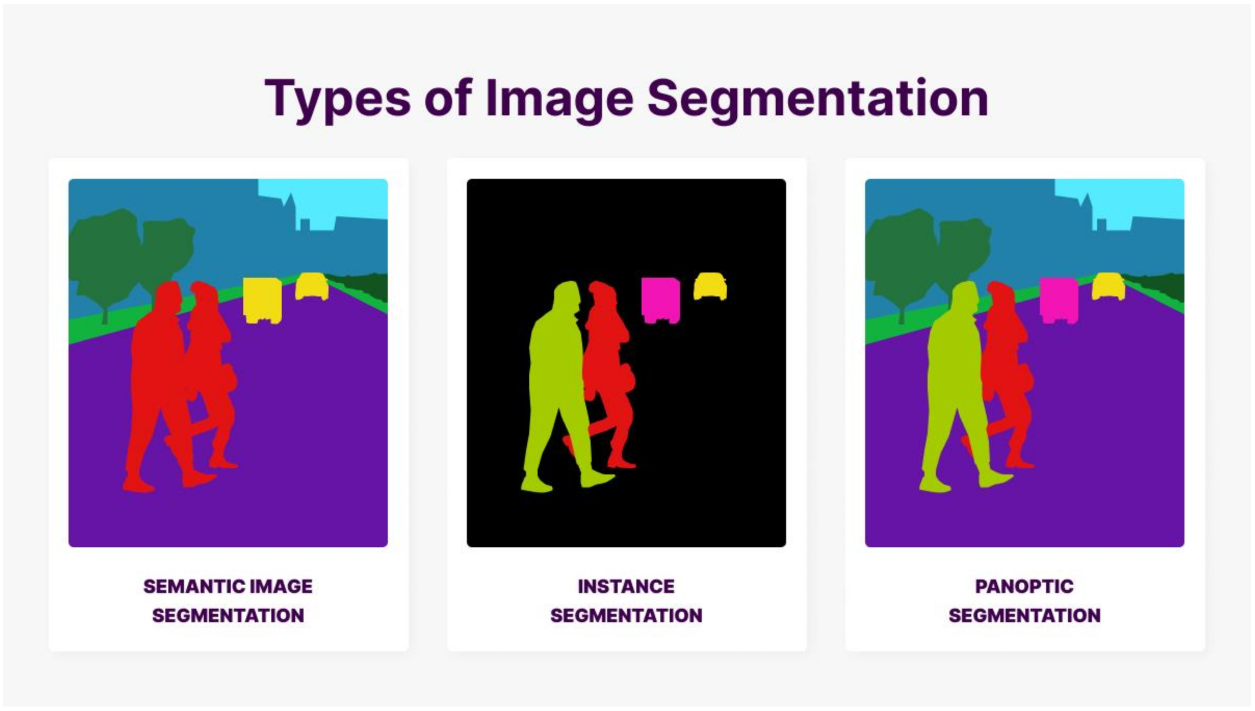
Deep Learning CSE676

rkhatik – 50485657

Final Project



Semantic segmentation is a computer vision technique that assigns a class label to each pixel or region in an image. It involves segmenting an image into meaningful regions based on their semantic category, providing a detailed understanding of the image's content. Deep learning models, particularly convolutional neural networks (CNNs), are commonly used for semantic segmentation. These models learn from labeled training data to predict the class label for each pixel, enabling applications such as autonomous driving, image editing, and medical imaging. Semantic segmentation helps in tasks that require pixel-level understanding and precise object localization.

The steps involved for semantic Segmentation are as as follows:

Here are the steps involved in semantic segmentation for road segmentation:

1. Dataset preparation: Collect a large dataset of images that includes road scenes. Annotate each image with pixel-level labels, marking road and non-road regions.

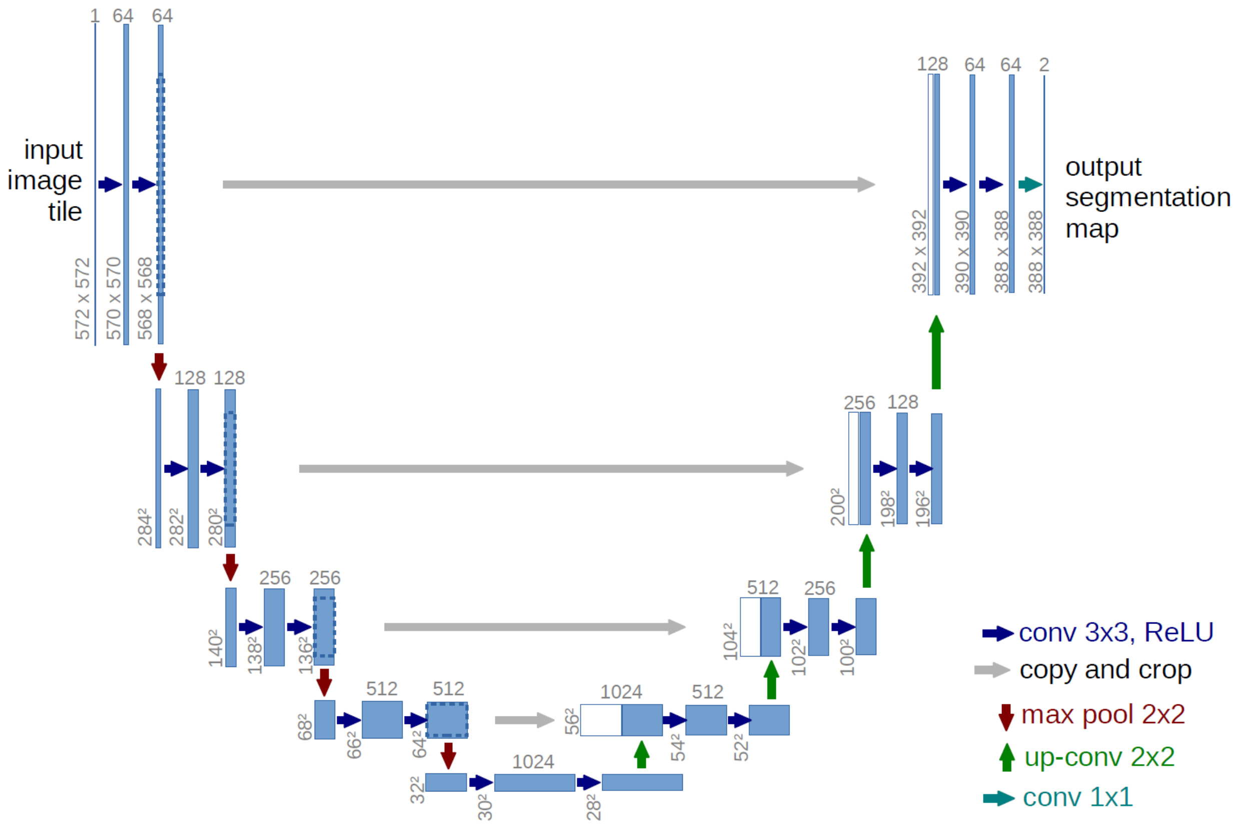
2. Network architecture selection: Choose a suitable deep learning architecture for semantic segmentation, such as U-Net, Fully Convolutional Networks (FCN), DeepLab, or Mask R-CNN. These architectures are designed to handle pixel-wise predictions effectively.

3. Training the model: Train the selected network architecture using the labeled dataset. The model learns to predict a segmentation mask that indicates road pixels based on the input images. The training process involves optimizing the model's parameters to minimize the difference between predicted and ground truth segmentation masks.

4. Evaluation and fine-tuning: Evaluate the trained model on a separate validation or test set to assess its performance. Fine-tune the model if necessary by adjusting hyperparameters or retraining with additional data to improve segmentation accuracy.

5. Inference and post-processing: Use the trained model for road segmentation on new, unseen images. The model outputs a pixel-level segmentation mask, where road pixels are highlighted. Apply post-processing techniques such as morphological operations or conditional random fields to refine the segmentation results and remove any noise or artifacts.

Network Architecture selected is U-NET architecture:



The U-Net architecture was introduced in 2015 and was specifically aimed at solving the problem of biomedical image segmentation. The U-Net architecture is a widely used deep learning model for image segmentation tasks. It consists of an encoder path that captures high-level features and a decoder path that recovers spatial resolution. The encoder path reduces the spatial dimensions of the input image, while the decoder path performs up sampling and concatenation operations to localize objects accurately. Skip connections connect the encoder and decoder paths, enabling the decoder to leverage both low-level and high-level features. The final layer produces a segmentation map where each pixel represents the probability of belonging to a specific class, such as road or non-road. The U-Net architecture is known for its effective contextual information capture, efficient parameter usage, and robust segmentation performance.

Encoder Network:

The encoder network serves as a feature extractor and uses a series of encoder blocks to learn an abstract representation of the input image. Each encoder block consists of two 3x3 convolutions, with a ReLU (Rectified Linear Unit) activation function coming after each convolution. The ReLU activation function adds non-linearity to the network, aiding in the training data's improved generalization. The appropriate decoder block receives a skip connection from the ReLU's output. The spatial dimensions (height and breadth) of the feature maps are then cut in half by a 2x2 max-pooling. This lowers the amount of trainable parameters, hence lowering the computational cost.

Skip Connections:

These skip links offer extra data that improves the decoder's ability to produce semantic features. Additionally, they serve as a quick link that facilitates the uninterrupted indirect passage of gradients to the lower levels. Simply said, skip connections aid in improved gradient flow during backpropagation, which in turn aids in the network learning better representation.

Bridge:

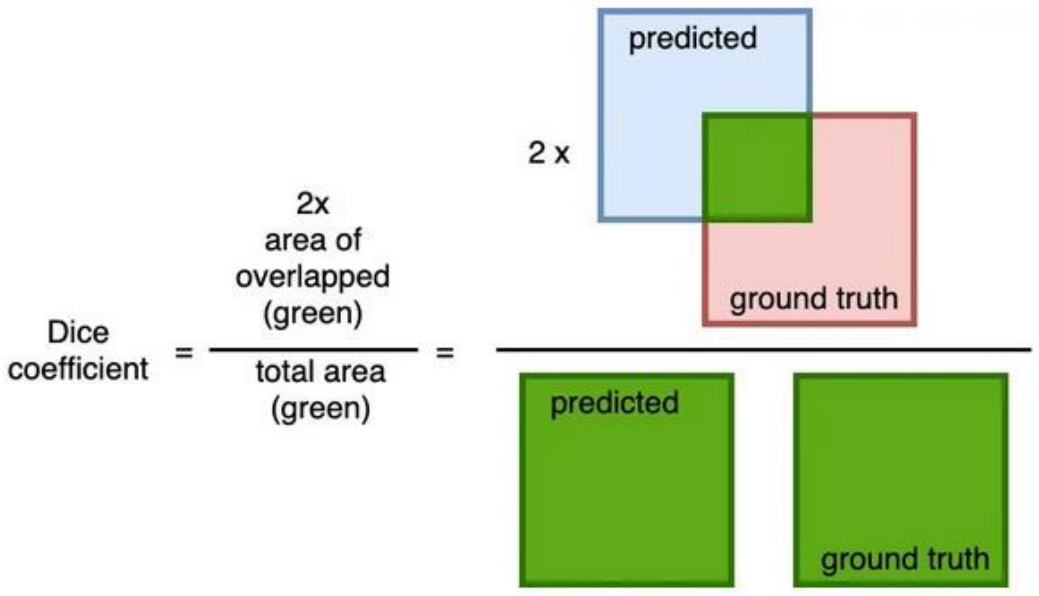
The bridge completes the information flow by connecting the network of encoders and decoders. It comprises of two 3x3 convolutions, with a ReLU activation function coming after each convolution.

Decoder Block:

The abstract representation is fed into the decoder network, which produces a semantic segmentation mask. The 2x2 transpose convolution is the first step in the decoder block. The relevant skip connection feature map from the encoder block is then concatenated with it. Due to the network's depth, these skip connections offer functionality from prior layers that are occasionally lost. Then, two 3x3 convolutions are employed, with a ReLU activation function coming after each convolution. The Final layer as softmax activation.

Dice Loss:

$$Dice = \frac{2\,|A \cap B|}{|A| + |B|}$$



The Dice loss is a variation of the Dice coefficient that can be used as a loss function for semantic segmentation tasks. The Dice coefficient measures the overlap between two sets, such as the predicted segmentation mask and the ground truth mask. The Dice loss is derived from the complement of the Dice coefficient and is defined as 1 minus the Dice coefficient. The Dice loss aims to minimize the dissimilarity between the predicted and ground truth segmentation masks. By optimizing this loss function during training, the model is encouraged to produce segmentation masks that closely match the ground truth, maximizing the overlap between the predicted and ground truth masks.

Datasets used:

- Lyft Udacity Challenge

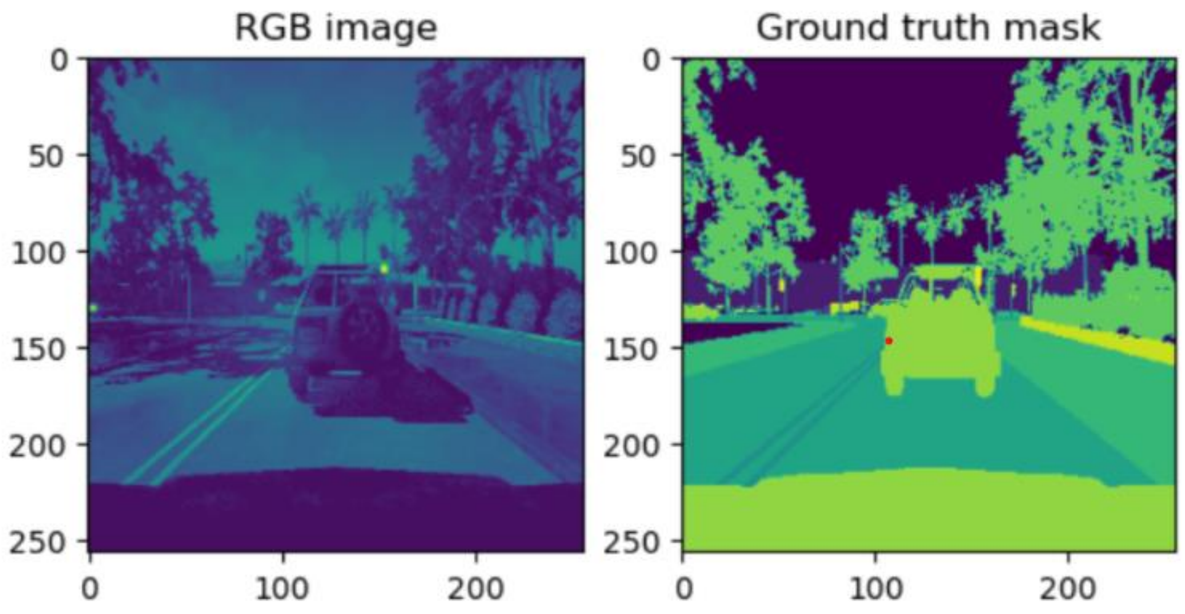
https://www.kaggle.com/datasets/kumaresanmanickavelu/lyft-udacity-challenge

The entire dataset is of 5.13 Gb. It contains 5 subfolders with and each subfolders has 1000 images of RGB and its corresponding mask image. The RGB image would be used as the input to the model. The predicted output would then be compared to the label mask.

The loss is then computed using Dice coefficient and back-propogation is performed to optimize the model.

The optimizer used is Adam and the learning rate selected is 0.0005

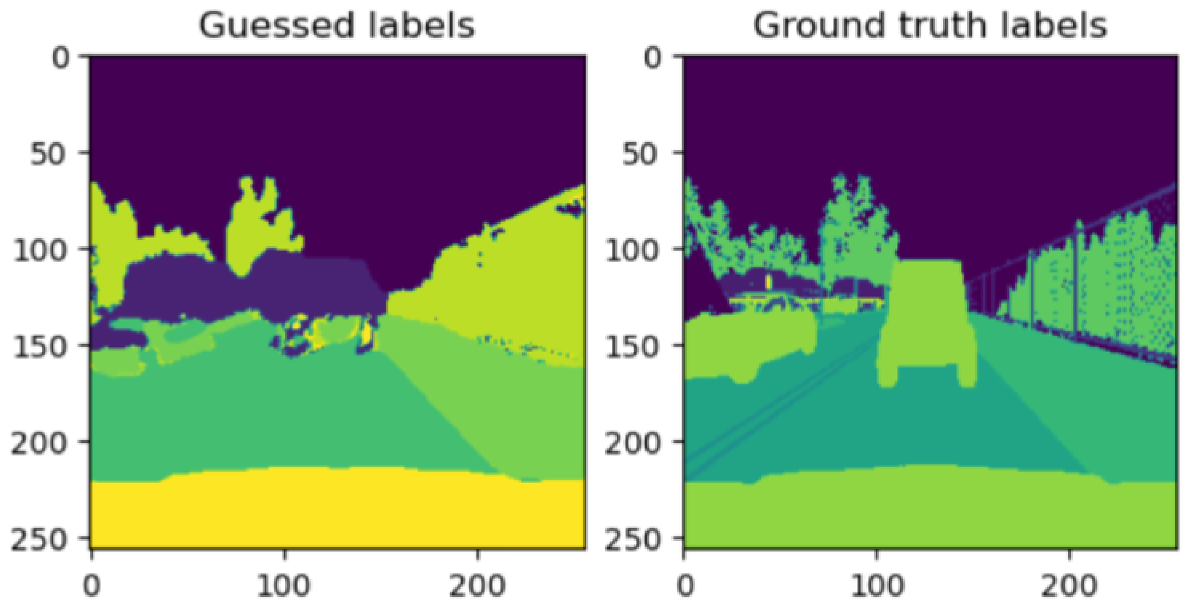
Visualization of the dataset:



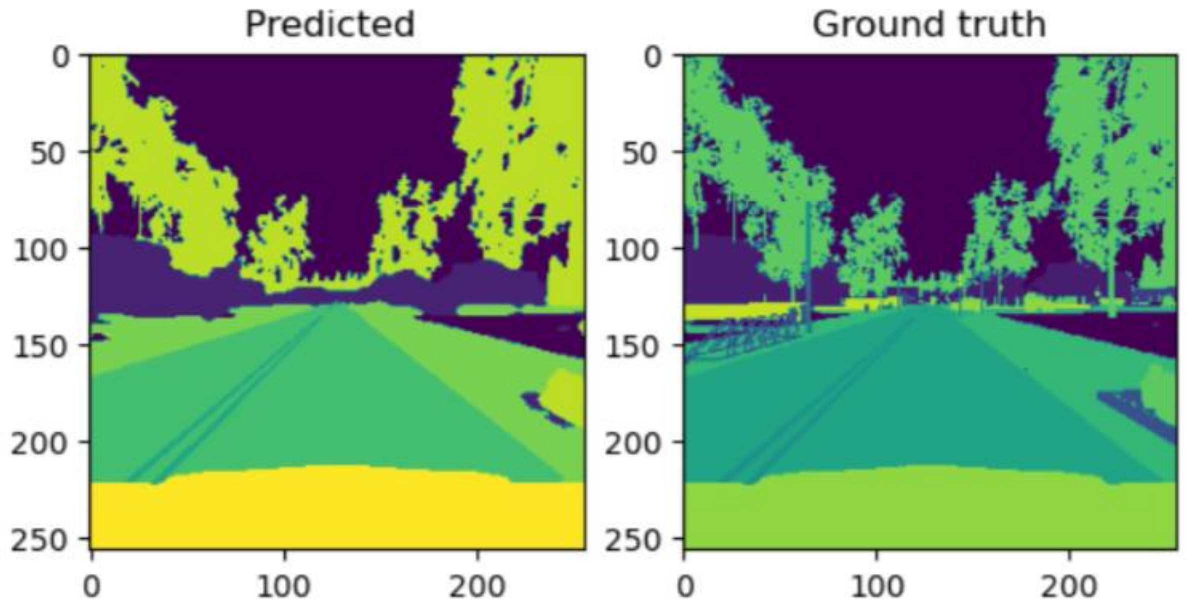
Results:

Comparison between Predicted and Mask images for Model made from scratch.

```
Epoch 1, loss:  0.7931134809147228
<Figure size 640x480 with 0 Axes>
```



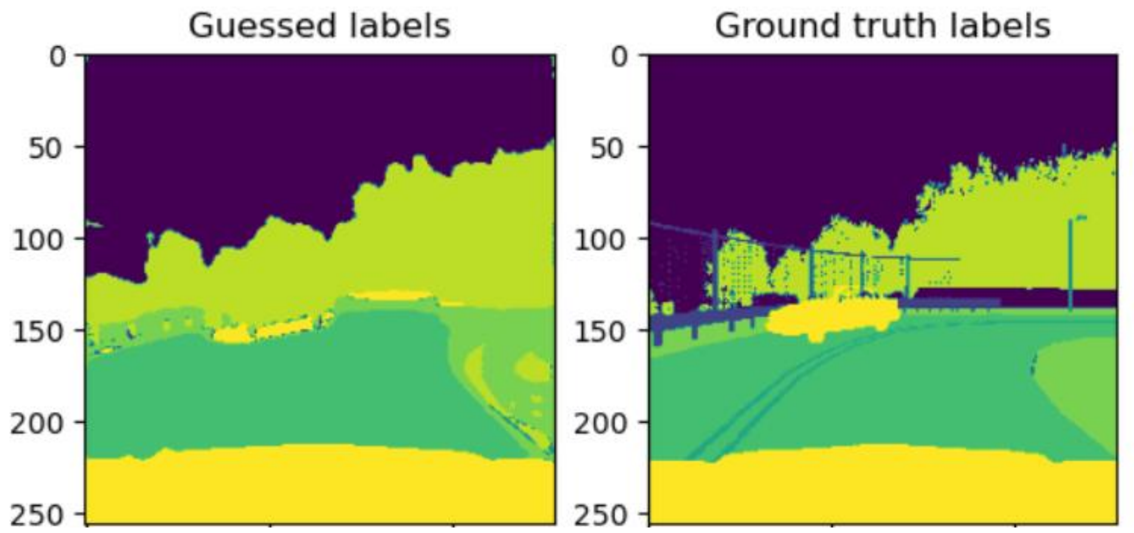
```
Epoch 100, loss:  0.7412789545275948
<Figure size 640x480 with 0 Axes>
<Figure size 640x480 with 0 Axes>
```



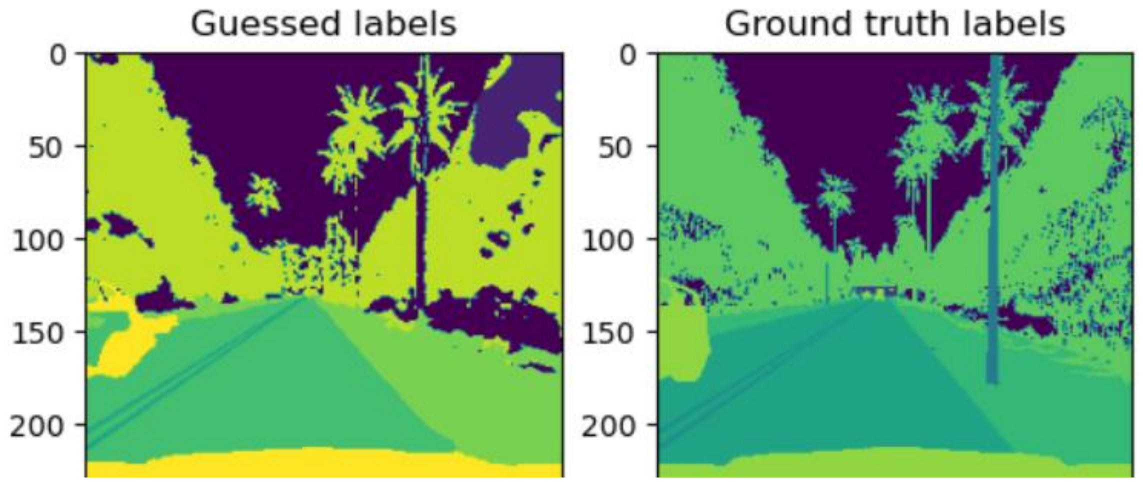
```
<Figure size 640x480 with 0 Axes>
```

Comparison between Predicted and Mask Images for pretrained ResNet34 backbone based UNet Architecture for Test dataset:

```
Epoch 1, loss:  0.8182893037796021
Epoch 1, validation loss:  0.8020266890525818

<Figure size 640x480 with 0 Axes>
```

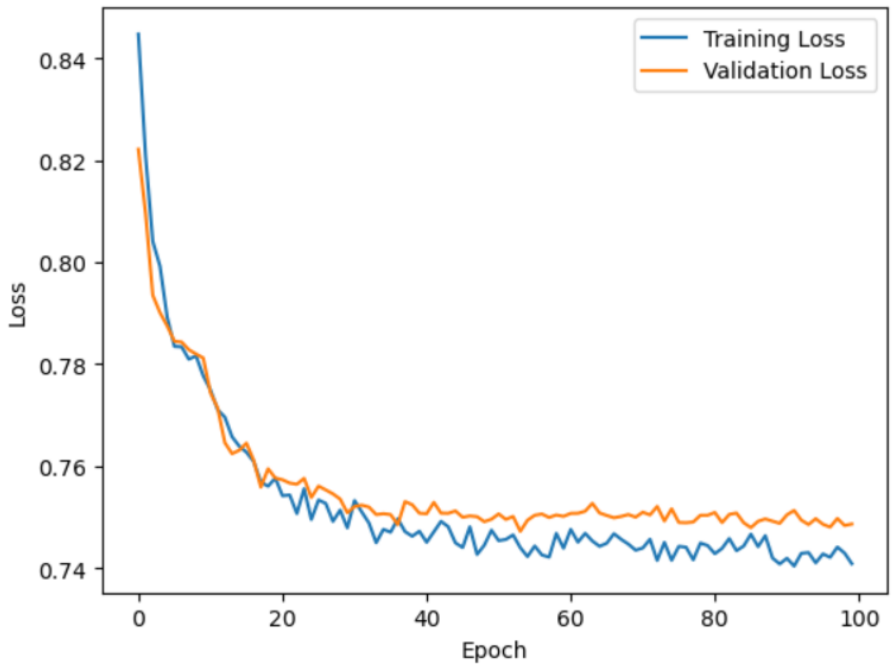


```
Epoch 100, loss:  0.7455451488494873
Epoch 100, validation loss:  0.7488870024681091

<Figure size 640x480 with 0 Axes>

<Figure size 640x480 with 0 Axes>
```



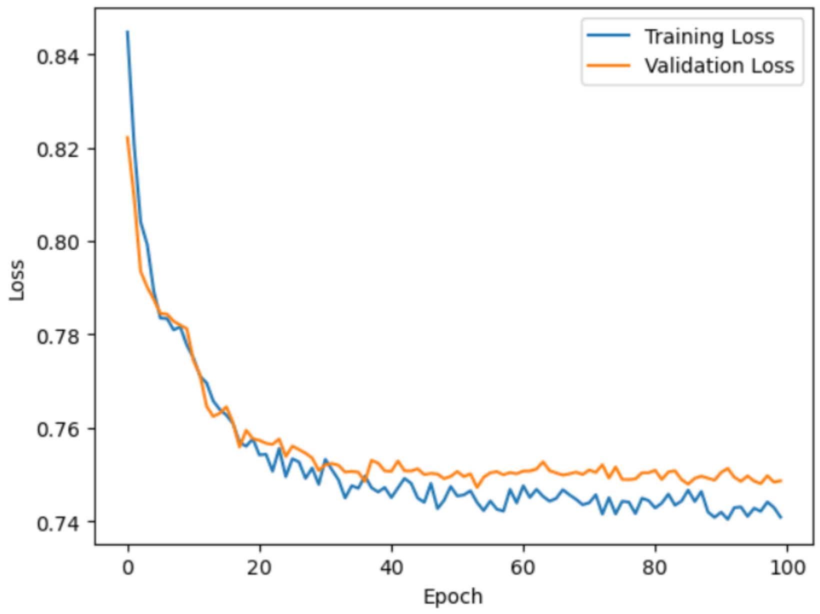
Plot comparison between Training and Validation loss for model made from scratch

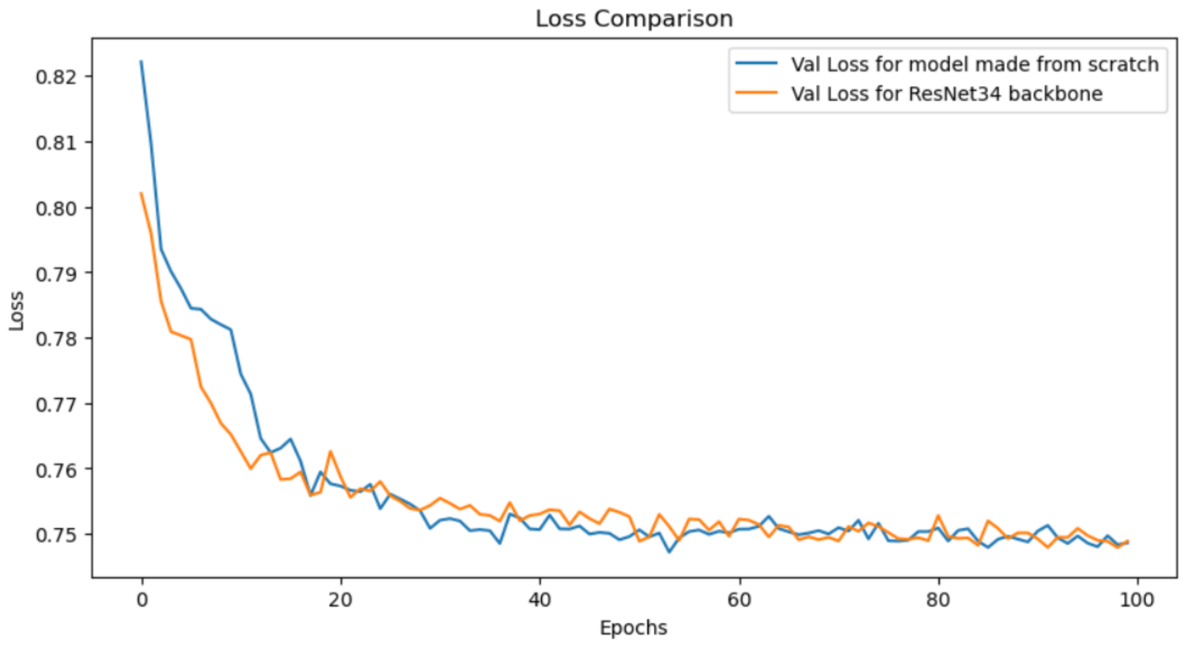
```
<Figure size 640x480 with 0 Axes>
```



Plot comparison between Training and Validation loss for pretrained ResNet34 UNet architecture

```
<Figure size 640x480 with 0 Axes>
```





References:

1. U-Net: Convolutional Networks for Biomedical Image Segmentation , Olaf Ronneberger, Philipp Fischer, Thomas Brox, arXiv:1505.04597 https://arxiv.org/abs/1505.04597
2. https://www.kaggle.com/datasets/kumaresanmanickavelu/lyft-udacity-challenge
3. https://medium.com/hackernoon/semantic-segmentation-datasets-for-autonomous-driving-1182ebd2aff0
4. https://github.com/qiaoxu123/Self-Driving-Cars/blob/master/Part3-Visual_Perception_for_Self-Driving_Cars/Module5-Semantic_Segmentation/Module5-Semantic_Segmentation.md
5. https://www.kaggle.com/code/gokulkarthik/image-segmentation-with-unet-pytorch
6. https://www.youtube.com/watch?v=IHq1t7NxS8k&t=2245s&ab_channel=AladdinPersson