3D Point Cloud Classification

DSA5201 DSML Industry Consulting and Applications Project

Wang Jiangyi, A0236307J

Department of Data Science and Machine Learning
National University of Singapore

13 Apr 2023

Outline

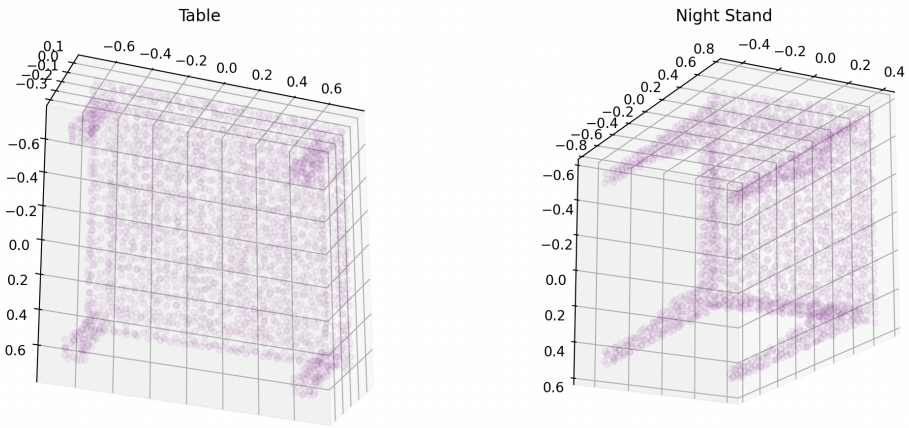
Outline

Point Cloud Data

What is Point Cloud Data?

One Point Cloud Data D_i uses a set of (x,y,z)-coordinates to characterize one object, namely $D_i = \{(x_i, y_i, z_i)\}_{i=1}^{N}$.

- Examples from ModelNet40[1] Dataset



- ModelNet40 Dataset is one benchmark of classification task, which contains 40 categories data.

Point Cloud Data

Why we need Point Cloud Data?

Real-life Application related to Point Cloud Data:

- Autonomous driving (Object Detection)
- Robotics (Object Detection)
- Industrial inspection and quality control (Classification)
- Healthcare (Classification)

In our project, we focus on Point Cloud Classification task since it is the basic of Detection task. Once we derive some efficient models in Classification task, we can easily transfer them into Detection task.

Existing Approaches

Tradition Computer Vision Approach

Given one point cloud data, tradition Computer Vision approaches can be decomposed into 2 stages:

- Stage 1: Keypoint Detection
  - We want to find those important points which construct the object.
- Stage 2: Keypoint Description
  - We want to give robust representation to those important points via some geometry quantities.

Existing Approaches

Deep Learning-based Approach

In Deep Learning-based approach setting, we mainly focus on Stage 2 problem: How to attain high-quality point representation?

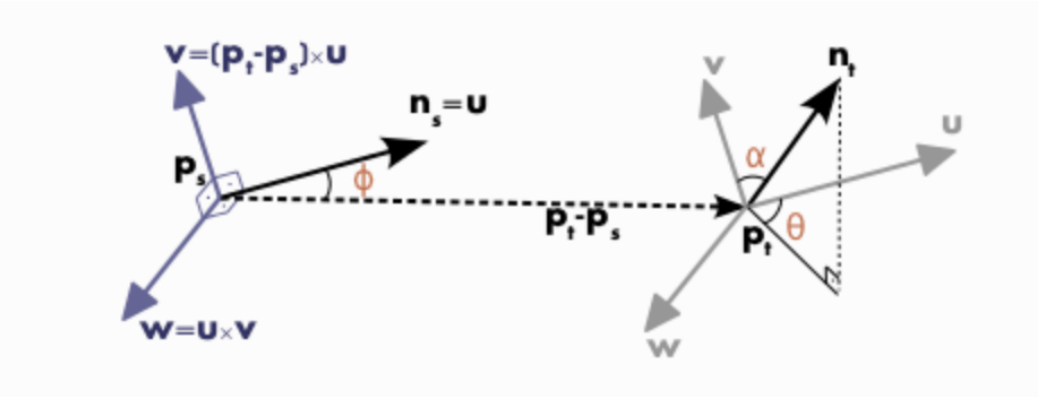
- General idea: we build a feature extractor via neural network.
- Difficulties of Point Cloud Data:
  - Unorderness, sparisity, irregularity.
  - Conclusion: it is difficult to apply previous network architecture directly.
- Two pioneering works: (discuss in detail later)
  - PointNet[2]
  - PointNet++[3]

Outline

Point Feature Histogram (PFH)

Brief Introduction

Intuition: PFH[4] captures local geometry information around one interested point p^* by computing a histogram of the relationships between the interested point p^* and its k-neighborhood.



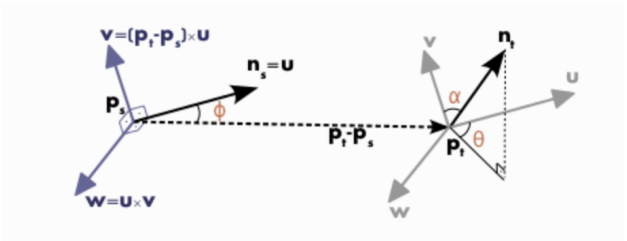
Point Feature Histogram (PFH)

Detailed Procedure

Detailed procedure for PFH feature with point p^*:

- Given point p^*, find its k-neighborhood.
- For each point pair (p_i, p_j), construct 3-angle features as follows (there are totally $\frac{k(k-1)}{2}$ pairs):

$$\begin{cases} u = n_s \\ v = u \times \dfrac{(p_t - p_s)}{||p_t - p_s||_2} \\ w = u \times v \end{cases} \qquad \begin{cases} \alpha = v \cdot n_t \\ \phi = u \cdot \dfrac{(p_t - p_s)}{||p_t - p_s||_2} \\ \theta = \arctan(w \cdot n_t / u \cdot n_t) \end{cases}$$



Point Feature Histogram (PFH)

Detailed Procedure

Detailed procedure for PFH feature with point p^*:

- Given point p^*, find its k-neighborhood.
- For each point pair (p_i, p_j), construct 3-angle features as follows (there are totally $\frac{k(k-1)}{2}$ pairs):

$$\begin{cases} u = n_s \\ v = u \times \dfrac{(p_t - p_s)}{||p_t - p_s||_2} \\ w = u \times v \end{cases} \qquad \begin{cases} \alpha = v \cdot n_t \\ \phi = u \cdot \dfrac{(p_t - p_s)}{||p_t - p_s||_2} \\ \theta = \arctan(w \cdot n_t / u \cdot n_t) \end{cases}$$

- For each α, θ, ϕ, we have $\frac{k(k-1)}{2}$ values from different pairs. Determine N_{bin} to generate histogram for each α, θ, ϕ based on these $\frac{k(k-1)}{2}$ values.
- Finally, we can derive PFH $f^* \in \mathbb{R}^{3 \cdot N_{bin}}$ for point p^*.

Fast Point Feature Histogram (FPFH)

Main limitation of PFH:

- For only one interested point p^*, we need to consider $O(k^2)$ number of neighbors to construct PFH feature.

Modification on PFH (FPFH[5]):

- Only consider those pairs directly connected with point p^* to construct Simplified PFH (SPFH).
- Within the k-neighborhood of p^*, use linear combination of SPFH to approximate PFH.
  - Weight is inversely proportional to direction with respect to point p^*.

Modified Point Feature Histogram (MPFH)

Main limitation of FPFH:

- Inversely weighted average scheme will make FPFH is no longer a well-defined histogram representation.
- α and ϕ are dot-product values, whose geometry information can be enhanced via non-linear function $arccos(\cdot)$

Our contribution is, Modified PFH (MPFH):

- Propose $\gamma-$weighted scheme, maintaining histogram property.

$$MPFH(p^*) = \gamma \cdot SPFH(p^*) + (1-\gamma) \cdot \sum_{i=1}^{k} \frac{1}{k} SPFH(p_i)$$

- γ balances the attention on interested point p^* and its k-neighborhood.

Modified Point Feature Histogram (MPFH)

Main limitation of FPFH:

- Inversely weighted average scheme will make FPFH is no longer a well-defined histogram representation.
- α and ϕ are dot-product values, whose geometry information can be enhanced via non-linear function *arccos*$(\cdot)$

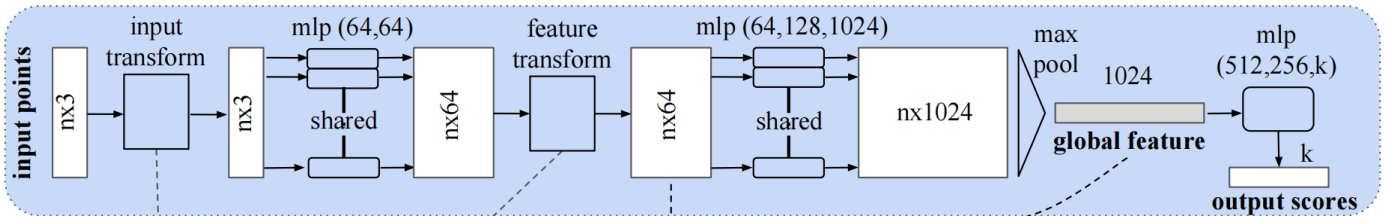
Our contribution is, Modified PFH (MPFH):

- Propose $\gamma-$weighted scheme, maintaining histogram property.
- Normalize previous 3 coordinates. Then achieve corresponding 3 angles via:

$$\begin{cases} \alpha = \arccos(v \cdot n_t) \\ \phi = \arccos(u \cdot \dfrac{(p_t - p_s)}{||p_t - p_s||_2}) \\ \theta = \arctan(w \cdot n_t / u \cdot n_t) \end{cases}$$

PointNet

Architecture:

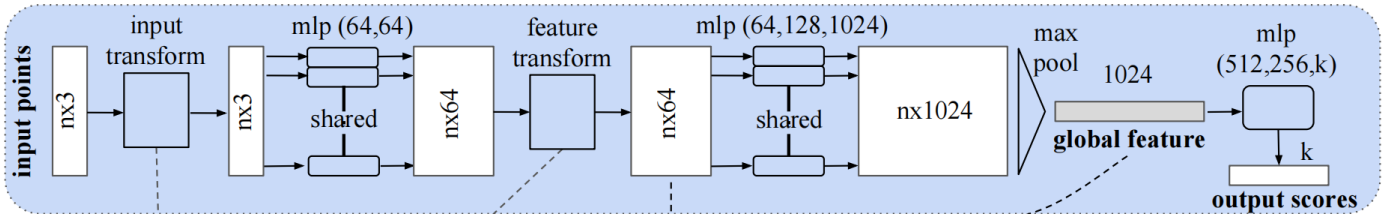


General idea:

- Utilize 3 shared-mlp (implement via 'conv1D') to encode features into richer semantic meaning.
- Highlight: introduce last pooling layer to achieve permutation invariance property.
  - Invariant with the order of input points.
  - Handle the undernoss and irregularity of point cloud data as we mentioned before.

PointNet

Architecture:

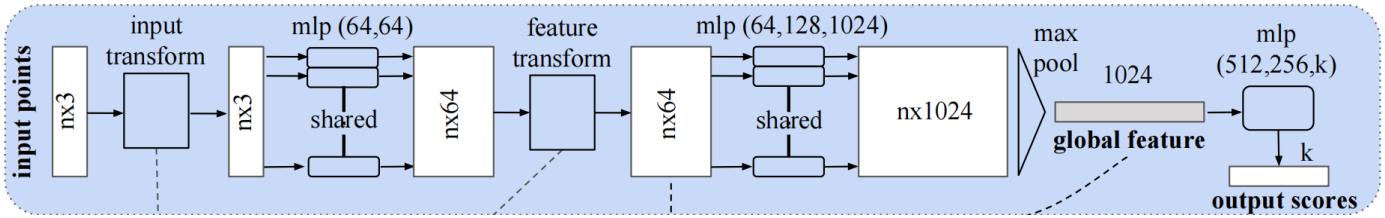


General idea:

- Utilize 3 shared-mlp (implement via 'conv1D') to encode features into richer semantic meaning.
- Highlight: introduce last pooling layer to achieve permutation invariance property.
- Design a learnable transformation matrix (T-net) to align coordinates into canonical order.
  - Removed in future variations.

PointNet

Architecture:

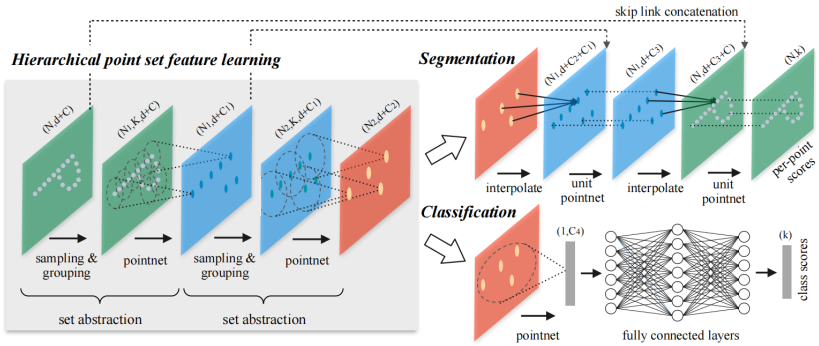


High-level understanding of Pointnet:

- Given n point cloud data as input, return one d-dim vector to represent these n point cloud data as output.
- It is similar with mechanism of kernel (convolution) in CNN.
  - Recap: $d-(3 \times 3$ kernel) will aggregate local 3×3 region channels into one d-dim vector

PointNet++

Architecture:



General idea (mimic CNN):

- Utilize k-NN + PointNet to mimic convolution operation.
- Utilize Farthest Point Sampling (FPS) to mimic pooling (down-sampling) operation.
- Apply hierarchical approach to capture local geometry.

Outline

Brief Introduction

The mainstream of recent research is to replace PointNet with some more sophisitcated local feature extractor. Generally speaking, it can be decomposed into two categories:

- Convolution-based
  - e.g., PointConv[6], PointCNN[7] e.t.c.
- Attention-based
  - e.g., Point Transformer[8], Point Cloud Transformer[9] e.t.c.

Proposed Method

Rethink the design of CNN, its convolution operation is indeed quite similar with 'filter' in tradition computer vision.

- It can be viewed as, introducing the inductive bias efficiently.

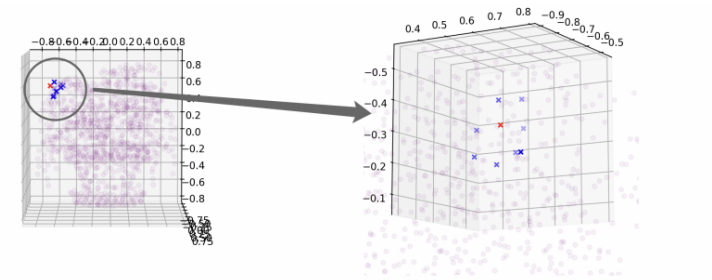
However, in point cloud task, tradition computer vision will not utilize architecture like MLP to capture local geometry.

- Therefore, we prefer not to let MLP learn how to generate high-level features from coordinates only.
- Our approach is, concatenating geometry features with coordinates and let MLP learn high-level features from them.

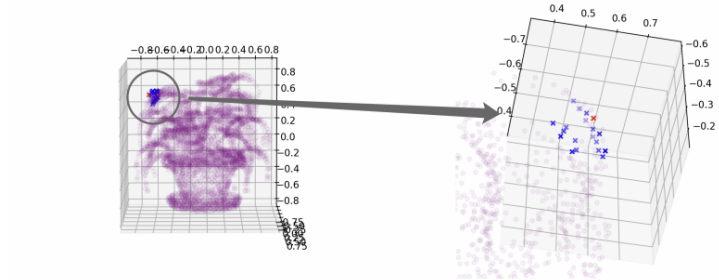
Outline

Effects of MPFH Hyper-parameters

Here, we mainly focus on, how N_{total} and k will influence the quality of MPFH feature representation.



(a) $N_{total} = 1000, k = 8$



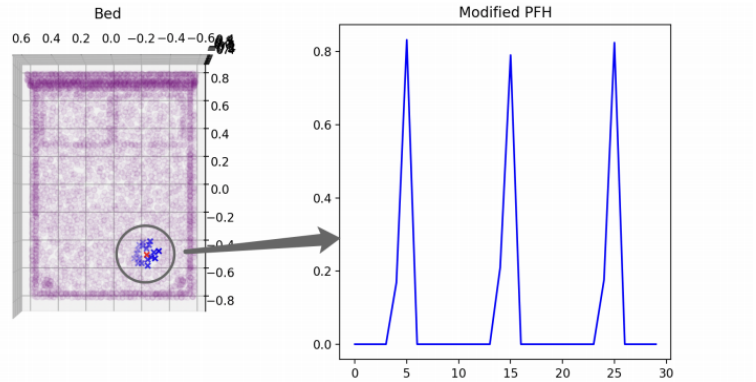
(b) $N_{total} = 4000, k = 18$

Conclusion:

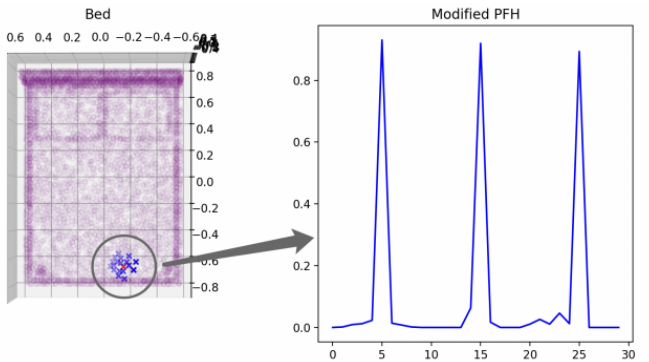
- N_{total} should be an intermediate value.
- k should be comparable with N_{total}

Interpretation of MPFH

Here, we fix $N_{total} = 4000, N_{bin} = 10, k = 18$ to generate MPFH feature. The interpretation of feature for the simplest case is:



(a) Center of Flat Plane



(b) Boundary of Flat Plane

Intuition:

- MPFH reflects curvature of local region.
- MPFH characterizes how local plane is oriented via angle histogram

Experiments on ModelNet40

Experiment Setting

Experiment setting:

- Dataset: ModelNet40
  - 9,843 training samples
  - 2,468 testing samples
- Optimizer: 'Adam'
  - $\alpha = 0.9, \beta = 0.999$
- Learning rate decay: exponential decay scheme
  - step size $d = 20$, decay rate $\gamma = 0.7$
- Data augmentation:
  - randomly rotate and dropout
  - inject Gaussian noise with 0 mean and 0.02 standard deviation

Experiments on ModelNet40

Performance measure

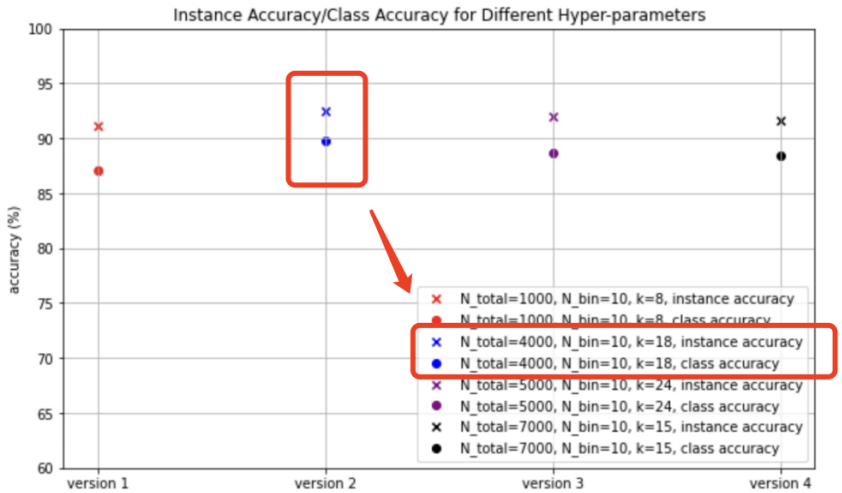
Performance measure for point cloud classification task:

- Instance Accuracy: The percentage of samples that are correctly classified over total samples.
- Mean Accuracy: Firstly compute instance accuracy for each classs. Secondly average it over all classes.
  - It is a more robust performance metric.

Experiments on ModelNet40

Experiment 1: Optimal Hyper-parameters for MPFH

Firstly, determine optimal hyper-parameters (N_{total}, k) for MPFH.



Conlusion:

- Optimal Hyper-parameters: $\hat{N}_{total} = 4000, \hat{k} = 18$.

Experiments on ModelNet40

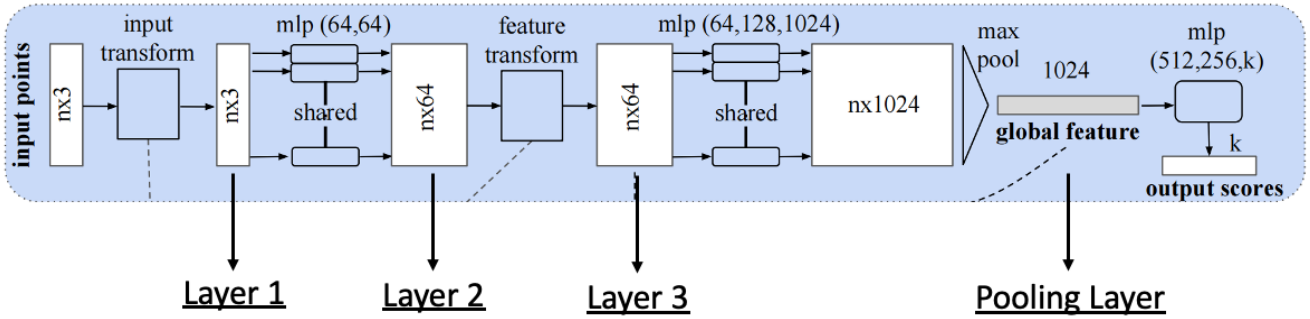
Experiment 2: Design of PointNet with MPFH

Secondly, we consider how to combine PointNet and MPFH in the optimal way.

Experiments on ModelNet40

Experiment 2: Design of PointNet with MPFH

Recap: PointNet Architecture:



Experiments on ModelNet40

Experiment 2: Design of PointNet with MPFH

Secondly, we consider how to combine PointNet and MPFH in the optimal way.

Method	ModelNet40 Dataset	
	Instance Accuracy	Mean Accuracy
PointNet (Baseline)	89.6	86.3
PointNet + Normal (Layer 1)	91.3	87.7
PointNet + MPFH (Layer 1)	91.9	89.2
PointNet + MPFH (Layer 3)	**92.4**	**89.9**
PointNet + MPFH (Pooling Layer)	90.9	87.6

Observation:

- Concatenate MPFH in Layer 3 will lead to best performance!
- With the help of MPFH, we improve the performance of PointNet from 89.6 to 92.4 in instance accuracy!

Experiments on ModelNet40

Experiment 2: Design of PointNet with MPFH

Secondly, we consider how to combine PointNet and MPFH in the optimal way.

Method	ModelNet40 Dataset	
	Instance Accuracy	Mean Accuracy
PointNet (Baseline)	89.6	86.3
PointNet + Normal (Layer 1)	91.3	87.7
PointNet + MPFH (Layer 1)	91.9	89.2
PointNet + MPFH (Layer 3)	**92.4**	**89.9**
PointNet + MPFH (Pooling Layer)	90.9	87.6

Intuition:

- We can enhance the feature representation of MPFH via MLP.
- We should process coordinates into high-level features first, followed by concatenating with MPFH.
  - Add MPFH in a hierarchical manner.

Experiments on ModelNet40

Experiment 3: Design of PointNet with MPFH and Normal

Furthermore, we want to verify the effectiveness of combining MPFH in a hierarchical manner.

Method	ModelNet40 Dataset	
	Instance Accuracy	Mean Accuracy
PointNet (Baseline)	89.6	86.3
PointNet + Normal (Layer 1)	91.3	87.7
PointNet + Normal (Layer 1) + MPFH (Layer 1)	92.1	89.7
PointNet + Normal (Layer 1) + MPFH (Layer 3)	**93.0**	**90.8**
PointNet + Normal (Layer 1) + MPFH (Pooling Layer)	92.0	87.6

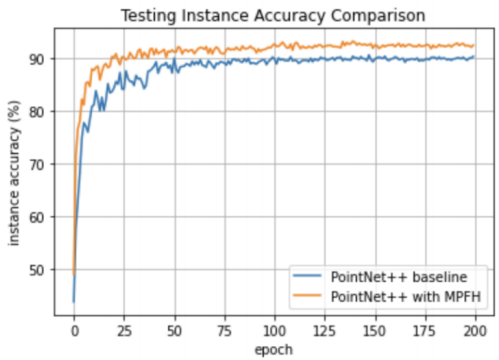
Guideline:

- Through experiments, we should add MPFH features in the last shared-MLP layer for better performance.

Experiments on ModelNet40

Experiment 4: Comparison between PoinNet++ baseline and PointNet++ with MPFH

Next, we apply the guideline from PointNet to design PointNet++ with MPFH. We conduct experiments to compare between PointNet++ baseline and PointNet++ with MPFH.



Model	Instance Accuracy
PointNet++ (Baseline)	90.6
PointNet with MPFH	92.4
PointNet++ with MPFH	**93.3**

Observation:

- With the help of MPFH, we improve the performance of PointNet++ from 90.6 to 93.3 in instance accuracy!

Experiments on ModelNet40

Experiment 5: Comparison between PoinNet++ with MPFH and RepSurf

Finally, we make comparison between our proposed PointNet++ with MPFH and SOTA RepSurf[10].

Model	ModelNet40 Dataset	
	Instance Accuracy	Mean Accuracy
PointNet++ with MPFH	93.3	**91.2**
RepSurf	**94.0**	91.1

Conclusion:

- Through utilizing local geometry from MPFH, our model is able to achieve comparable result, especially in mean accuracy.

Outline

Summary

What I have done?

Contribution:

- Propose novel Modified Point Feature Histogram, which is a more powerful feature for local geometry.
- Propose the approach to combine MPFH with PointNet and PointNet++ in a hierarchical manner.

Performance:

- Improve the instance accuracy of PointNet from 89.6 to 92.4 via MPFH.
- Improve the instance accuracy of PointNet++ from 90.6 to 93.3 via MPFH.
- Achieve comparable result with RepSurf, which is the SOTA in classification task.

Summary

What I have learnt?

Coding:

- Virtual Environment.
- Torch Framework.
- Linux Language.

Theory:

- Strong foundation in Deep Learning-based 2D and 3D computer vision tasks.
- Semi-supervised Learning techniques like knowledge distillation, self-ensemble e.t.c.

Summary and presentation ability.

Future Work

Due to the limit of time for internship, there still exists some works haven't finished yet, including:

- Conduct experiments on other benchmarks like ShapeNet40.
- Examine how to combine PointNet++ and MPFH in details.
- Explore the performance of MPFH on dense prediction task like segementation.

I will explore more on these topics in summer vacation.

Bibliography

[1] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao.
3d shapenets: A deep representation for volumetric shapes.
In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1912–1920, 2015.

[2] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas.
Pointnet: Deep learning on point sets for 3d classification and segmentation.
In CVPR, 2017a.

[3] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas.
Pointnet++: Deep hierarchical feature learning on point sets in a metric space.

Bibliography

[4] R. B. Rusu, Z. C. Marton, N. Blodow, and M. Beetz.
Persistent Point Feature Histograms for 3D Point Clouds.
In Proceedings of the 10th International Conference on Intelligent Autonomous Systems, 2008.

[5] Rusu, R. B., Blodow, N. and Beetz, M.
Fast point feature histograms (FPFH) for 3D registration.
In 2009 IEEE international conference on robotics and automation (pp. 3212-3217). IEEE.

[6] Wu, Wenxuan, Zhongang Qi, and Li Fuxin.
Pointconv: Deep convolutional networks on 3d point clouds.
In Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition (pp. 9621-9630).

Bibliography

[7] Li, Yangyan, et al.
Pointcnn: Convolution on x-transformed points.
Advances in neural information processing systems, 31.

[8] Zhao H, Jiang L, Jia J, et al.
Point transformer.
Proceedings of the IEEE/CVF international conference on computer vision. 2021: 16259-16268.

[9] Guo M H, Cai J X, Liu Z N, et al.
Pct: Point cloud transformer.
Computational Visual Media, 7, 187-199.

Bibliography

[10] Ran, Haoxi, Jun Liu, and Chengjie Wang.
Surface representation for point clouds.
In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 18942-18952).