Week 14 Report: Point Transformer V2 for Point-wise Segmentation: A Failed Experiment

Hatem Feckry (120220264)
Seifeldeen Mohamed Galal (120220252)
Abdelrahman Ashraf (120220292)
Ahmed Elsherbeny (120220271)

May 15, 2025

1 Introduction

This week, our main goal was to use the Point Transformer V3 model to perform point-wise segmentation. This was the next logical step for our project after using YOLO for set classification. We had to format our data to resemble a dataset compatible with the model. We then modified the configuration of the model, adjusted the architecture, and even changed the model version. Despite all our efforts, the results were extremely underwhelming. The model either guessed all points as type 1 or type 0, indicating a local minimum. We attempted to adjust the learning parameters and optimizer configuration, but to no avail. This report outlines our work in detail, speculates on the reasons for the model's failure, and proposes next steps.

2 The Training Environment and Resolving Technical Issues

Last week, we faced immense challenges when trying to run the model both locally and remotely on Colab. The primary issues were:

- VRAM requirements (8GB).
- Specific CUDA version compatibility.
- The need for an Ampere-based GPU or newer.

While Colab offers up to 16GB of VRAM and supports the required CUDA version, the provided NVIDIA T4's architecture is Tesla Turing, which is older than Ampere. To resolve this, we installed Linux on a laptop with an RTX 3070 mobile GPU, which barely met the VRAM requirements. We also familiarized ourselves with Conda environments, enabling us to run the model locally with acceptable performance after parameter tweaking.

3 Data Formatting

The data used for training this model was the same as that used for training the YOLO model, with a slight modification. Since we were training for segmentation, we used sets of 16 points that had no convex hexagon and added another random 16 points. It is proven that any set of points larger than 16 must contain a convex hexagon, so the additional points were considered noise. The original set was concatenated with the noise, and the resulting list was shuffled. We labeled the original points as class 0 and the noise as class 1. In total, we had:

- 2800 training sets, each containing 32 points.
- 700 validation sets.

To format the data into a structure compatible with the model, we chose the SemanticKITTI dataset for the following reasons:

1. It contains only LiDAR data, which intuitively translates to our data.
2. It has minimal metadata.

We downloaded a sample scene (scene 08) and cloned its point cloud data and labeling schemes. To preserve the model architecture, we used a 4D representation of points by replicating the x and y coordinates in the next two dimensions. This increased the relative difference between points, making them more "unique" in the latent space. We created two scenes (00 and 01) for training and validation, respectively. Each scene contained binary files storing the point data as `float32` and the labels as `unsigned int32`. We verified that the model correctly accessed the data before proceeding to training.

4 Training Setup and Adjustments

We initially aimed to use the latest version of Point Transformer (V3). However, no configuration for V3 with the SemanticKITTI dataset was available, and other datasets either included extra sensors or offered data in RGB-D format, which was not easily mappable to our data. Thus, we opted for the V2 model, which uses relative positional embedding and a traditional distance-based embedding.

The default model settings required adjustments for successful training:

1. Adjusted the decoder to output the same dimensions as the input.
2. Used a batch size of 32 and prevented data mixing.
3. Processed 15 point clouds simultaneously for better efficiency.
4. Adjusted the weights and classes to only two, with equal loss weights.

Despite these changes, the model training did not converge meaningfully.

5 Results of Training and Improvement Rounds

After 50 epochs of training, the model achieved a mean Intersection over Union (mIoU) of 0.25. However, further inspection revealed that the model assigned all points to a single class. This resulted in:

- 100% accuracy for the chosen class (50% IoU).
- 0% accuracy for the other class.

We attempted to penalize this behavior using asymmetric losses, but the model simply switched to the class with the lower loss. During training, the model explored other approaches but failed to find any correlation better than this local minimum. Extending the training to 70 epochs and reducing evaluation epochs to 10 did not improve results. Adjusting the learning rate and Adam optimizer decay parameters also proved ineffective.

6 Analysis

We overestimated the efficacy of LiDAR-based approaches for our problem. While the transformer architecture is permutation-invariant, LiDAR point cloud segmentation relies heavily on local information. Since we used an off-the-shelf model, we lacked full understanding of its inner workings. The relationship we tasked the model with finding—identifying empty set elements—was not based on proximity or direct positional relations. A deeper model with more self-attention layers might have uncovered a deeper relationship, but the current model shows little potential for improvement.

7 Mathematical Insights

The failure of the model can be partially explained by the nature of the data and the loss function. Let $y_i \in \{0, 1\}$ be the true label of point i, and $\hat{y}_i$ be the predicted probability of class 1. The binary cross-entropy loss is:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \left[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \right].$$

If the model predicts $\hat{y}_i = c$ for all i, where c is a constant, the loss becomes:

$$\mathcal{L} = -\left[p \log(c) + (1 - p) \log(1 - c) \right],$$

where p is the proportion of class 1 points. The minimum of this loss occurs at $c = p$. In our case, $p = 0.5$, so the model converges to predicting $\hat{y}_i = 0.5$ for all i, which is equivalent to random guessing.

8 Conclusion

This week, we successfully ran the model but achieved poor results. Moving forward, we will explore building a custom transformer or finding one specifically designed for graph problems.

9 Graphs

Below are visualizations of the data and model outputs:

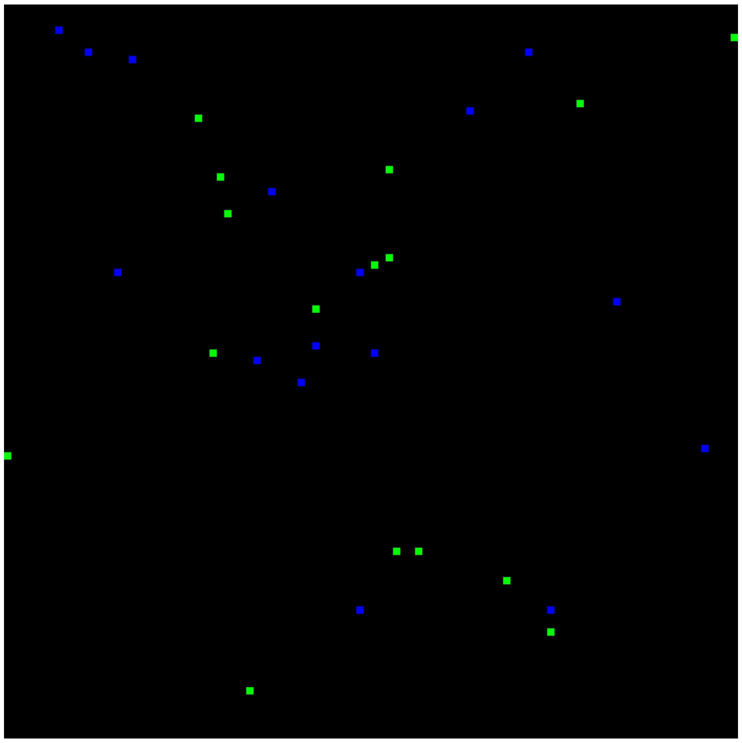
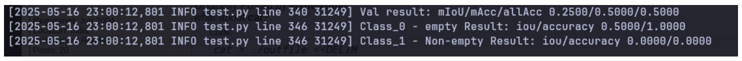
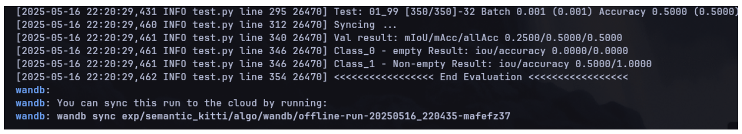


Figure 1: Input data with points colored by their true class (class 0: green, class 1: blue).



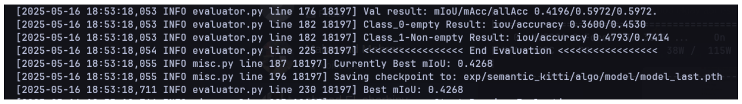
```
[2025-05-16 23:00:12,801 INFO test.py line 340 31249] Val result: mIoU/mAcc/allAcc 0.2500/0.5000/0.5000
[2025-05-16 23:00:12,801 INFO test.py line 346 31249] Class_0 - empty Result: iou/accuracy 0.5000/1.0000
[2025-05-16 23:00:12,801 INFO test.py line 346 31249] Class_1 - Non-empty Result: iou/accuracy 0.0000/0.0000
```

Figure 2: Model output where all points are predicted as class 0.



```
[2025-05-16 22:20:29,431 INFO test.py line 295 26470] Test: 01_99 [350/350]-32 Batch 0.001 (0.001) Accuracy 0.5000 (0.5000)
[2025-05-16 22:20:29,460 INFO test.py line 312 26470] Syncing ...
[2025-05-16 22:20:29,461 INFO test.py line 340 26470] Val result: mIoU/mAcc/allAcc 0.2500/0.5000/0.5000
[2025-05-16 22:20:29,461 INFO test.py line 346 26470] Class_0 - empty Result: iou/accuracy 0.0000/0.0000
[2025-05-16 22:20:29,461 INFO test.py line 346 26470] Class_1 - Non-empty Result: iou/accuracy 0.5000/1.0000
[2025-05-16 22:20:29,462 INFO test.py line 354 26470] <<<<<<<<<<<<<<<<< End Evaluation <<<<<<<<<<<<<<<<<
wandb:
wandb: You can sync this run to the cloud by running:
wandb: wandb sync exp/semantic_kitti/algo/wandb/offline-run-20250516_220435-mafefz37
```

Figure 3: Model output where all points are predicted as class 1.



```
[2025-05-16 18:53:18,053 INFO evaluator.py line 176 18197] Val result: mIoU/mAcc/allAcc 0.4196/0.5972/0.5972.
[2025-05-16 18:53:18,053 INFO evaluator.py line 182 18197] Class_0-empty Result: iou/accuracy 0.3600/0.4530
[2025-05-16 18:53:18,053 INFO evaluator.py line 182 18197] Class_1-Non-empty Result: iou/accuracy 0.4793/0.7414
[2025-05-16 18:53:18,054 INFO evaluator.py line 225 18197] <<<<<<<<<<<<<<<<< End Evaluation <<<<<<<<<<<<<<<<<
[2025-05-16 18:53:18,055 INFO misc.py line 187 18197] Currently Best mIoU: 0.4268
[2025-05-16 18:53:18,055 INFO misc.py line 196 18197] Saving checkpoint to: exp/semantic_kitti/algo/model/model_last.pth
[2025-05-16 18:53:18,711 INFO evaluator.py line 230 18197] Best mIoU: 0.4268
```

Figure 4: Intermediate validation output showing a mix of class 0 and class 1 predictions.