Driverless Car Nano degree – Project -2 – Semantic Segmentation

1. Objective

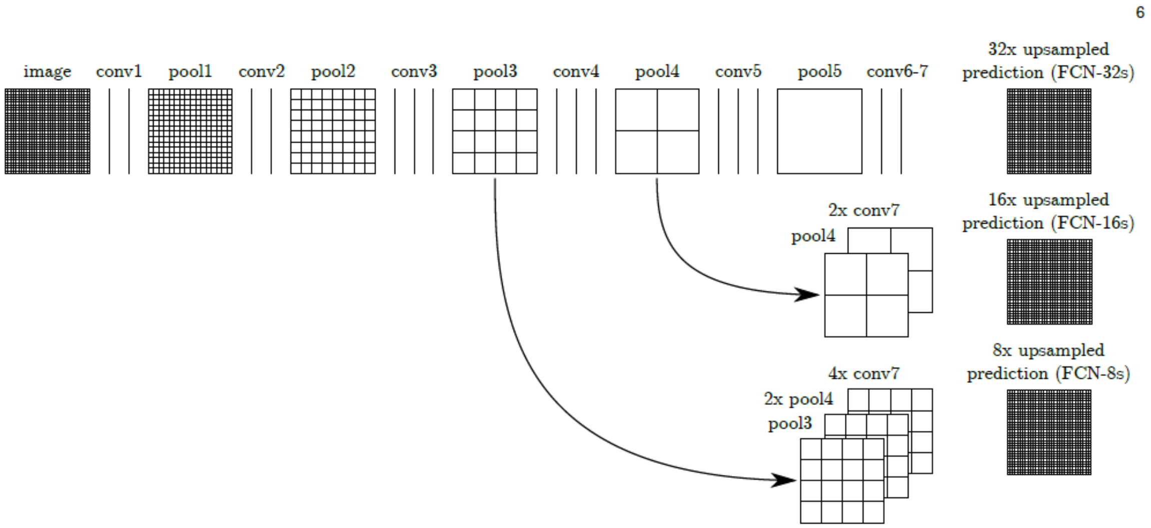
The objective of this project is to label the pixels of an image to identify the road or pathway using a Fully Convolutional network (FCN).

2. Architecture

The Semantic Segmentation architecture consists of two components of the Neural network namely:

1. Encoder – A pre-trained VGG16 is used as an encoder
2. Decoder – This piece of the neural network semantically project the discriminated features identified by the encoder onto the pixel space to get a dense classification

1. The FCN uses the pretrained VGG16 to perform the segmentation
2. The fully connected layer of VGG16 is converted to Fully convolutional layers using 1x1 convolution. In Tensorflow this is implemented as "tf.layers.conv2d"
3. Upsampling of these discriminated feature is done using the transposed convolution using tf.layers.conv2d_transpose


6
image
conv1
pool1
conv2
pool2
conv3
pool3
conv4
pool4
conv5
pool5
conv6-7
32x upsampled prediction (FCN-32s)
2x conv7
pool4
16x upsampled prediction (FCN-16s)
4x conv7
2x pool4
pool3
8x upsampled prediction (FCN-8s)

3. Program logic

- Load the pre-trained VGG16 model into Tensorflow using the function Load_VGG and get the tuple of tensors from VGG model including Input Image, Keep probability, Input3, 4 and 7
- Now we recreate the FCN as per the above architecture and this is done in function layers. The output of this function is the recreated FCN tensor with Encoder and Decoder with Softmax function
- We optimize the neural network using Cross entropy as loss function and Adam as the optimization algorithm. This is implemented in function Optimize. The objective of this function is to find the weights and parameters that would correctly identify the pixels
- We train the built neural network using various tuning parameters like no of epochs, batch size, loss function, optimizer using the keep probability and learning rate. The output is printed for analysis function
- Now the run function trains the neural network using above functions and then save the inference data in runs library
- The program downloads the Kitti Road dataset for this purpose.

4. Project Rubric

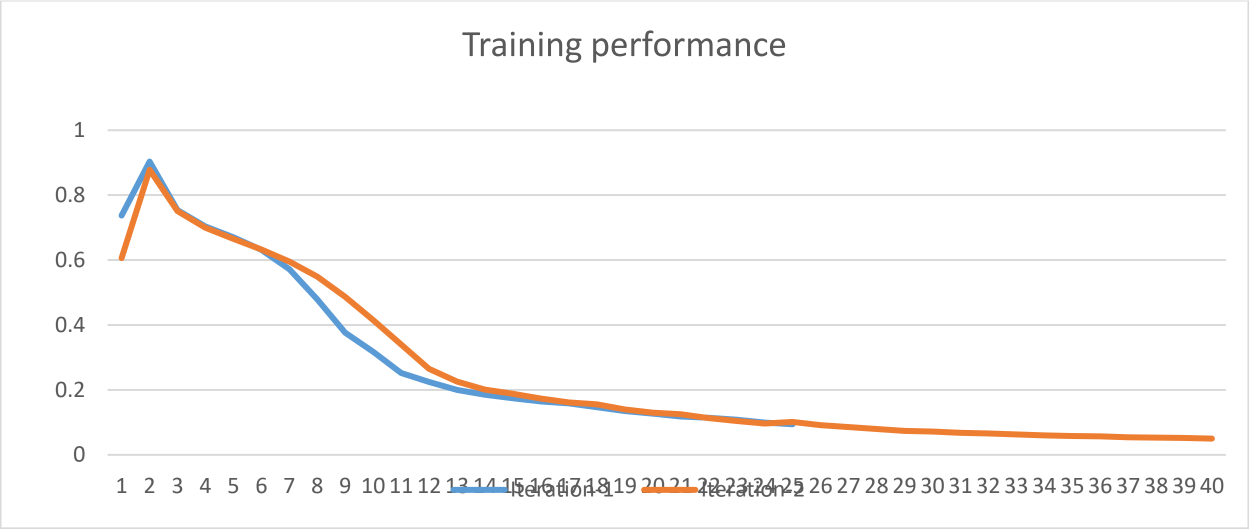
- Does the project load the pretrained vgg model? - Yes the trained model is loaded correctly
- Does the project learn the correct features from the images? – Yes the layers function is implemented as per architecture
- Does the project optimize the neural network? – Yes the project uses Adam Optimizer
- Does the project train the neural network? – Train_nn function is implemented correctly
- Does the project train the model correctly? – Yes shown in the results
- Does the project use reasonable hyperparameters? – Yes. Epoc – 40 and batch size is 5
- Does the project correctly label the road? – Yes shown in the results

5. Results

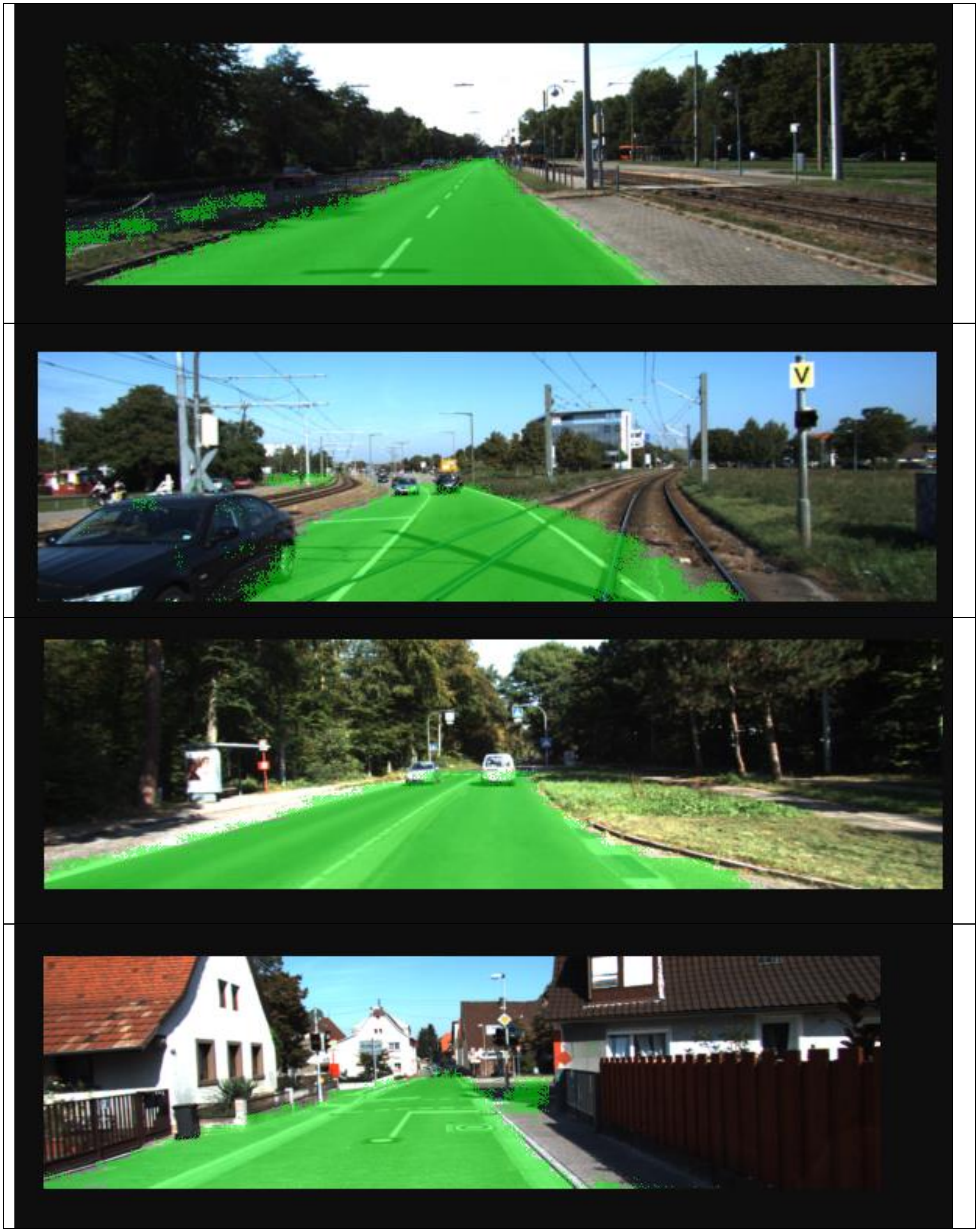
- Learning rate – 0.0001
- Epochs- 40
- Batch size – 5
- Keep Prob – 0.5

Iteration-1	Iteration-2
0.737097843	0.606443899
0.90354001	0.878099193
0.753569054	0.751187567

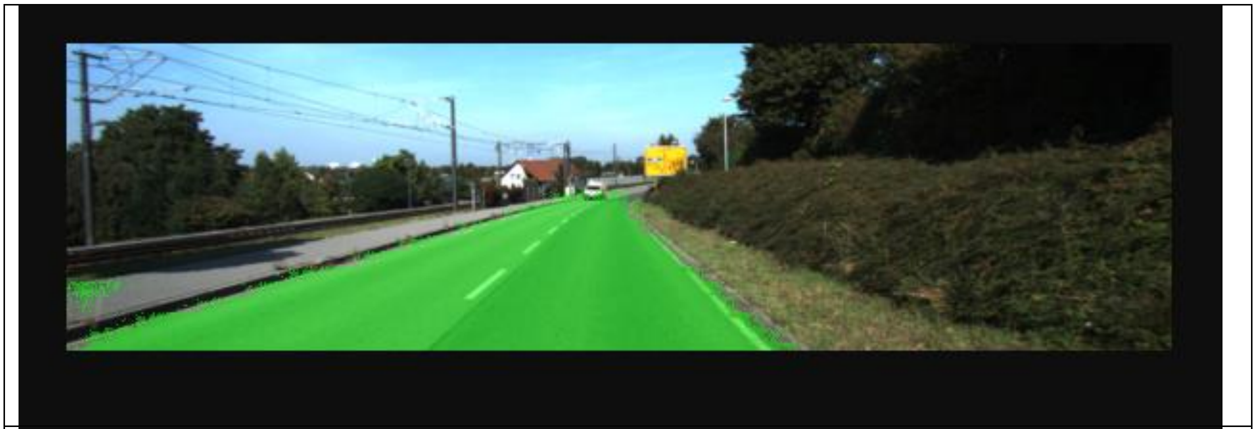
0.703267391	0.700064482
0.670522419	0.665018462
0.631629922	0.632881964
0.571205473	0.595368086
0.478675434	0.549060283
0.376708432	0.486378648
0.317448329	0.414790662
0.2518838	0.339827897
0.224352423	0.265316091
0.200130096	0.225351758
0.185655792	0.200781886
0.173925046	0.187713871
0.164427103	0.173281006
0.15824682	0.161183109
0.146567839	0.156063496
0.135345678	0.139410451
0.127286904	0.129758898
0.11851026	0.125268358
0.114630079	0.112895953
0.108419496	0.104005992
0.099221185	0.096917564
0.094330817	0.101755591
	0.092075732
	0.08520108
	0.079527064
	0.074137822
	0.071431443
	0.067929338
	0.066383724
	0.063353202
	0.060533462
	0.058100371
	0.05692553
	0.054261176
	0.053570403
	0.051846702
	0.050137407

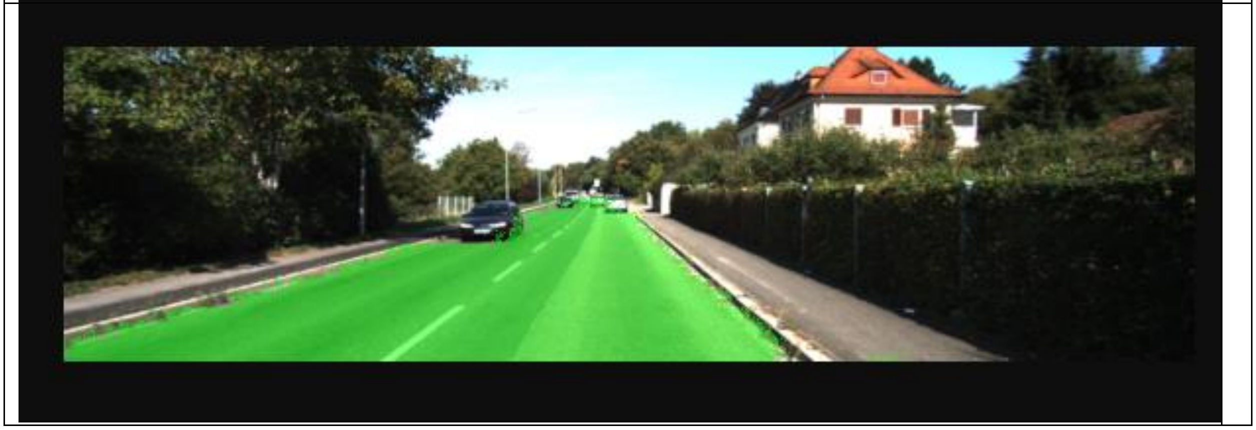

Training performance
1
0.8
0.6
0.4
0.2
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
Iteration-1
Iteration-2

```
TensorFlow Version: 1.3.0

Default GPU Device: /gpu:0

Testing for kitti datatset presence......

Tests Passed

Download pretrained vgg model....

Building NN using load_vgg, layers, and optimize function....

Starting Load VGG

Starting LAYERS

Starting OPTIMIZE

Built NN using load_vgg, layers, and optimize function....

Training NN using the train_nn function....


Starting Training

Epoch 1/40 - Batch 1 - Loss 49.13429641723633

…………………………………………………………………….

Epoch 40/40 - Batch 58 - Loss 0.04367953911423683

 Epoch 40/40 - MEAN LOSS : 0.05013740685736311

Trained NN using the train_nn function....

Saving inference data using helper.save_inference_samples....

Training Finished. Saving test images to: ./runs/1544121828.045032

Saved inference data using helper.save_inference_samples....
```



V





6. References

Most of the codes were already available in Udacity Lecture Video and FAQ