POINT CLOUD DEEP LEARNING

Innfarn Yoo, 3/29/2018
AGENDA

• Introduction
• Previous Work
• Method
• Result
• Conclusion
INTRODUCTION
2D OBJECT CLASSIFICATION
Deep Learning for 2D Object Classification

- Convolutional Neural Network (CNN) for 2D images works really well
  - AlexNet, ResNet, & GoogLeNet

- R-CNN $\rightarrow$ Fast R-CNN $\rightarrow$ Faster R-CNN $\rightarrow$ Mask R-CNN

- Recent 2D image classification can even extract precise boundaries of objects (FCN $\rightarrow$ Mask R-CNN)

3D OBJECT CLASSIFICATION

Deep Learning for 3D Object Classification

- 3D object classification approaches are getting more attentions
  - Collecting 3D point data is easier and cheaper than before (LiDAR & other sensors)
  - Size of data is bigger than 2D images
  - Open datasets are increasing
  - Recent researches approaches human level detection accuracy
    - MVCNN, ShapeNet, PointNet, VoxNet, VoxelNet, & VRN Ensemble

GOALS
The goals of our method

• Evaluating & comparing different types of Neural Network models for 3D object classification

• Providing the generic framework to test multiple 3D neural network models
  • Simple & easy to implement neural network models
  • Fast preprocessing (remove bottleneck of loading, sampling, & jittering 3D data)
PREVIOUS WORK
3D POINT-BASED APPROACHES

3D Points $\rightarrow$ Neural Nets

- **PointNet**
  - First 3D point-based classification
  - Unordered dataset
  - Transform $\rightarrow$ Multi-Layer Perceptron (MLP) $\rightarrow$ Max Pool (MP) $\rightarrow$ Classification

PIXEL-BASED APPROACHES

3D $\rightarrow$ 2D Projections $\rightarrow$ Neural Nets

- Multi-Layer Perceptron (MLP)
- Convolutional Neural Network (CNN)
- Multi-View Convolutional Neural Network (MVCNN)

VOXEL-BASED APPROACHES
3D Points → Voxels → Neural Nets

- VoxNet
- VRN Ensemble
- VoxelNet

METHOD
PREPROCESSING

Requirement

• Loading 3D polygonal objects

• Required Operations on 3D objects
  • Sampling, Shuffling, Jittering, Scaling, & Rotating
  • Projection, & Voxelization

• Python interface is not that good for multi-core processing (or multi-threading)
  • # of objects is notoriously for single-core processing
3D DATASETS

**MODELNET10**
- Princeton ModelNet Data
- 10 Categories
- 4,930 Objects (2 GB)
- OFF (CAD) File Format

**MODELNET40**
- Princeton ModelNet Data
- 40 Categories
- 12,431 Objects (10 GB)
- OFF (CAD) File Format

**SHAPENET CORE V2**
- ShapeNet
  - [https://www.shapenet.org/](https://www.shapenet.org/)
- 55 Categories
- 51,191 Objects (90 GB)
- OBJ File Format
NEURAL NETWORK MODELS

- Point-Based Models
- Pixel-Based Models
- Voxel-Based Models
POINT-BASED NEURAL NETWORK MODELS

Types of Models

- Preprocessing:
  - Rotate randomly
  - Scale randomly
  - Uniform sampling on 3D object surfaces
  - Sample 2048 points
  - Shuffle points

- Tested Models
  - Multi-Layer Perceptron (MLP)
  - Multi Rotational MLPs
  - Single Orientation CNN
  - Multi Rotational CNNs
  - Multi Rotational Resample & Max Pool Layers
  - ResNet-like
POINT-BASED NEURAL NETWORK MODELS

**MLP**

3D points → Flatten Vector → ReLU + Dropout → Fully Connected Layer → Softmax Cross Entropy → Class Onehot Vector
POINT-BASED NEURAL NETWORK MODELS

Multi Rotational MLPs

3D points → Random 3x3 Rotation → 3D Conv Layer → Max Pooling Layer → Flatten Vector → Fully Connected Layer → ReLU + Dropout → Softmax Cross Entropy → Class Onehot Vector
POINT-BASED NEURAL NETWORK MODELS

Single Orientation CNN
POINT-BASED NEURAL NETWORK MODELS

Multi Rotational CNNs

3D points

ReLU + Dropout

ReLU + Dropout

Random 3x3 Rotation

3D Conv Layer

Max Pooling Layer

Flatten Vector

Fully Connected Layer

Class Onehot Vector

Softmax Cross Entropy
POINT-BASED NEURAL NETWORK MODELS

Multi Rotational Resample & Max Pool Layers

3D points

Random 3x3 Rotation
Resample Layer
Max Pooling Layer
Flatten Vector
Fully Connected Layer
Class Onehot Vector

ReLU + Dropout
ReLU + Dropout

Softmax Cross Entropy
POINT-BASED NEURAL NETWORK MODELS

ResNet-like

3D points

ReLU + Dropout

ReLU + Dropout

ReLU + Dropout

Random 3x3 Rotation

Resample Layer

Max Pooling Layer

Flatten Vector

Fully Connected Layer

Class Onehot Vector

Softmax Cross Entropy
PIXEL-BASED NEURAL NETWORK MODELS

Types of Models

- Preprocessing:
  - Sample 8192 points
  - Same as point-based models
  - Depth-only orthogonal projection
    - 32x32 or 64x64
  - Generating multiple rotations
    - 64x64x5 & 64x64x10

- Tested Models:
  - MLP
  - Depth-Only Orthogonal MVCNN
PIXEL-BASED NEURAL NETWORK MODELS

MLP

Images (32x32x5) → Flatten Vector → Fully Connected Layer → Softmax Cross Entropy → Class Onehot Vector
PIXEL-BASED NEURAL NETWORK MODELS

Depth-Only Orthogonal MVCNN

Images (32x32x5) -> Image Separation -> 3D Conv Layer -> Max Pooling Layer -> Flatten Vector -> Fully Connected Layer -> Softmax Cross Entropy -> Class Onehot Vector
VOXEL-BASED NEURAL NETWORK MODELS

Types of Models

- Preprocessing:
  - Sample 8192 points
    - Same as point based models
  - Voxelization
    - 3D points → Voxels
    - Each voxel has intensity 0.0 ~ 1.0
      - how many points hit same voxel
    - 32x32x32 & 64x64x64

- Tested Models:
  - MLP
  - CNN
  - ResNet-like
VOXEL-BASED NEURAL NETWORK MODELS

MLP

Images (32x32x5) → Flatten Vector → Fully Connected Layer → Softmax Cross Entropy → Class Onehot Vector
VOXEL-BASED NEURAL NETWORK MODELS

CNN

Voxels 32x32x32

Softmax Cross Entropy

3D Conv Layer
Max Pooling Layer
Flatten Vector
Fully Connected Layer
Class Onehot Vector
VOXEL-BASED NEURAL NETWORK MODELS

ResNet-like

Voxels 32x32x32

Softmax Cross Entropy

3D Conv Layer
Avg Pooling Layer
Resample Layer
Max Pooling Layer
Flatten Vector
Fully Connected Layer
Class Onehot Vector
IMPLEMENTATION

System Setup

• System: Ubuntu 16.04, RAM 32 GB & 64 GB, & SSD 512 GB
• NVIDIA Quadro P6000, Quadro M6000, & GeForce Titan X
• GCC 5.2.0 for C++ 11x
• Python 3.5
• TensorFlow-GPU v1.5.0
• NumPy 1.0
HYPER PARAMETERS

• Object Perturbation
  • Random Rotations: -25 - 25 degree
  • Random Scaling: 0.7 - 1.0

• Learning Rate: 0.0001

• Keep Probability (Dropout layer): 0.7

• Max Epochs: 1000

• Batch Size: 32

• Number of Random Rotations: 20

• Voxel Dim: 32x32x32

• MVCNN Number of Views: 5
MODELNET10 ACCURACY

Iter: 1000

- PC MLP1
- PC CNN1
- PC MLPs
- PC CNNs
- PC MP
- PC ResNet
- PX MLP
- PX MVCNN
- VX MLP
- VX CNN
- VX ResNet

- Train Accu
- Test Accu
- mAP
MODELNET40 ACCURACY

Iter: 1000

%
MODELNET40 ACCURACY
7 CATEGORIES (# OF TRAIN OBJECTS > 400)
MODELNET40, 7 CATEGORIES ACCURACY

Iter: 1000

%
MODELNET40 ACCURACY
10 CATEGORIES (# OF TRAIN OBJECTS > 300)
MODELNET40, 10 CATEGORIES ACCURACY
MODELNET40 ACCURACY  
17 CATEGORIES (# OF TRAIN OBJECTS > 200)
MODELNET40, 17 CATEGORIES ACCURACY

Iter: 1000
MODELNET10 PERFORMANCE

Total Training Time

Inference Time Per Batch
MODELNET40 PERFORMANCE

- **Total Training Time**
  - PC MLP1
  - PC CNN1
  - PC MLPs
  - PC CNNs
  - PC MP
  - PC ResNet
  - PX MLP
  - PX MVCNN
  - VX MLP
  - VX CNN
  - VX ResNet

- **Inference Time Per Batch**
  - PC MLP1
  - PC CNN1
  - PC MLPs
  - PC CNNs
  - PC MP
  - PC ResNet
  - PX MLP
  - PX MVCNN
  - VX MLP
  - VX CNN
  - VX ResNet
• ShapeNet has pretty big dataset
  • 90 GB of dataset
  • Only tested 3 NN models
    • Point MP
    • Depth-Only MVCNN
    • Voxel CNN
CROSS ENTROPY CONVERGENCE GRAPH

PC MLP1
PC MLPs
PC CNN1
PC CNNs
PC MP
PC ResNet
PX MLP
PX MVCNN
VX MLP
VX CNN
VX ResNet
CONCLUSION
CONCLUSION

Models

• Voxel ResNet and Voxel CNN provide best result on 3D object classification

• Unordered vs. Ordered: Unordered data (point cloud) could be learned, but lower accuracy and higher computation cost than ordered data

• Projection vs. Voxelization: Voxelization provides better result with similar computation cost (converge faster, & more precise)

• Strict comparisons with previous methods are not done yet
  • Sampling and jittering methods are different (cannot directly compare yet)
CONCLUSION

Dataset Evaluation

- **ModelNet10 & ModelNet40**
  - Some categories have too small training and testing objects
  - Lowering classification accuracy
- **ModelNet40, 7 categories (# of training objects > 400)**
  - Achieved 98% accuracy
  - Partial dataset from ModelNet40 (Categories that have more than 400 3D objects)
  - # of train 3D objects in a category matters
- **ShapeNetCore.v2**
  - # of 3D objects are big enough, but many object but many duplicated objects
FUTURE WORK

• Running entire procedures in CUDA
  • Using NVIDIA GVDB → Will have advantages of Sparse Voxels

• Strict comparisons with previous methods
  • PointNet, VRN Ensemble, etc

• Extending methods to captured dataset (e.g. KITTI)

• Train set is too small
  • Need to investigate whether Generative Adversarial Networks (GAN) can alleviate this problem or not
• [1] He et al., 2017, Mask R-CNN
• [5] Qi et al., 2017, Pointnet: Deep learning on point sets for 3d classification and segmentation
• [7] Broke et al., 2016, Generative and Discriminative Voxel Modeling with Convolutional Neural Networks
• [8] He et al., 2016, Deep residual learning for image recognition
• [9] Goodfellow et al., 2014, Generative adversarial nets
• [12] Chang et al., 2015, ShapeNet: An Information-Rich 3D Model Repository
• [13] Wu et al., 2015, 3D ShapeNets: A Deep Representation for Volumetric Shapes
• [14] Wu et al., 2014, 3D ShapeNets for 2.5D Object Recognition and Next-Best-View Prediction