1. Set up the code

Before started, please take a look on the Python file, the user will need to set these values properly to run the Python code.

```
14  # choose action to train/create or test the model?
15  # notice: if the user only needs to testModel, plea
16  trainModel = True
17  testModel = True
```

These two boolean variables will tell the python file what to do:

If **trainModel** == True, then the Python will create / train / and save a model

If **testModel** == True, then the Python will test a given model on our HW test samples, which includes 195 png files

```
19  # file locations
20  # file name for create/train model
21  outputModelName = "D:\HW4_model/Model3LayersSuper.ckpt"
22  # file names for test model
23  inputModelName = "D:\HW4_model/Model3LayersSuper.ckpt"
24  annotationAddress = "D:\sample/annotation.txt"  # annotation.txt 's directory
25  outputAddress = "D:\sample/output.txt"  # output.txt 's directory, to print out result
26  testSampleAddress = "D:\sample/examples/"  # test sample png files's dirctory
```

As variable names suggest –

outputModelName is the directory (including file) name, where you want to save the trained model

inputModelName is the directory (including file) name, where you want to load and test model

annodatationAddress is a text file which contains name of png files, and their associated correct answers

outputAddress is the text file where you want to save the training info of your output model

testSampleAddress is the the given folder which contains 195 png files to test the model

Node: if you want to train and test your model at the same time, please make sure outputModelName is the same as inputModelName

```
28  # variables
29  steps = 20000
30  batchSize = 100
31  convolution = (1, 1)
32  kennelSize = (2, 2)
33  maxPoll = (2, 2)
34  #Optimizer = AdamOptimizer
35  #https://www.tensorflow.org
36  learningRate = 0.001
37  layer1Feature = 16
38  layer1Patch = 5, 5
39  layer2Feature = 32
40  layer2Patch = 5, 5
41  hiddenLayer = 100  # the m
42  dropoffRate = 0.5  # reduc
43  layer3Feature = 64
44  layer3Patch = 5, 5
```

These are the variables which set up the architecture of your model, and control the training of your model, and will eventually determine the specs of your model.

I did not list all of the variables which influence your model, for a more complete description, visits the tersorflow official guide.

2. Test of Different Architectures

The complete specs and test outcomes of models I set up and tested can be found in the Excel File **Model specs & logs.xlsx**

1. First, I tried the original setting of the model guided by tensorflow official tutorial. Its specs are:

	modelDefault.ckpt
Steps	3000
BatchSize	50
Convlution	1,1
KennelSize	2,2
MaxPool	2,2
Optimizer	AdamOptimizer
Learning Rate	0.0001
Layer 1 featur	32
Layer 1 patch	5
Layer 2 featur	64
Layer 2 patch	5
Hidden	1024
Dropoffrate	0.5
Layer 3 featur	NA
Layer 3 patch	NA

   Although after 3000 steps' training, the accuracy rate of the model reached 98% on the online test sample, its accuracy rate on the test sample created by our classmates is only 49%.
2. Second, I tried GradientDescentOptimizer which was used in the beginner's model on the official tutorial.
   The outcome is bad, with accuracy on our own test sample of 6%.
3. The third major model I tested is to micro tune BatchSize, Learning Rate, and the hidden layer number. Specs are:

	modelHiddenLayer.ckpt
Steps	3000
BatchSize	**100**
Convlution	1,1
KennelSize	2,2
MaxPool	2,2
Optimizer	AdamOptimizer
Learning Rate	**0.0010**
Layer 1 feature	32
Layer 1 patch	5
Layer 2 feature	64
Layer 2 patch	5
Hidden	**100**
Dropoffrate	0.5
Layer 3 features	
Layer 3 patch	

Although the change is not major, its performance is much better than the original one.

After 3000 steps' training, its accuracy on the online test sample reaches 99.21%, and its accuracy on our own created test sample reaches **72.82%**.

The result is satisfactory.

The major factor which distinguishes this model from the original one is its hidden layer numbers, which is reduced from 1024 to 100.

Reason behind it is that: our own created test sample is very different from the online training sample and test sample. Thus by reducing the hidden layer numbers, the model's ability to generalize is enhanced, thus perform much better on irregular hand writing digits by our students.

4. Before reaching my final architecture of the model, I also tried many other tunes, which can be found in the excel file. (I also tried change patch sizes, max_pool, batchSizes, etc, but did not include these in the excel file, because the outcome of these changes are not positive).

   One conclusion is that the training steps influence on the model's performance is not linear. The worse the architecture is, the bigger the training steps impact on the model is. In other words, if the model's architecture is already well designed, then the difference between training the model for 3000 steps and 20000 steps is not as big as I expected.

So I infer that the online training sample itself is flawed if the final purpose is to test the model on sample created by ourselves. Thus it is very hard to push the model's accuracy rate above 90%.

3. My Final Architecture

After reaching the conclusion that:

1. our sample is different from online test sample
2. To improve the model's accuracy on our own test sample, we need to improve the model's ability to generalize

The goal of the rest of my work is much clearer, which is to improve the model's ability to generalize.

The approach I adopted based on the last model I mentioned above, is to make the model longer and skinnier, which is to –

1. Add a third convolutional layer
2. Do not further increase the features on all 3 convolutional layers.
   I kept the features of the final layer to be maintained as 64
3. Keep the hidden layer number low as 100

A tricky part is to choose proper **max_Pool** values for 3 convolutional layers.

Because the size of our image are 28x28, so actually the choice are limited. The original model chooses **max_pool_2x2** for both layer 1 and layer 2, the final result image size becomes 7x7. Given the fact, we can only add a **max_pool_1x1** for one of the three layers, then the problem becomes to test to apply the max_pool_1x1 on which layer can we get the optimal result.

I tested all three possible choices. The best option is to apply the max_pool_1x1 on the third convolutional layer.

The final architecture and specs of my model is:

	model3Layer.ckpt	
Steps	3000	
BatchSize	**100**	
Convlution	1,1	
KennelSize	2,2	
MaxPool	2,2	
Optimizer	AdamOptimizer	
Learning Rate	**0.0010**	
Layer 1 feature	**16**	
Layer 1 patch	5	
Layer 2 feature	**32**	
Layer 2 patch	5	
Hidden	**100**	
Dropoffrate	0.5	
Layer 3 features	**64**	
Layer 3 patch	**5**	
	layer3_maxPool = 1,1	

After training for only 3000 steps, its test accuracy on the online test sample is which is probably better than most models I found online. Its accuracy on recognize our own hand writing digits reaches **87.18%**, consider it was only trained for 3000 steps, the outcome is pretty good!

I also tried train this model for 20000 steps, its performance on reading our own hand writing digits reaches **89.74%**.