PCT: Point Cloud Transformer

Fu Lian, M1
Prof. Oishi's Lab

2022/05/25

Introduction

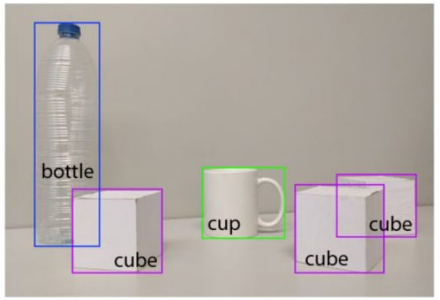
- I am **Fu Lian (傅 立安）** from **China**
- I am **Master 1 year** student in Electrical Engineering and Information Systems
- My Supervisor is **Prof. Takeshi Oishi**
- Our lab focus on **3D Vision** related topics
- My own research interest is **Deep Learning applied in RGB-D data**.

Tasks of Computer Vision

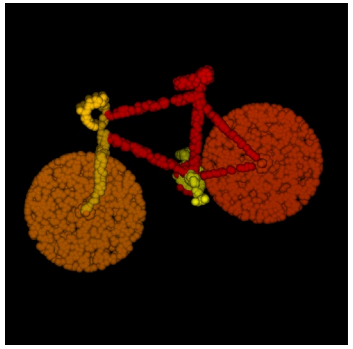
1. Image Classification
2. Object Localization
3. Segmentation
4. Pose Estimation
5. Articulation Estimation

[3][4] Garcia-Garcia A, Orts-Escolano S, Oprea S, et al. A review on deep learning techniques applied to semantic segmentation[J]. arXiv preprint arXiv:1704.06857, 2017.
[5][6] Hu Y, Fua P, Wang W, et al. Single-stage 6d object pose estimation[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020: 2930-2939.

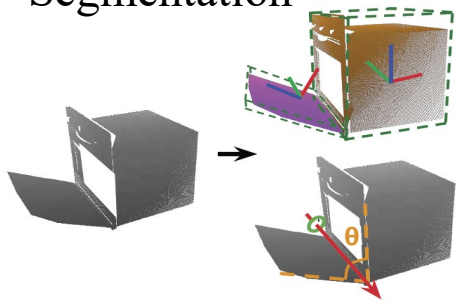
Image Classification[3]



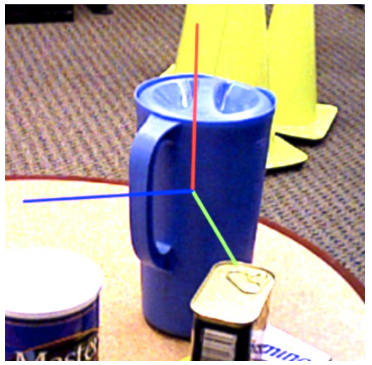
Object Localization[4]



Segmentation

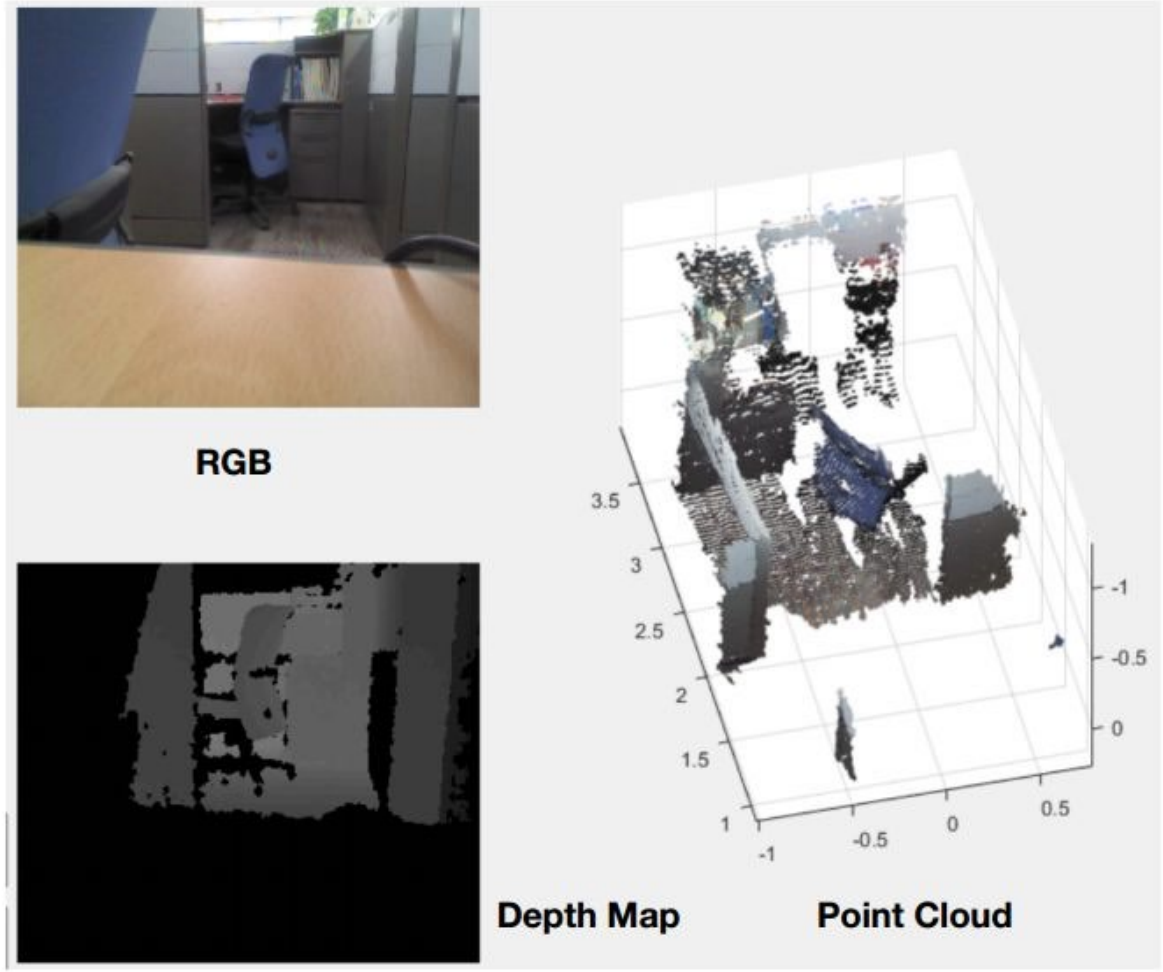


Articulation Estimation[5]



Pose Estimation[6]

2D or 3D?



Richer and more comprehensive environmental information

2D is good, but not enough

Depth information is key

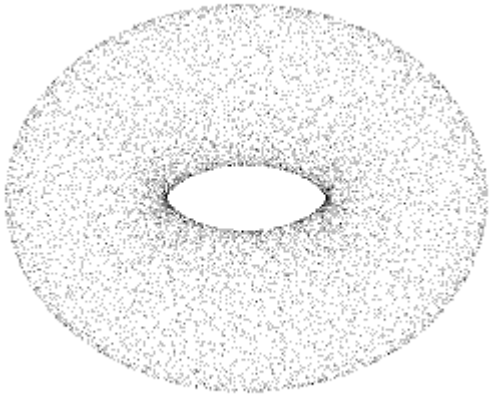
Figure cited from: http://www.tensorinfinity.com/paper_206.html

Point Cloud

Set of data points in space

Coordinates (x,y,z)

Have depth information, compared with RGB image

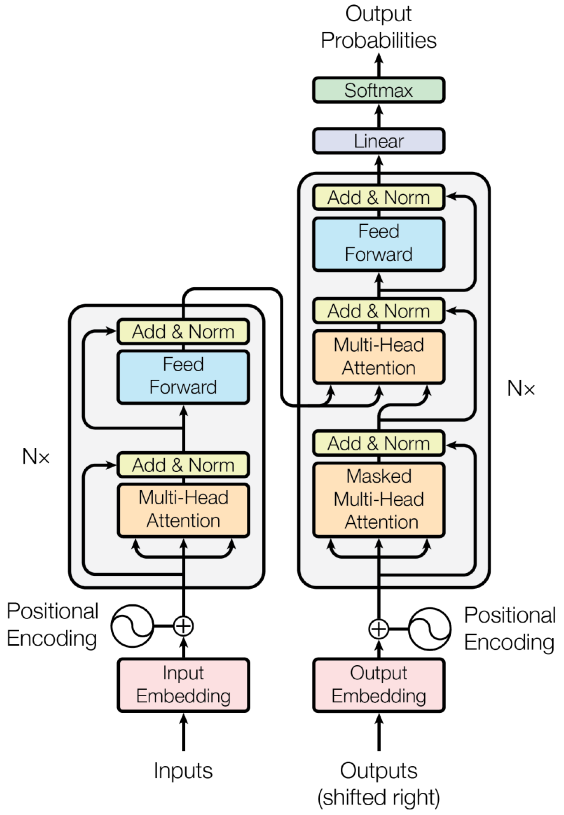


https://commons.wikimedia.org/wiki/File:Point_cloud_torus.gif

Transformer

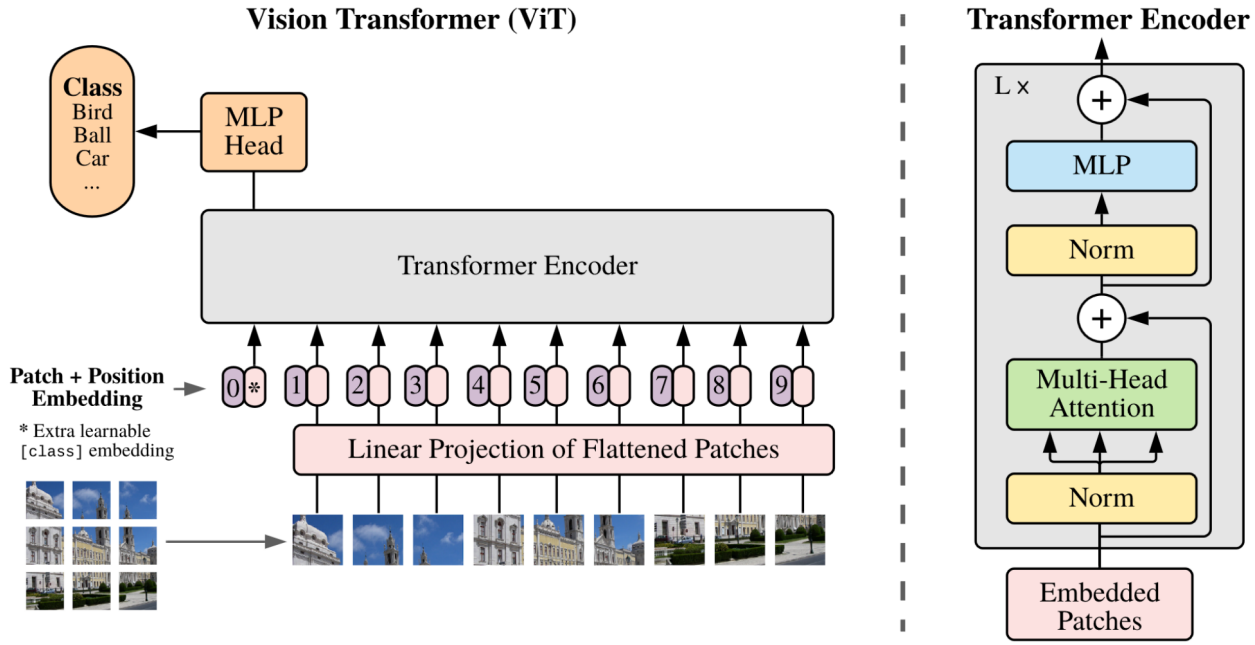
Attention Is All You Need, 2017

Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[C]//Advances in neural information processing systems. 2017: 5998-6008.



An image is worth 16x16 words: Transformers for image recognition at scale, 2020

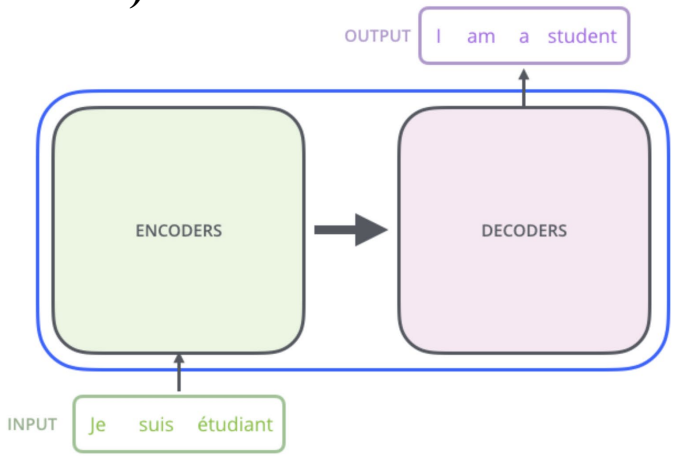
Dosovitskiy A, Beyer L, Kolesnikov A, et al. An image is worth 16x16 words: Transformers for image recognition at scale[J]. arXiv preprint arXiv:2010.11929, 2020.



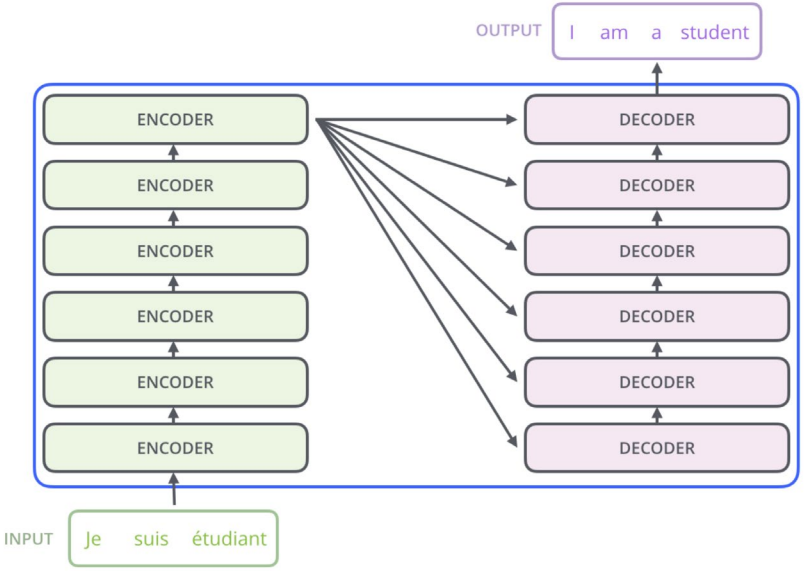
Transformer

Detailed Explanation of Attention & Transformer (Figures Cited from here):
http://jalammar.github.io/illustrated-transformer/

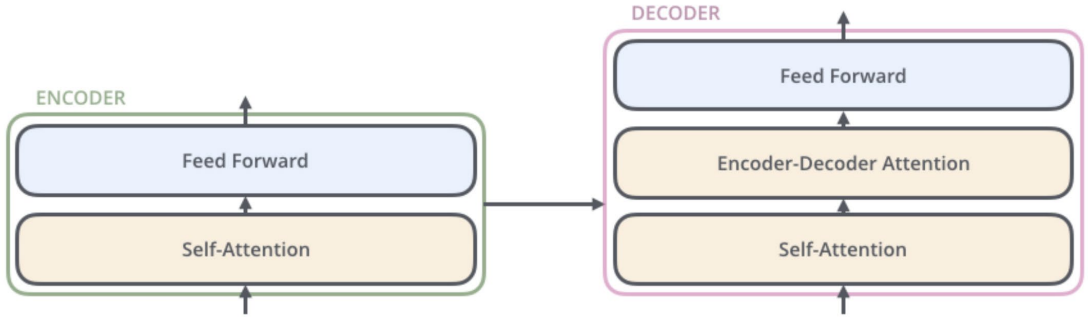
Input and Output of Transformer



Encoder-Decoder Structure



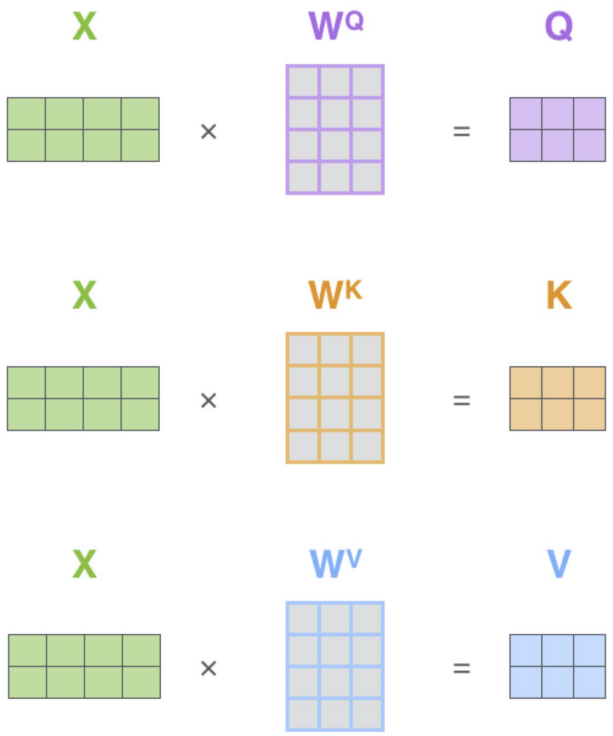
Stack of Encoder & Decoder



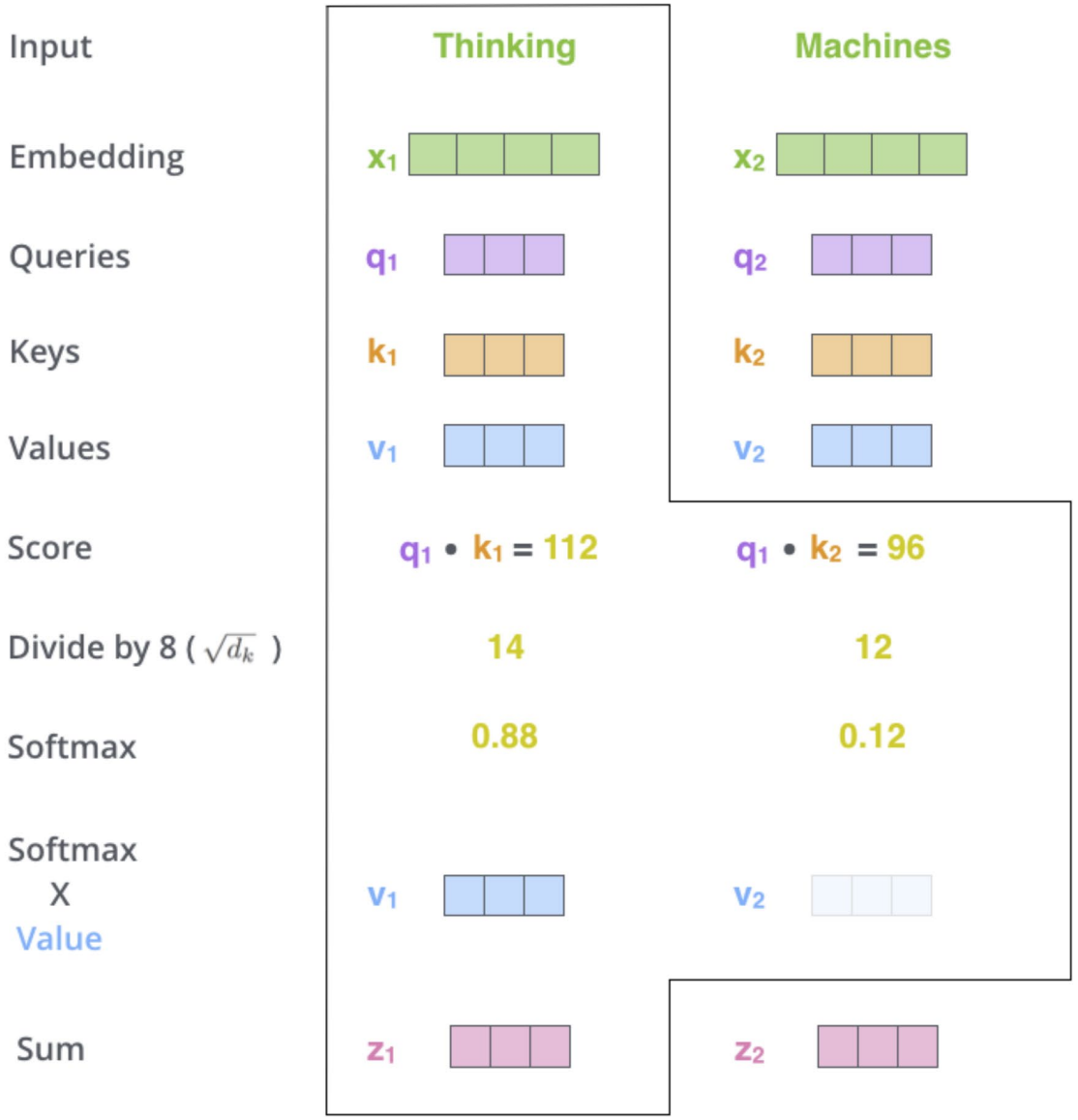
Structure of Encoder and Decoder

Self-Attention

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^{\text{T}}}{\sqrt{d_k}}\right)V$$

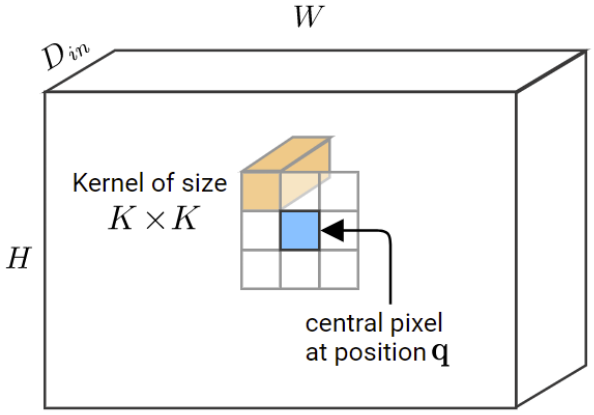


Parameters

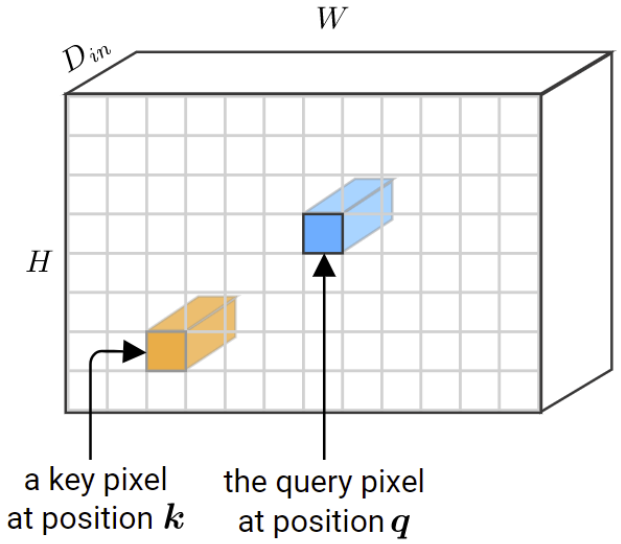


Process Procedure of Encoder

Self-Attention vs Convolutional Layer

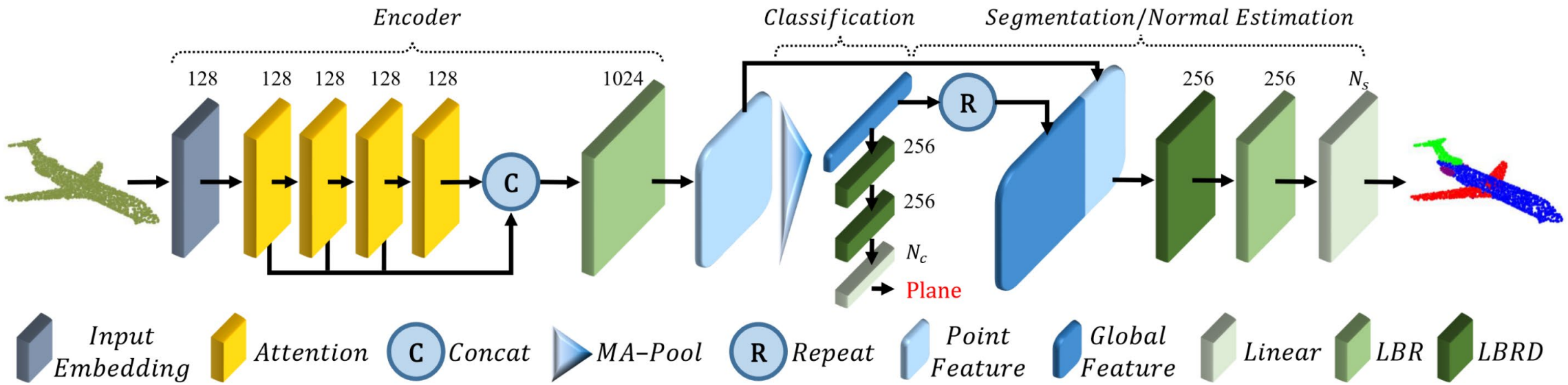


Receptive Field of Convolutional Layer



Receptive Field of Self-Attention

Point Cloud Transformer



- One of the first work applying transformer to point cloud.
- Encoder-Decoder Structure
- Use 4 attention layer as the encoder
- Decoder is Linear layer

PyTorch Lightning

https://github.com/PyTorchLightning/pytorch-lightning

Lightweight Wrapper of Pytorch

Focus on the Idea

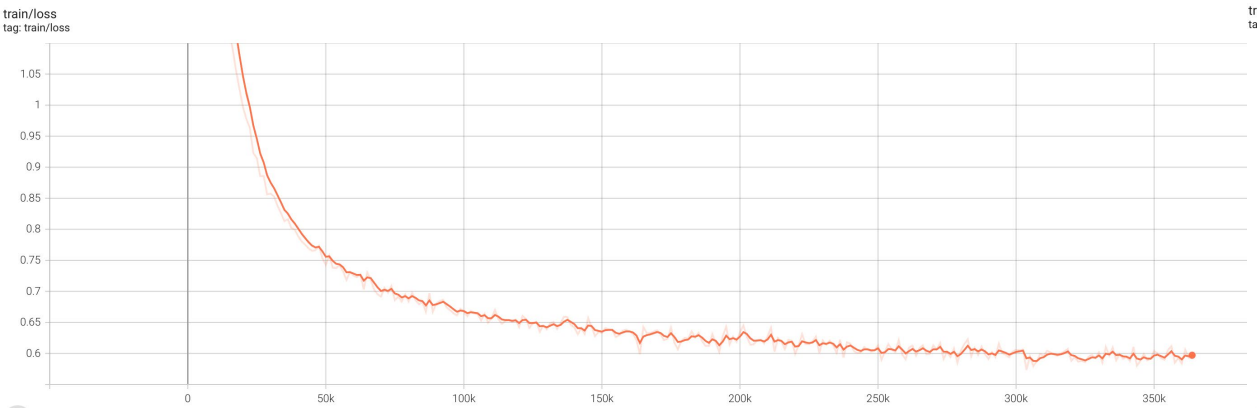
- Lightning Module: Define the pipeline
  - on_train_start: do anything
  - Training_step: here you use forward
  - training_epoch_end: do anything
  - …
- DataModule: Handles the datasets
  - Prepare the dataset, dataloader etc.
- Model: Same as Pytorch.

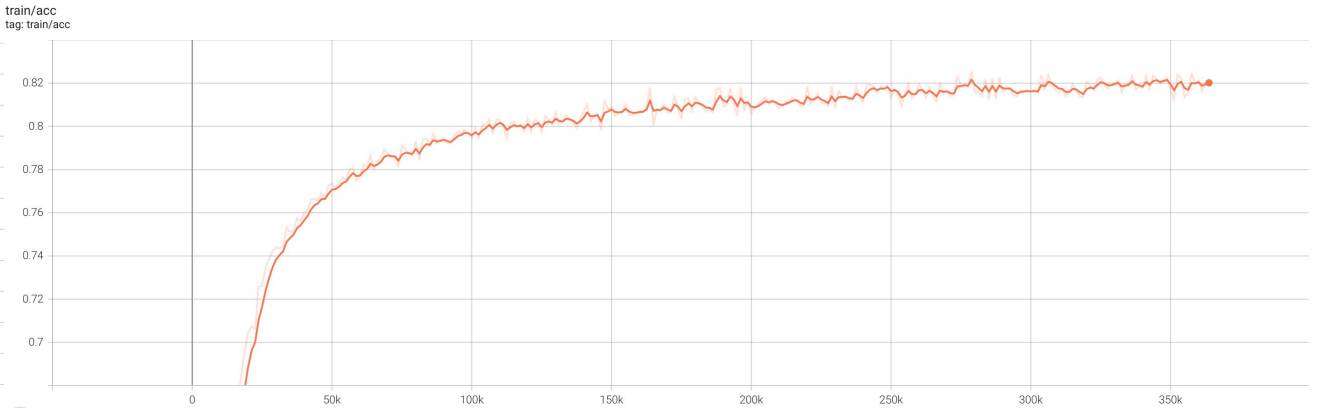
PyTorch Lightning + Hydra Template

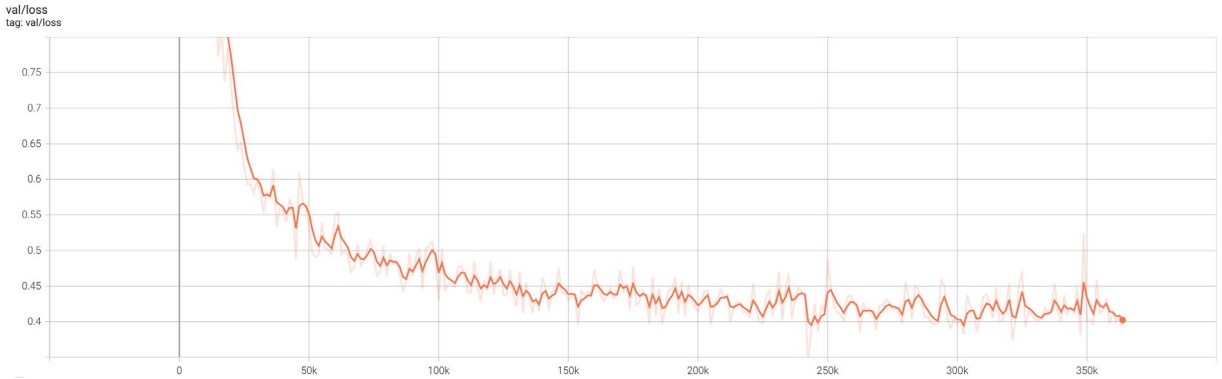
https://github.com/ashleve/lightning-hydra-template


```
├── configs                   <- Hydra configuration files
│   ├── callbacks                <- Callbacks configs
│   ├── datamodule               <- Datamodule configs
│   ├── debug                    <- Debugging configs
│   ├── experiment               <- Experiment configs
│   ├── hparams_search           <- Hyperparameter search configs
│   ├── local                    <- Local configs
│   ├── log_dir                  <- Logging directory configs
│   ├── logger                   <- Logger configs
│   ├── model                    <- Model configs
│   ├── trainer                  <- Trainer configs
│   │
│   ├── test.yaml             <- Main config for testing
│   └── train.yaml            <- Main config for training
│
├── data                   <- Project data
│
├── logs                   <- Logs generated by Hydra and PyTorch Lightning loggers
│
├── notebooks              <- Jupyter notebooks. Naming convention is a number (for ordering),
│                             the creator's initials, and a short `-` delimited description,
│                             e.g. `1.0-jqp-initial-data-exploration.ipynb`.
│
├── scripts                <- Shell scripts
│
├── src                    <- Source code
│   ├── datamodules              <- Lightning datamodules
│   ├── models                   <- Lightning models
│   ├── utils                    <- Utility scripts
│   ├── vendor                   <- Third party code that cannot be installed using PIP/Conda
│   │
│   ├── testing_pipeline.py
│   └── training_pipeline.py
│
├── tests                  <- Tests of any kind
│   ├── helpers                  <- A couple of testing utilities
│   ├── shell                    <- Shell/command based tests
│   └── unit                     <- Unit tests
│
├── test.py               <- Run testing
├── train.py              <- Run training
│
├── .env.example              <- Template of the file for storing private environment variables
├── .gitignore                <- List of files/folders ignored by git
├── .pre-commit-config.yaml   <- Configuration of pre-commit hooks for code formatting
├── requirements.txt          <- File for installing python dependencies
├── setup.cfg                 <- Configuration of linters and pytest
└── README.md
```

Results

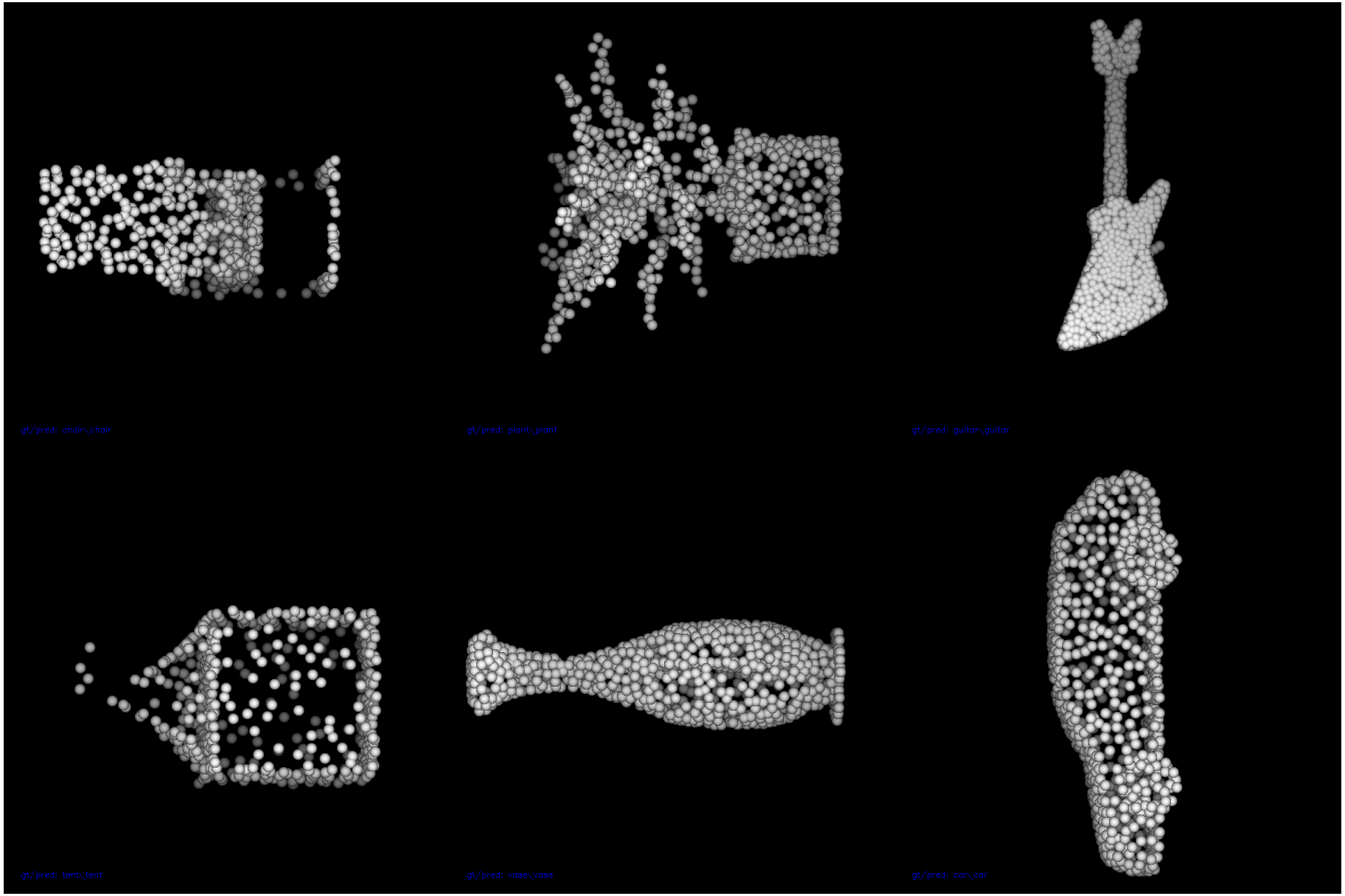

train/loss
tag: train/loss
1.05
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0
50k
100k
150k
200k
250k
300k
350k


train/acc
tag: train/acc
0.82
0.8
0.78
0.76
0.74
0.72
0.7
0
50k
100k
150k
200k
250k
300k
350k

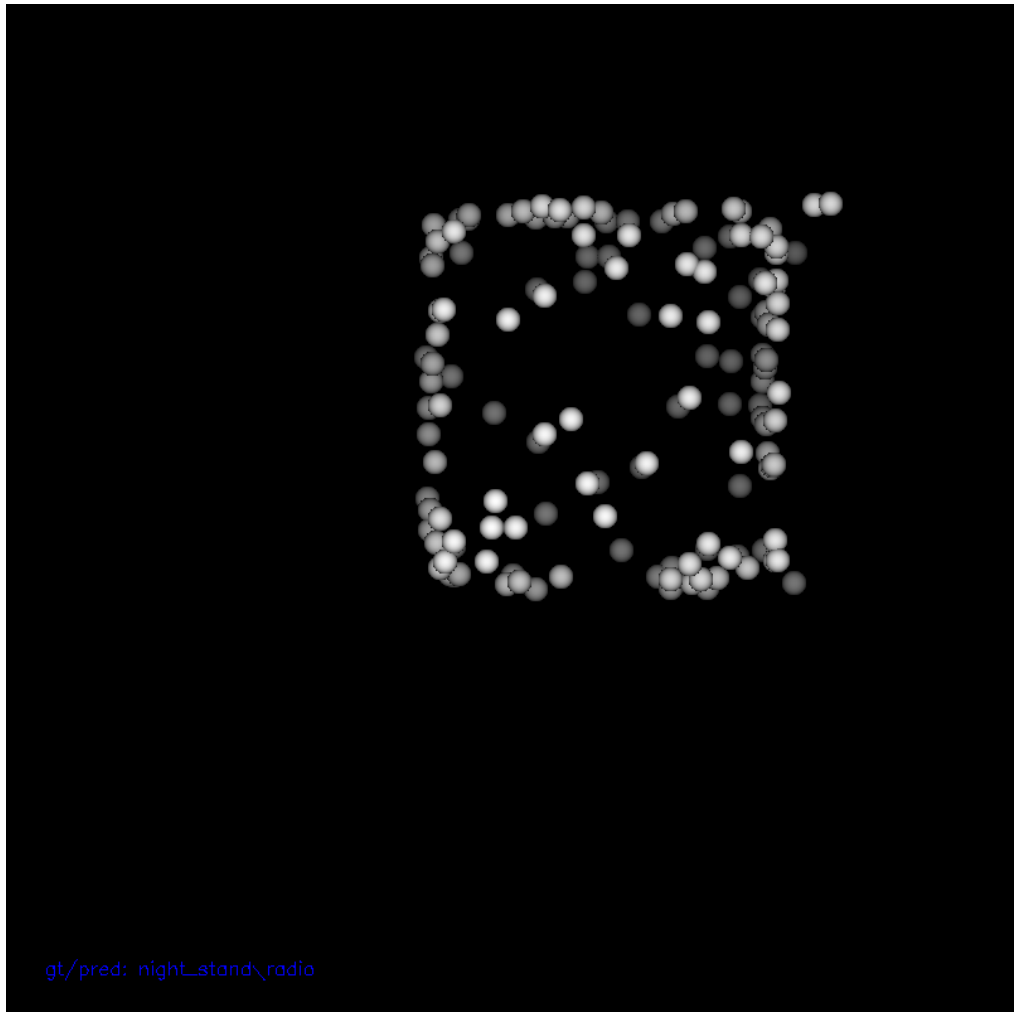

val/loss
tag: val/loss
0.75
0.7
0.65
0.6
0.55
0.5
0.45
0.4
0
50k
100k
150k
200k
250k
300k
350k

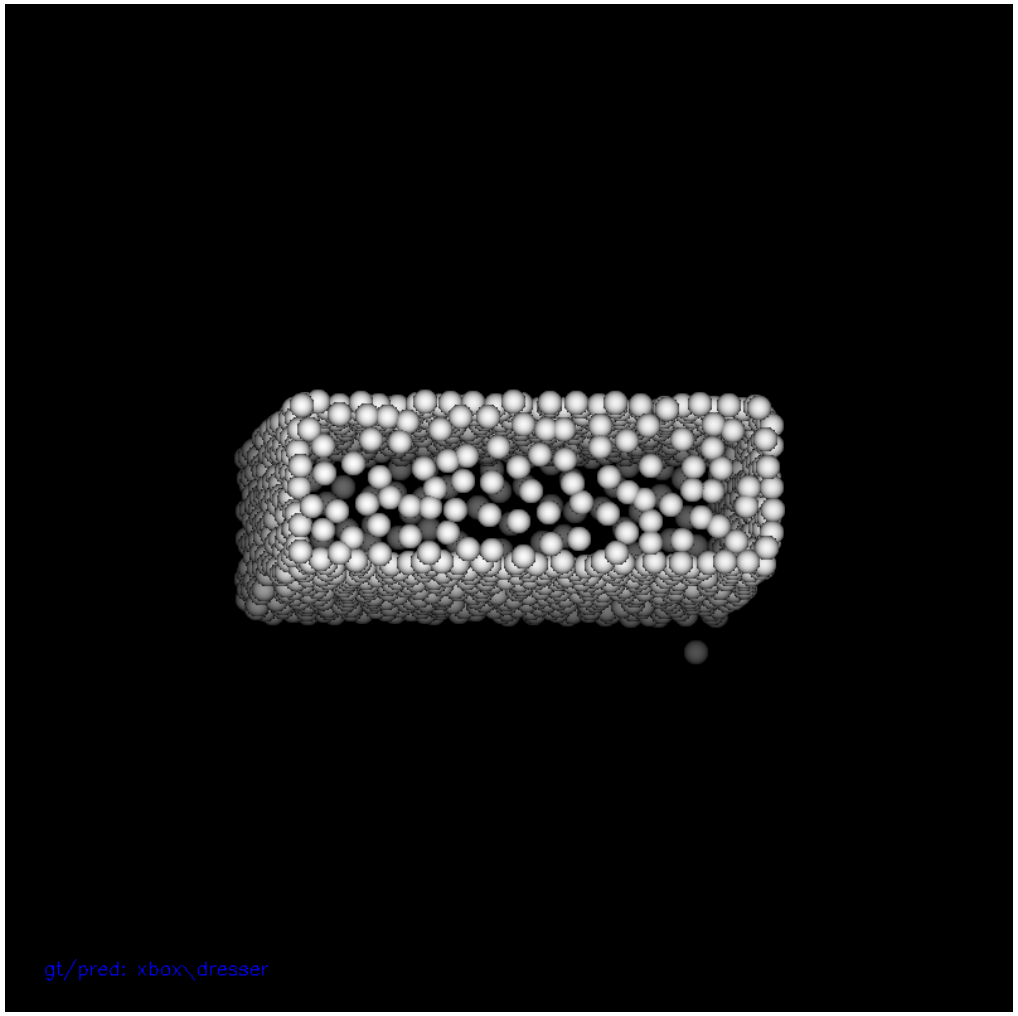

val/acc
tag: val/acc
0.88
0.87
0.86
0.85
0.84
0.83
0.82
0.81
0.8
0.79
0.78
0.77

Results


gt/pred: chair\chair
gt/pred: plant\plant
gt/pred: guitar\guitar
gt/pred: tent\tent
gt/pred: vase\vase
gt/pred: car\car

Some Missed Cases





Potential Pull Requests

- Only re-implement the classification branch. Why not try achieving segmentation with PCT
- Maybe better visualization methods.
    - Replace the C++ with python.
    - Better appearance.
- Using different Dataset! This involves writing some dataset modules.

PCT: Point Cloud Transformer

Fu Lian, M1
Prof. Oishi's Lab

2022/05/25