UDACITY

Back to Self-Driving Car Engineer

Semantic Segmentation

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Kudos ! I think you've done a perfect job of implementing a Fully Convolutional Networks for Semantic Segmentation. It's very clear that you have a good understanding of the basics.

Further Reading: Semantic Segmentation has evolved quite a lot, since FCNs came by. I would highly recommend reading, *Guide to Semantic Segmentation with Deep Learning*, to explore how the solutions evolved since FCNs and the current state of the art used in real world problems.

And Congratulations 🎉 once again, for successfully completing this project. I hope you had great learning experience working with FCNs :)

Build the Neural Network

The function `load_vgg` is implemented correctly.

Good work loading the VGG16 model and extracting placeholders from the graph.

Pro Tip: Visualizing this VGG16 model using Tensorboard can be extremely useful. Below I provide you with a snippet to convert `.pb` file into TF summary. After converting it, you can run `tensorboard --logdir=.` in the same directory to start Tensorboard and visualize the graph in your browser.

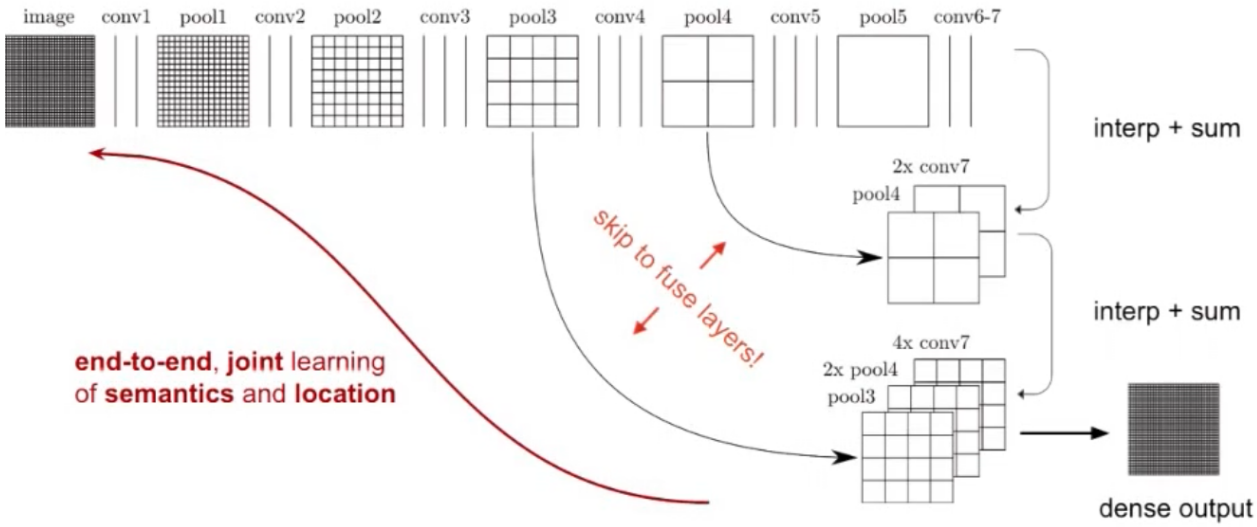
```
import tensorflow as tf
from tensorflow.python.platform import gfile
from tensorflow.core.protobuf import saved_model_pb2
from tensorflow.python.util import compat

with tf.Session() as sess:
    model_filename ='saved_model.pb'
    with gfile.FastGFile(model_filename, 'rb') as f:
        data = compat.as_bytes(f.read())
        sm = saved_model_pb2.SavedModel()
        sm.ParseFromString(data)
        g_in = tf.import_graph_def(sm.meta_graphs[0].graph_def)

LOGDIR='.'
train_writer = tf.summary.FileWriter(LOGDIR)
train_writer.add_graph(sess.graph)
```

The function `layers` is implemented correctly.

Overall you did a good job implementing `layers` for the model. Let me illustrate the pros of the architecture you chose.

Pros

- Correctly combined new layers with **VGG16** to construct your model.
- Skip connections are used from previous `vgg_layer3_out` and `vgg_layer4_out` layers. The authors of original paper highly suggest to use these skip connections to improve segmentation accuracy.



- Correctly added Deconvolution or Transpose Strided layers on top of provided VGG model.
- Used custom weight initialization. Xavier init is also proposed to work good when working with FCNs.

The function `optimize` is implemented correctly.

Clean and concise.
Good choice to use Adam. 👍

The function `train_nn` is implemented correctly. The loss of the network should be printed while the network is training.

Neural Network Training

The number of epoch and batch size are set to a reasonable number.

Given the network architecture, the hyperparams are reasonable. Both epochs and batch size are aligned to as proposed in original implementation.

Tips

- An important point to note is, batch size and learning rate are linked. If the batch size is too small then the gradients will become more unstable and would need to reduce the learning rate.

On average, the model decreases loss over time.

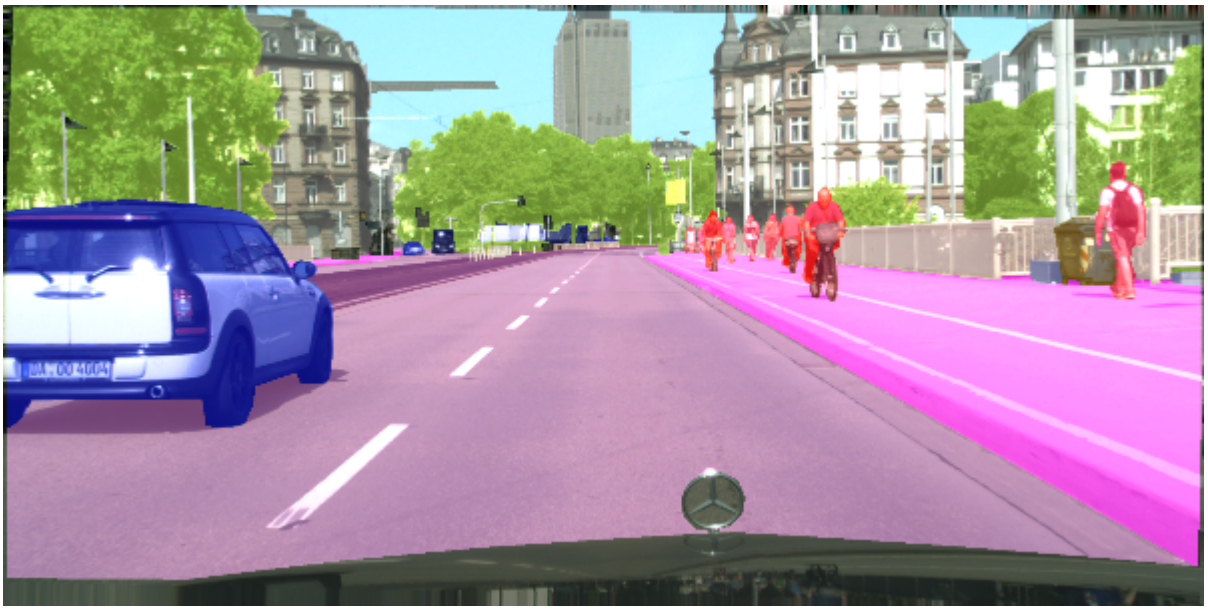
The project labels most pixels of roads close to the best solution. The model doesn't have to predict correctly all the images, just most of them.

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.

Woah ! Now that is something 👍
This finetuned VGG model is segmenting lanes to human-level accuracy. Wonder what will happen when trained on a much deeper network (think resnet?).

Do check out this talk from authors of the original paper and also the corresponding paper for more in depth details.

Further Experimentation: If you wish to work on a challenging dataset, you'll enjoy Cityscapes dataset. It has fine image annotations for 29 classes of objects. The images are video frames taken in German cities and there is around 11GB of them. This sample comes from the City Scapes dataset.



Reference

DOWNLOAD PROJECT

RETURN TO PATH