Semantic Segmentation in Autonomous Systems

Vaibhav Parikh (1217458)
Lakehead University
Thunder Bay, Canada
vparikh5@lakeheadu.ca

Hitarth Panchal (1214537)
Lakehead University
Thunder Bay, Canada
hpancha5@lakeheadu.ca

Saurav Patel (1220888)
Lakehead University
Thunder Bay, Canada
spatel184@lakeheadu.ca

Jatin Patel (1229894)
Lakehead University
Thunder Bay, Canada
jpatel136@lakeheadu.ca

***Abstract*— in this paper we focus on the implementation of semantic segmentation model for the road environments using the CityScapes dataset. CityScapes of roads of germany dataset is a large-scale dataset that provides high-quality pixel-level annotations for urban scenes, making it an perfect choice for that task. Our approach involves developing Fully Convolutional Neural Network (FCN) which segments each pixel of the image into eight different classes: void (background), flat, construction, object, nature, sky, human, and vehicle.**

Keywords*— *Semantic Segmentation, CityScapes Dataset, Fully Convolution Neural Network (FCN), ResNet50

I. INTRODUCTION

Semantic segmentation is critical task in computer vision specially for the autonomous systems, because it involves classifying of the each pixel in image into the predefined categories. This capability is important for the autonomous vehicles for navigation of their environment effectively. Further more, By segmenting road scenes and autonomous systems can identify various objects such as vehicles, pedestrians, and drivable regions which is extremely important for the safe and efficient driving. We focus on the implementation of semantic segmentation model for the road environments using the CityScapes dataset. CityScapes of roads of germany dataset is a large-scale dataset that provides high-quality pixel-level annotations for urban scenes, making it an perfect choice for that task. Our approach involves developing Fully Convolutional Neural Network (FCN) which segments each pixel of the image into eight different classes: void (background), flat, construction, object, nature, sky, human, and vehicle. To further enhance the performance of out model, we fine-tuned it using on ImageNet pre-trained weights with the ResNet50 model. Various data augmentation techniques were also used to reduce overfitting and improve the output of the dataset. This model was trained for 10 epochs on a GPU with a batch size of 48 Due to which we got the accuracy of 85%. For the training process, we used cross-entropy loss for loss criterion and Stochastic Gradient Descent (SGD) with weight decay and momentum for the weight updates. Furthermore, polynomial learning rate scheduler was used to reduce the learning rate progressively during training. This report details the previous work, methodology, implementation, results, and implications of our approach to semantic segmentation of road environments.

II. RELATED WORK

P In realm of autonomous driving, real-time semantic segmentation plays important role in the environment perception. Various studies have the advanced algorithms and models to enhance these systems' performance, accuracy, and efficiency. One notable study, "Design and Evaluation of a Real-Time Semantic Segmentation System for Autonomous Driving,"[1] evaluates algorithms across different hardware platforms which achieving impressive accuracy and frame rates. It focuses on optimizing edge devices of crucial for resource-constrained environments[2]. Another significant contribution is "Fast Semantic Image Segmentation for Autonomous Systems," which proposes a dual-branch network with attention-based fusion[3]. This method outperforms existing techniques, showing substantial performance improvements in both autonomous driving and aerial image analysis[4]. "A Real-Time Semantic Segmentation Algorithm Based on Improved Lightweight Network" introduces LightSeg, a model integrating dilated convolutions and attention mechanisms and adaptive pooling into MobileNet v2. LightSeg achieves 73.8% mIoU accuracy and 49.5 FPS on a single GPU that making it ideal for real-time applications in smart cars[5][6]. "S³M-Net: Joint Learning of Semantic Segmentation and Stereo Matching for Autonomous Driving" shows framework that combines semantic segmentation and stereo matching[7][8]. By sharing features and enforcing structural consistency with S³M-Net enhances both tasks' performance[9][10], demonstrating the advantages of multitask learning. "Real-Time Semantic Segmentation on Edge Devices with Nvidia Jetson AGX Xavier" evaluates segmentation algorithms on the Nvidia Jetson AGX Xavier platform which achieving high accuracy and speed, crucial for real-time autonomous vehicle applications[11][12][13]. "L2-LiteSeg: A Real-Time Semantic Segmentation Method for End-to-End Autonomous Driving" optimizes network architecture and incorporates attention mechanisms of maintaining high accuracy and speed with a small model size[14][15]. "Deep Learning Based Segmentation Approach for Automatic Lane Detection in Autonomous Vehicle" utilizes SegNet and U-Net models, trained on the tuSimple dataset, and evaluates them based on MSE, average miss rate, and computation time to identify the optimal model for real-time lane detection[16][17][18]. "A Semantic Segmentation Algorithm Based on Improved Attention Mechanism" introduces GSANet, which enhances boundary accuracy and real-time performance using global and selective attention mechanisms[19]. "HCINet: A Hierarchical Approach of Context Integration in Real-time Semantic Segmentation for Autonomous Driving" balances accuracy and efficiency with a multi-pathway architecture, feature fusion, and attention mechanisms[20][21]. Lastly, "An Autonomous Navigation Approach based on Bird’s-Eye View Semantic Maps" offers a camera-based system that generates Bird's-Eye view maps through real-time semantic segmentation, providing a cost-effective alternative to LiDAR sensors[22][23]. These studies collectively contribute to advancing real-time semantic segmentation systems for autonomous driving, addressing efficiency, accuracy, and application-specific needs.

III. METHODOLOGY

A. *CityScapes Dataset*

The CityScapes dataset was chosen for this project because of comprehensive and high-quality pixel-level annotations of urban scenes. Which includes a diverse set of images collected from 50 cities across Germany, which captured various urban scenarios in the different seasons (spring, summer, and fall). The dataset is divided into three subsets: training (2975 images), validation (500 images), and test (1525 images). Furthermore, we have also used validation data as our testing data because we were unable to acquire ground truth labels for the test data.

However, each image annotated with the polygonal shapes that aid in the generating dense a segmentation maps, providing high-quality pixel-level labels. This detailed annotation is essential for the training models to for accurately segment and for classify different regions within an image.

For this project, we focused on segmenting pixels into eight classes: void (background), flat, construction, object, nature, sky, human, and vehicle. Training data was collected from 18 cities in Germany: aachen, bochum, bremen, cologne, darmstadt, dusseldorf, erfurt, hamburg, hanover, jena, krefeld, monchengladbach, strasbourg, stuttgart, tubingen, ulm, weimar, and zurich. Whereas the validation set data was collected from different cities than the mentioned above: frankfurt, lindau, and munster. The dataset features has a large number of dynamic objects which is varying scene layouts and diverse backgrounds. Which contribute to its robustness and complexity.

Adding a graph of pixel distribution for eight classes will provide valuable insight into dataset's composition and model's performance. The subclasses inside category levels, according to dataset documentation are shown below

We believe it's meaningless to train and build a deep-learning model that can predict subcategories instead of top-level classes (filled bullet) shown above. There are several reasons for that. The amount of pixels for some subclasses is much less, in fact, the whole distribution is very skewed, resulting in a model that is only good at detecting a handful of classes. Secondly, the perception stack of self-driving cars doesn't need such precise information. It should care about which region is a safe drivable surface and which obstacles to avoid. This information is easily conveyed even if we train the model in high-level categories. This reduces the complexity of the deep learning model by a bit, slightly improving the forward pass runtime and post-processing time.

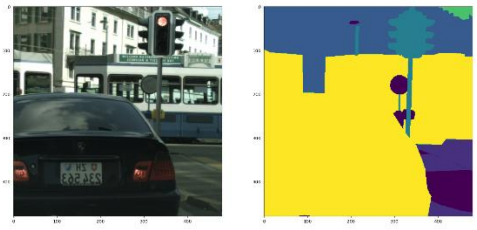
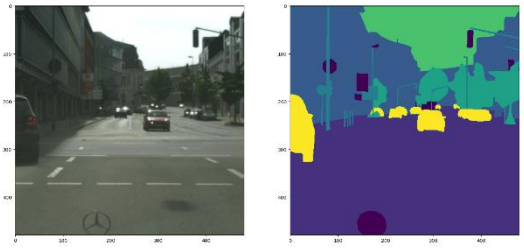


Figure1:- Image with annotation

Figure2:- Image1

Figure3: Image2



Firgure4:-Image 2 with annotation

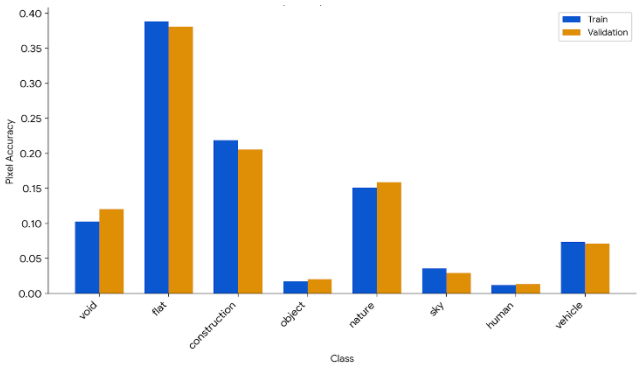


Figure5:- Different classes in image

B. *Training Data Pipeline*

The training data pipeline consists of several key stages, from loading and preprocessing data to feeding it into the model. This pipeline ensures that the data is in the correct format and is augmented appropriately to enhance model performance

C. *Data Loading*

Dataset Preparation: The CityScapes dataset was organized into directories corresponding to the training,

validation, and test subsets. Each image and its associated annotation were paired together.

Data Loader: A data loader was implemented to efficiently load images and annotations in batches. This helped in managing memory usage and speeding up the training process.

D. Data Augmentation

Random Resize:- Images were randomly resized to a fixed size between 520 to 640 pixels square shape.

Horizontal Flipping:- Images were randomly flipped horizontally with a probability of 0.5 to enhance the model's robustness to variations in image orientation.

Random Cropping:- 480x480 crop is randomly selected from the output of the previous step.

Pixel Normalization: Pixel values were normalized based on the mean and standard deviation of the ImageNet dataset to standardize inputs. This step was crucial for ensuring that the model training process was stable and efficient.

Batch Size: A batch size of 48 was used during training to balance memory usage and computational efficiency.

Shuffling: Data was shuffled before each epoch to ensure that the model was trained on a diverse set of examples in each epoch.

E. Data Pipeline Workflow

1) Load Images and Annotations: The data loader reads images and their corresponding mask images from the dataset.

2) Apply Augmentations: Data augmentation techniques are applied to the images and masks to generate a diverse set of training samples.

3) Normalize Data: Pixel values are normalized to prepare the data for model input.

4) Batch Formation: Data is organized into batches and shuffled to ensure varied training examples for each epoch.

This data pipeline setup ensures that the model receives well-prepared and diverse input, which is crucial for effective training and generalization. Apply the same transformation on the mask as well..

IV. FCN Architecture

Conversion to Fully Convolutional Layers: The fully connected layers of ResNet50 were converted to convolutional layers to enable pixel-wise prediction. This conversion allows the network to output a spatial heatmap rather than a single classification score, making it suitable for semantic segmentation tasks.

A. Output Layer

Softmax Activation: The final layer of the network applies a softmax activation function to produce a probability distribution over the eight classes for each pixel. This allows the model to output the most likely class for each pixel in the image.

The FCN model architecture was chosen for its ability to perform end-to-end learning and produce dense predictions that are essential for pixel-wise classification tasks like semantic segmentation. The use of ResNet50 as the backbone provided a strong foundation for feature extraction, while the fully convolutional design enabled the network to handle varying input sizes and produce detailed segmentation maps.

B. Architecture

	Input Channels	Output Channels	Filter size	Padding
Conv2D (s=2)	3	64	7, 7	3, 3
MaxPool2D(s=2)	64	64	3, 3	1, 1
Layer1				
BN1	64	256		
BN2	256	256		
BN3	256	256		
Layer2				
BN1	256	512	BN1 halves spatial dimensions here	
BN2	512	512		
BN3	512	512		
BN4	512	512		
Layer3				
BN1	512	1024		
BN2	1024	1024		
BN3	1024	1024		
BN4	1024	1024		
BN5	1024	1024		
BN6	1024	1024		
Layer4				
BN1	1024	2048		
BN2	2048	2048		
BN3	2048	2048		
FCNHead				
Conv2D	2048	512	3, 3	1, 1
Conv2D	512	8	1, 1	
Interpolate (Bilinear) to Input's Spatial Dimensions				

The first bottleneck module (BN1) of each layer.

Input	Downscale(1x1 Conv)	Retain (3x3 Conv)	Upscale (1x1 Conv)	Upscale (1x1 Conv) Input to match channels as the previous layer output	Add Outputs of the previous two layers

The rest of the bottleneck modules of each layer

Input	Downscale(1x1 Conv)	Retain (3x3 Conv)	Upscale(1x1 Conv)	Add Input and output of the previous layer

C. Training Process

The training process involves preparing the model, defining the loss function and optimizer. Furthermore, setting up the learning rate scheduler and training model over several epochs.

D. Loss Function

Cross-Entropy Loss: The cross-entropy loss function is used as criterion for training. This loss function well-suited for

multi-class classification tasks and helps in optimizing model to correctly classify each pixel into one of the eight classes.

E. Optimizer

Stochastic Gradient Descent (SGD): The SGD optimizer is used with weight decay and momentum to update model weights.

Weight Decay: A regularization technique which penalizes large weights that prevent overfitting.

Momentum: Accelerates convergence by considering previous update's direction and magnitude, which helps in smoothing of the optimization path.

F. Learning Rate Scheduler

Polynomial Learning Rate Scheduler: A polynomial learning rate scheduler is also used to progressively reduce learning rate during the training. This approach helps in stabilization of the training process and avoiding overshooting of the optimal solution.

G. Training Parameters

Epochs: The model was trained for the 10 epochs. Each epoch involved passing of the entire training dataset through the network, updating the model weights based on computed loss.

Batch Size: A batch size of 48 was used to balance memory usage and computational efficiency. The larger batch sizes allow for more stable gradient estimates. However they require more memory.

Hardware: The training process was carried out on a GPU to leverage parallel computation capabilities which significantly speeding up the training process

H. Training Workflow

Initialization: The model, loss function, optimizer, and learning rate scheduler were initialized.

Epoch Loop: For each epoch:

Batch Processing: The training dataset was divided into batches of 48 images.

Forward Pass: Each batch was passed through a network to compute predicted segmentation maps.

Loss Calculation: The cross-entropy loss was calculated based on the difference between the predicted and the true segmentation maps.

Backward Pass: Gradients were computed through backpropagation.

Weight Update: The optimizer updated the model weights based on the computed gradients and learning rate.

Learning Rate Adjustment: The learning rate scheduler adjusted the learning rate according to the polynomial decay schedule.

Evaluation: After each epoch, the model was evaluated on the validation (used as test) dataset to monitor its performance and adjust training strategies if necessary.

This structured training process ensured that the model was efficiently trained while minimizing overfitting and converging toward an optimal solution.

V. RESULT

A. Performance Analysis

The model achieved accuracy of 85% on validation dataset. This result demonstrates that the Fully Convolutional Neural Network (FCN) with ResNet50 can effectively segment urban road scenes into predefined classes. The use of pre-trained weights on ImageNet likely contributed to the model's strong performance by providing a solid foundation for the feature extraction.

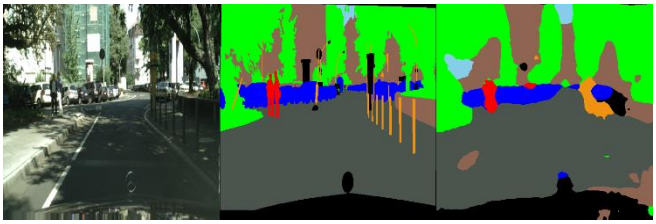


Figure 6:- Expected and predicted output of image 45

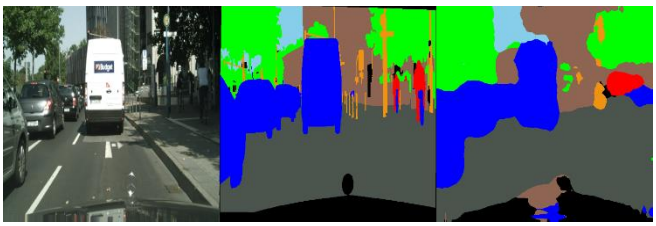


Figure 7:- Expected and predicted output of image 207

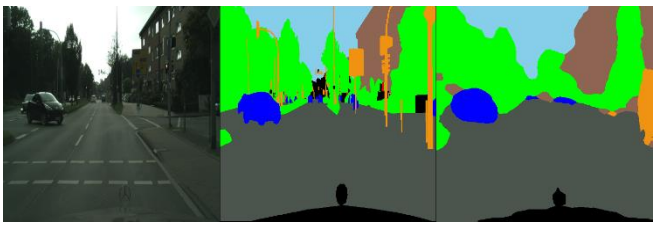


Figure 8:- Expected and predicted output of image 498

B. Strengths

High Pixel Accuracy: The achieved accuracy indicates that model is capable of accurately classifying a large proportion of the pixels, which is very important for the applications such as autonomous driving. Where precise segmentation is necessary for object recognition and scene understanding.

Diverse Data Handling: The use of various data augmentation techniques, such as random cropping, horizontal flipping, and color jittering, helped the model generalize better by exposing it to the different variations of the input data.

Efficient Training Process: Leveraging GPU of the training enabled the handling of large batches and complex computations and thereby speeding up the training process which allowed for the more epochs within a reasonable timeframe.

C. Limitations

Limited Validation: The initial approach used in the validation dataset as the test dataset, which may not be provide a fully unbiased evaluation of the model's performance. Re-evaluating with the actual test dataset is a necessary for a more accurate assessment.

Class Imbalance: Some classes may have fewer training samples than others, which could lead to lower accuracy for those classes. This can be mitigated by techniques such as class weighting or oversampling.

Overfitting: Despite data augmentation, the model might still overfit to the training data, especially given the relatively small number of epochs (10). Further experimentation with regularization techniques and more epochs might help.

VI. Future Work

Re-evaluate on Test Dataset: As mentioned, recalculating of the accuracy on a actual test dataset is crucial for obtaining the true measure of model performance.

Hyperparameter Tuning: Explore the different hyperparameters, such as a learning rate, batch size, and number of epochs, to optimize the model further.

Advanced Architectures: Experiment with more the advanced architectures like DeepLab, U-Net, or PSPNet, which have shown superior performance in the semantic segmentation tasks.

Additional Data Augmentation: Implement of the more sophisticated data augmentation techniques, such as random rotations, scaling, and perspective transformations, to enhance the model's robustness.

Class Balancing: Address class imbalance through techniques like class weighting, oversampling underrepresented classes, or using synthetic data generation.

VII. Conclusion

The project demonstrated the effectiveness of using a Fully Convolutional Network with a ResNet50 backbone for semantic segmentation of the road environments. With an 85% pixel accuracy, the model shows promise for the real-world applications. However, there are several avenues for of the improvement, particularly in the terms of validation, handling class imbalance, and exploring more of the advanced model architectures. Future work focusing on these areas could lead to even more accurate and robust segmentation models, contributing significantly to the development of autonomous systems.

VIII. Appendix

Code

Hyperparameter Settings

1. Training Hyperparameters:

Learning Rate: 0.01

Batch Size: 48

Epochs: 10

Optimizer: SGD with momentum=0.9, weight_decay=0.0001

Learning Rate Scheduler: Polynomial decay

2. Data Augmentation Parameters (Edit Details):

Random Crop: (480, 480)

Horizontal Flip Probability: 0.5

Hardware and Software Specifications

1. Hardware (Kaggle Notebook hosted on a Virtual Machine):

CPU: Intel(R) Xeon(R) CPU @ 2.00GHz

GPU: NVIDIA P100 (16GB)

RAM: 16 GB

2. Software:

Libraries: PyTorch 2.0, Torchvision, Numpy, Pandas, PIL, Matplotlib.

References

[1] *Design and Evaluation of a Real-Time Semantic Segmentation System for Autonomous Driving*. (2024, March 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/10511680

[2] *Fast semantic image segmentation for autonomous systems*. (2022, October 16). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/9897582

[3] *A Real-Time semantic segmentation algorithm based on improved lightweight network*. (2020, December 6). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/9378857

[4] *S$^{3}%%STRING%%#60;/tex-math>M-Net: Joint Learning of Semantic Segmentation and Stereo Matching for Autonomous Driving*. (2024, February 1). https://ieeexplore.ieee.org/document/10412168

[5] *Real-Time Semantic Segmentation on Edge Devices with Nvidia Jetson AGX Xavier*. (2022, October 26). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/9954835

[6] *L2-LiteSEG: a Real-Time semantic segmentation method for End-to-End autonomous driving*. (2023, August 21). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/10327554

[7] *Deep learning based segmentation approach for automatic lane detection in autonomous vehicle*. (2023, October 18). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/10331835

[8] *A Road scene Semantic Segmentation algorithm based on improved BISENET V2*. (2024, March 29). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/10581484

[9] *HCINET: A Hierarchical Approach of context integration in real-time semantic segmentation for autonomous driving*. (2023, December 14). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/10440773

[10] *An Autonomous Navigation Approach based on Bird's-Eye View Semantic Maps*. (2023, August 22). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/10242482

[11] *Design and Evaluation of a Real-Time Semantic Segmentation System for Autonomous Driving*. (2024b, March 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/10511680

[12] *Expert-driven rule-based refinement of semantic segmentation maps for autonomous vehicles*. (2023, June 4). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/10186738

[13] *Deep Learning based Dynamic-to-Static Image Translation for Accurate Localization of Autonomous Driving Vehicles*. (2023, August

27). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/10401764

[14] *Road Semantic Segmentation Oriented dataset for autonomous driving*. (2020, November 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/9277270

[15] *Deep learning based segmentation approach for automatic lane detection in autonomous vehicle*. (2023b, October 18). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/10331835

[16] *Speed bump recognition for autonomous vehicles based on semantic segmentation*. (2021, May 14). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/9446038

[17] *L2-LiteSEG: a Real-Time semantic segmentation method for End-to-End autonomous driving*. (2023b, August 21). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/10327554

[18] *Semantic Segmentation under Severe Imaging Conditions*. (2019, December 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/8945923

[19] *SSF-MOS: Semantic Scene Flow Assisted Moving Object Segmentation for autonomous vehicles*. (2024). IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/document/10399869

[20] *Visual Perception stack for autonomous vehicle using semantic segmentation and object detection*. (2021, August 27). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/9563527

[21] *A LIDAR Semantic Segmentation Framework for the Cooperative Vehicle-Infrastructure System*. (2023, October 10). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/10333790

[22] *Visual localization of intersections on autonomous vehicles based on HD Map*. (2023, October 17). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/10316841

[23] *Water Segmentation with Superior Guidance and Aligned Fusion Strategy for Unmanned Surface Vehicles in Maritime Environment*. (2022, October 28). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/9987160