Pointwise and Instance Segmentation for 3D Point Clouds

MS Thesis Presentation

Sanket Gujar

Worcester Polytechnic Institute

April 11, 2019

Schedule

Overview

Motivation

Uber's self driving vehicle hit bicyclist, Perception classification history: 1. Unknown, 2. Vehicle, 3. Bicycle (1.3 secs before impact)

Problem with Camera

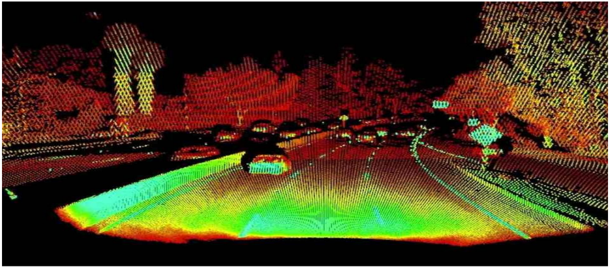
Some examples where use of camera for self-driving cars can be dangerous.

Problem with Camera

Cameras have limited dynamic range, making detection difficult.

Image Ref: Aurora's Approach to Development

Better is LiDAR



Innovusion LiDAR front projection image

- Range up to 200m → beneficial for high speed highway driving (9 secs at 50 miles/hr).
- Invariant to lighting conditions → same performance in day/night.
- 360°field of view → crucial for lane changing and monitoring vehicles behind.

Image Ref: An Introduction to LIDAR: The Key Self-Driving Car Sensor

Background

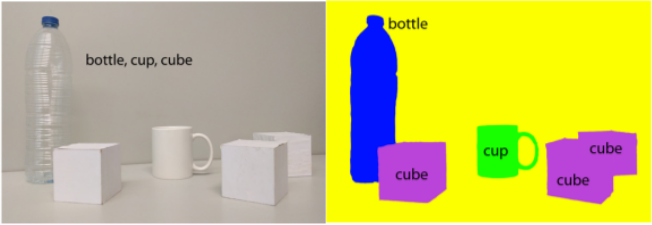
Point clouds

Point cloud of chair, car, table and airplane from ModelNet10 Dataset

- Point cloud: a collection of data points defined by a given coordinates system.
- Generally produced by 3D scanners, which measure a large number of points on the external surfaces of objects around them.
- Used to create 3D CAD models for manufactured parts, for quality inspection,animation, rendering and mass customization applications.

Image Ref: An Introduction to LIDAR: The Key Self-Driving Car Sensor

Semantic Segmentation

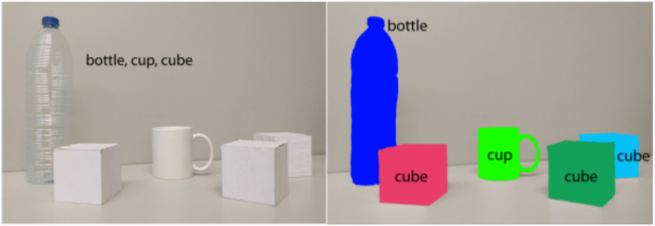


Semantic Segmentation example

- Semantic segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain characteristics.

Image Ref: A Review on Deep Learning Techniques Applied to Semantic Segmentation

Instance Segmentation

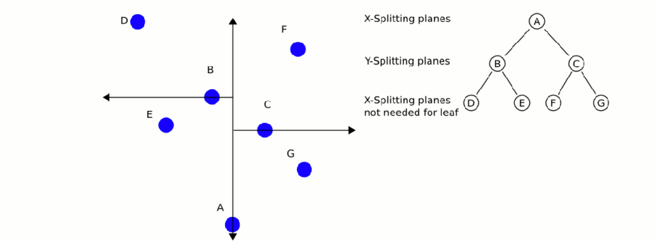


Instance Segmentation example

- Instance segmentation is the process of detecting and delineating each distinct object of interest appearing in an image.

Image Ref: A Review on Deep Learning Techniques Applied to Semantic Segmentation

K-d Tree

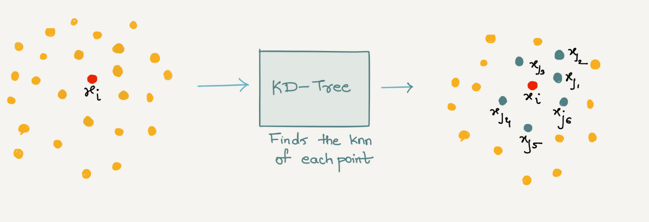


k-d tree example

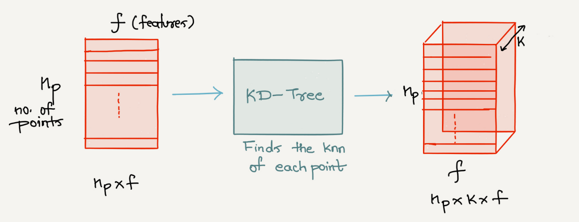
- k-d tree (short for k-dimensional tree) is a space-partitioning data structure for organizing points in a k-dimensional space.
- k-d trees are a special case of binary space partitioning trees.
- The complexity varies from $\log(N)$ to N depending on the pruning possible.

Image Ref:Using KD-Tree For Nearest Neighbor Search

K-nearest neighbors

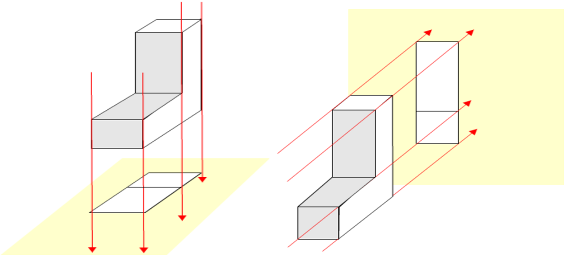


Nearest neighbours for a point for K = 6



Nearest neighbor operation for a Tensor of size N x f

Projections



Bird’s eye view and front projections of L shape

- Bird’s eye view is an elevated view from above, with a perspective as though the observer were a bird.
- Its a mapping of all point along z-axis on the x-y plane for our experiments.
- Front projection is mapping of all points along x-axis on y-z plane.

Image Ref: first angle - orthographic projection

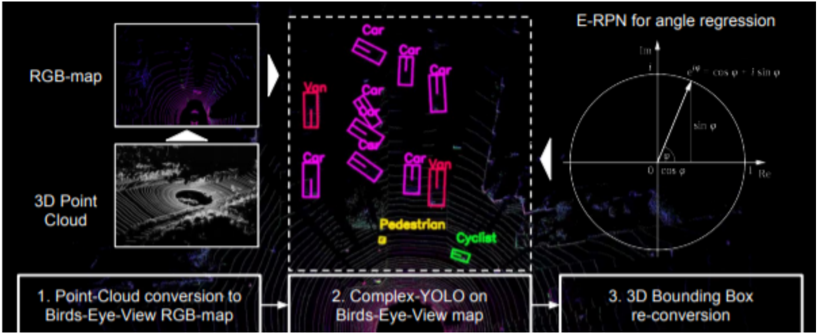
Problem Statement

Problem Statement

Develop an architecture to do end-to-end pointwise and instance segmentation for 3D point clouds which should be able to handle large point clouds for self-driving vehicle perception stack

Previous Approach

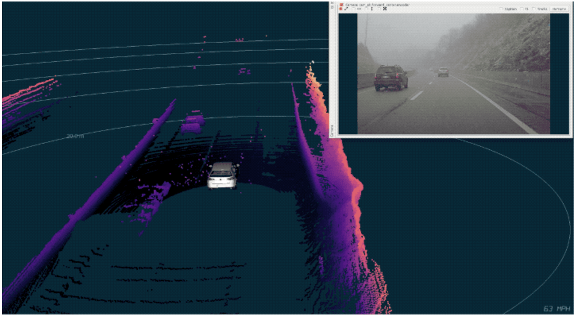
Projection Methods



Complex-YOLO [SMAG18] uses bird's eye view projection for detection

- Run detection and localization network on bird's eye view or front projection images from LiDAR.
- Projection methods are the fastest for detection and tracking for self-driving stack.

Projection Methods

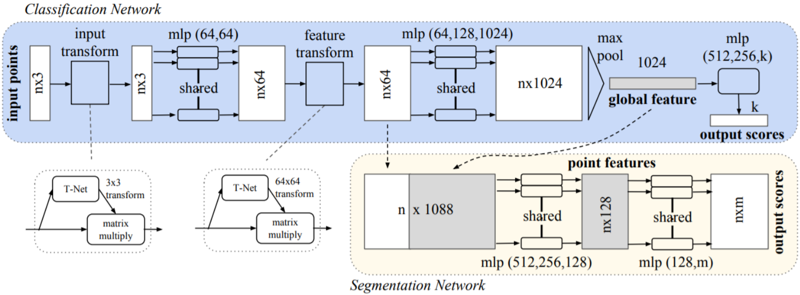


Lidar is sensitive enough to detect snow, making it more difficult to identify important objects.

- If rain drops or snow is picked by LiDAR sensor → noise distribution in projected images resulting in miss-classification.
- Miss-classification is a common issue when the vehicles are very close to each other.

Image Ref: Aurora's Approach to Development

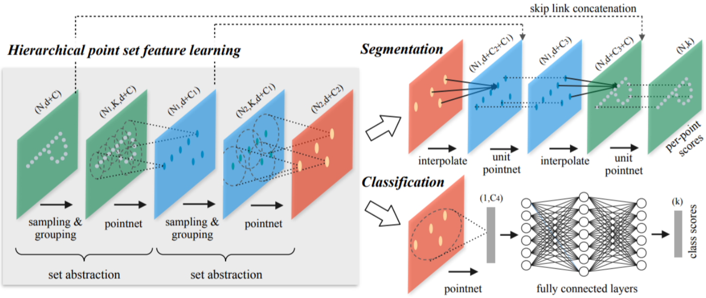
Pointnet



Pointnet Architecture [QSMG16]

- Pointnet was the most successful initial approach to apply deep learning to 3D point clouds.
- The important feature of the architecture to use symmetric function to get invariance to certain transformation like rotation and translation.
- The architecture used spatial and feature transformer to align input points and point features.

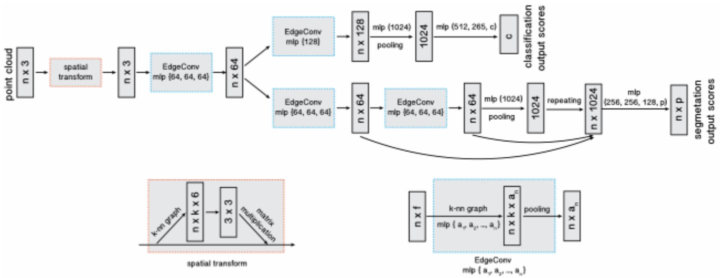
Pointnet++



Pointnet++ Architecture [QYSG17]

- Pointnet++ is a hierarchical network that applies Pointnet recursively on a nested portioning of the input point cloud.
- The hierarchical structure is composed of a number of set abstraction levels. The set abstraction layers consist of three layers: Sampling layer, Grouping layer and Pointnet layer.

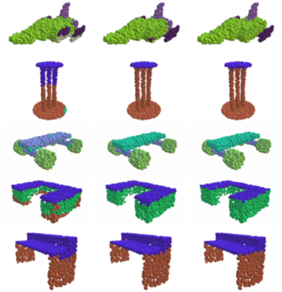
Edge Conv



Dynamic Graph CNN/ Edge Conv Architecture [WSL+18]

- EdgeConv appealing property is that it incorporates local neighborhood information as it can be stacked or recurrently applied to learn global shape properties.

Edge Conv Results

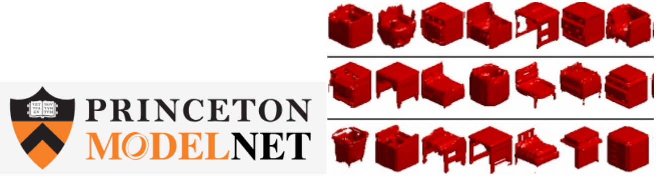


	Mean Class Accuracy	Overall Accuracy
3DShapeNets [54]	77.3	84.7
VoxNet [30]	83.0	85.9
Subvolume [35]	86.0	89.2
ECC [45]	83.2	87.4
PointNet [34]	86.0	89.2
PointNet++ [36]	-	90.7
Kd-Net (depth 10) [20]	-	90.6
Kd-Net (depth 15) [20]	-	91.8
Ours (baseline)	88.8	91.2
Ours	**90.2**	**92.2**

Comparisons of model on Modelnet40. [Ours is EdgeConv here]

Dataset

ModelNet40



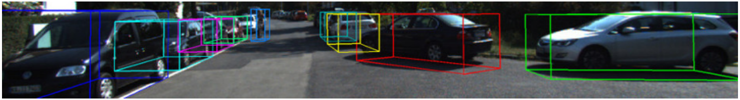
ModelNet40: Princeton 3D CAD model Dataset

ModelNet40 Dataset	
	Samples
Training	94k with 40 labels
Testing	24k with 40 labels

Image Ref:Princeton ModelNet Dataset

KITTI Vision Benchmark Suite

Karlsruhe Institute of Technology



3D bounding box annotations in Kitti Dataset [Gei12]

Kitti Dataset

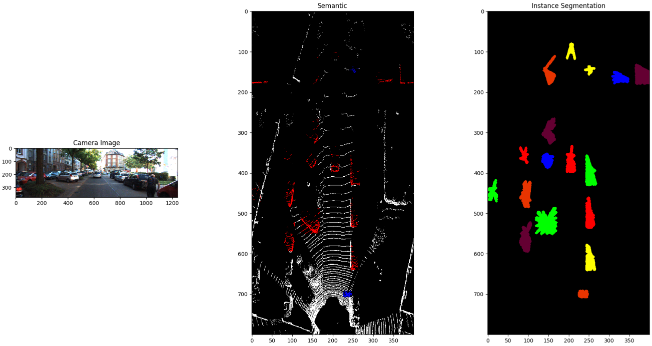
	Samples
Training	7481
Testing	7518

KITTI Reader

a. Camera Image, b. LIDAR front projection on image with labels

- The Kitti Dataset reader can provide dataset batches for training, does transformation with the caliberation matrix provided, create Birds eye view, provide instance and point segmentations labels

KITTI Reader



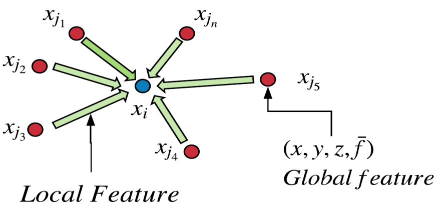
The Kitti Dataset reader can produce instance segmentation labels

Pointer

Approach to the problem

- The point can be represented by two properties which is its position in the frame (global) and the distribution of its neighboring points (local).
- Needed to develop an architecture that can embed both local and global features of the point cloud.
- Would learn to weight the importance of local and global features
- Would generate strong high level features that would make learning faster and easier.

Point cloud features



Point cloud features

Global Features

- Consist of real 3D world coordinates x, y, z and feature provided by the sensor like intensity, phase of wave, rgb value etc.
- Is a feature of a single point.

Local features

- Consist of unit vector pointing from its neighbours to the point. $x_i - x_j$, where x_j is the neighbors of the point x_i.
- Is a feature of a single point depending on its neighbors.

Pointer features

Pointer feature learning

- x_i is a point in the point clouds and x_j is the neighboring point in the point cloud. we can regard x_i as the central pixel and $x_j : (i, j) \in \varepsilon$ as a patch around it
- We define global features p_{ij} with function g_θ which is a parametric non-linear function parametrized by the set of learnable parameters θ

$$p_{ij} = g_\theta(x_i, x_j)$$

$$g_\theta : \mathbb{R}^F \times \mathbb{R}^F \rightarrow \mathbb{R}^{F'}$$

Pointer features

Pointer feature learning

- x_i is a point in the point clouds and x_j is the neighboring point in the point cloud. we can regard x_i as the central pixel and $x_j : (i, j) \in \varepsilon$ as a patch around it
- We define global features p_{ij} with function g_θ which is a parametric non-linear function parametrized by the set of learnable parameters θ

$$p_{ij} = g_\theta(x_i, x_j)$$

$$g_\theta : \mathbb{R}^F \times \mathbb{R}^F \to \mathbb{R}^{F'}$$

- We define local features q_{ij} with function h_θ which is also a parametric non-linear function parametrized by the set of learnable parameters θ

$$q_{ij} = h_\theta(x_i, x_i - x_j)$$

$$q_\theta : \mathbb{R}^F \times \mathbb{R}^F \to \mathbb{R}^{F'}$$

Pointer features

Pointer feature learning

- Global Feature

$$p_{ij} = g_\theta(x_i, x_j)$$

- Local Feature

$$q_{ij} = h_\theta(x_i, x_i - x_j)$$

- We define the fusion feature T_{ij} of local features q_{ij} and global feature p_{ij} with function M

$$T_{ij} = M(p_{ij}, q_{ij})$$

- Here M is a learnable function which can be weighted sum or a convolutional layer or a concatenation layer

Pointer features

Pointer feature learning

- Global Feature

$$p_{ij} = g_\theta(x_i, x_j)$$

- Local Feature

$$q_{ij} = h_\theta(x_i, x_i - x_j)$$

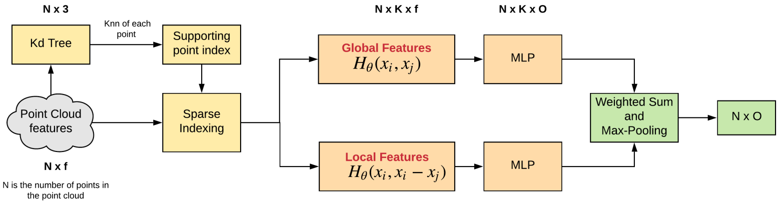
- Fusion feature

$$T_{ij} = M(p_{ij}, q_{ij})$$

- Finally, we define the Pointer operation by applying a channel-wise symmetric aggregation operation $\square$ ($\sum$ or max)

$$\boxed{x_i^{l+1} = \underset{j:(i,j)}{\square} T_{ij}^{l}}$$

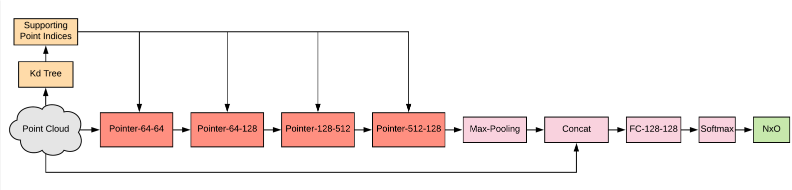
Pointer Block



Pointer Main Block

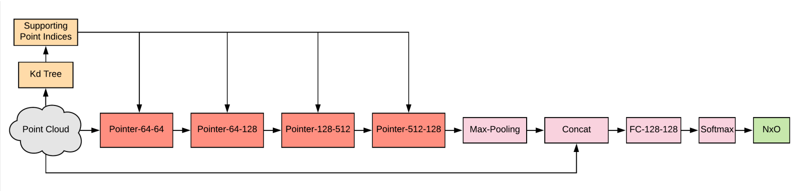
Pointer Semantic

Pointer Semantic Segmentation



Pointer Semantic Segmentation

Pointer Semantic Segmentation



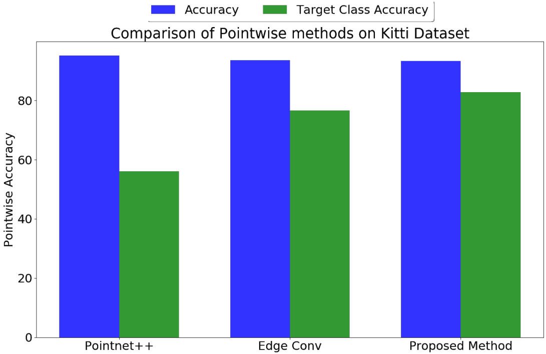
Pointer Semantic Segmentation

Implementation Details

- Used Skip connections to increase the accuracy $\rightarrow$ model size increases.
- Loss : Weighted cross-entropy $\rightarrow$ give more loss to target class due to class imbalance.
- Machine : Turing Cluster Nvidia Pascal P100
- Training times : approx 2 days

Pointer Semantic Segmentation

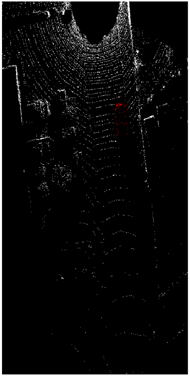
Model Accuracy on Kitti Dataset		
Model	Accuracy	Target Class Accuracy
Pointnet++	97.12	45.34
EdgeConv	95.20	75.20
Pointer	94.88	**83.40**



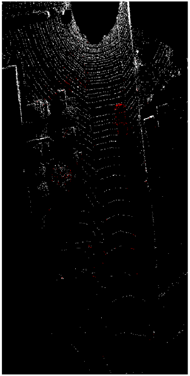
Pointer Semantic Segmentation Visuals

Camera Image

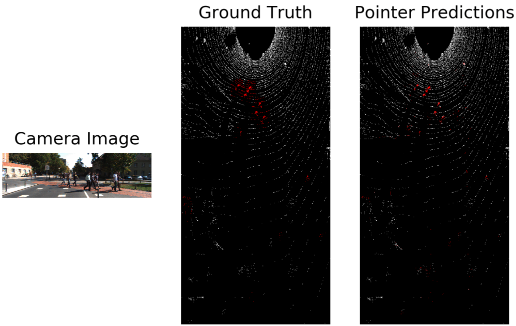
Ground Truth



Pointer Predictions



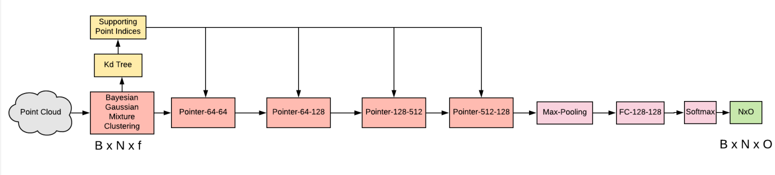
Pointer Semantic Segmentation Visuals (Pedestrians)



Pointer results (Pedestrains)

Pointer Instance

Pointer Clustering Instance Segmentation

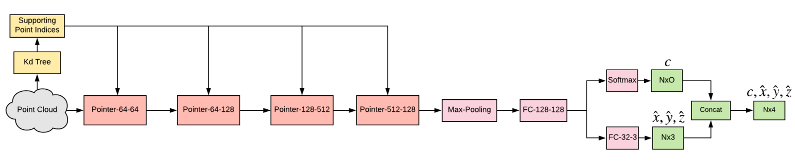


Pointer Instance Segmentation Architecture

- B is the number of clusters formed and batch size → which was 20 for experiments.

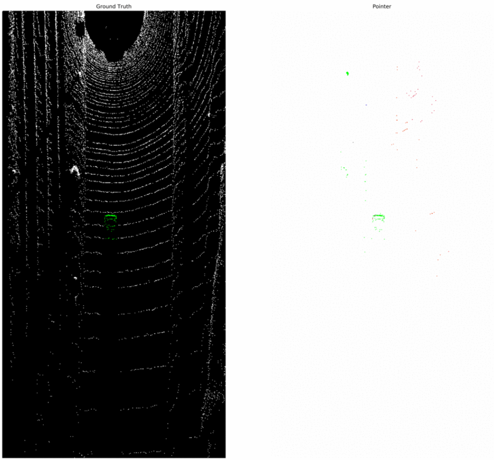
Bayesian Gaussian Mixture

Pointer Vector Instance Segmentation



Pointer Instance Segmentation Architecture

Pointer Clustering Instance Segmentation Visuals



Pointer Vector Instance Segmentation Visuals

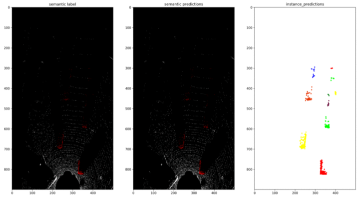


Figure: Pointer Instance Segmentation results

Pointer Instance Segmentation results link

Pointer Instance Segmentation

Model Accuracy on Kitti Dataset

Model	Accuracy	Target Class Accuracy
Pointer Clustering	91.59	40.62
Pointer Vector	93.38	82.91

Future Work

- Efficient sampling method to reduce the number of inputs points
- Reducing the size and inference time of the model
- Efficient clustering method for Instance Segmentation

Conclusion

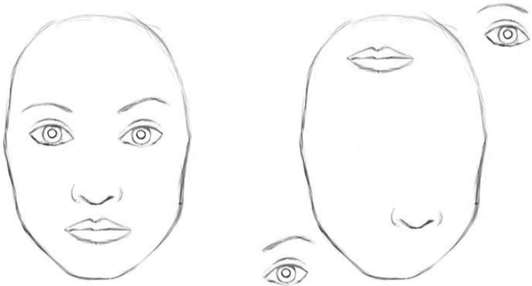
- Pointer is more robust and camera independent pipeline for segmenting vehicles and pedestrian for an autonomous vehicle perception stack.
- Pointer is invariant to lighting conditions.
- Pointer is one of the initial approach to do instance segmentation using LIDAR data alone.
- Pointer can contribute to the development of self-driving vehicle perception stack to make roads more safer for pedestrains.

References I

- Andreas Geiger, **Are we ready for autonomous driving? the kitti vision benchmark suite**, Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (Washington, DC, USA), CVPR '12, IEEE Computer Society, 2012, pp. 3354–3361.
- Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas, **Pointnet: Deep learning on point sets for 3d classification and segmentation**, CoRR **abs/1612.00593** (2016).
- Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas, **Pointnet++: Deep hierarchical feature learning on point sets in a metric space**, CoRR **abs/1706.02413** (2017).
- Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton, **Dynamic routing between capsules**, CoRR **abs/1710.09829** (2017).
- Martin Simon, Stefan Milz, Karl Amende, and Horst-Michael Gross, **Complex-yolo: Real-time 3d object detection on point clouds**, CoRR **abs/1803.06199** (2018).
- Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon, **Dynamic graph CNN for learning on point clouds**, CoRR **abs/1801.07829** (2018).

Pointer Capsnet

Capsule network



Convolutional neural network have the same prediction for both of the images.

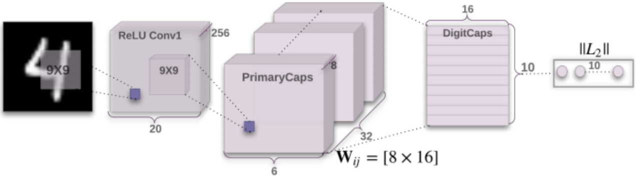
- Internal data representation of a convolutional neural network does not take into account important spatial hierarchies between simple and complex objects.
- Hinton argued that in order to correctly do classification and object recognition, it is important to preserve hierarchical pose relationships between object parts.

Capsule network

CNN do not have this capability to understand the change in orientation

- Capsules encode probability of detection of a feature as the length of their output vector and the state of the detected feature is encoded as the direction in which that vector points to.
- when detected feature moves around the image or its state somehow changes, the probability still stays the same (length of vector does not change), but its orientation changes.

Capsule network

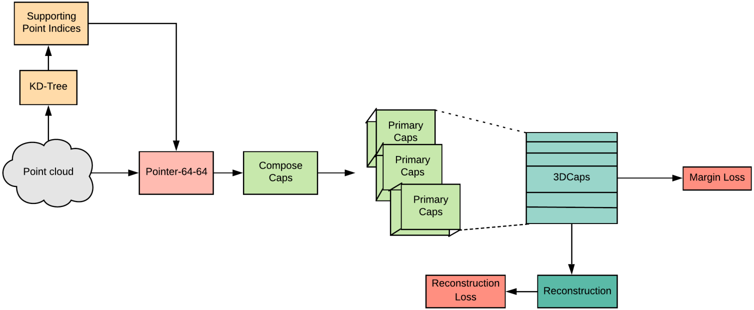


CapsNet Architecture [SFH17]

Approach

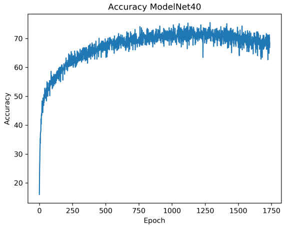
- The original capsule relies on the existence of a spatial relationship between elements in the feature map
- Whereas such features are lost in point permutation invariant formulation of 3D pointwise classification methods.
- We tried to extend capsule network for 3D point clouds by given the capsules the features extracted by pointer network.

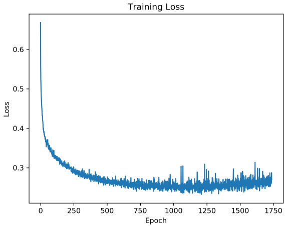
Pointer Capsnet Architecture



Pointer Capsnet Architecture

Pointer Capsnet Results on ModelNet40





Pointer Capsnet Accuracy and loss

Pointer Capsnet Results on ModelNet40

Model Accuracy on ModelNet40	
Model	Accuracy
Pointnet	89.2
Pointnet++	90.7
EdgeConv	92.2
3D Capsule (with Edgeconv)	92.7
Pointer Capsnet	**71.29**