I Have added two comments for students that took the deep learning path (we went over it during class, but it may be easier to copy and google lines of code to speed things up)

Extracting layers using tensorflow (to inject it into the smoothness analysis code)

Here there are two ways to extract layers from tensorflow.

First method: **as a return value from the model**.

As seen at MNIST example in the convolutional.py file in git, there is a "model" (defined in line 188) that defines the neural network that gets the input and return the logits layer:

```
return tf.matmul(hidden, fc2_weights) + fc2_biases
```

the call for the model object and retrieving the logits can be seen at line 230:

```
logits = model(train_data_node, True)
```

Therefore, if one wants to print additional layer (or layers) – he can just add them to the return value and catch them when you call them:
For example, in the discussed code, if somenone wants to retrieve the last conv layer he may change the line

```
return tf.matmul(hidden, fc2_weights) + fc2_biases
```

Into :

```
return tf.matmul(hidden, fc2_weights) + fc2_biases, conv
```

then, getting the new output is very simple:

changing

```
logits = model(train_data_node, True)
```

into:

```
logits, conv_returned = model(train_data_node, True)
```

The second method is done at activation time of the net and could be seen at the cifar10 example (also in the git).

as seen in cifar10_eval.py in line 85, we first need to get the structure of the layer we want to retrieve (in this case, local4):

```
local4 =sess.graph.get_tensor_by_name(r'local4/local4:0')
```

the name local4 was assigned in the cifar10.py code in line 252

```
with tf.variable_scope('local4') as scope:
  weights = _variable_with_weight_decay('weights', shape=[384, 192],
                                        stddev=0.04, wd=0.004)
  biases = _variable_on_cpu('biases', [192], tf.constant_initializer(0.1))
  local4 = tf.nn.relu(tf.matmul(local3, weights) + biases, name=scope.name)
```

Note that we have `r'local4/` because of the scope and the rest `local4:0'` because of `name=scope.name`.

Now, to get the values in local4, just add then as list into the run function, and place values to catch them as seen in line 100 at cifar10_eval.py:

```
local4s , predictions = sess.run([local4, top_k_op ] )
```

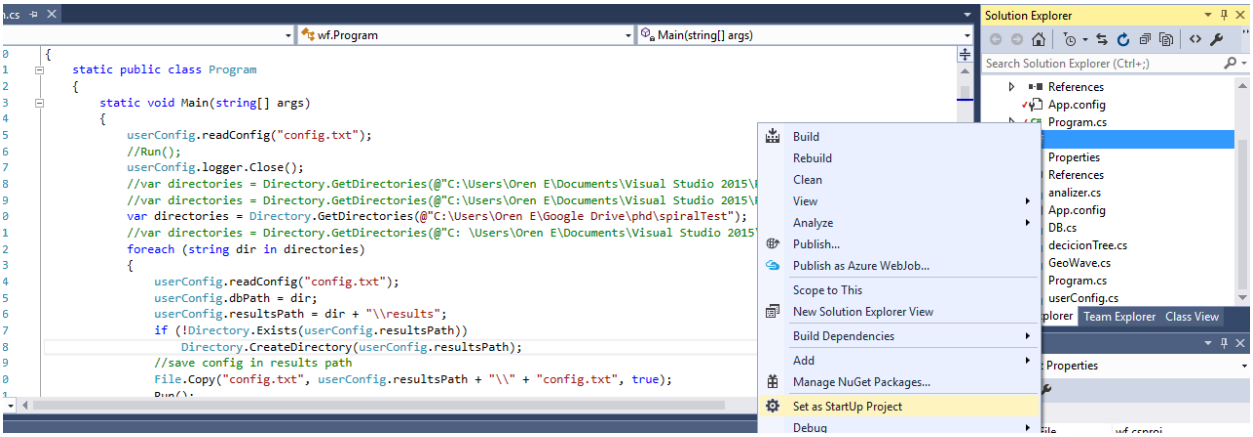
once you have values (usually in batch) you could either open a scv file for writing and print it batch by batch (google how to open file and print csv using python) or, you could append each batch during iteration into numpy array (actually matrix) and then only print it once the loop of extracting images representation ends.

Evaluating smoothness of many layers

Obviously, there is no sense of using the GUI for running 80 layers and you will need to use a for loop.

To do this go to the solution explorer and set the wf project as default (instead of the config project).

As seen at the image below:



Now the code will start from the main (rather than CONFIG project) and you could use the loop in line 22.

Things you need to edit:

1. You will need to open the config.txt file and edit it manually (once for all layers) – it is typically located next to the exe file. It has all the relevant fields that needs to be defined for the wf.
2. Note that in the loop only two things are modified:
    a. The directory – **you will need to enter your root directory** (of all layer directories) – this is done in line 20 (you can see I did it in the spiral example, but it may contain layers just as well)
    b. The path for writing the results - (you can keep line 26 as is)

That is it,

The results of all layers will be saved into the input directory you gave under “results” folder.

Good luck!