Project Report

Car Image Segmentation using U-Net

GitHub Repository

Nupur
Student Number: 1340285
Course: Deep Learning with PyTorch
Fanshawe College

August 4, 2025

Abstract

This project explores the application of deep learning in the field of semantic segmentation using the U-Net architecture. It focuses on segmenting cars iamges from background images in the Carvana dataset. The model uses convolutional encoders and decoders with skip connections to retain fine image details. Key metrics such as Binary cross entropy loss,IoU, Dice Score, and pixel accuracy are used to evaluate performance. The implementation demonstrates practical relevance in tasks such as vehicle extraction, background removal, and preprocessing for automotive image analysis, which are common in computer vision workflows.

1 Introduction

Semantic segmentation is a vital computer vision task, especially in autonomous systems and image processing. In this project, a U-Net model in PyTorch has been used to perform semantic segmentation of cars from the Carvana dataset. U-Net is known for delivering precise segmentation results and performs well even with limited data,in this project it is trained on a large dataset of 3675 car images from the Carvana dataset.

2 Dataset and Preprocessing

2.1 Dataset Description

The project uses the Carvana Image Masking Challenge dataset, which contains 5088 images along with corresponding masks.

- Car images in JPG format
- Corresponding binary masks in GIF format
- Data split: 15 percent of the total images used as the test set, and 15 percent of the remaining 85 percent used for validation.

2.2 Preprocessing Steps

- Convert input images to 3-channel RGB to make them compatible with U-Net input requirements, and convert masks to single-channel grayscale
- Resize both images and masks to 128×128 pixels for uniform input dimensions
- Apply transformations to convert PIL images to PyTorch tensors with pixel values normalized to the [0, 1] range
- Convert grayscale masks to binary format, where 0 represents the background and 1 represents car pixels
- Use a custom `Dataset` class and PyTorch's `DataLoader` for efficient data loading and batching during training

3 Methodology

3.1 Model Architecture: U-Net

U-Net comprises:

- Encoder: A series of convolutional blocks with ReLU activation and batch normalization, followed by max pooling for downsampling. Each block consists of two convolutional layers, enabling hierarchical feature extraction. The output of each block is stored for skip connections used in the decoder.
- **Decoder:** Transposed convolution layers for upsampling, followed by convolutional blocks with skip connections. At each stage, the upsampled feature map is concatenated with the corresponding encoder output, then passed through a double convolution block for refined feature reconstruction.
- **Skip Connections:** Preserve spatial details by directly connecting encoder feature maps to the corresponding decoder layers, enabling better localization and reconstruction.

3.2 Implementation Details

- **Framework:** PyTorch
- **Input Size:** 128x128 pixels
- **Batch Size:** 16
- **Epochs:** 40
- **Learning Rate:** 0.001
- **Loss Function:** BCEWithLogitsLoss
- **Optimizer:** Adam

4 Model Training

The U-Net model was trained using the Carvana dataset with images and corresponding masks resized to 128×128 pixels. The model was optimized using the Binary Cross-Entropy loss function and the Adam optimizer. Training was conducted for 40 number of epochs, with both training and validation losses monitored to ensure effective learning and to prevent overfitting. The loss curves shown illustrate a steady decrease in both training and validation loss, indicating successful model convergence.

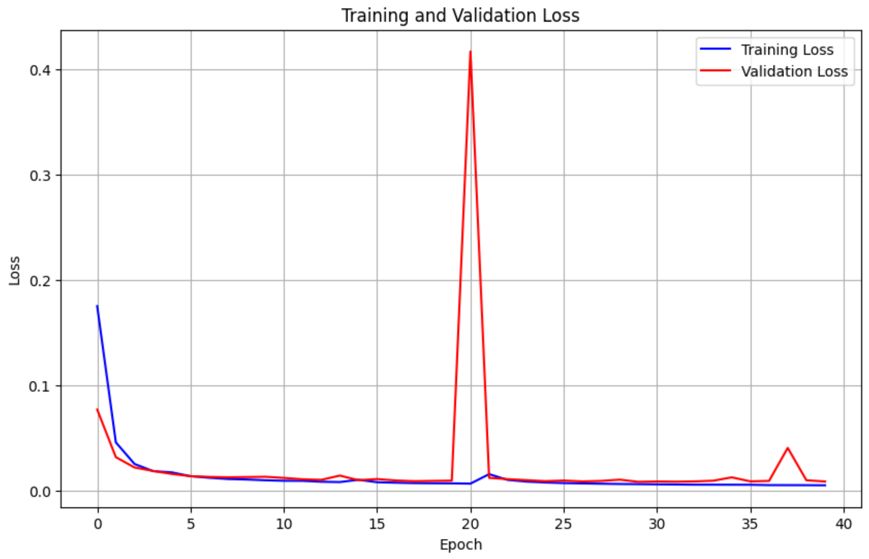

Figure 1: Training and validation loss curves for the U-Net model over epochs.

5 Evaluation Metrics and Results

– **Test Loss:** To measure the model's overall error on unseen test data, computed using Binary Cross-Entropy

– **Pixel Accuracy:** To measure the proportion of correctly predicted pixels over the total number of pixels

– **IoU (Intersection over Union):** To evaluate the overlap between predicted and ground truth masks

– **Dice Score:** To measure the similarity between the predicted and ground truth masks.

Metric	Value
Test Loss	0.0088
Pixel Accuracy	0.9966 (99.66%)
IoU (Intersection over Union)	0.9855
Dice Score	0.9927
Total Test Samples	764

Table 1: Performance metrics of the U-Net model on the test dataset

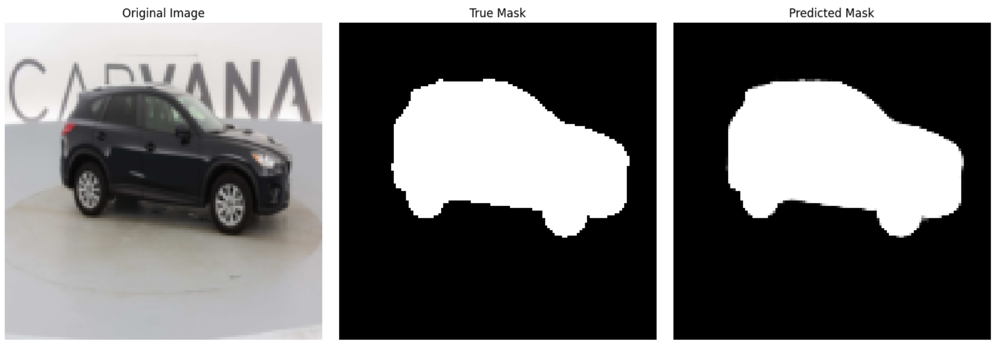

Figure 2: Prediction vs Ground Truth Samples

6 Car Background Removal Dashboard

A dashboard was developed using Streamlit to enable users to upload car images, automatically extract the car, and remove the background. This tool is deployed on Hugging Face Spaces and can be accessed at car-background-remover Dashboard.

The dashboard is integrated with Llama 3, allowing users to input the car's year, make, and model to fetch detailed information about the vehicle. This combined solution streamlines the process of creating car listings by providing both background removal and detailed car insights, enabling individuals to list their vehicles for sale in a faster and more user-friendly way.

7 Conclusion

This project demonstrates a complete pipeline of training a semantic segmentation model using U-Net in PyTorch. The approach achieves high high performance in separating cars from backgrounds in carvana dataset, and metrics like IoU and Dice Score confirm the model's robustness. Potential improvements include augmenting the training data, and adopting more advanced model architectures such as U-Net++ or other encoder-decoder variants to enhance segmentation performance.

References

[1] PyTorch Semantic Segmentation Tutorial. *PyTorch Documentation*. Available at: `https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html`

[2] Carvana Image Masking Challenge. *Kaggle*. Available at: `https://www.kaggle.com/competitions/carvana-image-masking-challenge/data`

[3] Streamlit: Turn data scripts into shareable web apps. *Streamlit Documentation*. Available at: `https://docs.streamlit.io/`

[4] Hugging Face. *Spaces: Deploy Machine Learning Apps*. Available at: `https://huggingface.co/spaces`