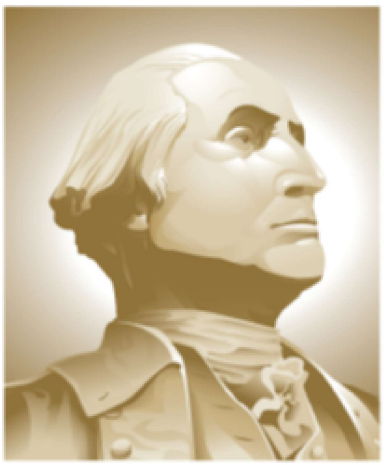


THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

Data Science Program

Project Report - Spring 2025
Group - 4

High-Performance Semantic Segmentation for Autonomous Urban Navigation

Abhilasha Singh

Pranjal Wakpaijan

Shabnam Rafat Ummey Sawda

Supervised By

Amir Jafari

TABLE OF CONTENT

1. Introduction

Semantic segmentation, a cornerstone of computer vision, plays a pivotal role in enabling autonomous vehicles to interpret complex urban environments with pixel-level precision. By assigning semantic labels to every pixel in an image, segmentation models provide a fine-grained understanding of scenes, distinguishing critical elements such as roads, pedestrians, vehicles, and infrastructure. This capability is essential for downstream tasks in autonomous driving, including navigation, obstacle avoidance, and real-time decision-making. As urban environments present diverse challenges ranging from class imbalance and occlusion to varying lighting and weather conditions developing robust and high-performance segmentation models remains a critical research frontier.

Recent advancements in deep learning have significantly enhanced semantic segmentation performance. Models like DeepLabv3 [1] and its successor, DeepLabv3+ [2], leverage atrous convolutions and encoder-decoder architectures with ResNet backbones to capture multi-scale contextual information while maintaining computational efficiency. Concurrently, Vision Transformers (ViTs) [3] have emerged as a powerful alternative, utilizing self-attention mechanisms to model long-range dependencies and achieve state-of-the-art results on complex datasets. These architectures offer distinct strengths: convolutional models excel in local feature extraction, while transformers capture global context, making them complementary approaches for urban scene understanding.

This report investigates the performance of three segmentation models DeepLabv3, DeepLabv3+ with ResNet support, and a Vision Transformer-based architecture in the context of autonomous urban navigation. By evaluating their accuracy, robustness, and ability to handle class imbalance in real-world urban scenarios, we aim to elucidate their practical applicability and limitations.

2. Dataset

For this study, we utilize the Audi Autonomous Driving Dataset (A2D2) [4], a comprehensive and publicly available dataset designed to advance research in autonomous driving. A2D2 provides a rich multimodal sensor suite, including data from six cameras and five Velodyne VLP-16 LiDAR units, offering full 360-degree environmental coverage. The dataset comprises 41,277 non-sequential frames with semantic segmentation labels for both camera images and corresponding LiDAR point clouds, annotated across 55 semantic classes. Of these, 12,497 frames also include 3D bounding box annotations within the field of view of the front-center camera. Additionally, A2D2 includes 392,556 sequential frames of unannotated sensor data, capturing diverse urban, highway, and country road scenarios across three cities in southern

Germany under varying weather conditions (sunny, cloudy, and rainy). Vehicle bus data, such as steering angle, throttle, and brake information, further enriches the dataset, enabling exploration of end-to-end learning and sensor fusion approaches.

A2D2 stands out among other autonomous driving datasets, due to its emphasis on full surround sensor coverage and the inclusion of vehicle bus data, which is often absent in comparable datasets. The dataset's semantic segmentation annotations, covering a taxonomy similar to Cityscapes, make it particularly suitable for pixel-level scene understanding tasks. To ensure compliance with GDPR and preserve anonymity, faces and vehicle license plates in the images are blurred. Released under the CC BY-ND 4.0 license, A2D2 permits commercial use, fostering both academic and industrial research.

For our experiments, we focus on the semantic segmentation subset of A2D2, utilizing 31,448 RGB images with a resolution of 1920×1208 pixels. Following standard practice, we split the data into training , validation and test sets. To align with common benchmarks, we map the 55-class taxonomy, including 18 foreground classes of interest (e.g., road, vehicle, pedestrian) and a background class, ensuring compatibility with Cityscapes' taxonomy. Data augmentation techniques, including random cropping, brightness and contrast adjustments, and horizontal flipping, are applied to enhance model robustness. The performance of the segmentation models DeepLabv3, DeepLabv3+ with ResNet support, and a Vision Transformer based architecture is evaluated using the mean Intersection over Union (mIoU) metric, providing a robust measure of segmentation accuracy across diverse urban scenarios.

3. Problem Statement

Autonomous urban navigation demands precise and robust semantic segmentation to enable pixel-level scene understanding for safe and efficient decision-making. However, urban environments pose significant challenges, including class imbalance, occlusion, and varying lighting conditions, which can degrade segmentation performance. This project aims to address the problem of achieving high-performance semantic segmentation in urban scenes by evaluating and comparing three models—DeepLabv3, DeepLabv3+ with ResNet50 support, and a Segmentation Transformer with a fallback to DeepLabv3+ (ResNet50) to determine their accuracy, robustness, and ability to handle real-world complexities, ultimately enhancing autonomous driving systems.

4. Model Architecture

4.1 Baseline Model

The initial baseline model implemented in this study utilizes the DeepLabV3 architecture with a MobileNetV3-Large backbone, aimed at achieving a balance between computational efficiency and high-resolution semantic segmentation. This configuration supports real-time inference and is

optimized for deployment in resource-constrained environments, such as edge devices used in autonomous vehicles. The MobileNetV3 backbone, known for its lightweight structure, complements DeepLabV3's atrous spatial pyramid pooling (ASPP) for contextual aggregation.

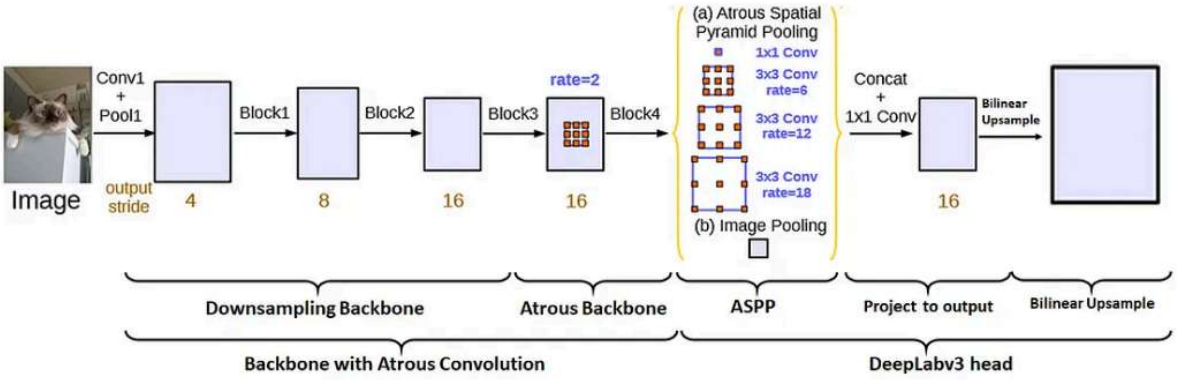


Fig.DeepLabV3

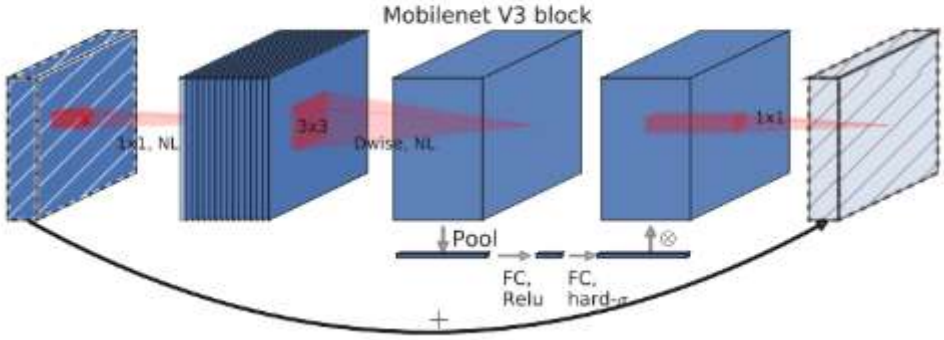


Fig.MobileNetV3

However, this architecture presents significant limitations for urban scene understanding, particularly in autonomous driving scenarios. Notably, DeepLabV3 lacks a decoder module, which restricts its capacity for fine-grained spatial reconstruction, a critical requirement for pixel-accurate segmentation. Additionally, MobileNetV3, while efficient, has reduced representational capacity compared to deeper backbones like ResNet50, making it suboptimal for complex, large-scale visual tasks like autonomous driving where dense object interactions and occlusions are common.

Training and Deployment Strategy

The training pipeline implements a custom learning rate scheduling strategy, where the learning rate increases gradually during the warm-up phase and decays exponentially after reaching a peak. This helps in achieving stable convergence and prevents issues like gradient explosion in the early epochs. Furthermore, the training leverages Automatic Mixed Precision (AMP),a feature in PyTorch that allows computations to run in a mix of float16 and float32 precision. By using lower precision where safe, AMP significantly improves training speed and reduces GPU memory usage, while maintaining model accuracy through dynamic loss scaling and careful casting.

For deployment, the model is saved regularly using a checkpointing mechanism. Each epoch generates a versioned snapshot and is maintained for quick access to the most recent model. This strategy ensures reliability, simplifies rollback if needed, and supports reproducibility during experimentation and deployment.

Evaluation Observations

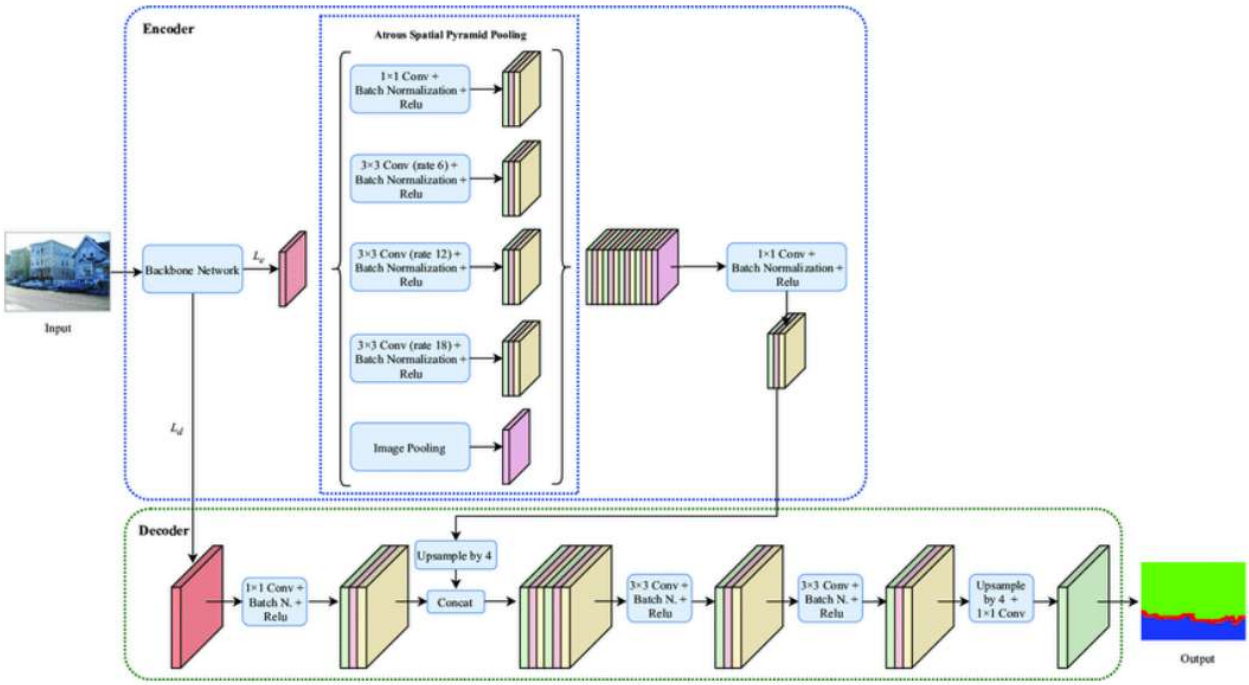
A number of critical limitations were observed during experimentation:

- **No pre-defined train-validation-test split** was included in the original configuration, necessitating custom dataset partitioning for fair evaluation.
- **Insufficient evaluation metrics, specifically** the lack of mean IoU, pixel accuracy, F1 score, Dice coefficient, and class-wise accuracy,restricted effective performance diagnosis.
- **Lack of architectural complexity** limited performance on fine-grained classes such as pedestrians, traffic lights, or lane markings.
- **Excessive checkpointing** led to high disk usage, as all epoch snapshots were retained without pruning.

This baseline configuration served as a benchmark reference to evaluate the performance of more advanced models such as DeepLabV3+ with ResNet50 and the Vision Transformer-based segmentation network. These models are explored to address the baseline's limitations by improving representational capacity, decoder integration, and metric coverage, ultimately contributing toward the development of more accurate and reliable segmentation systems for autonomous driving.

4.2 DeepLab V3+ (Resnet50)

DeepLabV3+ is an advanced semantic segmentation architecture that builds upon the strengths of DeepLabV3 by integrating a more effective decoder module for precise object boundary delineation. This design significantly improves the segmentation of fine-grained spatial details, which is crucial in complex urban scenes such as those encountered in autonomous driving datasets.



Architecture of DeepLabV3+ with backbone network.

At the heart of DeepLabV3+ is its **encoder-decoder structure**:

The encoder uses Atrous Spatial Pyramid Pooling (ASPP) to capture multi-scale contextual information without increasing computational cost.The decoder refines segmentation outputs, particularly at object edges, by combining low-level features from earlier layers with high-level semantic representations.

To enhance feature extraction, DeepLabV3+ in our project utilizes a ResNet-50 backbone: ResNet-50 is a deep convolutional neural network pre-trained on ImageNet, known for its residual connections that mitigate the vanishing gradient problem.It provides a strong foundation for learning robust hierarchical features, making it an excellent choice for transfer learning in segmentation tasks.This combination DeepLabV3+ with ResNet-50 offers a balance between accuracy and efficiency, making it suitable for high-resolution segmentation tasks in real-time or resource-constrained environments.
These architecture enhancements were accompanied by support for multiple loss functions (e.g., CrossEntropyLoss, Dice Loss), and optimizers such as SGD for baseline models and AdamW in the enhanced configurations to improve convergence stability and regularization. Additionally, confidence visualization capabilities were integrated to inspect prediction certainty across different spatial regions.

Enhancements Over the Baseline Model

Training Strategy Improvements

The initial baseline model lacked a structured evaluation framework, leading to inconsistent performance assessment. To address this, we implemented a customized data loader tailored to our dataset's format, enabling efficient and flexible data handling. Additionally, to enhance the setup with DeepLabV3+, we implemented several critical upgrades:

Introduced a structured 80/10/10 train-validation-test split with proper randomization to ensure consistent evaluation.
Integrated data augmentation techniques (e.g., horizontal flips, color jittering) to improve generalization and reduce overfitting.
Implemented class imbalance handling using weighted sampling and custom loss functions.
Switched to the AdamW optimizer to benefit from adaptive learning rates and better regularization.
Ensured computational feasibility by tuning batch sizes and memory usage for high-resolution image processing. A custom data loader was also designed to efficiently process the image-mask pairs according to this structure.

Expanded Evaluation Metrics

Early training stages lacked the quantitative tools needed to assess segmentation performance effectively. In the refined workflow, we introduced a suite of evaluation metrics:

Mean Intersection over Union (mIoU):
Measures the average overlap between predicted and ground truth masks across all classes

Pixel Accuracy:
Represents the proportion of correctly classified pixels out of the total, offering a broad measure of correctness.

F1 Score:
Balances precision and recall, which is especially useful when dealing with imbalanced datasets.

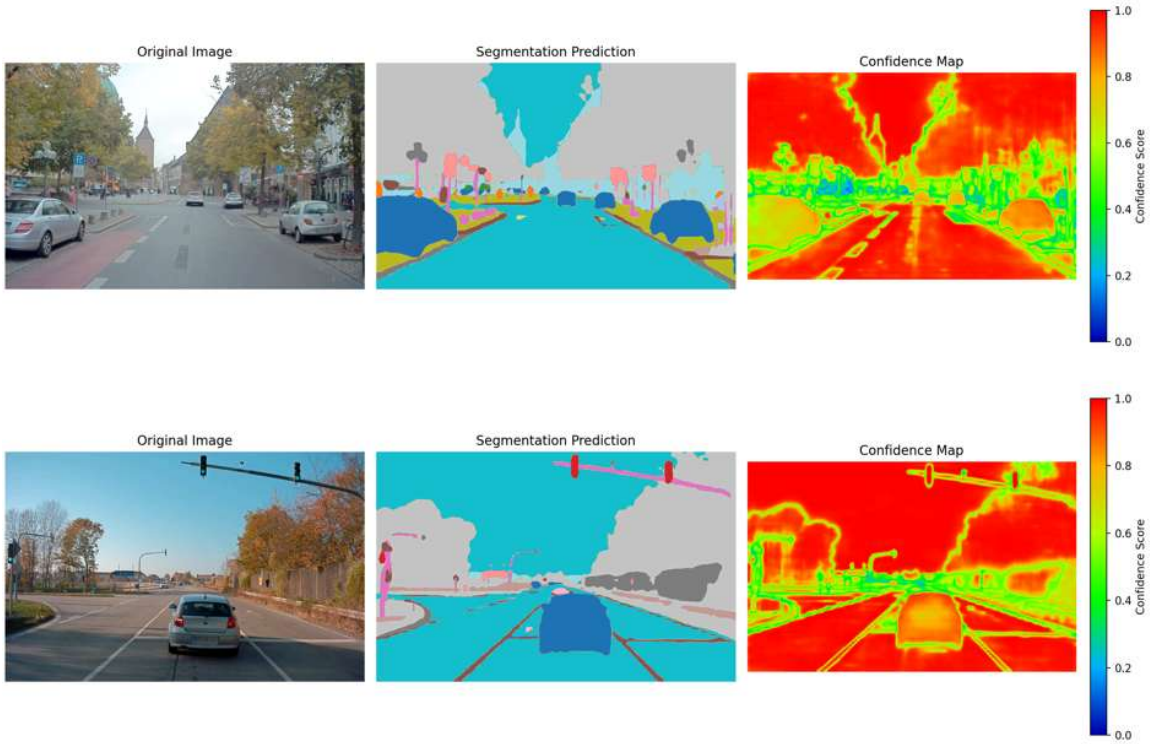
Dice Coefficient:
A similarity measure particularly effective for fine-grained and minority class predictions.

Loss Monitoring:
Training and validation losses were continuously tracked to monitor model convergence and detect overfitting trends.

These metrics enabled us to detect failure cases (e.g., classes with near-zero mIoU) and apply corrective strategies through targeted augmentation and sampling.

Deployment Pipeline

The final model iterations prioritized deployment-readiness by implementing efficient per-epoch and final version checkpointing, output visualization pipelines with segmentation overlays and per-class performance summaries, and color-mapped result saving for enhanced qualitative prediction interpretation.



The visualization compares the original image, segmentation output, and pixel-wise confidence map generated by the model. High-confidence areas (in red/yellow) align with well-represented classes like roads and vehicles, while lower-confidence regions (in green/blue) typically occur near object boundaries or less frequent classes. This helps assess both the accuracy and certainty of the model's predictions in complex driving scenes.

4.3 Vision Transformer

Vision Transformer (ViT)-based Segmentation Transformer model designed for semantic segmentation on the Audi Autonomous Driving Dataset (A2D2). The model leverages

transformer architecture to process high-resolution images and produce pixel-wise class predictions, critical for autonomous driving applications.

4.3.1 Model Architecture

The Segmentation Transformer is a Vision Transformer-based architecture tailored for semantic segmentation, combining patch-based image processing with transformer encoder-decoder layers to capture global context and produce high-resolution segmentation masks.

The model processes input RGB images of size (1208, 1920) and outputs segmentation masks with 55 classes, corresponding to objects and regions in the A2D2 dataset (e.g., road, car, pedestrian). The architecture consists of the following components:

1. **Input Image**:
    - Shape: (B, 3, 1208, 1920) (batch size, 3 RGB channels, height, width).
    - Represents high-resolution A2D2 images.
2. **Patch Embedding**:
   Patch embedding is a fundamental process in Vision Transformer (ViT) models, designed to transform a high-resolution input image into a sequence of fixed-size vector representations suitable for transformer processing. This sequence of patch embeddings serves as the input to the transformer's encoder, enabling the model to capture global relationships across the image while preserving local spatial information, a critical step for tasks like semantic segmentation in autonomous driving scenarios.
3. **Positional Embedding + CLS Token**:
   In the Segmentation Transformer model for the Audi Autonomous Driving Dataset (A2D2), positional embedding and the CLS token are critical components that enhance the transformer's ability to process patch-based image representations by incorporating spatial and global contextual information.
4. **Transformer Encoder**:
   Following the integration of positional embeddings and the CLS token, which transforms the input image's patch embeddings into a tensor of shape (B, 35,701, 768) with spatial and global contextual information, the Transformer Encoder in the Segmentation Transformer model for the Audi Autonomous Driving Dataset (A2D2) takes over to process this sequence and capture intricate global relationships across the image.
5. **Transformer Decoder**:
   The model features a decoder with four layers (decoder_depth=4), each comprising a sequence of operations: LayerNorm to normalize input, Cross-Attention using decoder input queries and encoder output (memory) keys/values, a residual connection, another LayerNorm, Self-Attention on decoder tokens, a residual connection, a final LayerNorm, an MLP akin to the encoder's, and a residual connection. It takes the encoder output

without the CLS token (B, 35,700, 768) as input and produces an output of the same shape (B, 35,700, 768).

6. **Output Head**:
    - Linear layer mapping embeddings to class scores: 768 -> 55 (num_classes).
    - Output: (B, 35,700, 55).
    - Reshapes to (B, 55, 150, 238) (channels, patch grid height, width).
7. **Upsampling**:
    - Uses bilinear interpolation to resize from (B, 55, 150, 238) to (B, 55, 1208, 1920).
    - Output: Final segmentation map with class probabilities.
8. **Final Output**:
    - After argmax(dim=1), produces (B, 1208, 1920) with class IDs.
    - Represents the segmentation mask, where each pixel is assigned one of 55 classes.

4.3.2 Training

The model is trained on the A2D2 dataset using a supervised learning approach with a combination of loss functions and advanced optimization techniques. The training pipeline is designed for efficiency and robustness, leveraging GPU acceleration and mixed-precision training.

Preprocessing:
Images resized to (1208, 1920) using bilinear interpolation.
Masks resized using nearest-neighbor interpolation to preserve class IDs.
Dataset Splitting:
Split into training (70%), validation (15%), and test (15%) sets.
Random seed (42) ensures reproducibility.
Example split: For 10,000 samples, 7,000 (train), 1,500 (val), 1,500 (test).
Data Loaders:
Batch size: 3 (configurable via args.batch).
Workers: 6 for parallel loading.
Prefetch factor: 2 for efficient data fetching.
Pin memory enabled for faster GPU transfer.
Training loader shuffles data; validation and test loaders do not.

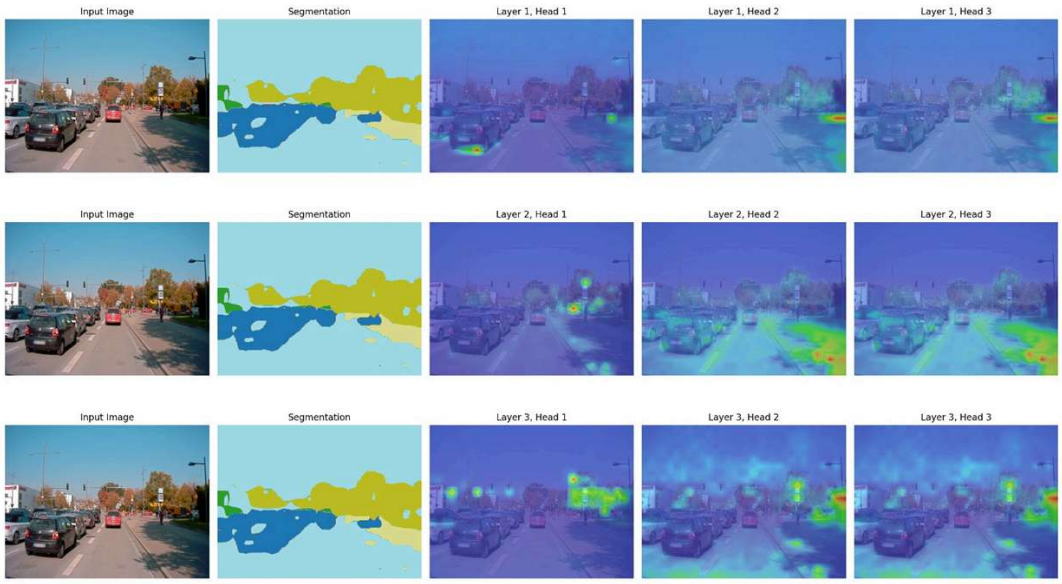
4.3.3 Loss Functions

The model supports multiple loss functions, selected via args.loss:
CrossEntropyLoss (ce**)**: Standard loss for multi-class segmentation.
FocalLoss: Focuses on hard examples using gamma=2.0 (focusing parameter) and alpha=0.25 (class balancing).
DiceLoss: Optimizes for overlap between predicted and ground truth masks with smooth=1.0.

CombinedLoss: Weighted combination of Focal and Dice losses (dice_weight=0.5, focal_weight=0.5).
Default: CrossEntropyLoss if not specified.

4.3.4 Optimization

The model employs the AdamW optimizer, customized for transformers, with a configurable learning rate of 0.005 (set via args.lr) and a weight decay of 0.05 to mitigate overfitting. It uses a learning rate scheduler with a warmup phase that linearly increases the learning rate over the first 10% of total steps, followed by a decay phase that linearly reduces it to zero. Mixed-precision training is enabled via args.amp="True", leveraging torch.cuda.amp with GradScaler to enhance training speed and reduce memory consumption.



This visualization shows how different attention heads in a Vision Transformer focus on various parts of the scene across layers. Early layers capture broad contextual features, while deeper layers focus more precisely on objects like vehicles and traffic signals. These attention maps help interpret how the model understands and segments complex urban environments.

5. Streamlit Applications for Semantic Segmentation Analysis

To support interactive exploration and evaluation of our semantic segmentation pipeline, we developed custom Streamlit web applications:

5.1 Mask Analysis App

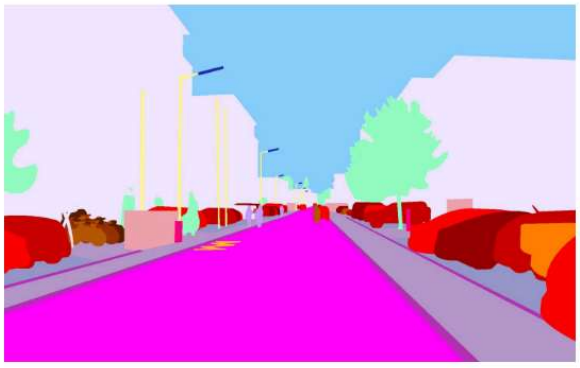
The Mask Analysis App was designed to facilitate detailed inspection of segmentation masks and class distributions across the dataset. This tool allows users to load pairs of RGB input images and their corresponding ground-truth masks, along with a JSON class map defining RGB-to-class associations.

Key functionalities include:

- Directory-based input handling for images, masks, and class mappings via a hex-to-label JSON configuration.
- Robust pixel-level mapping, converting RGB values in masks to class indices with built-in tolerance for slight color variations.
- Per-image statistical analysis, including:
    - Total vs. assigned pixel count
    - Class-wise distribution with label names
    - Identification of the dominant class
- Mismatch detection, which highlights unmatched RGB values in the mask, helping to identify annotation errors or unsupported colors.
- Side-by-side visual comparison of each original image and its corresponding label mask for intuitive inspection.

This tool proved essential for verifying dataset consistency, understanding class distribution trends, and informing both data augmentation and sampling strategies during model training.

Input Image



Label Mask

Mask Analysis:

- Total Pixels: 2319360
- Assigned Pixels: 2319360 (100.00%)
- Class Distribution:
- Class ID 0 (Car 1): 61970 pixels (2.67%)
- Class ID 1 (Car 2): 67306 pixels (2.90%)
- Class ID 2 (Car 3): 38066 pixels (1.64%)
- Class ID 3 (Car 4): 380 pixels (0.02%)

- Class ID 4 (Bicycle 1): 9005 pixels (0.39%)
- Class ID 5 (Bicycle 2): 5528 pixels (0.24%)
- Class ID 6 (Bicycle 3): 3966 pixels (0.17%)
- Class ID 8 (Pedestrian 1): 3004 pixels (0.13%)
- Class ID 11 (Truck 1): 21345 pixels (0.92%)
- Class ID 28 (Solid line): 2021 pixels (0.09%)
- Class ID 34 (Obstacles / trash): 2749 pixels (0.12%)
- Class ID 35 (Poles): 17865 pixels (0.77%)
- Class ID 38 (Grid structure): 27061 pixels (1.17%)
- Class ID 39 (Signal corpus): 1307 pixels (0.06%)
- Class ID 40 (Drivable cobblestone): 35160 pixels (1.52%)
- Class ID 43 (Nature object): 110089 pixels (4.75%)
- Class ID 44 (Parking area): 44712 pixels (1.93%)
- Class ID 45 (Sidewalk): 174015 pixels (7.50%)
- Class ID 50 (RD normal street): 488219 pixels (21.05%)
- Class ID 51 (Sky): 442940 pixels (19.10%)
- Class ID 52 (Buildings): 762652 pixels (32.88%)
- Dominant Class: 52 (32.88% of image)

5.2 Model Evaluation App

The Model Evaluation App was built to streamline post-training performance analysis of segmentation models. It takes as input a trained PyTorch model (.pth), a directory of test images, optional ground-truth masks, and a class list JSON.

Key Features:
Automatic model loading with seamless CPU/GPU detection for device-specific execution.

- Custom test dataset handler that dynamically converts RGB segmentation masks to class-index format using a user-provided class map.
- Comprehensive evaluation metrics, including:
    - Mean Intersection over Union (mIoU)
    - Pixel Accuracy
    - Mean Dice Coefficient

    - Precision, Recall, and F1 Score (both per-class and macro-averaged)
    - Boundary F1 Score (approximated using macro F1)

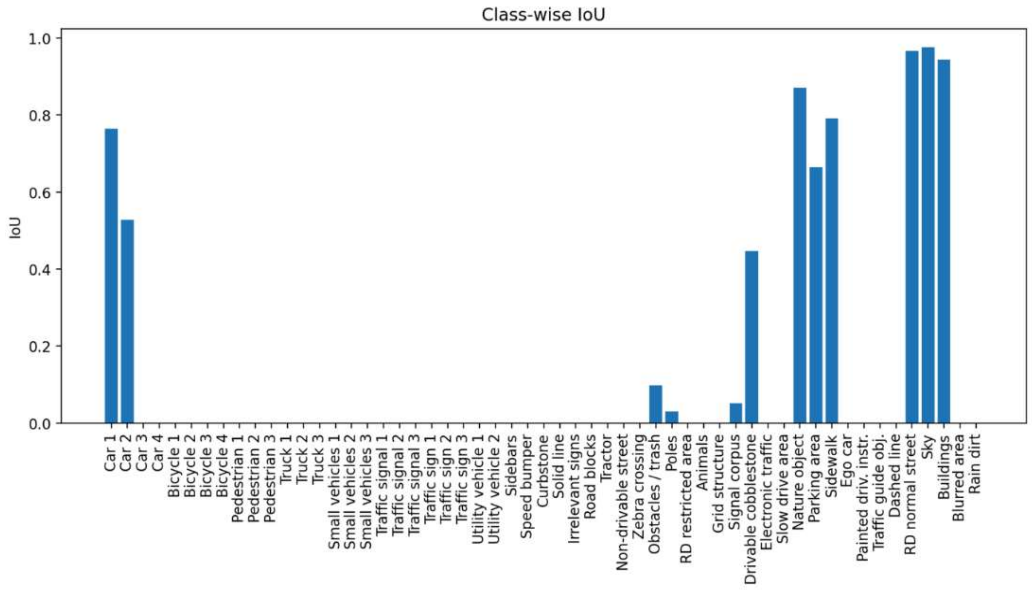
Model Evaluation on Test Set

Evaluation Completed!

Evaluation Metrics

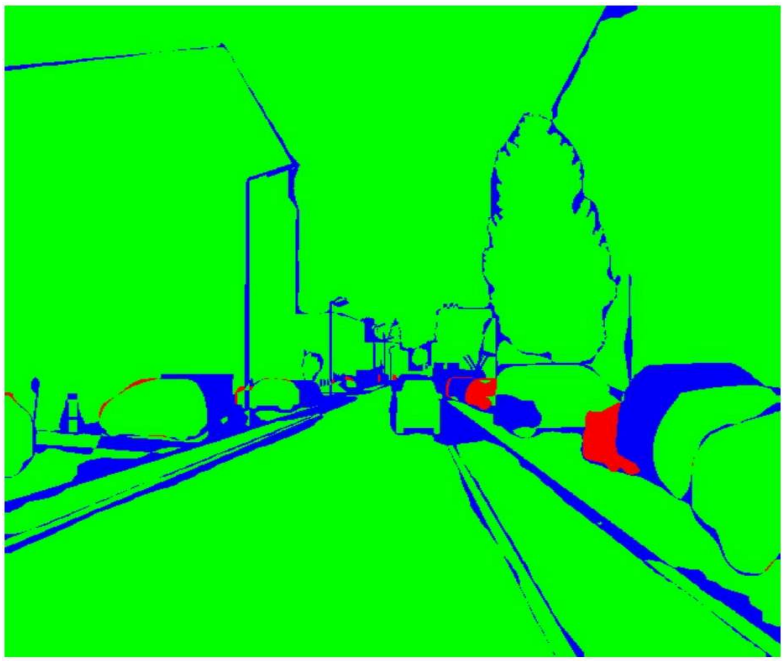
Mean IoU	Pixel Accuracy	Mean F1 Score
0.1297	0.9253	0.1467
Mean Dice	**Mean Recall**	**Mean Precision**
0.1467	0.1484	0.1514
Boundary F1		
0.1467		

Class-wise IoU bar chart for visual comparison of segmentation performance across categories.



Color-coded overlap visualization, using green (TP), red (FP), and blue (FN) to qualitatively assess prediction accuracy.

Overlap Visualization (Green=TP, Red=FP, Blue=FN)

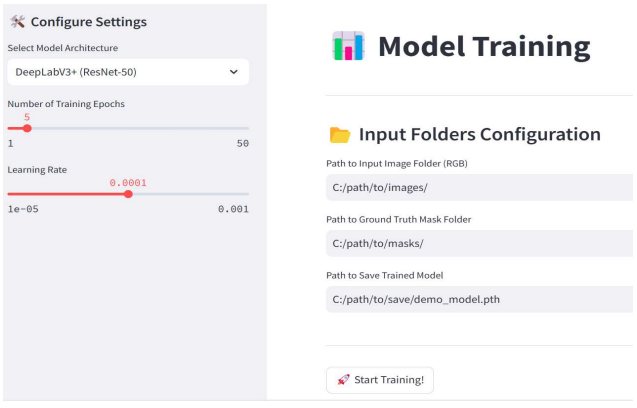


Overlap Map

This tool provided rapid feedback on model behavior, enabling efficient model comparison, metric-driven refinement, and identification of error patterns across test samples.

5.3 Model Trainer App

The Model Trainer App provides a simple interface to configure and train segmentation models directly through Streamlit. Users can select the model architecture (e.g., DeepLabV3+ with ResNet-50), set training parameters (epochs and learning rate), and specify directories for input images, masks, and model output.



The app automatically handles data preprocessing—resizing inputs, converting RGB masks to class indices using a predefined mapping, and batching the data for training. Once training is complete, the model is saved to the specified path for future inference or evaluation.

While lightweight in design, the app serves its purpose of confirming that the model pipeline functions end-to-end, and helps validate training configurations through visual logs and saved checkpoints.

5. Results

DeepLabV3 Performance Evaluation:


```
{
  "loss": 0.4547143042617038,
  "mean_iou": 0.19491489231586456,
  "pixel_acc": 0.6218803524971008,
  "mean_recall": 0.39099863171577454,
  "mean_precision": 0.30772536993026733,
  "f1_score":
  "mean_f1": 0.2770271301269531,
  "mean_dice": 0.2770271301269531,
  "bf_score": 0.2770271301269531,
```


High-performing classes (IoU > 0.8):
The model achieved a mean IoU of 0.19 and pixel accuracy of 62.18%, indicating limited segmentation accuracy. Performance was particularly poor on rare and fine-grained classes, with several categories showing extremely low or zero IoU scores. These findings highlight the limitations of the baseline DeepLabV3 architecture, underscoring the need for more robust models and targeted training enhancements to better handle the complexity of urban scenes.

DeepLab V3+ Resnet 50 Performance Evaluation:


```
{
  "loss": 0.06953168307256123,
  "mean_iou": 0.38949742913246155,
  "pixel_acc": 0.9504124522209167,
  "mean_recall": 0.46915170550346375,
  "mean_precision": 0.5304993987083435,
  "mean_f1": 0.47310715913772583, ,
  "mean_dice": 0.47310715913772583,
  "bf_score": 0.47310715913772583
}
```


The performance evaluation of DeepLabV3+ with ResNet-50 backbone demonstrates a significant improvement over the baseline model. The model achieved a mean IoU of 0.3895 and

a high pixel accuracy of 95%, indicating strong segmentation capability. The mean F1 score, Dice coefficient, and Boundary F1 score all reached approximately 0.473, reflecting better alignment between predictions and ground truth. Additionally, the balance between recall (0.4691) and precision (0.5305) suggests reliable detection with fewer false positives, while the low loss value (0.0695) confirms stable convergence during training.

Segmentation Transformer Performance Evaluation:


```
{
  "loss": 0.4038801786822315,
  "mean_iou": 0.09083440899848938,
  "pixel_acc": 0.8260788321495056,
  "mean_recall": 0.11086513847112656,
  "mean_precision": 0.21368317306041718,
  "mean_f1": 0.12182522565126419,
  "mean_dice": 0.12182523310184479,
  "bf_score": 0.12182522565126419
}
```


The Segmentation Transformer model underperformed significantly compared to the CNN-based architectures. It achieved a mean IoU of only 0.0908 and a pixel accuracy of 82.6%, indicating limited segmentation quality. Overlap-based metrics, including F1 score, Dice, and Boundary F1, were all around 0.12, while the recall (0.11) and precision (0.21) further highlight the model's struggle to detect and correctly classify objects. Despite the transformer's potential for global context modeling, the high loss value (0.4039) suggests ineffective learning, possibly due to underfitting or insufficient training adaptation for dense prediction tasks.

6. Conclusion and Limitations

This study evaluated DeepLabV3, DeepLabV3+ with ResNet-50, U-Net, and a Segmentation Transformer for semantic segmentation on the A2D2 dataset, aiming to support urban scene understanding in autonomous driving. While the Vision Transformer shows potential in modeling global context, the DeepLab models strike a better balance between efficiency and localized detail. However, all models require further enhancement for deployment readiness.

Key findings underscore the importance of training at full resolution (1920×1208) to capture fine-grained details like pedestrians, and extending training beyond a single epoch, ideally to 50+ epochs,to ensure convergence and improve metrics such as mIoU and F1 score, especially for rare classes. Learning rate scheduling proved beneficial for optimization, though the transformer's high computational demand calls for careful resource management.

Moreover, architectural complexity alone does not guarantee improved performance. Gains were more strongly linked to hyperparameter tuning, class balancing, and data augmentation. Particularly for rare objects such as bicycles, the lack of labeled samples hindered generalization, highlighting the need for dataset enrichment through synthetic data or oversampling.

Despite the strengths of the training pipeline, several limitations were encountered:

- High Computational Cost: Training with full-resolution inputs, longer epochs, and learning rate schedules significantly increased GPU memory consumption and training time, posing scalability challenges.

- Model Architecture vs. Performance: Transitioning to deeper or more complex architectures did not consistently yield better results, suggesting that hyperparameter tuning plays a more critical role than model depth alone.

- Class Imbalance & Data Scarcity: Rare object classes remained underrepresented, limiting model accuracy in those categories. Addressing this issue requires additional annotated samples or targeted augmentation strategies.

These insights emphasize the trade-offs between accuracy, efficiency, and data quality, all of which must be carefully balanced to advance real-world, scalable segmentation systems for autonomous driving applications.

8. Bibliography

[1] Chen, L.-C., Papandreou, G., Schroff, F., & Adam, H. (2017). Rethinking Atrous Convolution for Semantic Image Segmentation. arXiv preprint arXiv:1706.05587.
[2] Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., & Adam, H. (2018). Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. ECCV.
[3] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., & Houlsby, N. (2020). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. ICLR.
[4] Geyer, J., et al. (2020). A2D2: Audi Autonomous Driving Dataset. Available at http://www.a2d2.audi.