3D Semantic Segmentation with 3D LiDAR Point Clouds and 2D Camera Images for Autonomous Driving

Anze Liu
Stanford University
Stanford, CA
anzeliu@stanford.edu

Abstract

3D point clouds and 2D images are two different data representations of the objects and scenes in our physical world. Autonomous driving systems, a sector in rapid growth and development, have their data collected through various mediums and processed in various forms. Among which, LiDAR sensors and cameras are two primary ways of capturing the data for self-driving cars. Whether the vehicle's visual understanding of its surroundings is obtained from only 3D point clouds, or only 2D images, or both, the tradeoff in accuracy and efficiency is actively being explored by researchers. Leveraging the recent advancements in point cloud transformers and vision foundation models, this project aims to aggregate the information learned from the two different modalities, 3D LiDAR point cloud and 2D camera images, to perform 3D semantic segmentation on outdoor scenes. The final results showed an improvement in the average intersection over union score from 0.5935 by the baseline to 0.6975 by the best performing fusion model developed, on the test split of the down-scaled nuScenes.

1. Introduction

3D semantic segmentation enables autonomous vehicles to understand and track the surroundings. The two primary sources of data for such tasks are the 2D images taken by cameras facing different directions away from the self-driving vehicle and 3D point clouds collected by LiDAR sensors installed at the top of the vehicle. Different from ordered pixels as in image data, point clouds are sparse and unstructured. Since images are often collected alongside LiDAR data, how to leverage multiple modalities to further improve the 3D segmentation performance is an active area of research. While some 2D-3D fusion methods have been developed, the recent state-of-the-art methods for 3D segmentation are predominantly focused on 3D data. Inspired by the recent development in point cloud transformer models and vision foundation models, this project will be exploring the extraction of 3D point cloud features and 2D camera image features followed by effectively integrating 2D-3D features for outdoor scene 3D semantic segmentation. The fusion method developed to integrate the 2D and 3D features ultimately achieved an improvement on the primary mIoU metrics and convergence speed.

1.1. Problem Statement

This project aims to predict the semantic label of every point in the point clouds from the down-scaled nuScenes dataset, by extracting 3D LiDAR point features with Point Transformer V3 (PTv3), extracting 2D camera image features with DINOv2, and aggregating them to leverage the multi-modalities for 3D semantic segmentation. The input to the algorithm is the point clouds and images, of which features will be extracted with PTv3 and DINOv2, respectively. A fusion model is developed to encode the extracted features of the two modalities to the same latent space followed by merging the information learned to output predicted semantic segment labels for each point in the point cloud. This project explored the extent to which 3D semantic segmentation of outdoor scenes can be improved by incorporating both camera and LiDAR sensor data, instead of using the point clouds as the only data to learn from.

2. Related Work

Feature representations of point clouds is key to model learning and can be grouped into three categories. Projection-based methods project 3D points to a 2D plane and perform 2D CNN, which is runtime efficient but results in 3D information loss. Voxel-based methods transform point clouds into voxel grids and apply 3D convolution or sparse convolutions, but face scalability issues as they are constrained by kernel size but need to achieve large receptive fields. Point-based methods operate directly on the points, such as PointNet [8], but the unstructured nature of point clouds impact their computational efficiency.

Sparse Point-Voxel Convolution (SPVC) [10] uses both point-based and voxel-based methods but on two different streams of operations to, respectively, preserve geometrical information and enable large receptive fields, followed by fusing the information of both streams.

Applying self-attention networks to 3D point cloud processing is natural as self-attention is invariant to permutation and input cardinality, while 3D point clouds are essentially sets of sparse unordered points in 3D space. Point Transformer (PT) [14] is composed of 3 types of blocks: Point Transformer Block that performs self-attention among points, Transition Down Block as an encoder, and Transition Up Block as a decoder, symmetric to the Transition Down Block as in a U-net structure. The PT decoder in the final layer produces a feature vector for each point and an MLP follows to map it to the final logits. Superpoint Transformer [9] partitions point clouds into a hierarchical superpoint structure and applies the self-attention mechanism to capture relationships among superpoints, achieving a model with comparatively fewer parameters but similar performance with other larger models. Both transformer approaches aim to partition and structure the unordered point clouds in ways to be better understood by the encoder-decoder style architecture.

Along with the use of Transformer for point cloud segmentation, Vision foundation models use Vision-Transformer (ViT) [4] to train on large-scale image datasets and produce high-quality visual features. One foundation model is Distillation with No Labels (DINO) [3], composed of self-supervised Vision Transformers with knowledge distillation, trained on ImageNet without labels. Given an image, DINO creates global and local views via data augmentation to encourage the model to recognize the same content across different contexts and scales. The student-teacher framework is core to DINO, where the student network processes multiple augmented views of an image and the teacher processes global views only. While both networks have the same structure, student parameters are updated by backpropagation, whereas teacher parameters are updated with an exponential moving average of student parameters. The teacher is observed to have better performance than the student throughout training, acting as a guidance to the student by providing higher quality target features for the student. Both networks produce probability distributions over the dimension of "prototype scores" and cross-entropy loss is computed to measure difference in output distributions.

The combination of 3D point cloud and 2D images features to achieve 3D semantic segmentation are being explored more recently compared to single-modal data based segmentation. MSeg3D [5], which implements geometry-based and semantic-based fusion techniques, is one of the early models using both LiDAR and camera data to achieve comparable results with the state-of-the-art LiDAR only methods. BEVFusion [6] transforms camera and LiDAR features to the shared bird's-eye view (BEV) representation space and fuses the concatenated BEV features with fully-convolutional BEV encoder. DINO in the room (DITR) [13], an approach presented in a very recent paper with code not yet published, takes advantage of 2D foundation model to extract image features, projects them to 3D, and finally injects them into a 3D point cloud segmentation model. DITR has two variants: an injection approach that uses 2D features from DINOv2 and a distillation approach that aligns 3D features with DINOv2 features during a pretraining phase, which can be used for cases when images are unavailable during inference. The 2D image features extracted by DINOv2 are reverse projected to 3D point cloud and then max-pooled to different scales, such that they can be added to the skip connection between the pair of encoder-decoder blocks in a U-net style 3D point backbone.

Point Transformer V3 (PTv3) [12] and DINOv2 [7] are the two backbone models used in this project. PTv3, different from its previous versions, utilizes serializations to structure points into formats that enable the attention mechanisms to better capture spatial and contextual information. Scalability and efficiency improvement is core to PTv3. To more efficiently define the spatial proximity of points, space-filling curves are used to serialize point clouds. PTv3 also uses a patch attention mechanism that groups the points into non-overlapping patches to perform attention within each individual patch. As relative positional encoding is observed to be inefficient in point cloud transformers, PTv3 instead prepends a sparse convolution layer with skip connection before the attention layer as a conditional positional encoding to decrease latency. DINOv2 explores how to learn visual features that can be used to perform better on a variety of downstream tasks without fine-tuning compared to those task-specific models. Different from DINO, DINOv2 is trained on 142 million images, softmax normalization in the teacher is replaced with Sinkhorn-Knopp batch normalization, and a patch-level task iBOT is added to mask out parts of input image to student so that the student's guess of what is hidden can be compared with what the teacher sees from the complete image. In general, DINOv2 learned its features with a discriminative self-supervised method that combines DINO and iBOT losses with Sinkhorn-Knopp normalization, achieving more stabilized, accelerated learning when scaling up model and data sizes.

3. Methods

The proposed method runs the pretrained PTv3 backbone to extract point cloud features and runs the pretrained DINOv2 backbone to extract image features. To aggregate both features, fusion models are developed to combine the information learned from points and images, followed by transforming the joint features to semantic classes.

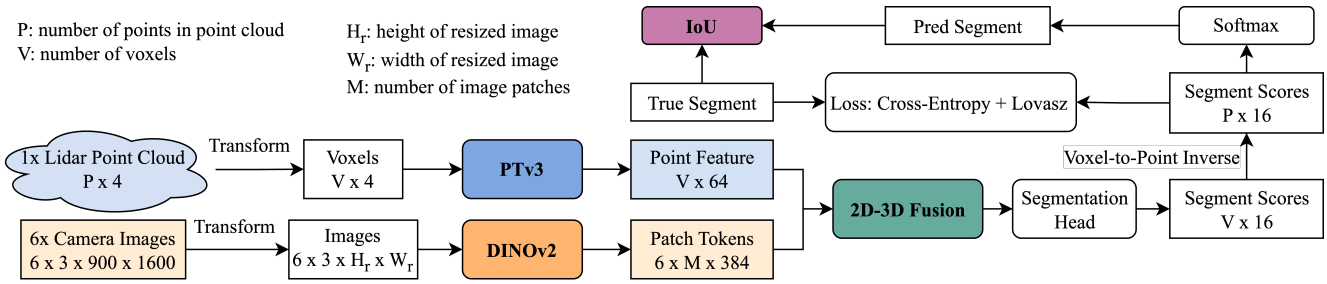


Figure 1: Architecture of the proposed method: point cloud feature extraction with PTv3, image feature extraction with DINOv2, and fusion of extracted features, followed by obtaining the normalized segment class scores and the actual predicted lidar segment index.

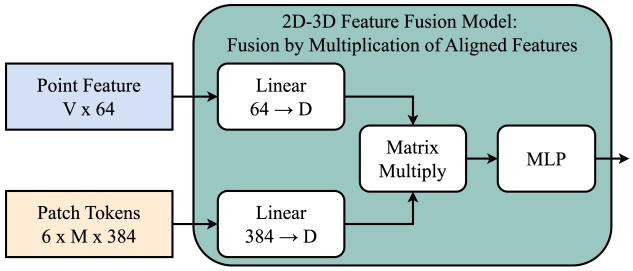


Figure 2: Direct Fusion model: fusion by multiplication of dimension-aligned features.

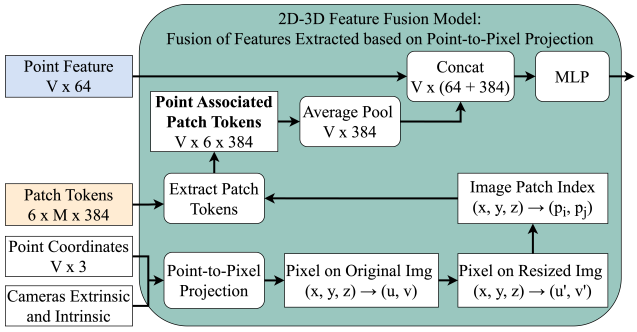


Figure 3: Projection Fusion model: fusion of features extracted based on point-to-pixel projection.

3.1. 3D LiDAR Point Cloud Feature Extraction

PTv3 is used as the primary backbone to extract point cloud features. The published pretrained model with weights has 46.16 million parameters, takes point cloud data as input, and outputs point features with embedding dimension 64. PTv3 is composed of three stages: (1) serialization, (2) a series of encoders, each with a grid pooling layer, serialized orders shuffling, and a Point Transformer block, ending with (3) a series of decoders, same structure as encoder but with different number of Point Transformer Blocks and mirroring to the encoders as in a U-Net style. The grid pooling layer in encoder first splits a set of points into non-overlapping partitions, then fuses the points from the same partition by max pooling the point features and average pooling the point coordinates.

When PTv3 was trained previously, model evaluation was computed after applying grid sampling (voxelization) on the points cloud to provide an initial performance assessment. While at test time, the point cloud is subsampled into a sequence of voxelized points clouds, which are then individually predicted and collected to form a complete prediction of the entire point cloud. In addition, test time data augmentation is performed by default in the original PTv3 to further enhance prediction stability. For this project, model evaluations for both training and inference are computed in the same way: the metrics, including intersection over union, are calculated based on the segment prediction on the entire set of the original points instead of the voxels. This gives a more precise and consistent evaluation results. Also, the validation and test datasets are preprocessed the same way as the training dataset, without any test time augmentation, to decrease inference durations.

3.2. 2D Camera Image Feature Extraction

As a separate stream of feature extraction, the pretrained DINOv2 ViT-S/14 backbone model with 22.06 million parameters, distilled from the large ViT-g/14, is used to produce visual features from camera images. This version of DINOv2 has 12 Transformer blocks and each layer outputs patch tokens of dimension 384. In the original approach for image semantic segmentation task, the patch tokens of the last 4 layers are concatenated to produce the class logits. However, for this project, only one of the last 4 layer's patch tokens will be used for feature fusion; experiments on using the 9th or the 12th layer's patch tokens will be conducted. As shown in Figure 4 and 5, the extracted image patch tokens encode information about the scene geometry.

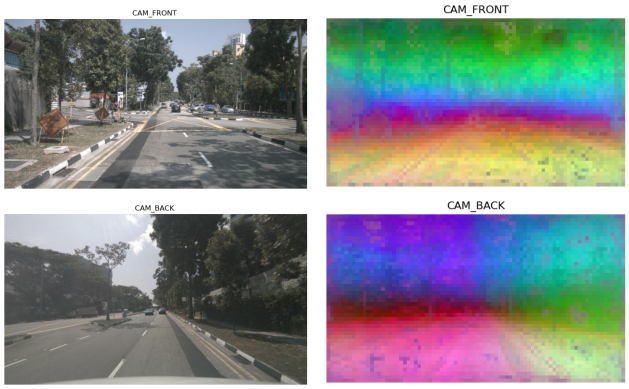


Figure 4: PCA visualization: first 3 principal components of patch tokens extracted by DINOv2 9th transformer layer

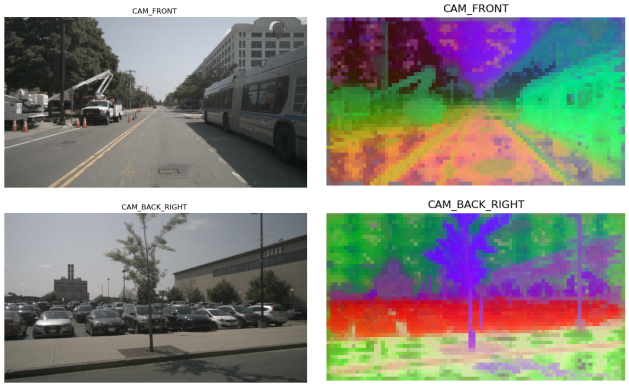


Figure 5: PCA visualization: first 3 principal components of patch tokens extracted by DINOv2 12th transformer layer

The 9th transformer layer encodes more general structure of the regions in scene, whereas the 12th layer encodes more specific object shapes and boundaries.

3.3. 2D-3D Feature Preparation

The point and image features are extracted and saved to .pt files on disk to enable more efficient development and training of the fusion model starting from the already extracted features rather than from the raw data. The feature dataset used as input to the fusion model does not only have the extracted voxel features and image patch tokens features, but also contains voxel coordinates, ground truth voxel segments, and voxel-to-point inverse vectors for point cloud related data, LiDAR point to camera coordinates transformation and LiDAR point to image pixels transformation matrices for image related data.

As different samples can have different numbers of points in the point cloud, the extracted point features are padded with zeros to enable batch fusion processing, thus the point feature vectors loaded in batch have shape $(\text{batch\_size}, \text{max\_num\_voxels}, 64)$. Every sample has 6 images of the same resolution and transformed with the same pipeline, so there is no additional processing when loading the batched extracted image features.

3.4. 2D-3D Feature Fusion Model

Two variants of the fusion model are developed: 1) Direct Fusion: fusion by multiplication of dimension-aligned features and 2) Projection Fusion: fusion of features extracted based on point-to-pixel projection.

In the Direct Fusion approach, shown from Figure 2, the extracted point feature and image feature are linearly projected to the same embedding dimension from their respective model output dim. Batch matrix multiplication is applied to the linearly transformed features to merge learned information from both point cloud and camera images. A two-layer MLP with ReLU activation is applied to output the final point embeddings.

In the Projection Fusion approach, shown from Figure 3, the 3D voxels centers are projected to the 2D camera image of the original 900 x 1600 resolution using the LiDAR to image rotation-translation matrix, which is constructed from world-to-camera extrinsic and camera intrinsic parameters. Each projected pixel is scaled such that its coordinates are in the resized image resolution; the resized images were taken as input to DINOv2 during the feature extraction process. The pixel coordinates are then converted to image patch index: patch at row i, col j of the image containing that pixel. The patch token at that specific patch index, which is associated with the voxel being projected, is extracted. This point-to-pixel projection and point-associated patch token extraction process repeats for each of the 6 camera images. Average pooling is applied to the associated patch tokens to obtain one patch embedding in dimension 384 for each projected voxel. 2-layer MLP with ReLU activation is applied to fuse the learned point and image representations.

More specifically on the point to pixel projection, given a point coordinate (x, y, z), as well as the precomputed LiDAR-to-camera and LiDAR-to-image rotation-translation matrices, the projection of this point to a 2D camera image can be performed with the following steps:

1. prepare point in homogeneous coordinate: $(x, y, z, 1)$
2. project point to pixel in image using the 4×4 rotation translation matrix $\text{RT}_{\text{lidar2img}}$

$$\begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} = \text{RT}_{\text{lidar2img}} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

3. normalize u, v, w by dividing by w to obtain the final projected pixel on image as $(u/w, v/w)$.

Additionally, depths is computed by projecting point to camera coordinates, similar step as above but with $\text{RT}_{\text{lidar2cam}}$ transformation matrix. Depths is used to check if

the projected pixel is a valid pixel in the image: not behind the camera. Depths along with the projected pixel coordinates in image are used to create a projection mask to mask out those points that have projected pixels outside of the image boundaries or behind the camera, for each of the 6 camera images per sample. For the invalid points for an image, there are no associated patch token features thus the visual features for those points from this specific image would be all zeros.

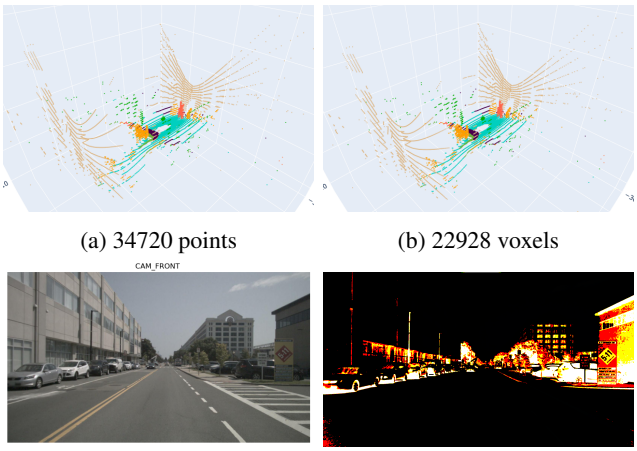
3.5. Prediction of Segment Class

As shown in Figure 1, after the 2D-3D fusion model outputs, a linear segmentation head is applied to transform learned joint embeddings to segment scores for each of the voxel, for all 16 classes. The voxel-to-point inverse is applied to map back to the original point space and thus obtaining the segment scores for each point. When training the fusion model, the sum of Cross-Entropy loss and Lovasz-Softmax loss [1] is computed from the ground truth point segment index and predicted point segment scores, as the objective to minimize with gradient descent. Lovasz-Softmax loss incorporates the softmax operation, computes a vector of pixel errors from the class probabilities, and constructs a loss surrogate to Jaccard index for each class, from which the average of the class-specific surrogates is taken, to offer better optimization of mIoU for the segmentation task.

4. Dataset and Features

nuScenes [2] is a public large-scale dataset collected by the sensor suite installed on self-driving cars. The original entire nuScenes dataset contains 1.4 billion annotated points across 40,000 point clouds and 1000 scenes. Each scene has 20 seconds of data sampled at 2Hz, thus 40 keyframes or samples per scene. For this project, a downscaled, mini version of the nuScenes and nuScenes-lidarseg [11] datasets are used: 323 training samples with a total of 10831744 lidar points to label, 40 validation samples with 1388928 points, and 41 test samples with 1423552 points. Each sample has one point cloud captured with a 32-beam LiDAR and 6 images each collected by a camera of different orientation, identified as front, front-left, front-right, back, back-left, and back-right. The semantic segmentation labels are annotated on the point clouds but not on the images.

Each LiDAR point is defined by 3D spatial coordinates and LiDAR intensity: (x, y, z, s). Different samples can have different numbers of points, thus the points for a sample i is represented in a $N_i \times 4$ matrix, where N_i is the number of points in sample i. Each camera image is represented as a $3 \times 900 \times 1600$ (channels $\times$ height $\times$ width) matrix. Every LiDAR point is annotated with one of 32 possible semantic labels, of which 16 classes are used for lidar segmentation task due to having limited samples for some



(a) 34720 points (b) 22928 voxels

(c) 900×1600 original image (d) 640×1138, normalized

Figure 6: Visualization of original point cloud and a camera image compared to the preprocessed point cloud and image to be taken by the feature extraction algorithm as input. (a) and (b) may look the same as voxel grid size is small, 5cm.

of the labels; similar classes were merged and rare classes removed.

4.1. Data Preprocessing and Augmentation

Point cloud preprocessing includes grid sampling (voxelization) with a grid size of 5cm to divide the point cloud into fixed-size voxels and each voxel has one representative point. In addition to the points coordinates and intensity, other point cloud information is preprocessed and loaded along during training and inference, including the inverse vector that maps which points are represented by the voxel, to be used after voxel segment prediction to revert from the voxels back to the points; the model predicts the segment for each voxel and all the original points represented by the same voxel would have the same segment.

Image preprocessing involves resizing to 640×1138 and normalization by subtracting mean (123.675, 116.28, 103.53) and divide by standard deviation (58.395, 57.12, 57.375), which are the defaults used for DINOv2 backbone pretraining. The image patch size is 14, resulting in $46 \times 82 = 3772$ patches per image. The transformation matrices that convert LiDAR points coordinates to camera coordinates and to image coordinates are also precomputed.

5. Results

Three evaluation metrics are used to assess and compare the model performance: mean intersection over union (mIoU), mean accuracy (mAcc), and frequency-weighted IoU (fwIoU), which are all computed from the True Positive (TP), False Positives (FP), and False Negatives (FN) of the predicted segments compared to ground truth segments.

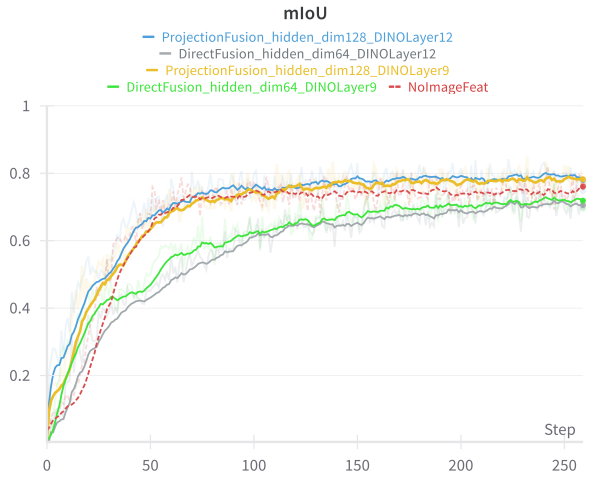


Figure 7: Training mIoU for each of the approaches with final evaluation results in Table 1.

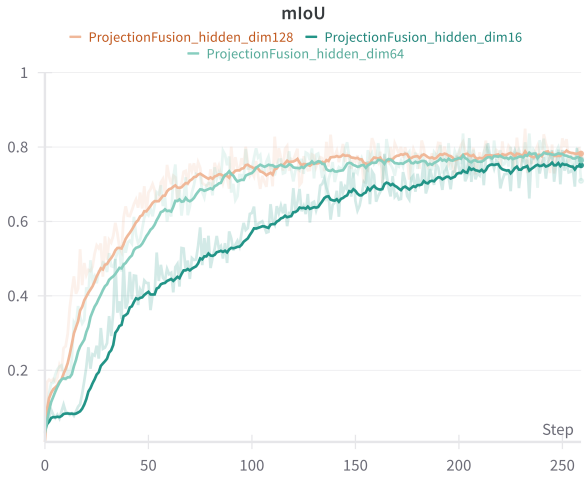


Figure 8: Training mIoU during hyperparameter tuning of the MLP hidden dimension for the projection-based fusion model with DINOv2's 9th layer patch tokens.

mIoU is the primary metrics for method comparisons. Lidar segmentation index 0 is ignored as 16 lidar segment labels of the general 32 nuScenes annotated labels are predicted.

1. IoU = $\frac{\text{TP}}{\text{TP+FP+FN}}$, mIoU = mean IoU over all classes.
2. Accuracy = $\frac{\text{TP}}{\text{TP+FN}}$, the total number of correctly predicted points divided by the total number of points; intersection over target. mAcc is the mean accuracy over all classes.
3. fwIoU is the sum of the IoUs of all classes, each weighted by the point-level frequency of its class.

The hyperparameters, such as learning rate, batch size, and MLP hidden dimension, are chosen by running iterations of experiments with different values. Example of tuning the MLP hidden dimension is shown in Figure 8, based on which 128 is chosen as the hidden dim. After the point associated patch tokens are identified, averaged and then concatenated with the point features, this joint feature vector has an embedding dim of $64 + 384 = 448$. If the MLP hidden dimension is too small, e.g. 16, there can be a greater loss of information when attempting to transform to the class logits and the convergence of learning is slower.

- optimizer: AdamW
- batch size: 12
- learning rate: 0.002
- weight decay: 0.005
- epochs: 10

5.1. Quantitative Results

As shown from evaluation results in Table 1, the best performing model variant is Projection Fusion: PTv3 + DINOv2 9th layer, with MLP hidden dimension of 128. This projection fusion model improved the mIoU on validation set by 5.7% compared to the baseline model, and reached 0.6975 mIoU on the test set compared to the baseline 0.5935. The baseline model is the PTv3 backbone and default linear segmentation head that have weights trained on all nuScenes training set and released publicly. Since a down-scaled version of the nuScenes dataset is used in this project, the data distributions and representativeness are different from the full dataset. Therefore, the baseline model results presented in the table are obtained by directly making inference on the down-scaled dataset.

Based on Figure 7, which shows the training mIoU for each of the model variant experimented, both variants of the Projection Fusion approach, one using patch tokens of DINOv2 layer 9 and another using that of layer 12, converge faster on the mIoU value during training and reached the highest final training mIoU, 0.77 and 0.75, respectively. In contrast, the Direct Fusion variants reached around 0.69 mIoU at the end of the 10 epochs. The metrics results of Projection Fusion being higher than baseline and Direct Fusion being lower than baseline is, to an extent, within expectations, as simply multiplying the point features and image patch tokens does not really encourage the fusion model to capture the key features of both modalities. Whereas by explicitly identifying the patch in image associated with each of the voxels enables the fusion model to learn how to let the image features influence the segment class predictions.

Furthermore, the evaluation results for Projection Fusion using DINOv2 layer 9's patch tokens and that using layer 12's patch tokens are quite similar. This can suggest that, although layer 12 extracts more detailed objects geometry compared to layer 9, the features that the fusion model pri-

Method	Val			Test		
	mIoU	mAcc	fwIoU	mIoU	mAcc	fwIoU
Baseline: Released PTv3 nuScenes Segmentor	0.6392	0.6910	**0.9641**	0.5935	0.6416	0.8566
Simple: PTv3 + Linear Seg Head	0.6483	0.6989	0.9600	0.6936	**0.7448**	**0.8571**
Direct Fusion: PTv3 + DINOv2 9th layer	0.5993	0.6391	0.9528	0.6549	0.6943	0.8317
Direct Fusion: PTv3 + DINOv2 12th layer	0.6180	0.6663	0.9545	0.6619	0.7130	0.8345
Projection Fusion: PTv3 + DINOv2 9th layer	**0.6757**	0.7238	0.9633	**0.6975**	0.7421	0.8568
Projection Fusion: PTv3 + DINOv2 12th layer	0.6730	**0.7266**	0.9638	0.6854	0.7338	0.8557

Table 1: Results on down-scaled validation and test dataset with the experimented approaches.

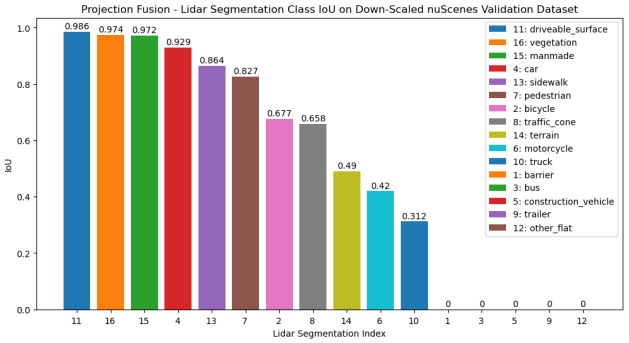


Figure 9: Lidar semantic segmentation class IoU of the best performing Projection Fusion on validation dataset. Due to using down-scaled dataset, the validation samples do not have any points labeled as barrier, bus, construction_vehicle, and trailer, so the 0 IoU for lidar segmentation index 1, 3, 5, 9 actually means both the ground truth and predicted segments do not have those classes: union is 0.

oritizes to learn from the patch tokens are actually quite similar. Nonetheless, further experiments would need to be conducted to see if a single layer's patch tokens would be sufficient to influence the point segmentation or multiple layers' output would need to be concatenated and used, as in the original DINOv2 image semantic segmentor.

5.2. Qualitative Results

The predicted semantic segmentation are, in general, quite successful, as visualized in Figure 10. However, there are still some challenging parts in the point cloud that are not classified correctly. For the sample in the second row, the ground truth render has the points on the L-shaped sidewalk labeled with purple, but the prediction has those about one-third of those points classified as drivable surface, in turquoise color. This may be misled by the lack of apparent street curb.

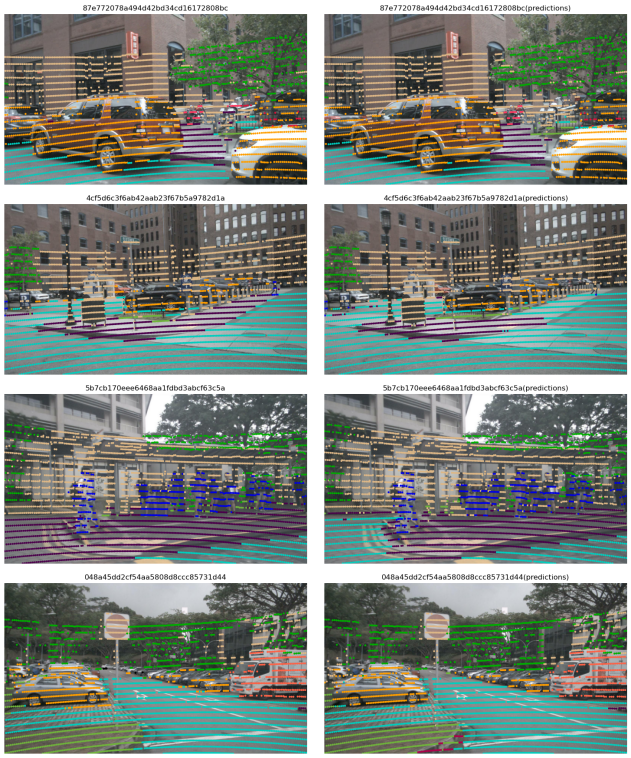
Compared to the Simple model, the best performing Projection Fusion achieved 4.2% increase in mIoU but the improvement of mIoU for test set is very small. This could be due to the challenging samples in the small test set. For instance, in the test sample shown in the third row of Figure 10, the ground truth segmentation has points in the bottom-left corner of image labeled as sidewalk in purple, but the predicted segmentation has that portion classified as drivable surface in turquoise. The street curb, in this case, could again be the misleading factor of this incorrect segmentation. To further add to the street curb issue, the prediction in the last row has the labels the curb in the foreground as other_flat, in pink color, where in reality, the curb is not flat.

Furthermore, compare to classes, such as drivable surface, vegetation, and car, that have plenty of points in the point cloud, classes with fewer frequency of appearing in the point cloud, including pedestrian, are more likely to be labeled incorrectly. This can be seen in the second row's example, with the small pedestrian figure near the right edge of the image. However, when there are crowds of pedestrians together in the foreground, the segmentation performs well, shown in the third row.

Finally, Figure 11 visualizes the first three principal components of the final MLP layer's feature embedding, in the best performing Projection Fusion model. Regions of similar RGB color indicates similar semantics, for instance, the red points on the ground have distinctly different color values from those points forming some vertical, standing structure in green. These final feature embeddings are the input to the linear segmentation head, which then outputs class logits.

6. Conclusion

The results of this project demonstrate how data from different modalities can be leveraged together for 3D semantic segmentation and the advantage of such fusion approach. The Projection Fusion, which aggregate the DINOv2 extracted patch tokens with PTv3 extracted point feature based on point-to-pixel projection, achieved the highest intersection over union averaged across all lidar segmentation classes. This projection-based approach is better than



(a) Ground truth. (b) Predictions by Projection Fusion: PTv3 + DINOv2 9th layer.

Figure 10: Visualized segmentation results by projecting 3D points to image. Each row has the ground truth on the left and prediction on the right. Top two rows are two samples from validation set and the bottom two rows are two samples from test set.

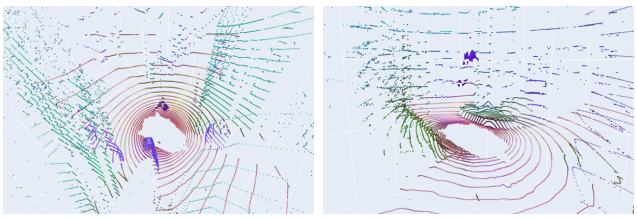


Figure 11: PCA visualization of two samples' final layer point feature embeddings, input to the linear segmentation head, from the best performing projection fusion model.

the direct, multiplication-based approach because it more explicitly associates the potentially relevant point cloud and image features, from which the fusion MLP can better capture the relationship between the points and pixels.

Future work can be carried out from different perspectives. In terms of scalability, the full nuScenes dataset can be used to train and validate the fusion approach. Instead of using a single DINOv2's Transformer layer output patch tokens as image features, multiple layers' outputs can be used. Data augmentation, such as random jitter, can be applied, but maintaining the geometry consistency across point clouds and images as they get augmented would be an area of challenge. With regards to improving how model would learn the relationship between points and pixels, we can apply less explicit association, instead try encouraging the model to figure out how each point would correlate to the different parts of the image itself, for example, by using attention mechanism. Furthermore, we can experiment applying fusion at different locations or implementing synergistic pre-training instead of fusing the features extracted by separately pre-trained models.

In conclusion, this projects offers a solid starting framework for integrating models that extract point cloud features and models that extract visual image features. In the future, if there are better point-based or image-based models being developed, especially those in the context of autonomous driving, outdoor scenes, or self-supervised models trained on very large dataset, the PTv3 and DINOv2 implemented in the current framework can be replaced. This, along with enhancement or different variation of the fusion model, can contribute to further improvement in 3D semantic segmentation.

7. Contributions and Acknowledgments

This project is independent from other classes and no collaborators were involved. The following public code was used or referenced:

1. nuScenes devkit: github.com/nutonomy/nuscenes-devkit
2. Point Transformer v3: github.com/Pointcept/Pointcept/releases/tag/v1.5.1
3. PTv3 backbone weights: huggingface.co/Pointcept/PointTransformerV3
4. DINOv2 and pretrained backbone model weights: github.com/facebookresearch/dinov2

References

[1] M. Berman, A. R. Triki, and M. B. Blaschko. The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks, 2018.

[2] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020.

[3] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. Emerging properties in self-supervised vision transformers, 2021.

[4] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image

is worth 16x16 words: Transformers for image recognition at scale, 2021.

[5] J. Li, H. Dai, H. Han, and Y. Ding. Mseg3d: Multi-modal 3d semantic segmentation for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21694–21704, June 2023.

[6] Z. Liu, H. Tang, A. Amini, X. Yang, H. Mao, D. Rus, and S. Han. Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation, 2024.

[7] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P.-Y. Huang, S.-W. Li, I. Misra, M. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski. Dinov2: Learning robust visual features without supervision, 2024.

[8] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation, 2017.

[9] D. Robert, H. Raguet, and L. Landrieu. Efficient 3d semantic segmentation with superpoint transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 17195–17204, October 2023.

[10] H. Tang, Z. Liu, S. Zhao, Y. Lin, J. Lin, H. Wang, and S. Han. Searching efficient 3d architectures with sparse point-voxel convolution, 2020.

[11] J. H. L. Z. H. C. O. B. A. V. W. Fong, R. Mohan. Panoptic nuscenes: A large-scale benchmark for lidar panoptic segmentation and tracking. In *ICRA*, 2022.

[12] X. Wu, L. Jiang, P.-S. Wang, Z. Liu, X. Liu, Y. Qiao, W. Ouyang, T. He, and H. Zhao. Point transformer v3: Simpler faster stronger. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4840–4851, June 2024.

[13] K. A. Zeid, K. Yilmaz, D. de Geus, A. Hermans, D. Adrian, T. Linder, and B. Leibe. Dino in the room: Leveraging 2d foundation models for 3d segmentation, 2025.

[14] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16259–16268, October 2021.