2D Floor Plan Reconstruction Using Cool Deep Learning Methods

Jitendra Pandey
Stanford University
Stanford, CA
jnpandey@gmail.com

Gandharv Mahajan
Stanford University
Stanford, CA
gandharv@stanford.edu

Sahil Tadwalkar
Stanford University
Stanford, CA
stadwalk@stanford.edu

Abstract

Point cloud data brings unique challenges for segmentation in deep learning models due to its irregular format, however it captures additional features such as point density which are not available in images. 3D voxelization of point clouds leads to a bloated dataset that is largely sparse. There has been recent effort in applying convolutions directly to the point cloud scenes for the segmentation of real world objects. In this paper, we apply convolutional neural networks to the problem of floor plan reconstruction from 3D point clouds of large buildings. We explore and compare two distinct approaches. U-Net is used to segment the 2D image projection of point cloud data. In this approach we fine tune a pretrained segmentation model with our dataset. Another approach uses the PointNet segmentation network directly on the 3D point cloud. In this method we train a PointNet from scratch. We demonstrate that PointNet performs better than U-Net because 3D models fit better for large geometrical structures in the layout of the floor plan. PointNet takes advantage of the point density of the point cloud.

1. Introduction

Existing buildings often don't have complete floor plan information, for reasons such as age, misplaced records, or miscommunication during a transfer of ownership. Without these floor plans, building owners cannot make informed decisions on upgrades, retrofits, and maintenance for code compliance, which can result in excess cost and delays. Today, ubiquitous and cost effective technologies such as smartphones and Lidar scanners can generate point clouds for the interior of a building.

An important step in floor plan reconstruction is segmentation of different components, such as doors, walls, floors, etc. The most critical component are walls, which create the basic outline of a floor plan. In this paper, we perform binary classification, and evaluate two distinct convolutional neural networks to classify walls from point cloud data. The input data consists of raw point cloud data as well as labels for the points obtained from ground truth files. The output of this network are all the points labeled as walls. With this work, we can empower building owners to have accurate and precise records of their existing buildings.

2. Related Works

2.1. Features

Features in a floor plan such as doors, walls, stairs and floors need to be determined from the point cloud data in order to begin the floor plan reconstruction process. Without this preliminary step, neural networks have a difficult time differentiating between noise and classes.

Mahmood et al. [11] proposed a smartphone assisted solution that exploits deep learning to automatically generate the detailed layout of the floor plan. Point-clouds were obtained from a simple scan by a mobile phone like Google's Tango device. The first step of their process was to identify walls of different rooms from the given point-clouds and the second, was room detection from those walls.

Then they adopted a 2D histogram based technique to identify only the points representing the walls from the entire point-cloud of size N. The histogram was produced by projecting all points to the floor and counting the number of points that fall into each bin. This allowed an image of wall-like structures to be developed, which was used for room extraction.

Two methods for feature extraction were evaluated; a Pointer Network (PtrNet) [17], and a Mask R-CNN [5]. The models were trained on synthetic data, and the experiments show satisfactory results in real-world environments. However, this approach relies on appropriate thresholding to identify higher density wall structures.

Liu et al. [10] has proposed FloorNet, which aims to automatically reconstruct a floor plan simply by walking through a house with a smartphone in a pocket. They used a novel hybrid DNN architecture for extracting features from RGBD videos, and developed a new floorplan reconstruction benchmark.

2.2. Clustering

Clustering is an important step to segmenting point cloud data into their appropriate classes, as occlusion, rotation, noise and other obstacles can interfere with accuracy of the results. Algorithms like connected component analysis and region growing algorithms are not effective in cluttered environments [1].

Klassing et al. [7] proposed a method for the efficient segmentation of 3D laser range data. They use an agglomerative nearest neighbor clustering algorithm to segment the raw data into meaningful portions and filter noise. The obtained clusters are then processed by further supervised or unsupervised classification algorithms and augmented with information obtained from other sensors, such as cameras or other laser scanners.

2.3. Floorplan Reconstruction

Phalak et al. [12] has proposed a method of floor plan reconstruction which uses a deep network based on the Pointnet++ architecture [15] to cluster walls/rooms and then estimate their perimeter.

An effective method for floor plan reconstruction is the random sample consensus (RANSAC) algorithm [4], as used by Pouraghdam et al. [13] Using RANSAC they were able to successfully extract almost 90% of the walls in their test area. There was an average error of 3 cm for the extracted walls, which are adequate for the proposed method for building floor plan modeling.

3. Dataset and Features

The dataset used was provided by the Computer Vision In The Built Environment For The Design, Construction, And Operation Of Buildings Workshop [1], at the The Conference on Computer Vision and Pattern Recognition. It contains a total of 31 buildings with multiple floors each and dozens of rooms on each floor, of which, 20 buildings are designated as the training set, with a total of 49 point clouds. The validation and testing sets contain 5.5 buildings with 21 point clouds each. For each model, there is a point cloud in LAZ format. For the training and validation sets, a corresponding floor plan aligned with the coordinate system of the point cloud is also provided. This 2D ground truth is provided as a .txt file in JSON format, which provides

1. number of layers
2. number of structures in each layer
3. layer details
   (a) layer name
   (b) number of points
   (c) x y coordinates for each point

See Fig. 1 below, for a visualization of the point cloud data.

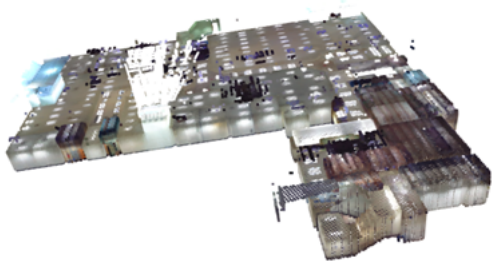


Figure 1. Visualization of point cloud data for one area in the dataset

3.1. Training/Validation

The point clouds from the provided dataset did not contain a sufficient number of doors, stairs and other pertinent features. We trained PointNet on the training dataset and evaluated it on the validation dataset. Since ground truth data was not available for the test dataset, the results shown are on the validation dataset.

3.2. Imbalance of Classes

The provided dataset is heavily imbalanced with more than 70% of the data being background clutter. Wall structures constitute less than 30 percent of the data and the rest of the classes are in the 1% to 5% range. Given this imbalance, we focused on wall segmentation only making it a binary segmentation task, as walls alone give a good idea about the overall layout of a building. It also helped to effectively segment background clutter.

4. Methods

In our dataset the point cloud consists of 3D coordinates, while the ground truth contains 2D coordinates of the structures, therefore, we used two distinct approaches:

1. PointNet [14]: We projected the 2D ground truth to the 3D point cloud for labeling and used that to train a PointNet architecture from scratch. We believed that the 3D model would capture the geometry of large structures like walls better. The existing PointNet models train on smaller features in indoor environment (eg. furniture) e.g [Reference]. This model [Reference] didn't work on our dataset, because it was trained

[1] https://cv4aec.github.io/

on objects in a small room setting, and motivated us to train the PointNet model on our preprocessed dataset from scratch.

2. U-Net [16]: We projected the 3D point cloud to a 2D image and fine tuned a U-Net model for 2D image segmentation for our data set. We used pre-trained U-Net model with the ResNet [6] backbone with imagenet [3] weights on the encoder. This approach significantly reduced the data size and training time complexity.

4.1. Pre-processing

We used a cuboidal grid to create a uniformly downsampled point cloud from the given input point cloud. The algorithm works by first bucketing points into boxes of fixed size and then averaging all points inside the box to generate one point. The data is sparse because empty grid boxes are ignored. This process reduces the size of the data which makes training the model faster and removes feature redundancy in the point cloud.

The algorithm then removes points that are further away from their nearest neighbor based on the threshold on standard deviation of the average distances across the point cloud. This was useful in cleaning the point cloud and removing any stray points in the model. We encountered many noisy and sparse features (eg. cars, trees and streets around the building) in the dataset so it was beneficial to remove them before feeding to the model.

Finally the 2D ground truth was projected in higher dimensions and labeled the downsampled data by setting the threshold on box size in 3D. This way we could segment the point cloud in 3D and label them based on the 2D ground truth. The 2D ground truth represents the walls, therefore it was important when projecting it to 3D, the grid was kept aligned to the wall.

4.2. PointNet Implementation

The PointNet architecture introduced by Qi et al. [14] is a novel deep net architecture that consumes raw point cloud data without voxelization or rendering. It is a unified architecture that learns both global and local point features. PointNet architecture includes a segmentation network that concatenates the global features with the local features to generate point wise scores. The point cloud input lacks order/structure and consists of significant density variations. PointNet is designed to handle the unordered set of points. We tweaked the PointNet implementation for 6 channel input that includes 3 channels for x, y and z coordinates and 3 RGB channels. Additionally, we simplified the output layer to produce single logit for binary segmentation. For every iteration of training or validation the points are randomly chosen from an input scene. A cuboidal box is created around a randomly chosen center and a given number of points are selected. The number of points ***(npoints)*** is a hyperparameter of the model. The positional coordinates are centered around the mean and scaled by the standard deviation. The RGB values are normalized between 0 and 1. A representation of the PointNet architecture can be seen in Fig. 2.

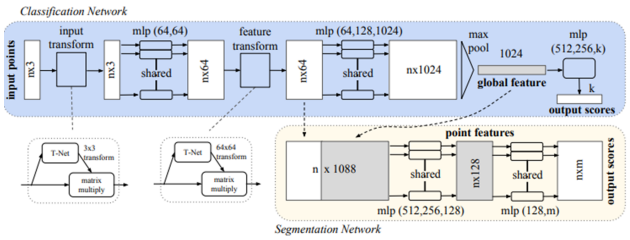


Figure 2. PointNet architecture

4.2.1 Loss Function

Since our data set was heavily imbalanced we used a weighted binary cross entropy loss, as shown in Eq. (1). The weight for the wall structure was calculated by dividing the number of clutter points by the number of wall points, as shown in Eq. (2).

We also experimented with focal loss [9] and dice loss [8], which are designed to handle imbalance in the training data. However, weighted binary cross entropy performed just as well with more stable training.

$$l_n = -w_n[y_n \log(\sigma(x_n)) + (1 - y_n) \log(1 - \sigma(x_n))] \quad (1)$$

and,

$$w_n = \frac{ClutterPoints}{WallPoints} \quad (2)$$

4.3. U-Net Implenentation

U-Net, introduced by [16], is a fully convolutional neural network that captures both features of the context as well as the localization. It consists of an encoder block that has a max-pooling layer of strides 2. It also has repeated convolutional layers with an increasing number of filters in the encoder architecture. The U-Net architecture can be visualized in Fig. 3 below.

The original U-Net paper [16] highlights the ability of the architecture to perform well on neural structures and biological cell membranes [2]. There exist some parallel in segmenting cell membranes and segmenting walls in the 2D layout of the building. Therefore, we chose this vanilla implementation of U-Net with Jaccard loss, which can be understood in the context of focal loss where we want to minimize both losses and gradient of the correct prediction. U-Net performs well in localizing the segmentation

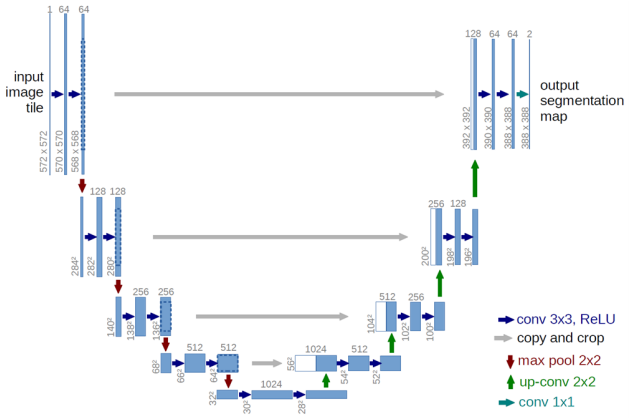


Figure 3. U-Net Architecture

on patches but there is a trade-off between localization accuracy and context. U-Net also performs well on patches of data, hence we fed patches of data as there were less training data points.

5. Experiments/Results/Discussion

5.1. PointNet

We trained PointNet from scratch with only 49 training scenes. The scenes had different sizes, therefore, the number of iterations for different scenes was scaled based on their size. Back propagation was performed using Adam optimizer.

5.1.1 npoints Hyperparameter

The PointNet training required tuning of a few hyperparameters. As noted in an earlier section the points fed into the network are sampled by creating a box around a randomly chosen center in every iteration. The number of points to be used is a hyperparameter which we refer to as $npoints$. We observed significant variation in model performance for different choices of npoints. For our data set, a smaller npoints worked better than what we saw in other PointNet implementations. We believe this was due to the fact that our dataset was really sparse, and therefore, we needed to expand the box sizes to capture those many points. Very large box sizes skew the local features being learned in every iteration. In other implementations, the box size was kept constant and points were sampled with replacement. We didn't use this approach because it would lead to over training for background clutter in the point cloud as those points were more likely to get over-sampled, rather than denser wall structures. Learning rate and batch size also needed some tuning for a stable progression of training. Batch size 16 showed best results. We experimented with larger batch sizes but performance deteriorated.

5.1.2 Evaluation Metric

We tracked IoU for wall segmentation as the primary evaluation metric. The imbalance in the classes can make an accuracy metric misleading. However, IoU gives a good representation of the goal we are tracking as it mitigates the effects of the imbalance.

5.1.3 PointNet Results

Fig. 4 shows the progression of our training for npoint = 128 which gave a mean IoU of 65% for wall segmentation. Fig. 5 shows the progression of training and validation losses with epochs indicating a stable learning process. Fig. 6 shows the training for npoints=1024 which shows that for high npoint values the training model is unable to learn well. We repeated the experiment with many different values of npoint, and found best performance for npoint = 128 and 256, which gave an IoU of around 65%. We also tried npoint = 64 but it didn't perform well, which we believe was because the model had difficulty learning global features with too small of an npoint.

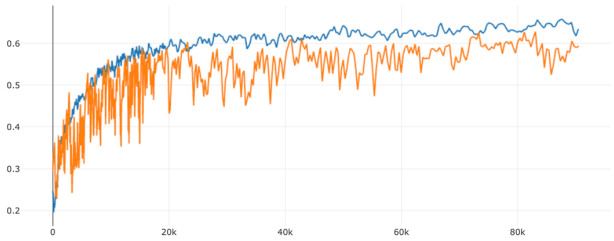


Figure 4. Number of epochs versus loss for an npoint of 128

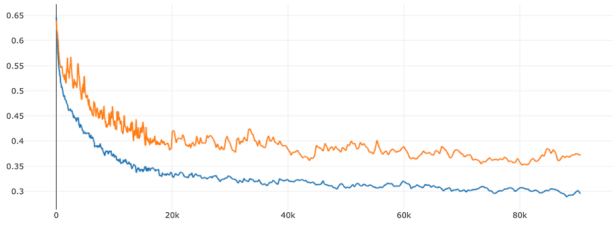


Figure 5. Training and Validation IoUs for npoint = 128

Fig. 7 shows the output of our segmentation on some of the evaluation data scenes.

The results show that PointNet has captured the geometry of the structures pretty well. The model is successful in identifying and eliminating background clutter. The wall structures predicted are a bit more sparse than the ground truth, but an extrapolation algorithm should be able to provide perfect reconstruction of the walls, as the geometry and

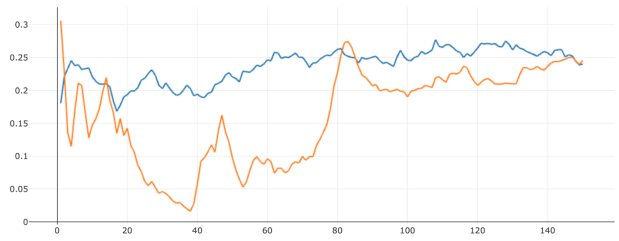


Figure 6. Training and Validation IoUs for npoint = 1024. For large npoint the model training and performance deteriorate.

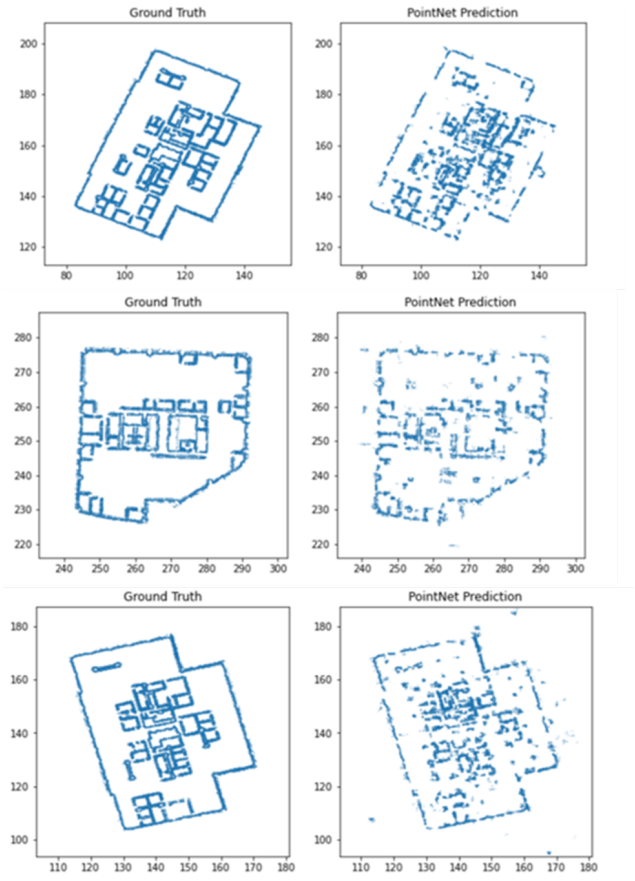


Figure 7. Ground truth vs PointNet

alignment is captured well. Fig. 8 shows a scene which is not well segmented. Even in this case, the overall geometry looks visibility accurate. However, the input scene itself is noisier on some of the edges causing the model not to perform well around those structures. It seems that noise on some parts of the scene affects the global features and impacts performance on other parts of the scene as well. This is a case of non-uniform noise distribution and seems to impact the performance of the model.

Figure 8. Ground truth vs PointNet: Noisy Input

5.2. U-Net

For building the dataset to be trained on U-Net, the 3D point cloud was projected in the 2 dimensional space and pixelated with a grid size of 384x384. The RGB value of all the points in the pointcloud that fell in the grid were averaged over the number of points. Initially the entire training data set contained 60 images of floor plans, and the corresponding masks for walls were fed to the U-Net. We soon realized that it performed badly because of the high amount of noise in the dataset. This motivated us to manually clean the dataset to achieve the best accuracy through U-Net. After cleaning up the noisy input images, we were left with 57 images of size 384x384. For augmentation, all the images with their corresponding masks were patched into 4 smaller images of size 192x192. Other data augmentations applied on the dataset include - rotation by 90, shear, zoom, vertical shift, horizontal shift, vertical flip and horizontal flip. The dataset was then divided into train and test sets and the U-Net was trained for 60 epochs.

5.2.1 U-Net Results

The best IOU on the training and validation set achieved was around 50% and 40% respectively after training for 60 epochs as can be seen in Fig. 9. 60 epochs were enough to achieve saturation in both training and validation losses as can be seen in Fig. 9. The best average IOU achieved on the test data was 25%. It is evident that due to scarcity of training data, U-Net isn't able to learn general wall features. Fig. 10a and Fig. 10b shows the results on the test images and compares the prediction vs ground truth. U-Net performed well in detecting the large geometrical shape and more prominently the outer edges Fig. 10a, however it didn't work well for internal or shorter wall structures Fig. 10b. The outer walls being larger and more dense were easier to segment for U-Net, however U-Net's 2D segmentation approach turned out to be insufficient for internal structures of the layout. This is because there is no distinction in the dense and sparse parts of the layout. We lose the density information which is inherent to point clouds, when

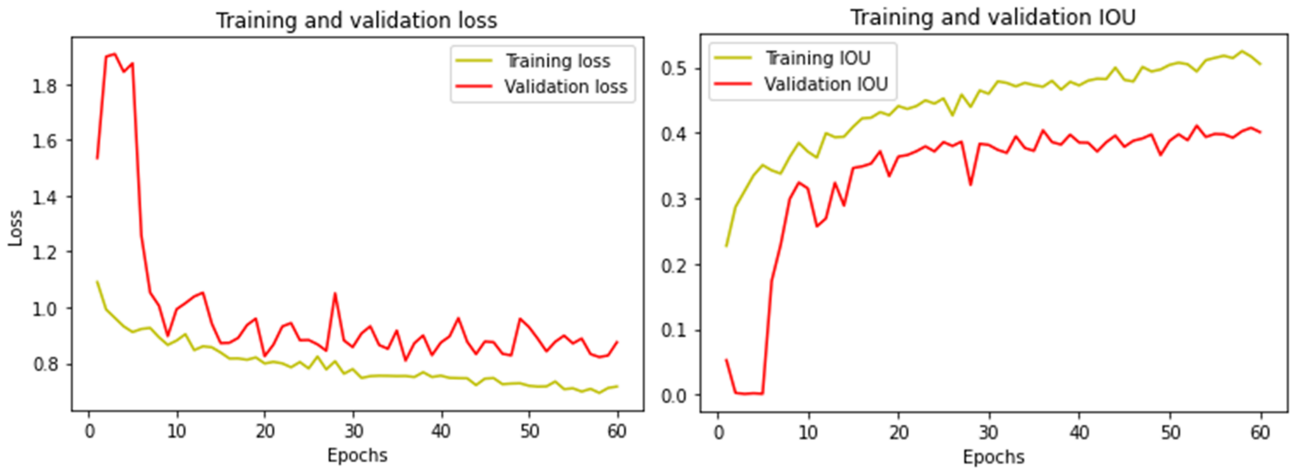


Figure 9. Training and Validation losses and IOUs, vs number of epochs

we project the data in 2D.

5.3. Evaluation Metric

An Intersection over Union (IoU) approach was used to evaluate the amount of overlap between the predicted and ground truth. It was found that the walls were segmented with a high accuracy leaving other classes in the data with a sub-standard accuracy and IoU scores. This occurs due to the biases towards walls in the training dataset. Dimensionality reduction can be used to project the pointcloud and extract features from x,y and rgb channels along with density to mitigate this problem.

6. Conclusion

In conclusion, we have demonstrated that the PointNet and U-Net architectures can be used to segment walls from point cloud data of building floors. The challenge we faced segmenting these 3D point clouds with a 2D layout of the floorplan is a novel problem and we need to combine models from both the regimes of the 2D and 3D segmentation architectures. Projecting the point cloud to 2 dimensions and training a vanilla U-Net proved to be unsuccessful for the given task. This motivated us to use the PointNet with a higher number of channels to accommodate RGB values associated with the points.

The PointNet segmentation network gives much better results and we are able to achieve 65% IoU on the validation dataset. This is particularly remarkable because we had only 49 scenes in our training data set and the model was trained from scratch. This demonstrates that PointNet works well to segment large geometrical structures even in a noisy point cloud setting. We successfully demonstrate the promise of deep learning for floor plan reconstruction tasks from 3D point clouds, and we believe that deep learning can achieve better performance than traditional image processing techniques for this important application domain.

7. Future Work

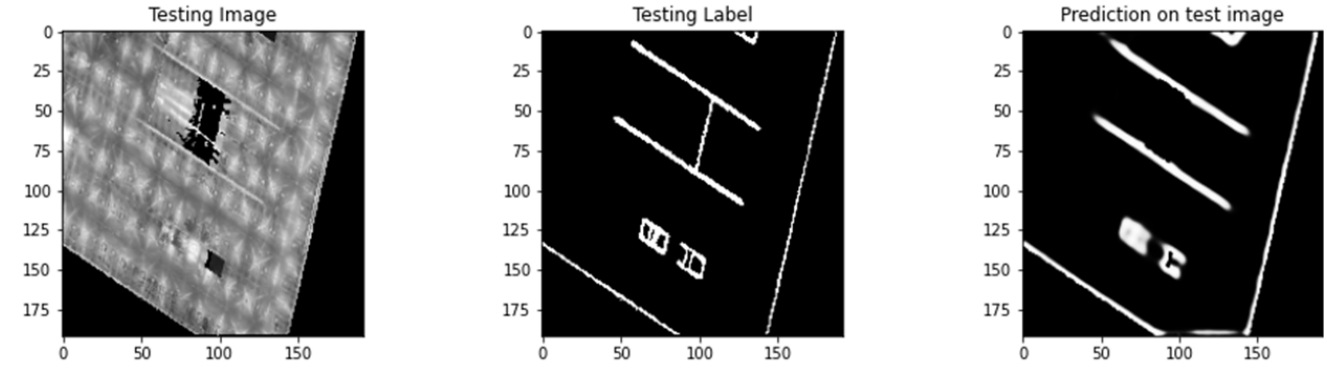
It is evident that one of the biggest challenges with floor plan reconstruction is properly segmenting many types of classes, including stairs, doors and other relevant features. Heavily imbalanced data can cause the model's performance to deteriorate. It is recommended that more data be collected for various layout structures like doors, stairs and different types of walls. PointNet can be used for multi-class segmentation and our results indicate that PointNet can perform well on identifying various other layout structures from a given point cloud scene as well.

In our experiments we down-sampled the point cloud to reduce the size of scenes to reduce the computational complexity. However, we do expect better results with more granular point cloud data, given that PointNet is able to reject the background clutter effectively. It is also clear that projecting the 3D layout into lower dimensions loses a lot of crucial information hence it is preferable to use point cloud feature space to train the segmentation model.

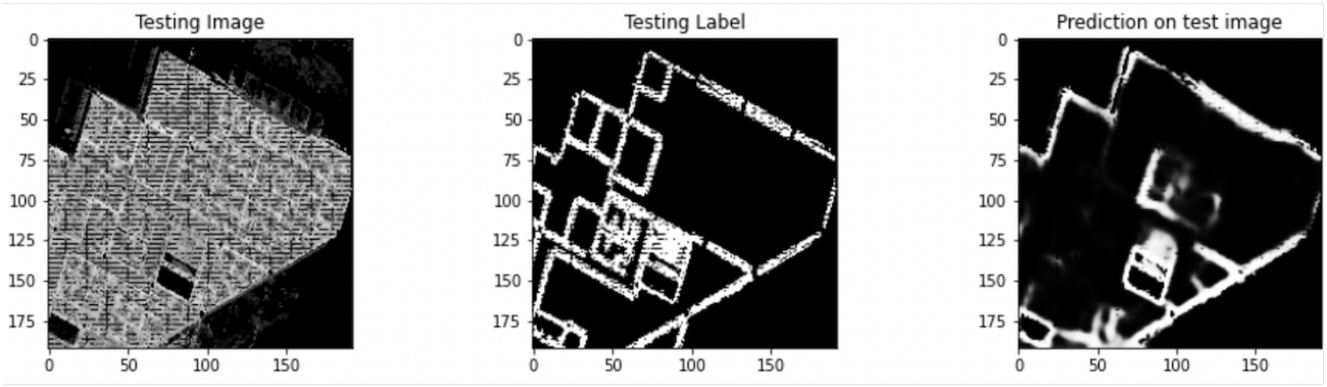
Point cloud scenes are often noisy, so it would be worthwhile to study the impact of non-uniform noise distribution on the scene.

Acknowledgments

We'd like to acknowledge William Shen and Sumith Kulal for their support with this work, as well as the CS231n teaching staff for providing general guidance with technical concepts. We have used some of the code from Point-



(a) Ground truth vs U-Net



(b) Ground truth vs U-Net: Noisy Input

Figure 10. Ground Truth vs U-Net

Net implementations at [2], [3], [4], [5]. We are also thankful to Fan Wang for helping us understand the data layout for the ground truth.

References

[1] K Babacan, L Chen, and G Sohn. Semantic segmentation of indoor point clouds using convolutional neural network. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, 4, 2017.

[2] Dan Ciresan, Alessandro Giusti, Luca Gambardella, and Jürgen Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. *Advances in neural information processing systems*, 25, 2012.

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[4] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[5] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[7] Klaas Klasing, Dirk Wollherr, and Martin Buss. A clustering method for efficient segmentation of 3d laser data. In *2008 IEEE international conference on robotics and automation*, pages 4043–4048. IEEE, 2008.

[8] Xiaoya Li, Xiaofei Sun, Yuxian Meng, Junjun Liang, Fei Wu, and Jiwei Li. Dice loss for data-imbalanced nlp tasks. *arXiv preprint arXiv:1911.02855*, 2019.

[9] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[10] Chen Liu, Jiaye Wu, and Yasutaka Furukawa. Floornet: A unified framework for floorplan reconstruction from 3d scans. In *Proceedings of the European conference on computer vision (ECCV)*, pages 201–217, 2018.

[2] https://github.com/nikitakaraevv/pointnet/blob/master/nbs/PointNetClass.ipynb

[3] https://github.com/yanx27/Pointnet_Pointnet2_pytorch

[4] https://github.com/fxia22/pointnet.pytorch/

[5] https://github.com/meder411/PointNet-PyTorch

[11] Md Mahmood, Mohammed Eunus Ali, et al. Learning indoor layouts from simple point-clouds. *arXiv preprint arXiv:2108.03378*, 2021.

[12] Ameya Phalak, Vijay Badrinarayanan, and Andrew Rabinovich. Scan2plan: efficient floorplan generation from 3d scans of indoor scenes. *arXiv preprint arXiv:2003.07356*, 2020.

[13] M Hossein Pouraghdam, M Saadatseresht, H Rastiveis, A Abzal, and M Hasanlou. Building floor plan reconstruction from slam-based point cloud using ransac algorithm. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 42:483–488, 2019.

[14] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[15] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[16] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[17] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. *Advances in neural information processing systems*, 28, 2015.