Semantic Segmentation

Building Neural Network

The file main.py contains `load_vgg, layers, optimize and train_nn`. I implemented them as was described in project walkthrough video. But I had to add other functions such ass `generate_protobuf, load_graph` and `run_graph`, they will be discussed soon .

Training Neural Network

I used AWS instance for nn training. But there were some limitations of memory, so I reduced the number of batches from 5 to 2. I used g2.2 instance and the nn was trained after 2 hours with 30 epochs, I noticed that after 30 epoch the loss did not reduce drastically, so I did not see the reason for the larger amount of epochs. I also modified `train_nn function` and added `tf.train.Saver().save` function.

Optimising Neural Network

After I saved model I generated protobuf file with the help of generate_protobuf function. You can call this function from the shell like this:

```
python
from main import generate_protobuf
generate_protobuf()
```

It will generate `binary_fcn.pb` file which could be used later, but first let me explain how could you optimize it. Call

```
~/tensorflow/bazel-bin/tensorflow/python/tools/freeze_graph \
--input_graph=binary_fcn.pb \
--input_checkpoint=fcn.ckpt \
--input_binary=true \
--output_graph=frozen_fcn.pb \
--output_node_names=logits
```

This command will freeze graph and you could use it for testing. But if you add another one:

```
~/tensorflow/bazel-bin/tensorflow/python/tools/optimize_for_infe
rence \
--input=frozen_fcn.pb \
--output=optimized_fcn.pb \
--frozen_graph=True \
--input_names=image_input \
--output_names=logits
```

You will optimize the model and reduce the number of operations in the graph.

Miscellaneous

I tried to use **`udacity-carnd-advanced-deep-learning`** ami but faced some problems with tensorflow version, I had to install and activate my own environment, you could use it also, I added anaconda **`environment-gpu2.yml`** to project rep.

Results

After we trained, froze and optimized the model we could check its performance.

```
python
from main import run_graph
run_graph('optimized_fcn.pb')
```

I added `timeit` decorator to all functions, so you could see the difference in performance.

For some reason I faced OOM error on GPU instance, I tried to run graphs on my laptop but the memory consumption was too high, so I reduced the number of test images to five. I did not solve this issue due to fact that this is not the topic of this project.

Model	Description	Number of operations	Processing time in ms(g2.2 amazon GPU instance)
binary_fcn.pb	The file which would be generated after train() and generate_probuf function would be called.	1577	none
frozen_fcn.pb	Call freeze_graph operation	244	3436.40 2513.82 2602.50 2681.99 2747.47
optimized_fcn.pb	Call optimize_for_inference operation	199	2991.81 1859.05 1854.50 1938.93 1946.55

Images