Road Segmentation with U-Net

Introduction

Problem to Be Solved

The goal of this program is to utilize a U-Net architecture for precise image segmentation involving roads and vehicles. The model developed should accurately detect vehicles within road environments and delineate their boundaries.

Relevance of Problem

The ability to detect where vehicles are on roads could significantly improve traffic flow. I am currently interning at the Department of Transportation and one major issue is managing the inflow and outflow of traffic at intersections on different freeways. Utilizing a model to leverage segmentation could allow engineers to act quicker when there are large build ups of traffic on freeways.

Plan to Solve Problem

To solve the problem, a pipeline will be developed that is capable of preprocessing, augmentation, training, prediction, and validation, and uses a U-Net for segmentation. This pipeline will be a deep learning solution to solve segmentation tasks for road and vehicle images.

Justification for a DL Solution

A random forest classifier modified for segmentation tasks was tested on the Road Segmentation dataset, and the accuracy score for validation was 0.6. This score was poor in correctly segmenting the cars on the roads in the images, no matter which hyperparameters were changed. In addition to the validation score, the random forest made predictions on the masks. As seen below, the mask predictions were not that accurate as the RF failed to mask the vehicles on the road.

This lead to the conclusion that the random forest classifier was not well-versed enough to solve this problem, and that a deep learning solution was required.

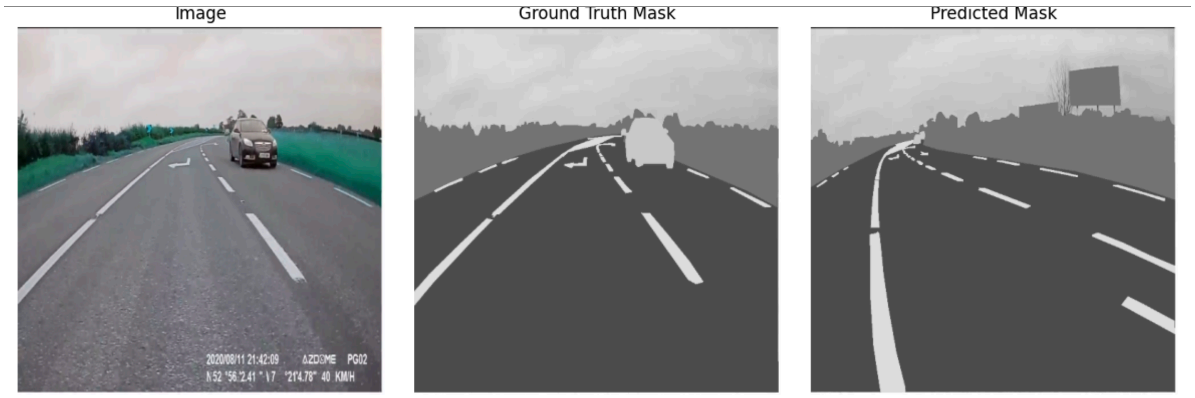


Figure 1: 'Figure 1'

Data

Describing the Data

This dataset comprises a collection of images captured through DVRs (Digital Video Recorders) showcasing roads. Each image is accompanied by segmentation masks bounding different entities (road surface, cars, road signs, marking and background) within the scene.

Below is the result from exploratory data analysis, where the first 6 images and masks from the training dataset are displayed.

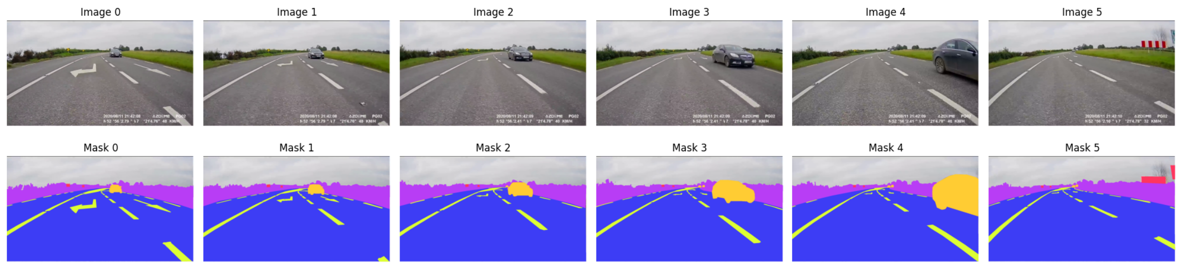


Figure 2: 'Figure 2'

Methods

Preprocessing and Augmentation

Initially, the script loads image and corresponding mask paths from predefined directories, ensuring that the dataset is ready for preprocessing. It employs pandas DataFrame to pair images with their respective masks, which is crucial for maintaining the correspondence between images and their segmentation masks during training. The dataset is divided into training, validation, and testing sets using train_test_split from sklearn, a standard practice to evaluate the model's performance on unseen data and mitigate overfitting. The preprocessing phase includes a custom function, color_to_id_mapping, designed to convert the color-coded masks into a format suitable for training. This function is vital for semantic segmentation tasks where each pixel's label is determined by its color in the mask. The augmentation pipeline is defined using transforms.Compose from torchvision. It includes resizing both images and masks to a uniform size (512x512), followed by converting images to tensor format. These transformations are essential for standardizing input size and format, allowing the model to process the data efficiently. A custom Dataset class, ImageMaskDataset, is implemented to handle the loading and transformation of images and masks. This class ensures that each data item undergoes the defined preprocessing and augmentation steps before being passed to the model, maintaining the integrity and consistency of the data throughout the training process. DataLoader from PyTorch is utilized to batch the data, providing additional features like shuffling and parallel data loading. This enhances the training efficiency and introduces randomness in the training process, which can help in reducing overfitting.

Model

The model employed is a U-Net architecture from the segmentation_models_pytorch library, designed specifically for image segmentation tasks. I chose a U-Net because of its efficient encoder-decoder structure, which is enhanced with skip connections to facilitate detailed segmentation at the pixel level. Configured to identify five distinct classes, the model employs a DiceLoss function to focus on the overlap between predicted and actual segments. An Adam optimizer was used with a learning rate of 0.01 and a weight decay of 0.0001. The training regimen unfolds over 10 epochs, employing a batch size of 1, which impacts memory usage and convergence.

Results and Discussion

To measure the performance of the model, IoU was calculated over each epoch and mask predictions were made after training the model. Unfortunately, the performance of the model is substandard at best. The IoU fluctuated over each epoch between relatively low values no matter what alterations to the model were made. I tried to change the model itself, change its hyperparameters, change the preprocessing, but nothing seemed to improve the score. This lead me to research into why my model was performing not only badly, but unpredictably, which I will discuss further on.

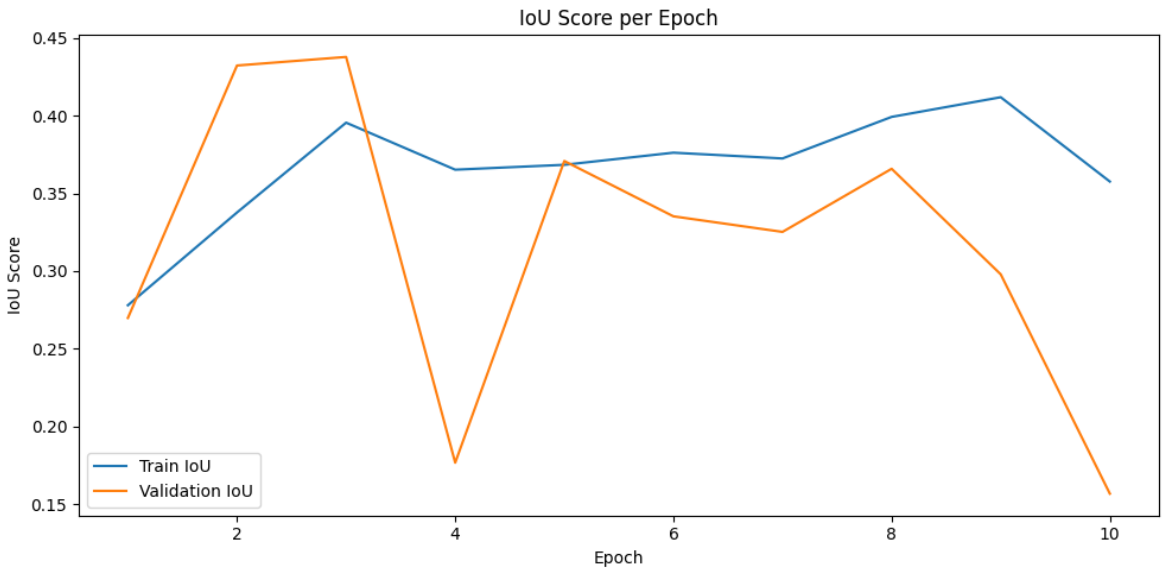
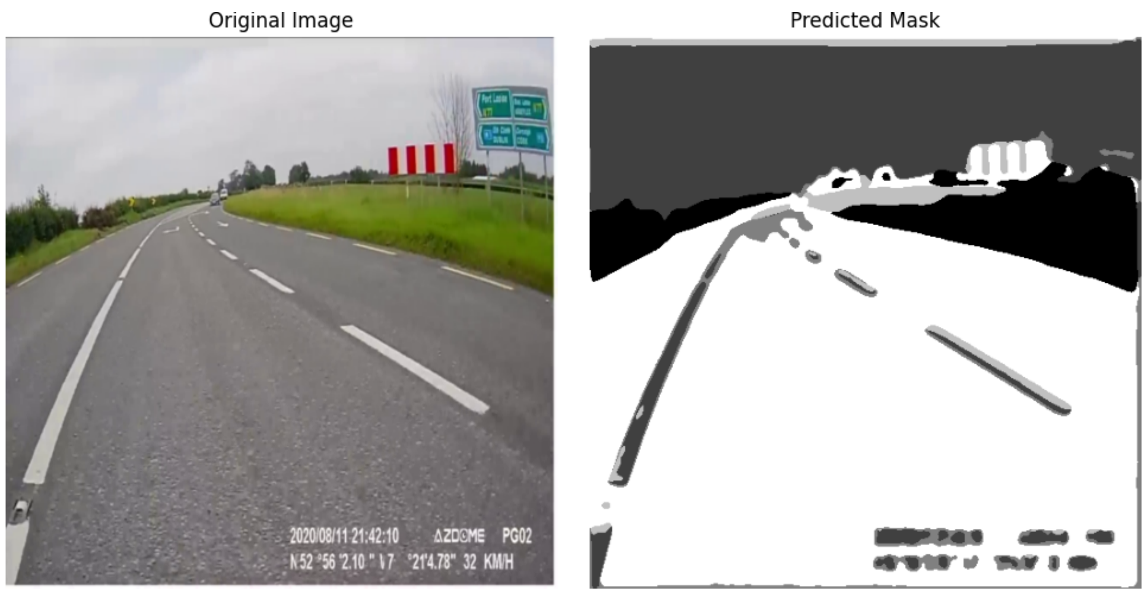
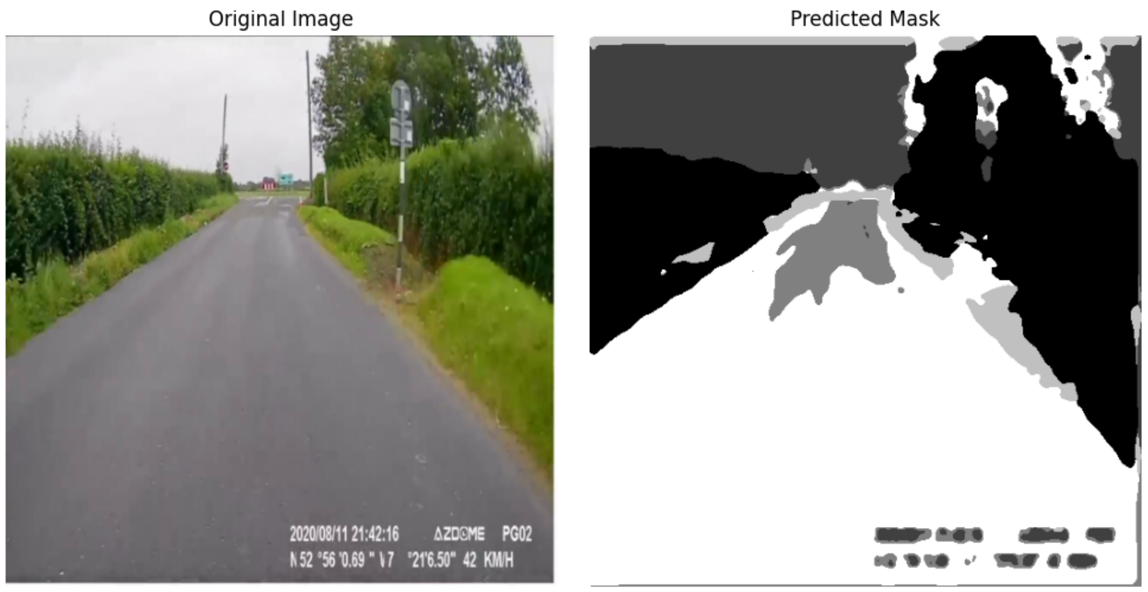


Figure 3: ‘Figure 3’

The figure aboves depicts the training and validation IoU scores over each epoch. As discussed, the IoU fluctuates drastically and does not seem to hold a recognizable pattern at any point. Additionally, the score is low indicating that the model is uneffective.


Original Image
Predicted Mask
2020/08/11 21:42:10 ΔZDOME PG02
N 52 °56 '2.10 " W7 °21'4.78" 32 KM/H


Original Image
Predicted Mask
2020/08/11 21:42:16 ΔZDOME PG02
N 52 °56 '0.69 " W7 °21'6.50" 42 KM/H

The two figures above were predictions made by the model of the predicted masks. The blurry and incorrect masking highlights the model's ineffectiveness.

Model Effectiveness

With inconsistent and poor IoU scores and poor predictions, the model is not effective. I am currently unsure of what is directly causing this but my research lead to a few possible conclusions. One answer may be the data itself; working with 5 classes is a difficult segmentation class and more importantly this dataset is only a trial dataset and contains a limited amount of images. Another possibility is how the model is trained. The loss function might not be best suited for this task and may be causing the model to train inaccurately. However after researching, the Dice loss function was considered a good option for U-Net architecture, so it may be the case that the loss function is not at fault in this situation.

Future Works and Lessons Learned

There were quite a few mistakes made completing this assignment which I will try to amend in the future.

I started with a different dataset initially and that one did not work so I changed to this one. After choosing this dataset and building the model, I realized too late that it did not contain the full contents of the data, but due to time constraints it was the only option. Therefore, next time I will put much more careful analysis into choosing a dataset that is accurate for the task and works with what is trying to be achieved.

I struggled understanding U-Net architecture and spent most of the time reading about how it works and how to implement rather than acutally implementing and tuning it to be effective. This is probably the main reason why my model is not as effective as I hoped it would be. Next time more emphasis will be placed on tuning the model to perform better.

Another aspect I struggled with was understanding segmentation and masks. Figuring out how to make mask predictions using a U-Net was difficult and consumed a significant amount of time. Next time, I hope my background knowledge has improved to be able to complete the task.