Semantic Segmentation with Waymo LiDAR Dataset

Thomas Slavonia, Anderson Compalas, Logan Bledsoe

Abstract

Semantic segmentation is becoming increasingly critical as autonomous vehicles and robotics gain a stronger presence in physical environments. For our project we want to determine what methods are best for LiDAR based semantic segmentation. Our project compares several neural network architectures, including diffusion-based approaches, to determine which performs best on the Waymo LiDAR dataset.

1 Introduction

Our project was inspired by a Waymo competition in 2024, where contestants were judged on their ability to achieve the highest classification rate on the Waymo Open Dataset. The dataset includes both LiDAR range images and camera images. For this project, we used only LiDAR images, since our scope was focused on models for LiDAR point clouds rather than multimodal camera–LiDAR fusion. LiDAR data was also significantly less storage-intensive, making the project feasible with our available resources: approximately 127 GB for LiDAR versus over 2 TB for camera data.

We evaluated three methodologies. First, we projected LiDAR data into a 2D representation and applied a U-Net architecture for semantic segmentation. Second, we tested 3D point cloud segmentation architectures, including PointNet[1], PointNet++[2], and EdgeConv[3]. Third, inspired by Qu et al.[4], we implemented a diffusion-based semantic segmentation approach in which the reverse denoising process was learned using both a 2D U-Net on range images and a 3D point cloud network (EdgeConv).

Our initial hypothesis was that the diffusion model combined with a point cloud architecture would perform best, motivated by the success of diffusion methods in 2D image denoising and generation.

For diffusion, we used a Denoising Diffusion Probabilistic Model (DDPM) architecture. This architecture consists of a forward process in which noise is gradually added to an image or point cloud, and a learned reverse process that reconstructs the target structure.

2 Dataset

We use the Waymo Open Dataset LiDAR subset, represented as range images and point clouds. We preprocess the raw inputs into formats suitable for both 2D and 3D pipelines, and use consistent train/validation/test splits for fair model comparison. There are 23 unique classes in the dataset, with one class consisting of unclassified points.

3 Methodology

3.1 2D Baseline

The 2D baseline converts LiDAR observations into range-image-like projections and applies a U-Net segmentation backbone.

3.2 3D Point Cloud Models

3.2.1 Eigen Feature Model

The Eigen Feature Model is our baseline 3D model. It is based on explicit neighborhood feautres rather than learned ones. The model explores how far semantic segmentation can be driven by hand-designed geometric features alone, before moving to more expressive learned point-cloud architectures such as PointNet, PointNet++, and EdgeConv.

For each point, the model constructs k-nearest-neighbor neighborhoods at multiple scales, computes a 3×3 covariance matrix from the neighbor coordinates, and extracts features from its eigenvalues and eigenvectors. Let $\lambda_1 \geq \lambda_2 \geq \lambda_3$ denote the sorted covariance eigenvalues of a local neighborhood. At each scale, the model computes an 11-dimensional engineered descriptor. This contains four eigen-value based shape descriptors:

$$\text{linearity} = \frac{\lambda_1 - \lambda_2}{\lambda_1}, \qquad \text{planarity} = \frac{\lambda_2 - \lambda_3}{\lambda_1},$$

$$\text{scattering} = \frac{\lambda_3}{\lambda_1}, \qquad \text{surface variation} = \frac{\lambda_3}{\lambda_1 + \lambda_2 + \lambda_3},$$

along with three normal components (estimated by the eigenvector associated with λ_3), the absolute vertical normal component, the point's height and distance from the

neighborhood mean, and the mean and maximum neighbor distances. These descriptors are then concatenated across each neighboorhood scales and added to the raw coordinates (x, y, z) and LiDAR intensity and elongation to form the full feature vector for each point.

Unlike the learned point-cloud models considered later, this model does not learn the neighborhood graph or the local feature formulas: the KNN neighborhoods, covariance construction, eigendecomposition, and descriptor definitions are fixed.

3.2.2 PointNet

Before PointNet, 3D point clouds were voxelized, which means they are converted to the equivalent of a 3D grid before the point cloud is classified. PointNet is a significantly more efficient model than voxelization that operates directly on the point clouds with each input being the (x, y, z) coordinates of the points. The points in point clouds are unordered, unlike a gridded image where the u, v pixels can be ordered. Also the metric for the distance between betweens is the Euclidean metric on $\mathbb{R}^n$:

$$\sqrt{\sum_{i=1}^{N}(x_i - y_i)^2}$$

for $\boldsymbol{x} = [x_1, \ldots, x_n]^T \in \mathbb{R}^n$ and $y \in \mathbb{R}^n$. The distance between points is an absolutely critical part of semantic segmentation using LiDAR point clouds. To attain information from all of the poins in the point cloud a max pooling layer is applied to a MLP network. Max pooling is permutation invariant meaning that for a function with n points $x_i, i \in [n]$, and any permutation σ, the function

$$f(x_1, \ldots, x_n) = f(x_{\sigma(1)}, \ldots, f(x_{\sigma(n)})).$$

The use of the permutation invariant max pooling, is a response to the fact that the point cloud consists of unordered points. The result of the max pooling on the transformed points $h(x_i)$ is

$$f(x_1, \ldots, x_n) = \gamma\left(\max_{i=1,\ldots,n}\{h(x_i)\}_i\right)$$

with γ and h being MLPs. Ordering the points falls to the curse of dimensionality, making this alternative the best option. From [1] the funtion is approximated by a symmetric function of transformed points

$$f(\{x_1, \ldots, x_n\}) = g(h(x_1), \ldots, h(x_n))$$

with g being the symmetric function.

3.2.3 PointNet++

PointNet++ takes a more localized view of the dataset than PointNet. We still want to learn functions f to perform our semantic segmentation. For PointNet++, we first sample points, then these sampled points are grouped into sets of points. Farthest point sampling is used for the point sampling, whereby given two subsets of points [2] $\{x_{i1}, \ldots, x_{im}\}$ and $\{x_{i1}, \ldots, x_{ij-1}\}$ x_{im} is the furthest point from x_{ij-1}. Selecting like this has the effect of sampling points that farthest possible from one another, so ensuring that all of the localized objects in the point cloud are sampled. Lastly, PointNet is applied to these grouped sets of points rather than the whole point cloud.

3.2.4 EdgeConv

The central goal of EdgeConv is to maintain the permutation invariance inherent to PointNet, while introducing local neighborhood information. So far, the models have ignored the relationship between points that are close to one another using the Euclidean metric, but no more. Instead of working with points, [3] instead creates local neighborhood graphs where the edge features are defined

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

with h_θ is a nonlinear function. The output is the x_{ij} vertex. The edge function used in our project was the assymetric edge function

$$h_\theta(x_i, x_j) = h_\theta(x_i, x_j - x_i)$$

Here x_i represents the global structure of x_i in the point cloud, while $x_i - x_j$ represents the local structure of $x_i's$ relationship to x_j. Then SiLU is applied and a max pooling of the edge features is taken as a permutation invariant aggregator. Therefore, the function used for the edge features is

$$e_{ij} = SiLU(\theta_m \cdot (\boldsymbol{x}_j - \boldsymbol{x}_i) + \phi_m \cdot \boldsymbol{x}_i)$$

with θ_m and ϕ_m as learned parameters. With the EdgeConv model, we attain permutation invariance, translation invariance, local information, and global information. EdgeConv is an excellent combination of all of the best qualities of PointNet, with the foresight of where the deficiencies of its precursors lay.

3.3 Diffusion-Based Segmentation

Our diffusion-based segmentation builds on the Denoising Diffusion Probabilistic Model (DDPM) framework [5] and trains models with both 2D and 3D context. We compare variants to isolate the contribution of architecture choice and data representation.

3.3.1 Forward Process

The forward process gradually corrupts a clean signal $\mathbf{x}_0$ into Gaussian noise over T timesteps via a Markov chain:

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}\Big(\mathbf{x}_t;\ \sqrt{1-\beta_t}\,\mathbf{x}_{t-1},\ \beta_t\mathbf{I}\Big) \quad (1)$$

where $\{\beta_t\}_{t=1}^T$ is a fixed variance schedule. Using the reparameterization trick and letting $\alpha_t = 1-\beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^t \alpha_i$, one can sample $\mathbf{x}_t$ directly from $\mathbf{x}_0$ in closed form:

$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}\big(\mathbf{x}_t;\ \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0,\ (1-\bar{\alpha}_t)\mathbf{I}\big) \quad (2)$$

so that $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. As $t \to T$, $\bar{\alpha}_T \to 0$ and the signal is fully destroyed.

3.3.2 Reverse Process

The reverse process learns to denoise step by step:

$$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}\Big(\mathbf{x}_{t-1};\ \boldsymbol{\mu}_\theta(\mathbf{x}_t, t),\ \tilde{\beta}_t\mathbf{I}\Big) \quad (3)$$

where $\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$. Following [5], we parameterize the model to predict the added noise $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ directly, which gives the simplified training objective:

$$L^{\text{simple}} = \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\Big[\big\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\big(\sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon},\ t\big)\big\|^2\Big] \quad (4)$$

We use $T = 1000$ timesteps throughout all experiments.

3.3.3 Segmentation via Diffusion

We adapt the DDPM framework to semantic segmentation by treating the label map as the signal to be denoised. Each training frame produces a soft one-hot label map $\mathbf{x}_0 \in \mathbb{R}^{C\times H\times W}$ with $C = 23$ semantic classes, $H = 64$ laser beams, and $W = 2650$ azimuth steps:

$$\mathbf{x}_0 = 2 \cdot \text{onehot}(\mathbf{y}) - 1 \quad (5)$$

where $\mathbf{y} \in \{0, \ldots, 22\}^{H\times W}$ is the ground truth label map and values are scaled to $[-1, 1]$. Pixels outside the valid LiDAR mask are zeroed. The diffusion model learns to reverse the noising process on this label map, conditioned on the LiDAR point cloud. At inference time, we start from $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and iteratively apply the learned reverse process to recover $\mathbf{x}_0$, from which the final segmentation is obtained via $\hat{\mathbf{y}} = \arg\max_c [\mathbf{x}_0]_c$.

In our dataset, dominant classes such as road and vegetation account for over 40% of valid pixels while rare classes such as cyclists account for under 0.1%. To handle class imbalance, we apply inverse-frequency class weighting to the training loss:

$$L = \sum_{c=1}^{C} w_c \left\|\boldsymbol{\epsilon}_c - [\boldsymbol{\epsilon}_\theta]_c\right\|^2 \quad (6)$$

where $w_c \propto N/(C \cdot n_c)$, n_c is the pixel count for class c, and weights are normalized so their mean equals one. The undefined class (class 0) receives weight zero.

3.3.4 Denoising Network

The denoising network $\boldsymbol{\epsilon}_\theta$ takes as input the noisy label map $\mathbf{x}_t$, the timestep t, and a LiDAR point cloud, and outputs a predicted noise map of the same shape. We compared two denoising architectures, described in the following sections.

4 Experiments

4.1 Evaluation Metric

To measure the quality of our semantic segmentation algorithms we use the metric mean intersection over union (mIoU). Using our predicted labels assigned to the points of the points of the point cloud and the actual labels assigned to the points of the point cloud, the the IoU is meant to measure the ratio of the overlap between the intersection over the points and the union of the points. So, if we had perfect overlap between a set of points in our predicted point cloud semantic map with the ground truth semantic map, the intersection would equal the ground truth semantic map. For two sets A and B IoU is calculated as

$$IoU = \frac{|A \cap B|}{|A \cup B|}.$$

Note that the union is always at least as large as the intersection, thus the ratio will be between 0 and 1. Hence, it represents the proprotion of overlap. Mean IoU or mIoU is just the mean of all of the IoUs calculated for each

4.2 Diffusion Experiments

4.2.1 2D U-Net DDPM (Range Image Baseline)

As a diffusion baseline, we implemented a DDPM model where both the diffusion target and the conditioning signal live entirely in range image space, with no 3D point cloud processing. The *LiDAR condition encoder* ,a strided Conv2D, compresses the raw 4-channel range image into context tokens that condition the 2D U-Net denoiser via cross-attention.

Self-attention is removed entirely, even at the deepest level ($N \approx 10{,}608$ tokens) it would cause OOM on the 6 GB GPU. Cross-attention over the 664 downsampled context tokens is $O(N \times 664)$ and remains tractable at the deepest encoder and decoder level only.

4.2.2 PointNet + DP3 1D U-Net

Building on the baseline, we next introduced 3D point cloud conditioning by replacing the LiDAR condition encoder with a PointNet encoder [1]. The pipeline has two distinct components: the **PointNet encoder** ingests $N = 16{,}384$ subsampled 3D points and produces a global conditioning vector; the **1D U-Net denoiser** ingests the noisy label map and predicts the noise, with the PointNet feature injected at each layer via FiLM [6]:

$$\text{FiLM}(\mathbf{h}) = \gamma(\mathbf{f}) \odot \mathbf{h} + \beta(\mathbf{f})$$

The noisy label map $\mathbf{x}_t \in \mathbb{R}^{23\times64\times2650}$ is reshaped into $(2650, 1472)$ where $1472 = 23 \times 64$ is the flattened elevation and class channel dimension per azimuth column. Code for both the PointNet feature extractor and the 1D U-Net denoiser were sourced verbatim from the DP3 repository [7].

4.2.3 PointNet + 2D U-Net

The third experiment retains the PointNet 3D encoder but replaces the 1D denoiser with a 2D convolutional U-Net to test whether the 1D architecture's failure stems from destroying the 2D spatial structure of the label map. As in the previous experiment, the **PointNet encoder** produces a global conditioning vector from the 3D point cloud; the **2D U-Net denoiser** operates Conv2D layers directly over the elevation and azimuth dimensions of $\mathbf{x}_t \in \mathbb{R}^{23\times64\times2650}$. The PointNet feature is projected to a single context token $\mathbf{c} \in \mathbb{R}^{1\times256}$ and injected via cross-attention rather than FiLM. Each SpatialTransformerBlock at the deepest level applies self-attention over the $2{,}648$ spatial tokens, followed by cross-attention with the PointNet context token and a GEGLU feed-forward layer.

Attention is restricted to the deepest level (level 3, 8×331, $\sim$0.45 GB). Level 0 would require $>$1.8 TB and level 1 causes OOM on an A100 80 GB GPU. The range image width $W = 2650$ is zero-padded to $W = 2656$ (nearest multiple of 8) and trimmed at output.

4.2.4 Discussion

Table 1 and Figure 1 together reveal two key findings. First, **2D spatial structure matters for denoising**: the DP3 1D U-Net (1.14%) and PointNet + 2D U-Net (1.19%) achieve similar mIoU, but their validation loss curves tell a clearer story. The DP3 val loss plateaus near 22–23 throughout training, barely below the random noise baseline, while the 2D U-Net val loss reaches $\sim$20.9 before overfitting sets in, confirming that Conv2D better captures the spatial structure of the range image label map. Second, **dense 2D conditioning outperforms global 3D conditioning**: the baseline's encoder provides $\sim$664 spatially-grounded context tokens per frame, while PointNet collapses $16{,}384$ points to a single $\mathbb{R}^{256}$ vector via global max-pool. This explains the baseline's dramatic advantage on road (22.73% vs. $\sim$4%) and building (9.23% vs. $\sim$4%), classes with consistent spatial positions in the range image that dense conditioning exploits directly.

The confusion matrices in Figure 5 reinforce this: the baseline shows structured collapse to dominant classes, a sign the model learned something about class frequency and spatial layout, while the PointNet experiments show near-random scatter, a sign they learned almost nothing. The baseline result also carries important confounds, 1,000 inference steps vs. 50, smaller model capacity (5.6M vs. 48–72M params), and not converged at epoch 50, so direct comparison should be interpreted cautiously. A stronger PointNet conditioning using multi-scale spatial features projected back onto the range image grid, rather than a single global token, could close this gap.

4.3 3D Model Experiments

4.3.1 PointNet

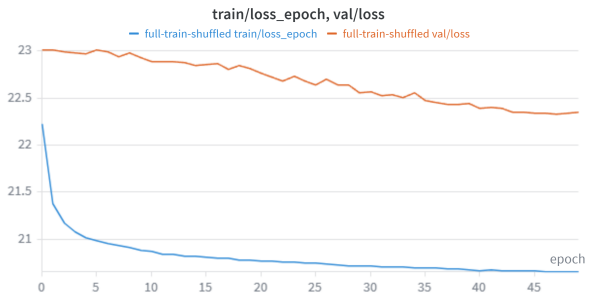
The PointNet algorithm uses $16,384$ 3D points. The points first go through a linear layer, and then six MLP layers. Afterwards, it is then max pooled across every point. The global feature afterwards is added to every point after a linear projection layer.

4.3.2 PointNet++

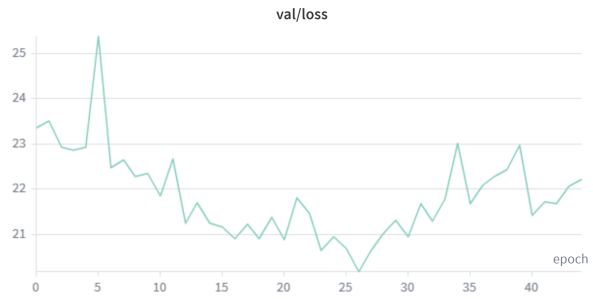
For our PointNet++ algorithm farthest point sampling is used for each set abstraction layer. The number of sampled points decreases for each layer, while the radius for where the sampled points come from increases. We use ball query for the sampling where given a radius wherre all data points are sampled within a radius r around an initially chosen point using the Euclidean metric. Four set abstraction layers are used for our experiments.

4.3.3 EdgeConv

Instead of the ReLU activatation function used in the paper for the edge features, our edge feature function is SiLU. EdgeConv performed the best of all of our models achieving a mIoU of 30%. A focal loss of $f = 2$ was used for the model with 16384 input points.



(a) PointNet + DP3 1D U-Net

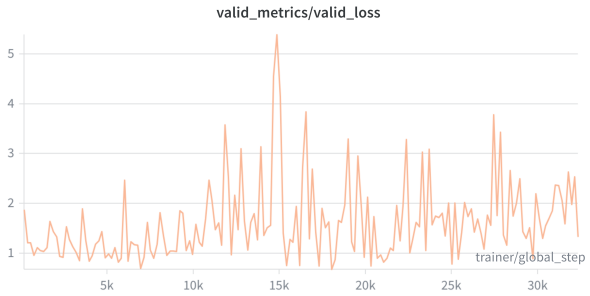


(b) PointNet + 2D U-Net

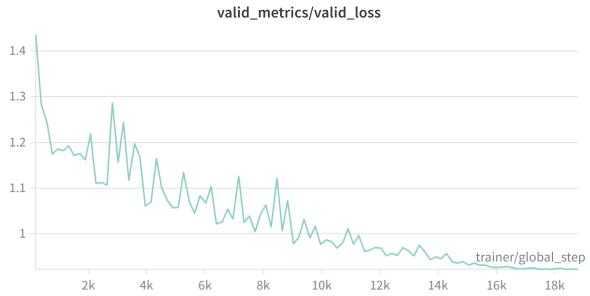
Figure 1: Validation loss curves for the PointNet diffusion experiments. (a) DP3 plateaus in the range 19–24, barely below the random noise baseline of ∼23, indicating the model failed to learn meaningful structure. (b) PointNet + 2D U-Net bottoms at ∼20.9 at epoch 26 before rising, indicating overfitting.

5 Conclusion

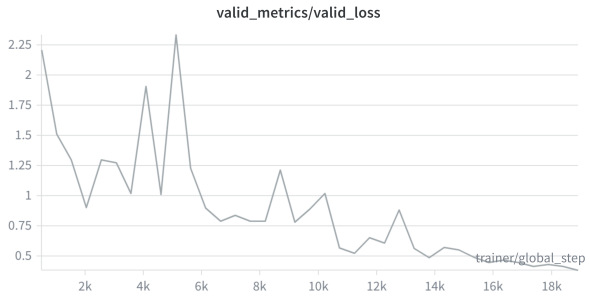
From our studies, we can conclude that, as expected, EdgeConv performed very well and if we had more time to tune the hyperparameters of the model, such as focal loss, learning rate, and class weights we could've achieved better results. For instance, the EdgeConv results displayed in the paper (mIoU = 30.12%) used a focal loss of $f = 2$ and a learning rate of $\alpha = 4^{-3}$. But, a subsequent run, that unfortunately couldn't finish in time, using a focal loss of $f = 1$ and a learning rate of $\alpha = 4^{-4}$ was performing even better. The Eigen Feature model also performed quite well, achieving a maximum mIoU of 25.86%. The base UNET architecture on the 2D range images also worked surprisingly well, achieving a mIoU of 19%. We wanted to do the project with diffusion, but our project was limited in three areas. One is that we didn't train for enough time. The Waymo LiDAR open dataset is over 120GB, which made it very difficult to train without large amounts of RAM available. To work around this we used Google Colab resources to run our models. But, we were at the mercy of Google Colab disconnecting The EdgeConv is still improving considerably at the moment that the Google Colab instance disconnected. Being able to run more epochs on models such as EdgeConv could've yielded significantly improved results. Unfortunately, certain models such as PointNet++ and the diffusion models achieved extremely low mIoU. The results indicate that the models were not implemented correctly. In the wise words of Thomas Edison "I have not failed. I've just found 10,000 ways that don't work."



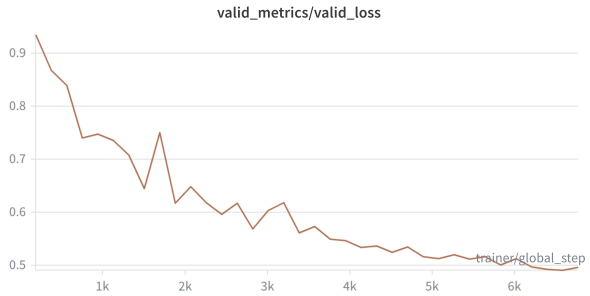
(a) 2D U-Net



(b) PointNet

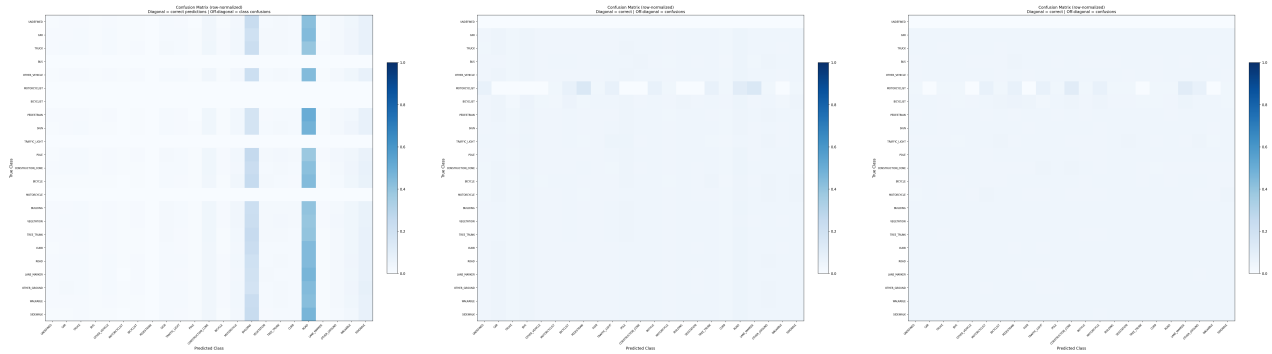


(c) EdgeConv



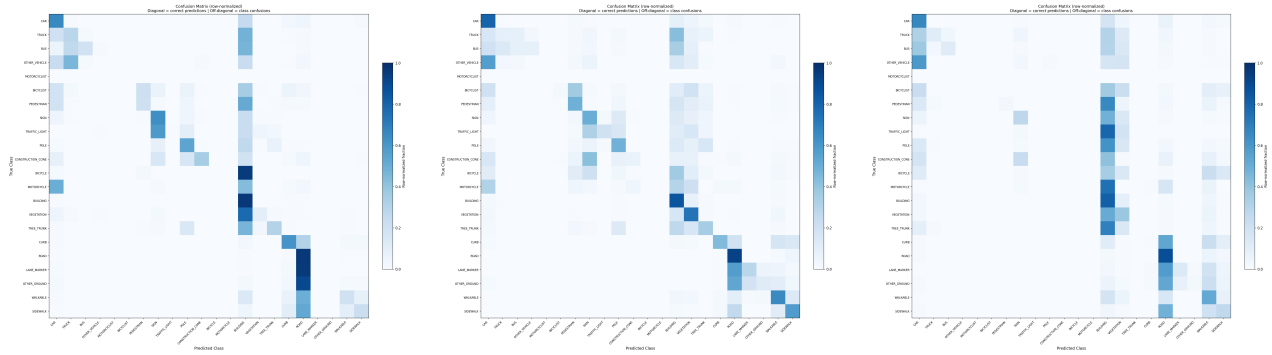
(d) Eigen Features MLP

Figure 2: Validation loss curves for the four non-diffusion models.



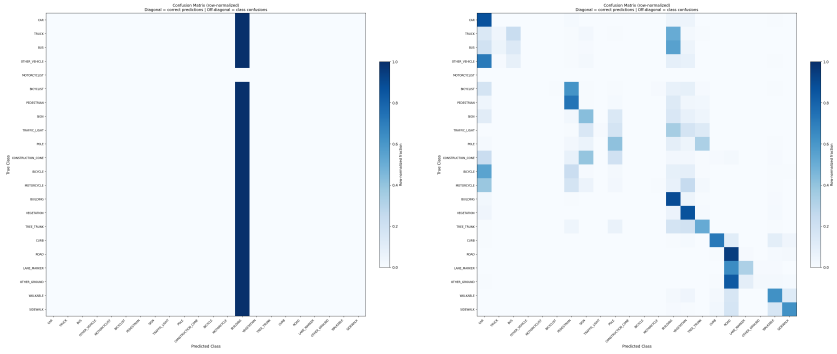
(a) 2D U-Net Diffusion (b) PointNet + DP3 1D U-Net (c) PointNet + 2D U-Net

Figure 3: Row-normalized confusion matrices for all three diffusion experiments (rows = true class, columns = predicted class). (a) The baseline shows a structured collapse: nearly all true class rows concentrate on the building and road columns, reflecting the model's exploitation of their dominant spatial positions in the range image. (b,c) The PointNet experiments exhibit near-uniform distributions across predicted classes, consistent with random noise rather than structured collapse, confirming the models failed to learn class-discriminative features.



(a) 2D U-Net (b) Eigen Feature (c) Pointnet

Figure 4: Row-normalized confusion matrices for 2D U-Net on range images, baseline eigen feature MLP on point-clouds, and pointnet (rows = true class, columns = predicted class).



(a) Pointnet++ (b) Edgeconv

Figure 5: Row-normalized confusion matrices for PointNet++ and EdgeConv (rows = true class, columns = predicted class).

Table 1: Per-class IoU (%) across experiments.

Class	**2D U-Net**	**Eigen Feature**	**PointNet**	**PointNet++**	**EdgeConv**	**2D U-Net Diffusion**	**PointNet DP3 1D U-Net Diffusion**	**PointNet 2D U-Net Diffusion**
Car	50.39	61.52	46.49	00.00	63.52	01.09	02.88	03.06
Truck	10.42	05.59	07.60	00.00	03.85	00.82	00.71	00.70
Bus	16.15	06.88	10.29	00.00	09.36	00.00	00.02	00.02
Other Vehicle	00.02	00.16	00.01	00.00	00.01	00.71	00.26	00.26
Motorcyclist	00.00	00.00	00.00	00.00	00.00	00.00	00.00	00.00
Bicyclist	00.01	01.90	00.00	00.00	00.39	00.00	00.00	00.00
Pedestrian	18.78	28.28	02.43	00.00	40.25	00.06	00.09	00.09
Sign	26.35	29.94	21.61	00.00	31.39	00.33	00.35	00.36
Traffic Light	00.49	10.29	00.00	00.00	00.92	00.00	00.01	00.01
Pole	37.24	29.60	00.01	00.00	31.47	00.38	01.21	01.26
Construction Cone	26.97	05.88	00.04	00.00	00.23	00.21	00.02	00.02
Bicycle	00.21	00.00	00.00	00.00	00.22	00.15	00.02	00.02
Motorcycle	00.00	00.00	00.00	00.00	00.15	00.00	00.02	00.03
Building	57.47	73.16	53.40	30.49	82.35	09.23	04.10	04.21
Vegetation	09.99	62.99	31.50	00.00	76.71	01.64	03.26	03.27
Tree Trunk	24.98	21.92	00.00	00.00	32.85	00.84	00.98	01.00
Curb	26.43	31.98	00.49	00.00	52.53	00.72	00.83	00.81
Road	70.57	81.88	70.40	00.00	83.32	22.73	03.87	04.63
Lane Marker	00.18	21.84	11.10	00.00	24.14	00.38	00.96	00.98
Other Ground	00.30	07.36	01.93	00.00	01.51	00.53	00.79	00.81
Walkable	17.80	51.29	31.33	00.00	51.82	01.89	01.94	01.91
Sidewalk	17.74	36.40	18.22	00.00	45.53	01.98	02.66	02.71
mIoU	19.64	25.86	14.61	01.45	30.12	01.91	01.14	01.19

References

[1] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," 2017. [Online]. Available: https://arxiv.org/abs/1612.00593

[2] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," 2017. [Online]. Available: https://arxiv.org/abs/1706.02413

[3] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," 2019. [Online]. Available: https://arxiv.org/abs/1801.07829

[4] W. Qu, J. Wang, Y. Gong, X. Huang, and L. Xiao, "An end-to-end robust point cloud semantic segmentation network with single-step conditional diffusion models," 2025. [Online]. Available: https://arxiv.org/abs/2411.16308

[5] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," 2020. [Online]. Available: https://arxiv.org/abs/2006.11239

[6] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. Courville, "Film: Visual reasoning with a general conditioning layer," 2017. [Online]. Available: https://arxiv.org/abs/1709.07871

[7] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu, "3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations," 2024. [Online]. Available: https://arxiv.org/abs/2403.03954

[8] C. Luo, "Understanding diffusion models: A unified perspective," 2022. [Online]. Available: https://arxiv.org/abs/2208.11970

[9] Waymo, "Waymo-related document (google drive)," https://drive.google.com/file/d/1AB_SjteWdvqK-0Bv5QQAHO8093zxUTCh/view?pli=1, 2024, accessed: 2026-03-17.

[10] Medium, "Diffusion model from scratch in pytorch (ddpm)," https://medium.com/data-science/diffusion-model-from-scratch-in-pytorch-ddpm-9d9760528946, 2026, accessed: 2026-03-17.

[11] ——, "Dynamic graph cnn (edgeconv) overview," https://medium.com/@sanketgujar95/dynamic-graph-cnn-edge-conv-2582c3eb18d8, 2026, accessed: 2026-03-17.