Cloud based High-Dimensional Object Detection using Deep Neural Network

Authors: Junkang Gu, Chunda Song

1. Introduction

The project is to Explore and compare Deep Learning methodologies for 3D Object Recognition. In this project we will use PointNet and MV-CNN and compare their performance on 3D Object Recognition. This experiment helps us find effective approaches to recognize 3D objects.

There are two models being used to test 3D detection, which are PointNet and MV-CNN. PointNet is in pointnet.py, and MV-CNN is in mvcnn\_pytorch.

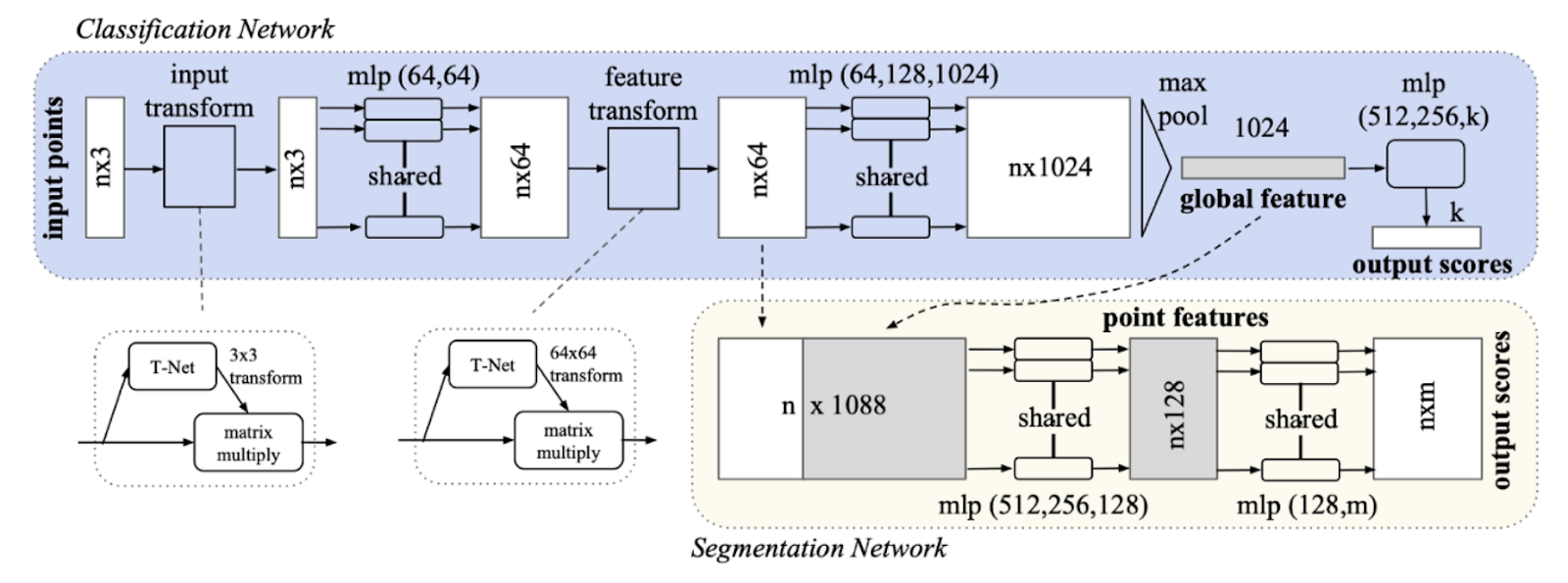
The PointNet architecture proposed for point clouds classification and semantic segmentation tasks are shown in the below image. The top blue path specifies the classification network while the bottom yellow path is the semantic segmentation network.

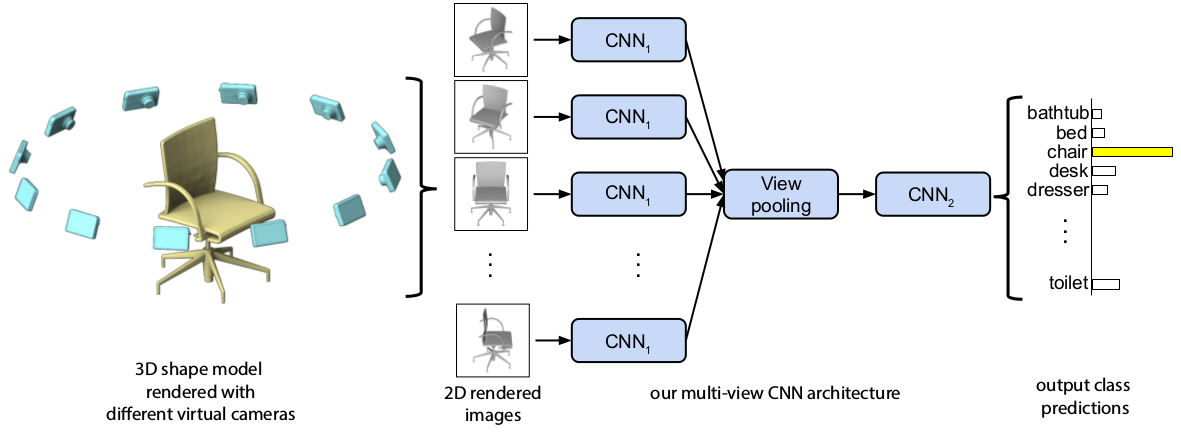
MV-CNN is a network topology that combines information from different views into fully connected layers to classify the voxel where the planes cross. The multi-view approach can be considered as a 2.5D CNN given that it incorporates information from each image plane, but does not use the full 3D neighborhood of the queried voxel. This results in a lower computational complexity when compared to 3D-kernel methods

2. Proposed Architecture

In this project we have two architecture for 3D recognition.

Solution Approach/Technical Details

–PointNet



–MVCNN

First we use PointNet to recognize 3D object datasets. The PointNet architecture proposed for point clouds classification and semantic segmentation tasks are shown in the below image. The top blue path specifies the classification network while the bottom yellow path is the semantic segmentation network. In the classification network, the layers from the input layer up to the max-pooling layer form function *h* while max-pooling layer is selected as symmetric function *g*. The embedding dimension *d*is chosen to be 1024. In PointNet, function *h* which is applied in a point-wise manner to each lidar point is formed by fully-connected layers (MLP) and [spatial transformer networks](https://arxiv.org/abs/1506.02025). (Multilayer Perceptrons, or MLPs for short, are the classical type of neural network. They are comprised of one or more layers of neurons).

Secondly, we will introduce MV-CNN architecture, which is based on local information fusion network. it can aggregate the multi-view feature maps into a shape super matrix, and it can extracting its local information.

3. Proposed Solutions

Data representation:

Point cloud, Polygon Mesh and Volumetric can be used to represent 3D modal, and we would like to use point cloud. A point cloud is a collection of points in 3D space, each point specified by an (XYZ) coordinates, optionally along with other attributes (like RGB color). Point cloud representation is more preferred since the conversion Point cloud to other formats and vice versa is easy. Point representation preserves the original geometric information in 3D space without any discretization.

MVCNN May neglect internal structure information which can be useful for classification sometimes (e.g. wardrobes & cupboards look similar from the outside but differentiable by internal structures)

Dataset:

First we will consider data processing,

Convert CAD models to point cloud format with random sampling and normalization

Render CAD models into multi-view images with virtual cameras

The original test set split into two sets: Set 1 & Set 2.

Set 1 contains simple categories with minimal or no internal structures (e.g. cups, tables)

Set 2 contains categories with complex internal structures modeled (e.g. cars)

Method:

Use ModelNet40 (40 categories of CAD models) as the data set. Manually split the test set of 40 categories into two groups based on complexity of internal structures. Then we will process data and implement models. Finally we will start to evaluate model performance.

Experiment Procedure:

(1)

Model Hyperparameters:

PointNet: 15 Epochs, 1024-point-cloud

MVCNN: 15 Epochs, 12 views

(2)

Train the two models on the training set

(3)

Evaluate accuracies over three test sets:

Test Set 0: The general test set for overall accuracy

Test Set 1: A subset of Test Set 0 and only includes those objects with minimal to no internal structures

Test Set 2: A subset of Test Set 0 and only includes those objects with complex internal structures

(4)

Measure training time

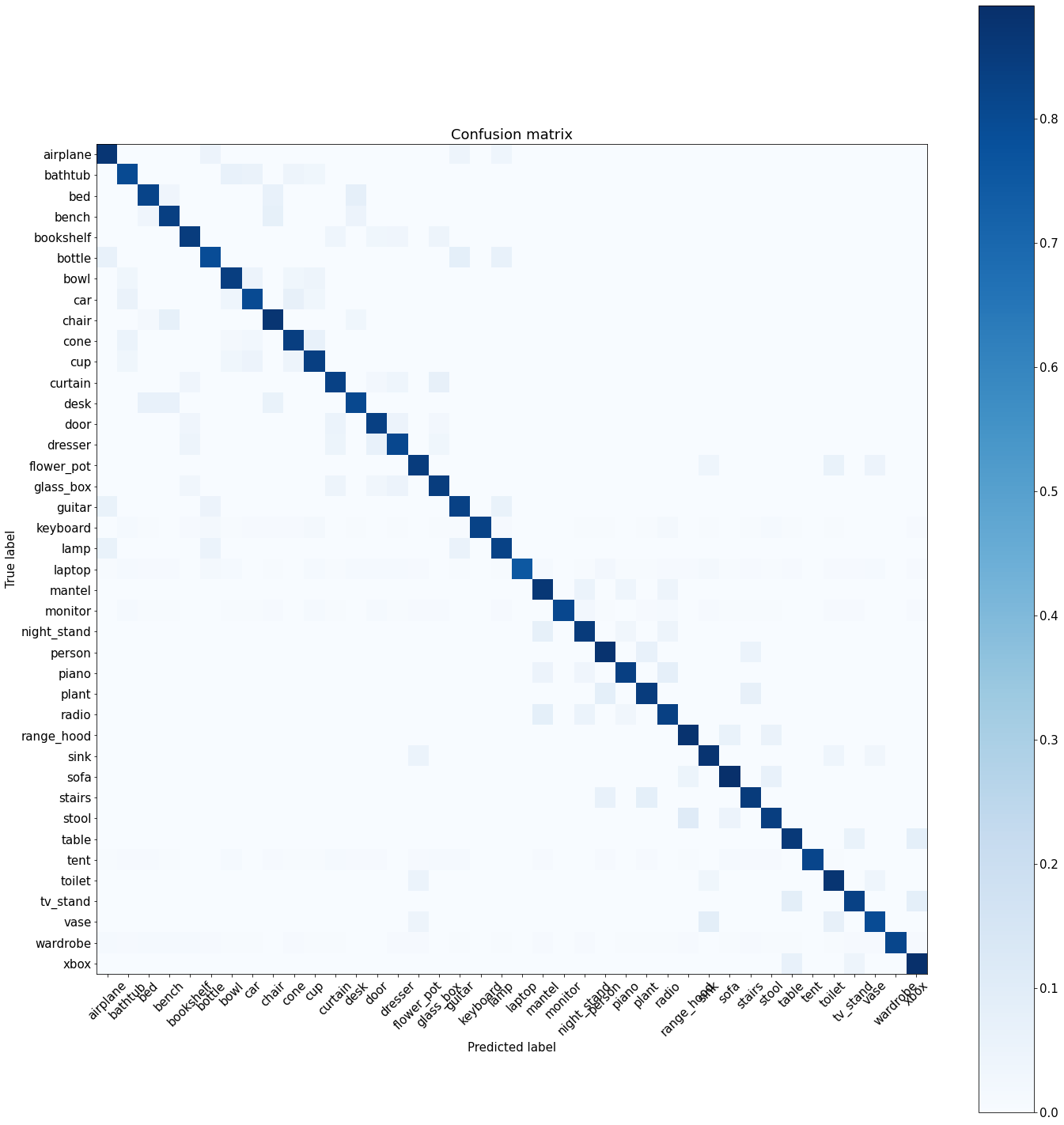
Experimental evaluation:

Overall accuracy on ModelNet40:

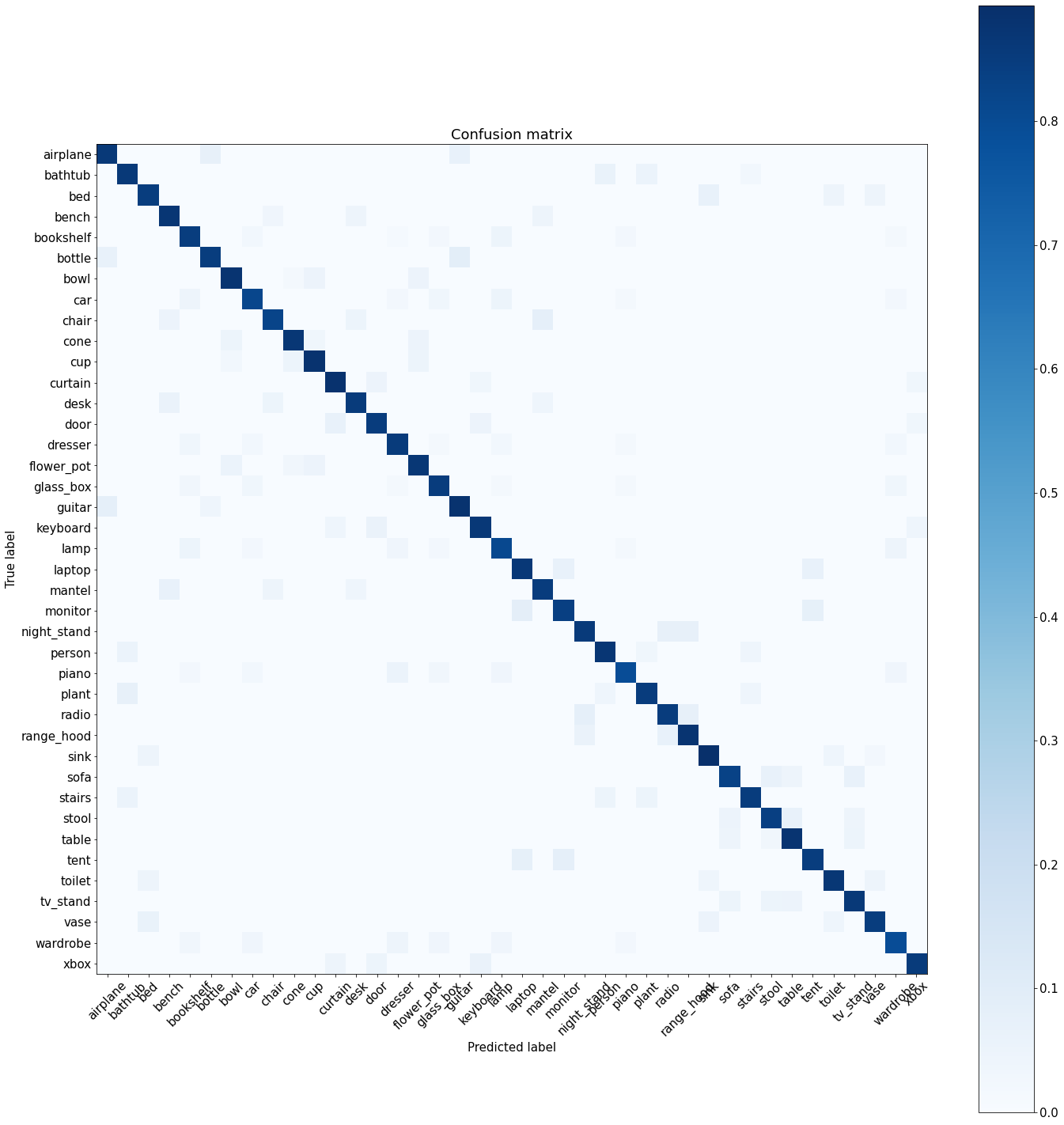
PointNet: 0.88243

MVCNN: 0.90072

Confusion Matrix:



MVCNN



PointNet

Accuracies over sub-categories:

|  |  |  |
| --- | --- | --- |
|  | Test Set 1:  Limited Internal Structures | Test Set 2:  Complex Internal Structures |
| PointNet | 0.88165 | 0.89032 |
| MVCNN | 0.90032 | 0.88014 |

Conclusion:

For overall accuracy with ModelNet40 dataset, MVCNN seems to perform better than PointNet

We believe it is likely because sparse point clouds lose lots of information (while dense point clouds are computation expensive), and for most 3D objects having 2D views is enough to classify

We proposed a hypothesis that with the advantage of having internal structure information, PointNet may perform better than MVCNN on those objects with complex internal structures.

Result shows slightly improvement. MVCNN performs a bit worse when it tries to distinguish between wardrobes and bookshelves, while PointNet seems to capture their internals and predict slightly better.

MVCNN trains faster than PointNet.

Related Work:

references:

Qi et. al., PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation

https://github.com/gkadusumilli/kaolin\_1/tree/master/Documents/kaolin-0.1

https://arxiv.org/abs/1612.00593

<https://towardsdatascience.com/deep-learning-on-point-clouds-implementing-pointnet-in-google-colab-1fd65cd3a263>

https://www.nature.com/articles/s41598-021-93905-2

Github repo:

jongchyisu/mvcnn\_pytorch

nikitakaraevv/pointnet