accuracy	# Layers
GAT	0.303109	10
GCN	0.292511	10
GraphSAGE	0.377710	10
MLP	0.302429	2
TAGCN	0.304149	10

Figure 4: Top accuracy for each model type.

We use the Adam optimizer with a learning rate of 0.004 and a exponential rate scheduler with a gamma of 0.8. A batch size of 128 is used across 100 epochs. Cross entropy loss is used as our loss function. After competition, we save the best model by validation accuracy (per epoch) to disk.

All models were trained on an NVIDIA GeForce RTX 3080 GPU.

3 Results

To compare our models, we use the pre-portioned test dataset from Cubicasa5k and evaluate the models on the top-level room categories. Figure 3 shows the accuracy of each model on the test set by number of hidden layers. The best seen combination came from the 10-layer GraphSAGE model, with an accuracy of 37.78%. Even when restricted to 2 or 5 layers, the GraphSAGE model preformed best with accuracies of 30.27% and 31.10% respectively. The top reported accuracy for each model can be found in Table 4.

4 Discussion

These results were surprising to us, as the original paper this study was based on reported much higher accuracies for nearly all models. While our models were barely able to scratch 40% accuracy, Paudel et al. [8] reported accuracies of 50% and above, with the best models approaching 82%. We believe that much of this is due to the difference in dataset. The dataset used in Paudel et al. [8] was much larger, with a training set of over 20,000 floor plans, compared to our 4,194. Their dataset was also less diverse, with only 8 different room categories, compared to our 10. Finally, their methods were able to extract two additional attributes: whether a room is a parent or child room. We believe

that these differences in dataset size and complexity are the primary reason for the discrepancy in results.

In addition, the ordering of model performance was different then reported by Paudel et al. [8]. By 10-layers, a pretty clear model hierarchy of:

1. TAGCN
2. GraphSAGE
3. MLP
4. GCN
5. GAT

established itself. This is in contrast to our results, which have all models but GraphSAGE performing roughly the same. Most surprising is the TAGCN model, which we recorded as the worst performing with 5 layers (not doing much better then chance) as opposed to Paudel et al. [8]'s best performing model. We can only speculate as to why this is, but believe it may be due to the differences in dataset discussed prior.

Regardless, we were still surprised at how well each model was at representing the data. Despite only having a grand total of 842 parameters (including a fully connected classifier!), the 2-layer GraphSAGE model was able to achieve an accuracy of 30.27% - well above chance.

5 Conclusion & Future Work

Overall, our experiments showed that graph neural networks are a promising approach to floor plan analysis. While we were unable to match the results of Paudel et al. [8], we were still able to achieve a respectable accuracy of 37.78%.

We believe that with a larger dataset and more complex models, we could achieve even better results. Given more time, we would like to explore a number of possible ways to better represent the data, including:

- Encoding the shape of the room through a neural net
- Including recognized appliances
- Using different methods for calculating room adjacencies
- Experimenting with deeper models

Perhaps more practically, our methods could be combined with other methods of floor plan analysis, such as image recognition on the raster floor plan. Raster versions of floorplans are far more common then their graphical counterparts, and could be used to provide additional information about the floor plan. For example, the image could be used to identify appliances and extract the true dimensions of a room (instead of relative pixels).

6 Bibliography

[1] Apple Developer. Roomplan overview. URL `https://developer.apple.com/augmented-reality/roomplan/`.

[2] Jian Du, Shanghang Zhang, Guanhang Wu, Jose M. F. Moura, and Soummya Kar. Topology adaptive graph convolutional networks, 2017. URL `https://arxiv.org/abs/1710.10370`.

[3] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[4] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs, 2017. URL `https://arxiv.org/abs/1706.02216`.

[5] Ahti Kalervo, Juha Ylioinas, Markus Häikiö, Antti Karhu, and Juho Kannala. Cubicasa5k, March 2019. URL `https://doi.org/10.5281/zenodo.2613548`.

[6] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2016. URL https://arxiv.org/abs/1609.02907.

[7] Chen Liu, Jiajun Wu, Pushmeet Kohli, and Yasutaka Furukawa. Raster-to-vector: Revisiting floorplan transformation. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2214–2222, 2017. doi: 10.1109/ICCV.2017.241.

[8] Abhishek Paudel, Roshan Dhakal, and Sakshat Bhattarai. Room classification on floor plan graphs using graph neural networks, 2021. URL https://arxiv.org/abs/2108.05947.

[9] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2017. URL https://arxiv.org/abs/1710.10903.