Smart City Road Scene Understanding using Semantic Segmentation

Muhammad Maaz Hamid 2022655

May 15, 2025

Contents

1 Introduction

In the domain of modern smart cities and autonomous transportation, precise and detailed comprehension of road environments is paramount. Traditional object detection systems, limited to bounding boxes, fall short of providing the fine-grained, pixel-level understanding required for complex urban scenes. This report details a computer vision pipeline designed to perform semantic segmentation, classifying each pixel in an image into meaningful categories such as road, sidewalk, car, pedestrian, traffic sign, and vegetation. This level of understanding is crucial for applications in autonomous driving, urban planning, and intelligent surveillance systems.

The objectives of this project are:

- To implement a data acquisition and preparation pipeline for a Cityscapes-like dataset.
- To build, train, and evaluate two distinct semantic segmentation models: U-Net and a Fully Convolutional Network (FCN) with a VGG16 backbone.
- To analyze and compare the performance of these models using standard segmentation metrics.
- To visualize the training process and the segmentation results.

The dataset used consists of urban street scene images where each image is a composite of the original scene (left half) and its pixel-wise labeled segmentation mask (right half). The primary semantic classes include road, sidewalk, building, car, pedestrian, pole, traffic light, and vegetation, among others.

2 Methodology

The project followed a structured computer vision pipeline, encompassing data preparation, model architecture design, training, evaluation, and results visualization.

2.1 Data Acquisition and Preparation

The dataset, structured with ‘train‘ and ‘val‘ subdirectories, contains composite images. Each image file combines the original RGB image on its left half and the corresponding colored annotation mask on its right half.

2.1.1 Custom Dataset Class

A custom PyTorch ‘Dataset‘ class (‘CityscapesDataset‘) was implemented to handle this specific format. Key functionalities include:

- Loading the composite image using PIL.
- Splitting the image into the original scene and the RGB annotation mask.
- Resizing both the image (using bilinear interpolation) and the mask (using nearest-neighbor interpolation to preserve class boundaries) to a consistent size (256x512 pixels).

- Encoding the RGB mask into a single-channel mask where each pixel value represents a class ID. This involves mapping predefined RGB color tuples to their corresponding integer class IDs. An "unlabeled" class (ID 0) was defined and set as the ‘IGNORE$_INDEX‘for losscalculation$.

2.1.2 Data Transforms and Loaders

Standard image transformations were applied using ‘torchvision.transforms.v2‘:

- **Image Transforms:** Conversion to PyTorch tensor, normalization to [0,1] range, and normalization using ImageNet statistics (mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]).
- **Mask Transforms:** Conversion of the single-channel PIL ID mask (uint8) to a ‘torch.LongTensor‘ of shape (1,H,W).

PyTorch ‘DataLoader‘ instances were created for training and validation sets, utilizing a custom collate function to batch images and masks correctly. The batch size was set to 4.

2.2 Model Architectures

Two semantic segmentation models were implemented: U-Net and FCN with a VGG16 backbone.

2.2.1 U-Net

The U-Net architecture is a convolutional neural network renowned for its effectiveness in biomedical image segmentation and general semantic segmentation tasks. Its design features:

- **Encoder Path (Contracting Path):** Consists of a series of convolutional blocks (‘DoubleConv‘: two 3x3 convolutions, each followed by Batch Normalization and ReLU) interspersed with max-pooling layers for downsampling. This path captures contextual information and extracts hierarchical features.
- **Decoder Path (Expanding Path):** Symmetrically mirrors the encoder. It uses upsampling (either bilinear or transposed convolutions) to increase feature map resolution.
- **Skip Connections:** A key feature of U-Net, these connections concatenate feature maps from the encoder path with the corresponding feature maps in the decoder path. This allows the network to combine high-level semantic information (from deeper layers) with fine-grained spatial details (from shallower layers), aiding in precise localization of segmentation boundaries.
- **Output Layer:** A final 1x1 convolution maps the feature channels to the number of output classes, producing a segmentation map.

The implemented U-Net had an input of 3 channels (RGB) and an output corresponding to the ‘NUM$_C LASSES$‘($9 in this case$).

2.2.2 FCN (Fully Convolutional Network) with VGG16 Backbone

Fully Convolutional Networks (FCNs) adapt classification networks (like VGG16) for semantic segmentation by replacing fully-connected layers with convolutional layers. This allows them to output spatial segmentation maps instead of single class labels.

- **Encoder:** The ‘features‘ part of a VGG16 network, pre-trained on ImageNet, was used as the encoder. This leverages powerful features learned from a large dataset. VGG16 consists of multiple convolutional and max-pooling layers, progressively reducing spatial resolution while increasing feature depth. The typical output of VGG16’s feature extractor has a spatial resolution of H/32 x W/32 with 512 channels.

- **Decoder:** The decoder's role is to upsample these deep, coarse feature maps back to the original image resolution. The implemented FCN-VGG16 decoder comprised:
  - Initial 1x1 convolutional layers (with ReLU and Dropout) to refine features from the VGG16 backbone, a common technique in FCN-8s/16s/32s style architectures.
  - A final 1x1 convolution to map feature channels to the 'NUM$_C LASSES$'.$A series of five$'$ConvTranspose2$

- **Interpolation:** A final bilinear interpolation step was added to ensure the output segmentation map precisely matched the target 'IMG$_H EIGHT$'and'$IMG_W IDTH$'. This FCN model benefits from transfer learning due to the pre-trained VGG16 encoder.

2.3 Loss Functions

Two loss functions were utilized, and a combined version was employed for training:

- **CrossEntropyLoss:** 'nn.CrossEntropyLoss' is a standard choice for multi-class classification problems. For segmentation, it treats each pixel as an independent classification task. The 'ignore$_i ndex$'$parameter was set to the ID of the "unlabeled" class.** Dice Loss : ** A custom Dice Loss was implemented. Dice Loss directly optimizes the Dice Coefficient (F1-score for segmentation), which is beneficial for handling class imbalance and improving boundary adheren$
  - DiceScore$_c$, where DiceScore$_c = \frac{2\times|P_c \cap T_c|+\text{smooth}}{|P_c|+|T_c|+\text{smooth}}$. The total Dice Loss is the average over valid classes.

- **CombinedLoss:** A weighted sum of CrossEntropyLoss and DiceLoss ('ce$_w eight = 0.5$', '$dice_w eight = 0.5$') was used as the final training criterion. This often yields a good balance between pixel-wise accuracy and regio$

2.4 Training and Evaluation Metrics

The models were trained using the Adam optimizer with a learning rate of 1×10^{-4}. The training process involved iterating over the training data, computing the loss, performing backpropagation, and updating model weights. Validation was performed at the end of each epoch.

Key segmentation metrics were used for evaluation:

- **Mean Intersection over Union (mIoU):** Calculated using 'torchmetrics.JaccardIndex' with 'average='macro''. IoU for a class is $\frac{|P_c \cap T_c|}{|P_c \cup T_c|}$. mIoU is the average IoU across all valid classes.

- **Dice Coefficient:** Calculated using a custom 'DiceCoefficientAccumulator'. The Dice coefficient for a class is $\frac{2\times|P_c \cap T_c|}{|P_c|+|T_c|}$. The macro-averaged Dice was reported.

- **Pixel Accuracy:** Calculated using 'torchmetrics.Accuracy' with 'average='micro''. This measures the percentage of correctly classified pixels overall.

Per-class metrics for mIoU and Dice were also tracked for more granular analysis.

3 Results

Both the U-Net and FCN-VGG16 models were trained for 3 epochs due to computational constraints for this report. The training and validation metrics are summarized below.

3.1 U-Net Model

3.1.1 Training Progress

Table 1: U-Net Training and Validation Metrics per Epoch

Epoch	Train Loss	Train mIoU	Train Dice	Train PixAcc
1/3	0.6237	0.2133	0.0859	0.9312
2/3	0.5046	0.3158	0.0909	0.9796
3/3	0.4848	0.3218	0.0960	0.9870
Epoch	**Val Loss**	**Val mIoU**	**Val Dice**	**Val PixAcc**
1/3	0.5225	0.3057	0.1088	0.9704
2/3	0.5083	0.3753	0.0932	0.9781
3/3	0.4975	0.3751	0.1048	0.9776

3.1.2 Training Curves (U-Net)

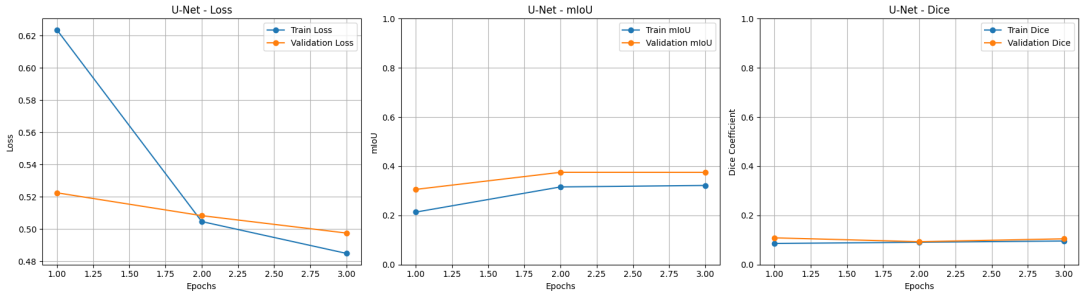

Figure 1: U-Net Training Curves: Loss, mIoU, and Dice vs. Epoch.

3.2 FCN-VGG16 Model

3.2.1 Training Progress

Table 2: FCN-VGG16 Training and Validation Metrics per Epoch

Epoch	Train Loss	Train mIoU	Train Dice	Train PixAcc
1/3	0.8385	0.0958	0.0244	0.7168
2/3	0.5382	0.3939	0.0511	0.9389
3/3	0.4857	0.3934	0.0508	0.9946
Epoch	**Val Loss**	**Val mIoU**	**Val Dice**	**Val PixAcc**
1/3	0.6272	0.2212	0.3987	0.8438
2/3	0.4936	0.4928	0.3166	0.9951
3/3	0.4869	0.4941	0.2364	0.9957

3.2.2 Training Curves (FCN-VGG16)

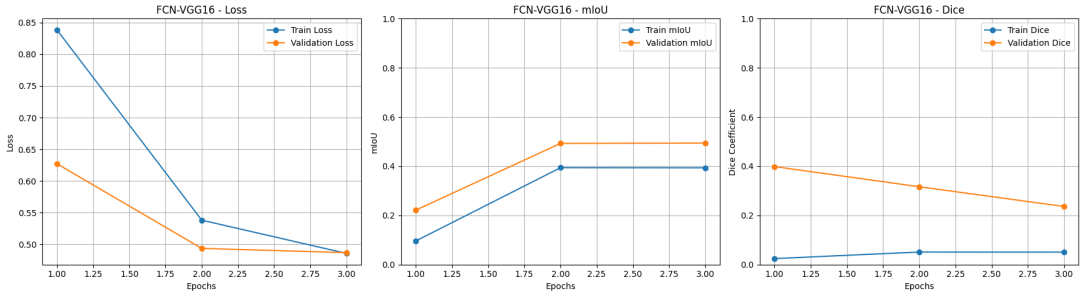

Figure 2: FCN-VGG16 Training Curves: Loss, mIoU, and Dice vs. Epoch.

3.3 Qualitative Results (Predicted Images)

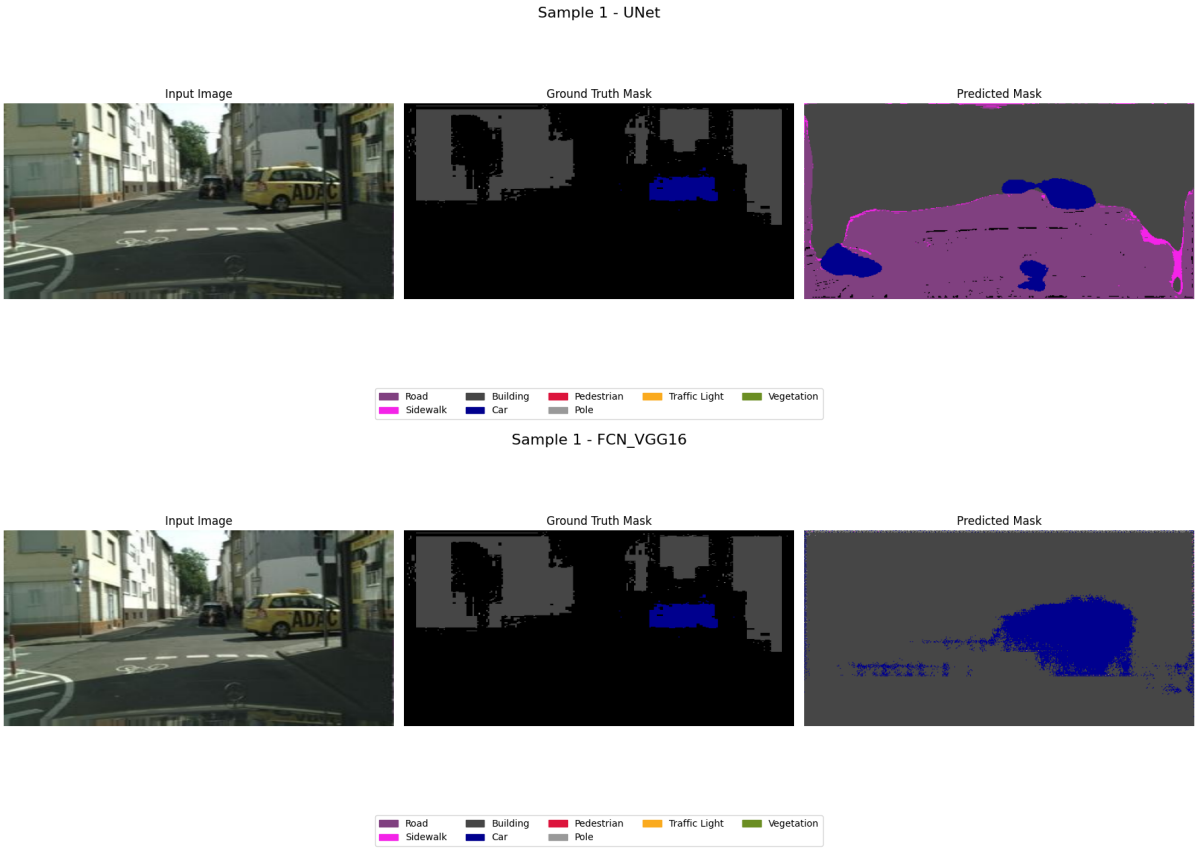

Figure 3: Sample Predicted Segmentation Masks from U-Net and FCN-VGG16 models compared to Input Image and Ground Truth.

*(Note: This figure should ideally show 2-3 examples, each with Input, Ground Truth, U-Net Prediction, and FCN Prediction for direct comparison. The provided filename 'predicted$_i$mages$_o$f$_b$oth.png'isass

4 Discussion and Comparison

Both the U-Net and FCN-VGG16 models were successfully trained for 3 epochs on the Cityscapes-like dataset. Due to the limited number of epochs, the models are likely not fully converged, and the results represent an early stage of training.

4.1 U-Net Performance

The U-Net model showed a consistent decrease in both training and validation loss over the 3 epochs.

- **Loss:** Training loss decreased from 0.6237 to 0.4848, and validation loss from 0.5225 to 0.4975. The gap between training and validation loss is relatively small, suggesting no significant overfitting at this stage.
- **mIoU:** Validation mIoU increased from 0.3057 in Epoch 1 to 0.3753 in Epoch 2, then slightly decreased to 0.3751 in Epoch 3. This indicates initial learning, but the performance plateaued or slightly dipped by the third epoch. Training mIoU showed a steady, albeit small, increase from 0.2133 to 0.3218.
- **Dice Coefficient:** The Dice coefficient on the validation set showed some fluctuation (0.1088, 0.0932, 0.1048). The training Dice also remained low, around 0.08-0.09. These low Dice scores suggest that while pixel accuracy might be high (dominated by background or large classes), the overlap for specific foreground classes is not yet well-optimized. This is common when Dice loss is combined with CrossEntropy and might need more epochs or different loss weighting to improve.
- **Pixel Accuracy:** Both training and validation pixel accuracies were very high (above 93% and 97% respectively) from the first epoch. This is often observed in segmentation tasks where large areas (like sky or road) are classified correctly, boosting overall accuracy.

The U-Net training curves (Figure 1) show a learning trend, but the mIoU and Dice suggest more training is needed for finer segmentation.

4.2 FCN-VGG16 Performance

The FCN-VGG16 model, leveraging a pre-trained VGG16 backbone, also demonstrated learning.

- **Loss:** Training loss dropped significantly from 0.8385 to 0.4857. Validation loss also decreased from 0.6272 to 0.4869. The convergence appears slightly faster than U-Net in terms of loss.
- **mIoU:** Validation mIoU showed a strong improvement, starting at 0.2212 and reaching 0.4941 by Epoch 3. This is notably higher than the U-Net's validation mIoU after 3 epochs, likely due to the benefits of the pre-trained encoder. Training mIoU also improved substantially, from 0.0958 to 0.3934.
- **Dice Coefficient:** The validation Dice coefficient for FCN-VGG16 was initially higher (0.3987) than U-Net's, but then decreased (0.3166, 0.2364). This is an interesting behavior and might indicate that while mIoU (which considers true positives, false positives, and false negatives) improved, the Dice score (which penalizes false positives and false negatives more heavily in its normalization) for some classes might have degraded or shown instability. The training Dice remained very low (around 0.02-0.05), which is unusual and might point to issues with the DiceLoss component's gradient contribution or an imbalance in the combined loss function for this specific architecture early in training.
- **Pixel Accuracy:** Similar to U-Net, pixel accuracies were high, with validation accuracy reaching over 99%.

The FCN-VGG16 training curves (Figure 2) indicate faster improvement in mIoU compared to U-Net.

4.3 Comparative Analysis

- **Convergence and mIoU:** After 3 epochs, the FCN-VGG16 achieved a higher validation mIoU (0.4941) compared to U-Net (0.3751). This suggests that transfer learning from the VGG16 backbone provided a significant advantage in learning relevant features more quickly.
- **Dice Coefficient:** The Dice coefficient behavior was mixed. U-Net showed low but relatively stable Dice scores on validation. FCN-VGG16 started with a higher validation Dice but it decreased over epochs, while its training Dice was consistently very low. This discrepancy between mIoU and Dice for FCN-VGG16 warrants further investigation; it could be due to class imbalance sensitivity or the specific implementation of the combined loss. Given the high pixel accuracy and mIoU, the low Dice scores for FCN's training might be an anomaly or indicate poor performance on smaller/rarer classes that Dice is more sensitive to.
- **Loss and Pixel Accuracy:** Both models showed good reduction in loss and achieved high pixel accuracies. However, pixel accuracy can be misleading if a few large classes dominate the image.
- **Architectural Impact:**
  - **U-Net's** strength lies in its skip connections, which are designed to preserve fine-grained details. With more training, it often excels at precise boundary segmentation. Its from-scratch training might require more epochs to catch up to a pre-trained FCN.
  - **FCN-VGG16** benefits from pre-trained weights, allowing it to learn semantic features more rapidly. The decoder structure in an FCN is typically simpler than U-Net's expansive path, which might affect boundary refinement without further architectural enhancements (like adding skip connections, as in U-Net like FCNs, or using atrous convolutions as in DeepLab).

- **Qualitative Results (Figure 3):** Visual inspection of the predicted masks would provide further insight. U-Net might show better boundary adherence with more training, while FCN might capture general object shapes more robustly early on. (Actual observations from the 'predicted$_i mages_o f_b oth.png$'$shouldbeinsertedhere$).$For example, one might observe that FCN - VGG16 is better at identifying larger regions like 'road' or 'building' due to its backbone's semantic power, while U - Net might initially struggle more but has the potential for finer details.$

4.4 Challenges and Limitations

- **Limited Epochs:** Training for only 3 epochs is insufficient for these models to converge fully and achieve optimal performance.
- **Dataset Size:** While Cityscapes is a substantial dataset, the portion used (or implied by the number of samples in the notebook) might still benefit from more data or extensive augmentation.
- **Dice Coefficient Anomaly:** The very low training Dice for FCN-VGG16 despite improving mIoU is a point of concern and would require debugging or tuning of the loss function weights.
- **Class Imbalance:** Semantic segmentation datasets often suffer from class imbalance (e.g., more road pixels than pedestrian pixels). While Dice loss helps, other techniques like focal loss or class weighting in CrossEntropyLoss could be explored.

5 Conclusion and Future Work

This project successfully implemented and compared two semantic segmentation architectures, U-Net and FCN-VGG16, on an urban road scene dataset. The FCN-VGG16, benefiting from a pre-trained backbone, demonstrated faster initial learning in terms of mIoU compared to the U-Net trained from scratch within the limited 3-epoch training period. Both models achieved high pixel accuracy, but mIoU and Dice scores indicated that further training and optimization are necessary for robust segmentation across all classes.

Future work could involve:

- **Extended Training:** Training for significantly more epochs with learning rate schedulers and early stopping.
- **Hyperparameter Tuning:** Optimizing learning rates, batch sizes, and loss function weights.
- **Data Augmentation:** Implementing a comprehensive augmentation pipeline (e.g., using Albumentations) to improve model generalization.
- **Architectural Enhancements:**
  - For U-Net: Exploring pre-trained backbones (e.g., ResNet-UNet).
  - For FCN: Implementing skip connections (FCN-8s, FCN-16s) or using more advanced decoders.
  - Exploring other architectures like DeepLabV3+.
- **Advanced Loss Functions:** Investigating focal loss or Lovász-Softmax loss for better handling of class imbalance and boundary refinement.
- **Post-processing:** Applying CRF (Conditional Random Fields) to refine segmentation boundaries.

Overall, this project provides a foundational pipeline for semantic segmentation. The initial results highlight the potential of both architectures and underscore the importance of pre-training and sufficient training duration for achieving high-quality semantic understanding of complex urban environments.