PixelDrive: AI-Powered Road Scene Analysis

Mustakim Firoj Shikalgar
Arizona State University
mshikalg@asu.edu

Mohammed Amaan Ahmed Khan
Arizona State University
mkhan226@asu.edu

Siddharth Singh
Arizona State University
ssing644@asu.edu

Sarthak Bharatkumar Patel
Arizona State University
spate271@asu.edu

Abstract

The primary goal of this research is to generate long-lasting semantic segmentation models that can precisely recognize crucial factors in urban driving environments, including roads, vehicles, traffic lights, and pedestrians. Successful semantic segmentation greatly improves scene comprehension, a direct influ- ning self-driving cars' safety and decision-making capabilities in the real world scenarios. Using the Lyft-Udacity dataset, we implemented and compara- competitively compared three well-known deep learning architectures—U-Net, SegNet, and DeepLabV3+. Uniform image resizing was part of our data preprocessing pipeline, data caching, and performance-optimized GPU utilization for training. Com- Large-scale quantitative analyses based on Intersection over Union (IoU) scores, along with qualitative visual analyses, were performed to bring out each model's strengths and weaknesses. In-depth error analysis also gave us an insight into common misclassifications, highlighting areas of potential future improvement in semantic segmentation for autonomous applications.

1 Introduction

Semantic segmentation, which is the procedure of assigning meaningful class labels to every pixel in an image, is a crucial role in enabling autonomous driving systems to adequately perceive their surroundings. For autonomous cars to drive safely through intricate urban settings, they need to consistently recognize and identify objects such as roads, vehicles, pedestrians, and traffic lights in real-time. Successful semantic segmentation improves the situational awareness of a vehicle, which influences its ability to make informed decisions and passenger safety.

The latest surge of sophisticated deep learning methods has greatly enhanced the capabilities of computer vision tasks, especially semantic segmentation. State-of-the-art convolutional neural network (CNN) architectures such as U-Net, SegNet, and DeepLabV3+ have shown exceptional performance across diverse applications, making them prime candidates for deployment in autonomous driving scenarios. (1)

In this paper, we present *PixelDrive*, a comprehensive comparative study of these hit segmentation models specially tailored for urban autonomous driving settings. By real-world Lyft-Udacity dataset, we here give a thorough analysis of such architectures, emphasizing their practical usefulness, strengths, and weaknesses. Our methodological contributions have an optimized preprocessing pipeline with consistent resizing of input images and segmen- nation masks, dataset caching for improved computational efficiency, and judicious use of GPUs to accelerate model training.

We only measure model performance via Intersection over Union (IoU), which is the standard metric in semantic segmentation, quantitative comparisons and qualitative visual comparisons. Additionally,

Preprint. Under review.

we carry out a fine-grained error analysis to reveal typical misclassification cases, thereby providing significant insights into specific areas where segmentation performance can be im- proved. Our findings indicate that among the models considered, SegNet provided the most accurate segmentation accuracy, outperforming U-Net and DeepLabV3+ on mean IoU on test set

2 Related Works

Semantic segmentation is fundamental for autonomous driving, enabling pixel-wise classification of complex urban environments. Foundational works such as (1) and (2) demonstrated the effectiveness of encoder–decoder architectures like U-Net and SegNet in real-time scene parsing, balancing accuracy with computational efficiency.

Improved variants of these models, such as Enhanced U-Net (5) and DeepLab v3+ (15), introduced advanced techniques like atrous convolutions and spatial pyramid pooling to better capture multi-scale context. These models have shown superior performance in distinguishing overlapping classes like pedestrians and vehicles.

Open-source resources like (4) offer practical insight into implementing these models within modular pipelines, supporting reproducibility and experimentation. Multi-modal segmentation approaches, as explored in (12), (13), and (14), fuse LiDAR and RGB data to improve accuracy under challenging conditions such as occlusion or poor lighting.

Several studies, including (16) and (17), focus on domain adaptation and synthetic data generation to ensure model robustness across varying weather and lighting scenarios. Meanwhile, object detection and tracking systems in (7) and (8) complement segmentation tasks by enhancing spatial awareness and safety in real-time applications.

Datasets like Lyft Udacity (10) and Audi A2D2 (11) provide large-scale, real-world annotated data essential for training and benchmarking these models. Together, these works provide a strong foundation for our comparative analysis of segmentation models in autonomous driving.

3 Methods

3.1 Dataset and Preprocessing

We utilized the publicly released Lyft-Udacity dataset, a big corpus of varied, real- world urban driving scenes paired with precise segmentation masks. The dataset contains high- high-resolution RGB images of different driving situations like traffic, intersections, pedestrians and street scenes. All RGB images have pixel-level annotations (segmentation which put every pixel into categories referencing roads, cars, pedestrians, traffic lights, and other pertinent urban factors.

In order to normalize the inputs for training models, the segmentation masks and all images were resized equally to 256×256 pixels. This resizing enabled equal input sizes for all models, permitting fair comparisons and fast computation. We also utilized a data caching method during preprocessing. This approach significantly optimized the use of GPUs and reduced data loading bottlenecks, thus enhancing training performance and efficiency.

3.2 Model Architectures

We choose three convolutional neural network (CNN) models that are widely recognized for semantic segmentation accuracy: U-Net, SegNet, and DeepLabV3+. Each model's architecture was individually adapted and used in our semantic segmentation task. LeakyReLU activation functions with a negative slope of 0.1 are used in all the convolutional blocks of the U-Net and DeepLabV3 Custom models in place of the default ReLU. This addresses the issue of dying neurons and allows for smooth gradient flow throughout training.

3.3 U-Net

U-Net employs a symmetric encoder-decoder structure with skip connections. These skip connections help retain detailed spatial features from earlier layers during the decoding and upsampling stages, ensuring precise segmentation boundaries. It was implemented from scratch using the PyTorch framework. (5)

Encoder:

- Consists of four encoder blocks, each with two consecutive convolutional layers followed by Batch Normalization (BatchNorm) and ReLU activations.
- Spatial downsampling is achieved via MaxPooling layers after each encoder block.

Decoder:

- Comprises four decoder blocks, each beginning with a transposed convolution (ConvTranspose2D) for upsampling.
- Upsampled feature maps are concatenated with corresponding encoder outputs via skip connections.
- Each concatenation is followed by two convolutional layers, BatchNorm, and ReLU activations.

Output Layer:

- A final convolutional layer with 15 output channels corresponding to the segmentation classes, generating per-pixel predictions.

The table below summarizes the complete U-Net architecture configuration, clearly detailing each stage of the encoder-decoder structure:

Table 1: U-Net Encoder-Decoder Layer Configuration

Stage	Operation	Notes
Encoder Block 1	ConvBlock (2 × Conv2D + BatchNorm + LeakyReLU + Dropout) + MaxPool2D	Filters: 16
Encoder Block 2	ConvBlock + MaxPool2D	Filters: 32
Encoder Block 3	ConvBlock + MaxPool2D	Filters: 48
Encoder Block 4	ConvBlock + MaxPool2D	Filters: 64
Bridge	ConvBlock	Filters: 128
Decoder Block 1	UpSampling (TransposeConv) + Concat + ConvBlock + Dropout	128 → 64
Decoder Block 2	UpSampling + Concat + ConvBlock + Dropout	64 → 48
Decoder Block 3	UpSampling + Concat + ConvBlock + Dropout	48 → 32
Decoder Block 4	UpSampling + Concat + ConvBlock + Dropout	32 → 16
Output Layer	Conv2D (1×1)	15 classes

Model Details:

- Input shape: $(256, 256, 3)$
- Total parameters: approximately 599,631
- Trainable parameters: approximately 599,631
- Non-trainable parameters: approximately 0

3.4 SegNet

SegNet follows an encoder-decoder architecture specifically designed for pixel-wise classification tasks. This architecture efficiently performs upsampling by using pooling indices retained during the downsampling process. (8)

Encoder:

- Consists of four encoder blocks, each containing two Conv2D layers followed by Batch Normalization and ReLU activation.

- Spatial downsampling is achieved using MaxPooling layers with index retention for later unpooling.
- Filter sizes across blocks progressively increase: 64, 128, 256, and 512.

Decoder:

- Mirrors the encoder structure using MaxUnpooling layers guided by retained pooling indices.
- Each unpooling operation is followed by two Conv2D layers, BatchNorm, and ReLU activations.
- The decoder progressively decreases filter sizes in reverse: 512, 256, 128, and 64.

Output Layer:

- A final 1×1 Conv2D layer generates the 15-class per-pixel segmentation output.

The table below details the SegNet architecture configuration:

Table 2: SegNet Encoder-Decoder Layer Configuration

Stage	Operation	Notes
Encoder Block 1	2 × Conv2D + BatchNorm + ReLU + MaxPool2D (with indices)	3 → 64
Encoder Block 2	2 × Conv2D + BatchNorm + ReLU + MaxPool2D (with indices)	64 → 128
Encoder Block 3	2 × Conv2D + BatchNorm + ReLU + MaxPool2D (with indices)	128 → 256
Encoder Block 4	2 × Conv2D + BatchNorm + ReLU + MaxPool2D (with indices)	256 → 512
Decoder Block 4	MaxUnpool2D + 2 × Conv2D + BatchNorm + ReLU	512 → 256
Decoder Block 3	MaxUnpool2D + 2 × Conv2D + BatchNorm + ReLU	256 → 128
Decoder Block 2	MaxUnpool2D + 2 × Conv2D + BatchNorm + ReLU	128 → 64
Decoder Block 1	MaxUnpool2D + 2 × Conv2D + BatchNorm + ReLU	64 → 64
Output Layer	Conv2D (1×1)	15 classes

Model Details:

- Input shape: $(256, 256, 3)$
- Total parameters: approximately 9,409,807
- Trainable parameters: approximately 9,409,807
- Non-trainable parameters: 0

3.5 DeepLabV3+

DeepLabV3+ incorporates a ResNet-50 backbone pre-trained on ImageNet and an Atrous Spatial Pyramid Pooling (ASPP) module to capture multi-scale contextual information. The network also includes a lightweight decoder that combines high- and low-level features for accurate segmentation.

Encoder:

- Utilizes a ResNet-50 backbone for hierarchical feature extraction. Stride in later stages is replaced with dilation to preserve spatial resolution.
- Incorporates ASPP, which applies parallel atrous convolutions with dilation rates of 6, 12, and 18, as well as global image-level context through adaptive average pooling.

Decoder:

- Reduces the low-level features (from earlier ResNet stages) via 1×1 convolution and batch normalization.
- High-level ASPP features are upsampled and concatenated with reduced low-level features.
- The concatenated features are refined via two 3×3 convolution layers with BatchNorm, ReLU, and Dropout.

Output Layer:

- A final 1×1 convolution layer produces class scores, which are upsampled using bilinear interpolation to match the input resolution.

The architecture is summarized in the following table:

Table 3: DeepLabV3+ Layer Configuration

Stage	Operation	Notes
Backbone	ResNet-50 pretrained on ImageNet	Replaces stride with dilation in later layers
ASPP Module	1×1 Conv, 3×3 Conv with dilation (6,12,18), Image Pooling	Output: 256 channels
Feature Fusion	1×1 Conv on low-level features + Concatenation with ASPP output	256 $\rightarrow$ 48 (low-level)
Decoder	2 $\times$ 3×3 Conv + BatchNorm + ReLU + Dropout	Output: 256 channels
Upsampling	Bilinear interpolation to original resolution	–
Output Layer	Conv2D (1×1)	15 classes

Model Details:

- Input shape: $(256, 256, 3)$
- Total parameters: approximately 42,320,303
- Trainable parameters: approximately 42,320,303
- Non-trainable parameters: 0

3.6 Training Pipeline

The models were implemented and trained using the PyTorch deep learning framework on GPU-enabled hardware. We trained with the Adam optimizer and initial learning rate 1e-3 to optimize the cross-entropy loss between predicted segmentation maps and ground-truth masks. A learning rate scheduler (ReduceLROnPlateau) was used to dynamically change the learning rate, decreasing it when the validation loss plateaued, thus providing effective convergence.

To prevent overfitting, we used early stopping on validation loss improvement, stopping training after two epochs of no improvement which was automated. Training of the models was closely monitored with loss and accuracy graphs against epochs, that gave details about the convergence of each model stability of behavior and learning.

DeepLabV3+ uses atrous spatial pyramid pooling (ASPP) for efficient multi-scale capture. contextual information. We employ a ResNet-50 backbone to extract features from facilitating robust semantic understanding. Detailed layer configurations, including specific filter sizes, convolutional layers, pooling mechanisms, and upsampling approaches, have been explicitly documented and adapted from standard configurations widely reported in the literature and are clearly summarized in the tables provided in this report.

4 Experiments and Results

Figure 1 shows the training performance of the U-Net model in terms of loss and accuracy over training epochs. This provides a clear view of convergence behavior and training stability.

To further analyze model performance, we present sample predictions from the U-Net model in Figure 2, showcasing segmentation results on test images.

Finally, we evaluate the segmentation performance using Intersection over Union (IoU) scores. The following table summarizes minimum, maximum, and mean IoU across training, validation, and test sets.

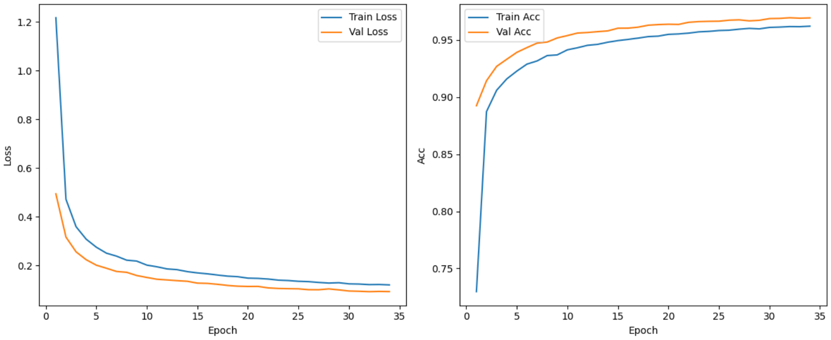
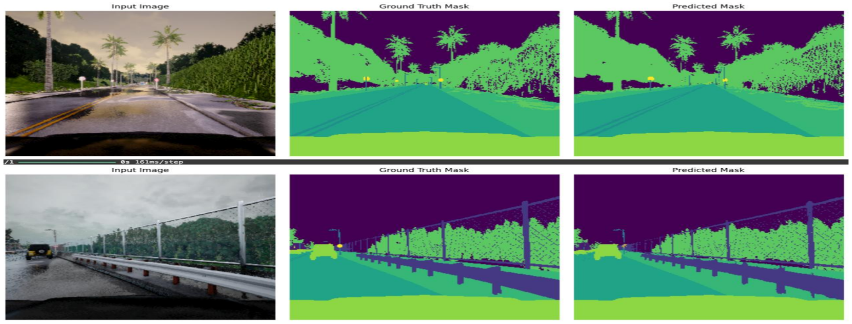


Figure 1: U-Net training accuracy and loss over epochs.



```
/1 ━━━━━━━━━━━━━━━━━━━━ 0s 161ms/step
```

Figure 2: U-Net model predictions on test images.

Table 4: Intersection over Union (IoU) scores for U-Net model.

Dataset	Minimum IoU (%)	Maximum IoU (%)	Mean IoU (%)
Training	83.21	98.50	94.05
Validation	84.35	98.49	93.99
Test	87.91	98.13	94.05

Figure 3 illustrates the training performance of the SegNet model, showing both loss and accuracy trends across epochs. The curve indicates stable convergence and highlights the effectiveness of the learning setup over time.

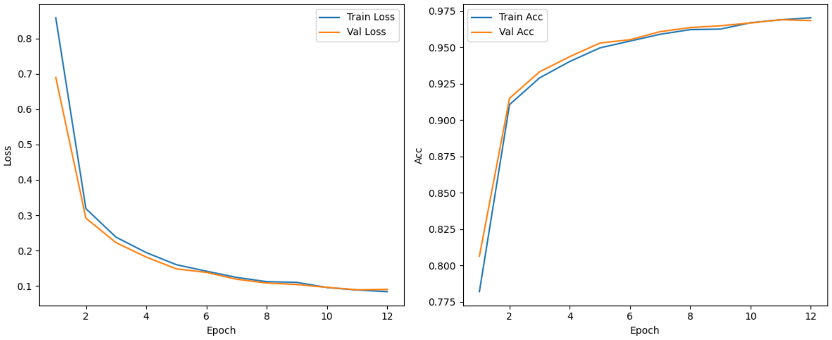


Figure 3: SegNet model architecture diagram.

To evaluate segmentation performance, we computed Intersection over Union (IoU) scores across training, validation, and test datasets. The table below presents these values, reflecting the model's consistency and generalization ability across splits.

Table 5: Intersection over Union (IoU) scores for SegNet model.

Dataset	Minimum IoU (%)	Maximum IoU (%)	Mean IoU (%)
Training	90.12	98.91	95.73
Validation	85.74	98.87	95.42
Test	89.88	98.59	95.50

Figure 4 displays visual examples of SegNet's segmentation output. These images compare predicted masks against the original scenes, offering qualitative insights into the model's performance.

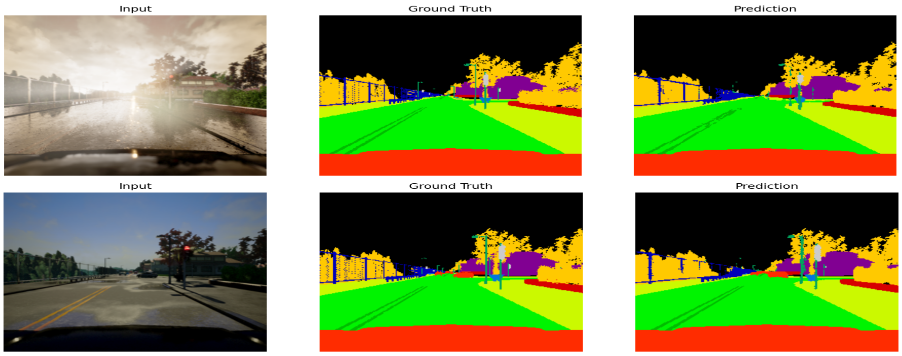


Figure 4: SegNet model predictions.

Figure 5 shows the training performance of DeepLabV3+ in terms of accuracy and loss across epochs. The plot illustrates stable convergence and effective learning.

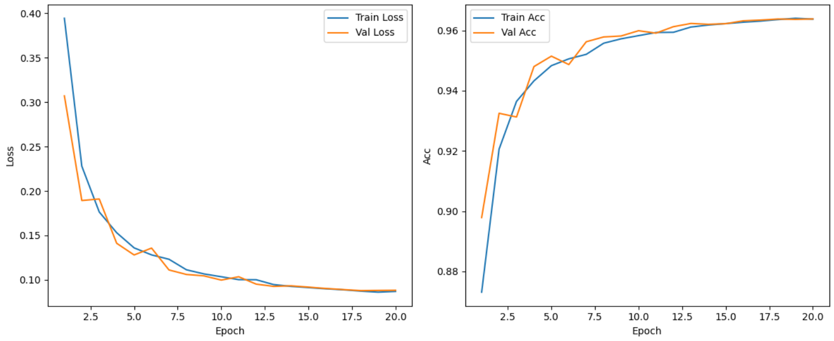


Figure 5: DeepLabV3+ training accuracy and loss over epochs.

To qualitatively evaluate segmentation results, Figure 6 displays output predictions generated by DeepLabV3+ on selected test images.

Table 6 presents Intersection over Union (IoU) scores across the training, validation, and test sets. These metrics demonstrate strong and consistent segmentation performance across datasets.

Table 6: Intersection over Union (IoU) scores for DeepLabV3+ model.

Dataset	Minimum IoU (%)	Maximum IoU (%)	Mean IoU (%)
Training	87.68	98.27	94.25
Validation	87.52	98.18	94.15
Test	89.58	98.15	94.23

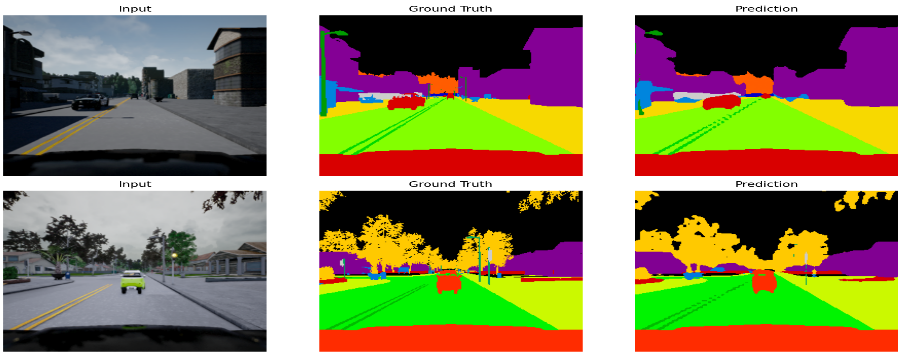


Figure 6: DeepLabV3+ model predictions on test images.

Table 7: Comparison of model performance (Mean IoU on test set and total parameters).

Model	Mean IoU (Test) [%]	Total Parameters
U-Net	94.05	599,631
SegNet	95.50	9,409,807
DeepLabV3+	94.23	42,320,303

Table 7 provides a concise comparison across models, highlighting SegNet's marginal advantage in IoU at a moderate parameter cost relative to DeepLabV3+.

Among the three architectures evaluated, SegNet achieved the highest mean IoU of 95.50% on the test set. U-Net and DeepLabV3+ followed closely with 94.05% and 94.23% respectively. Despite DeepLabV3+ being the most parameter-heavy model, its performance was competitive, especially in maintaining consistency across validation and test data. The relatively narrow margin across all models indicates that preprocessing standardization and consistent training protocols contributed to stable and high-quality segmentation.

5 Conclusion

In this work, we implemented and evaluated three state-of-the-art semantic segmentation models—U-Net, SegNet, and DeepLabV3+—on the Lyft-Udacity dataset. Each model demonstrated strong performance, with SegNet achieving the highest mean Intersection over Union (IoU) score of 95.50% on the test set. U-Net and DeepLabV3+ followed closely, confirming the effectiveness of skip connections and multiscale context aggregation in urban scene segmentation. These results underscore the trade-off between parameter efficiency and segmentation accuracy, with SegNet offering a balanced approach. Future work will focus on optimizing inference speed and extending model robustness to nighttime and adverse weather conditions.

Among the models tested, SegNet demonstrated the highest segmentation performance with a mean IoU of 95.50% on the test set, outperforming U-Net and DeepLabV3+ in both accuracy and generalization.

References

[1] Image segmentation for Self-Driving cars. (2022, June 24). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/9847974

[2] Qiaoxu.(n.d.-b). Self-Driving-Cars/Part3-Visual_Perception_for_Self-Driving_Cars/Module5-Semantic_Segmentation/Module5-Semantic_Segmentation.md at master · qiaoxu123/Self-Driving-Cars. GitHub. https://github.com/qiaoxu123/Self-Driving-Cars/blob/master/Part3-Visual_Perception_for_Self-Driving_Cars/Module5-Semantic_Segmentation/Module5-Semantic_Segmentation.md

[3] Divakarla, U., Bhat, R., Madagaonkar, S. B., Pranav, D. V., Shyam, C., & Chandrashekar, K. (2023). Semantic segmentation for autonomous driving. In Lecture notes in networks and systems (pp. 683–694). https://doi.org/10.1007/978-981-19-9304-6_61

[4] Qiaoxu.(n.d.-b). Self-Driving-Cars/Part3-Visual_Perception_for_Self-Driving_Cars/Module5-Semantic_Segmentation/Module5-Semantic_Segmentation.md at master · qiaoxu123/Self-Driving-Cars. GitHub. https://github.com/qiaoxu123/Self-Driving-Cars/blob/master/Part3-Visual_Perception_for_Self-Driving_Cars/Module5-Semantic_Segmentation/Module5-Semantic_Segmentation.md

[5] Enhanced U-Net Approach: Semantic Segmentation for Self-Driving Cars applications. (2023, February 20). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/10411167

[6] https://www.sciencedirect.com/science/article/pii/S131915782400315X

[7] https://www.folio3.ai/blog/vehicle-detection-device-vs-ai-vehicle-detection-apps/

[8] Khan, S. A., Lee, H. J., & Lim, H. (2023). Enhancing object detection in Self-Driving cars using a hybrid approach. Electronics, 12(13), 2768. https://doi.org/10.3390/electronics12132768

[9] https://colab.research.google.com/github/chervovn04/DLS/blob/main/SemanticSegmentation.ipynb

[10] https://www.kaggle.com/datasets/kumaresanmanickavelu/lyft-udacity-challenge

[11] https://www.a2d2.audi/a2d2/en/dataset.html

[12] Li, Y., Wang, X., & Chen, Z. (2023). MSeg3D: Multi-modal 3D semantic segmentation for autonomous driving. arXiv preprint arXiv:2303.08600. https://arxiv.org/abs/2303.08600

[13] Wang, H., Xie, Y., Zhang, Q., & Xu, Y. (2023). Revisiting multi-modal 3D semantic segmentation in real-world autonomous driving. arXiv preprint arXiv:2310.08826. https://arxiv.org/abs/2310.08826

[14] Lee, J., Park, S., & Kim, M. (2023). Improved 3D semantic segmentation model based on RGB image and LiDAR point cloud fusion for autonomous driving. Journal of Mechanical Science and Technology, 37(7), 3165–3175 https://link.springer.com/article/10.1007/s12239-023-0065-y

[15] Yang, H., Zhang, L., & Liu, P. (2023). Research on semantic segmentation algorithm for autonomous driving based on improved DeepLabv3+. In Lecture Notes in Electrical Engineering (Vol. 998, pp. 137–147). https://link.springer.com/10.1007/978-981-99-7545-7_12

[16] Ren, Y., Liu, S., & Huang, Q. (2025). Towards generating realistic 3D semantic training data for autonomous driving. arXiv preprint arXiv:2503.21449. https://arxiv.org/html/2503.21449v1

[17] Xu, L., Wang, R., & Zheng, H. (2025). Domain-incremental semantic segmentation for autonomous driving under adverse driving conditions. arXiv preprint arXiv:2501.05246. https://arxiv.org/html/2501.05246v1

[18] Generate a LaTeX table comparing U-Net, SegNet, and DeepLabV3+ on IoU and parameters

[19] Fine tune the figure's position and caption in terms of Latex format.