Build the Neural Network

|  |  |
| --- | --- |
| Does the project load the pretrained vgg model? | The function load\_vgg is implemented correctly. |
| Does the project learn the correct features from the images? | The function layers is implemented correctly. |
| Does the project optimize the neural network? | The function optimize is implemented correctly. |
| Does the project train the neural network? | The function train\_nn is implemented correctly. The loss of the network should be printed while the network is training. |

Neural Network Training

|  |  |
| --- | --- |
| Does the project train the model correctly? | On average, the model decreases loss over time. |
| Does the project use reasonable hyperparameters? | The number of epoch and batch size are set to a reasonable number. |
| Does the project correctly label the road? | The project labels most pixels of roads close to the best solution. The model doesn't have to predict correctly all the images, just most of them.  A solution that is close to best would label at least 80% of the road and label no more than 20% of non-road pixels as road. |

python main.py

**Submission**

1. Ensure you've passed all the unit tests.
2. Ensure you pass all points on [**the rubric**](https://review.udacity.com/#!/rubrics/989/view).
3. When you're done with the project, please submit the following in a zip file.
   * helper.py
   * main.py
   * project\_tests.py
   * Newest inference images from runs folder (all images from the most recent run)

**Tips**

* The link for the frozen VGG16 model is hardcoded into helper.py. The model can be found [**here**](https://s3-us-west-1.amazonaws.com/udacity-selfdrivingcar/vgg.zip), but is already downloaded if using the Udacity workspace.
* The model is not vanilla VGG16, but a fully convolutional version, which already contains the 1x1 convolutions to replace the fully connected layers. Please see this [**post**](https://s3-us-west-1.amazonaws.com/udacity-selfdrivingcar/forum_archive/Semantic_Segmentation_advice.pdf) for more information. A summary of additional points, follow.
* The original FCN-8s was trained in stages. The authors later uploaded a version that was trained all at once to their GitHub repo. The version in the GitHub repo has one important difference: The outputs of pooling layers 3 and 4 are scaled before they are fed into the 1x1 convolutions. As a result, some students have found that the model learns much better with the scaling layers included. The model may not converge substantially faster but may reach a higher IoU and accuracy.
* When adding l2-regularization, setting a regularizer in the arguments of the tf.layers is not enough. Regularization loss terms must be manually added to your loss function. otherwise regularization is not implemented.
* The VGG model may be larger than some you have used before, so be conscious of your memory usage - you may need to use small batch sizes in order to avoid running out of memory when training.
* The VGG model, as well as semantic segmentation, may be slower than previous, simpler models you have constructed. You may want to print additional information during training to be more aware of progress - depending on batch size and number of batches per epoch, each epoch could be 2-3 minutes or more even with the GPU.
* Given the above, you may consider prototyping on just a subset of the data until you are sure your model works correctly!

https://classroom.udacity.com/nanodegrees/nd013/parts/6047fe34-d93c-4f50-8336-b70ef10cb4b2/modules/595f35e6-b940-400f-afb2-2015319aa640/lessons/1b046c47-76e3-45de-8be7-8bc6b4361b18/concepts/2a478851-eebb-47c9-acc3-c6a92f1f3e61