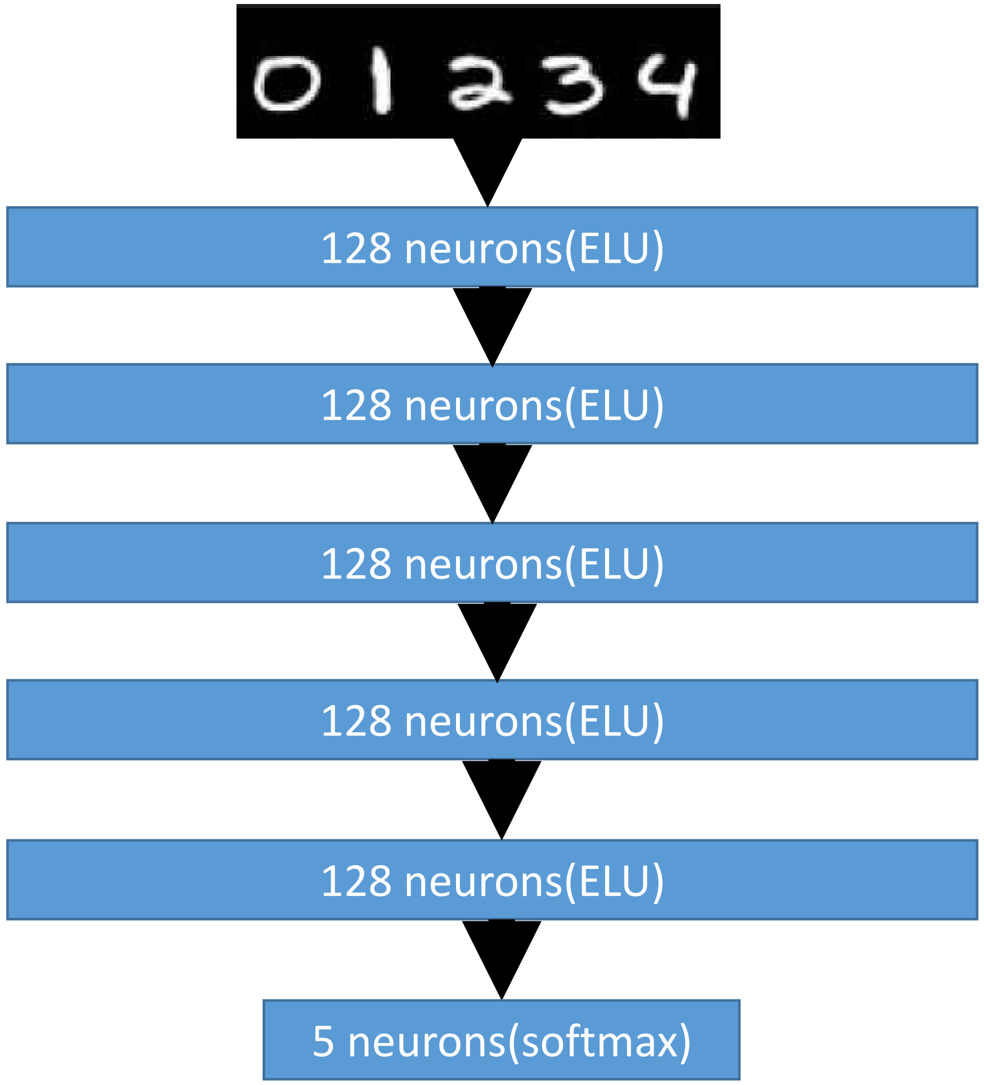
HW3 Transfer Learning

Recall your HW2?

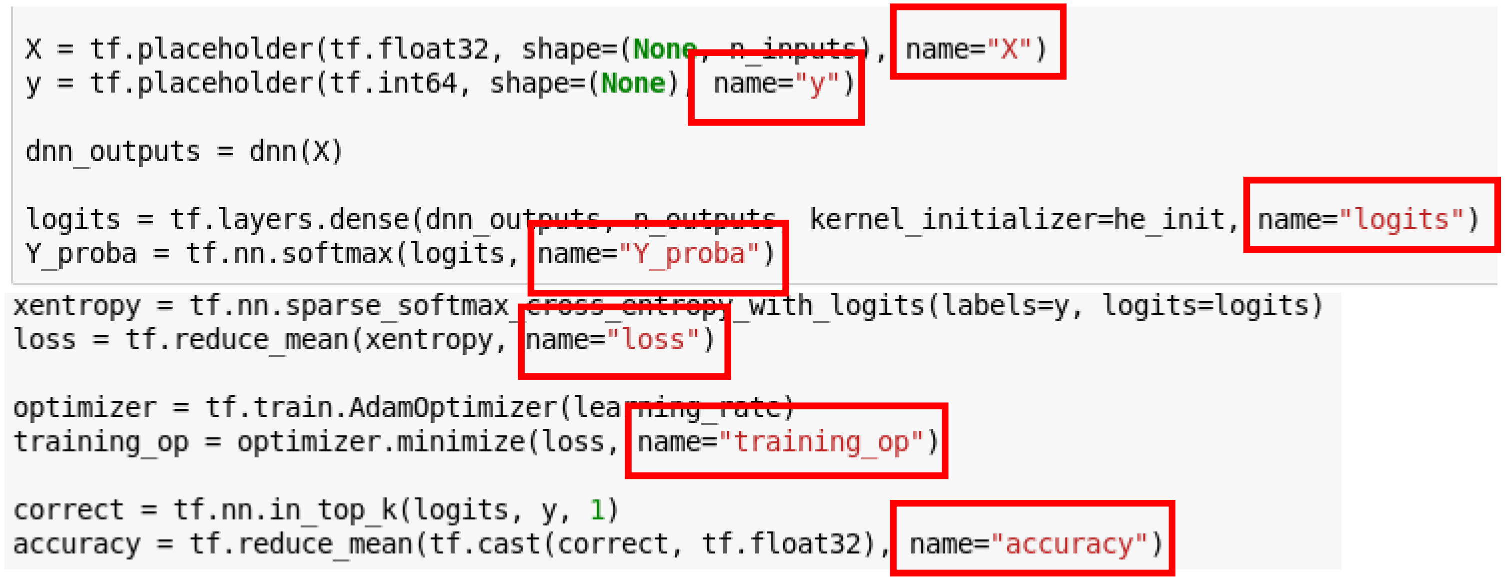
- Build a DNN with five hidden layers of 128 neurons each
- Training on MNIST but only on digits 0 to 4

- We’ll reuse your model in HW2



Recall your HW2?

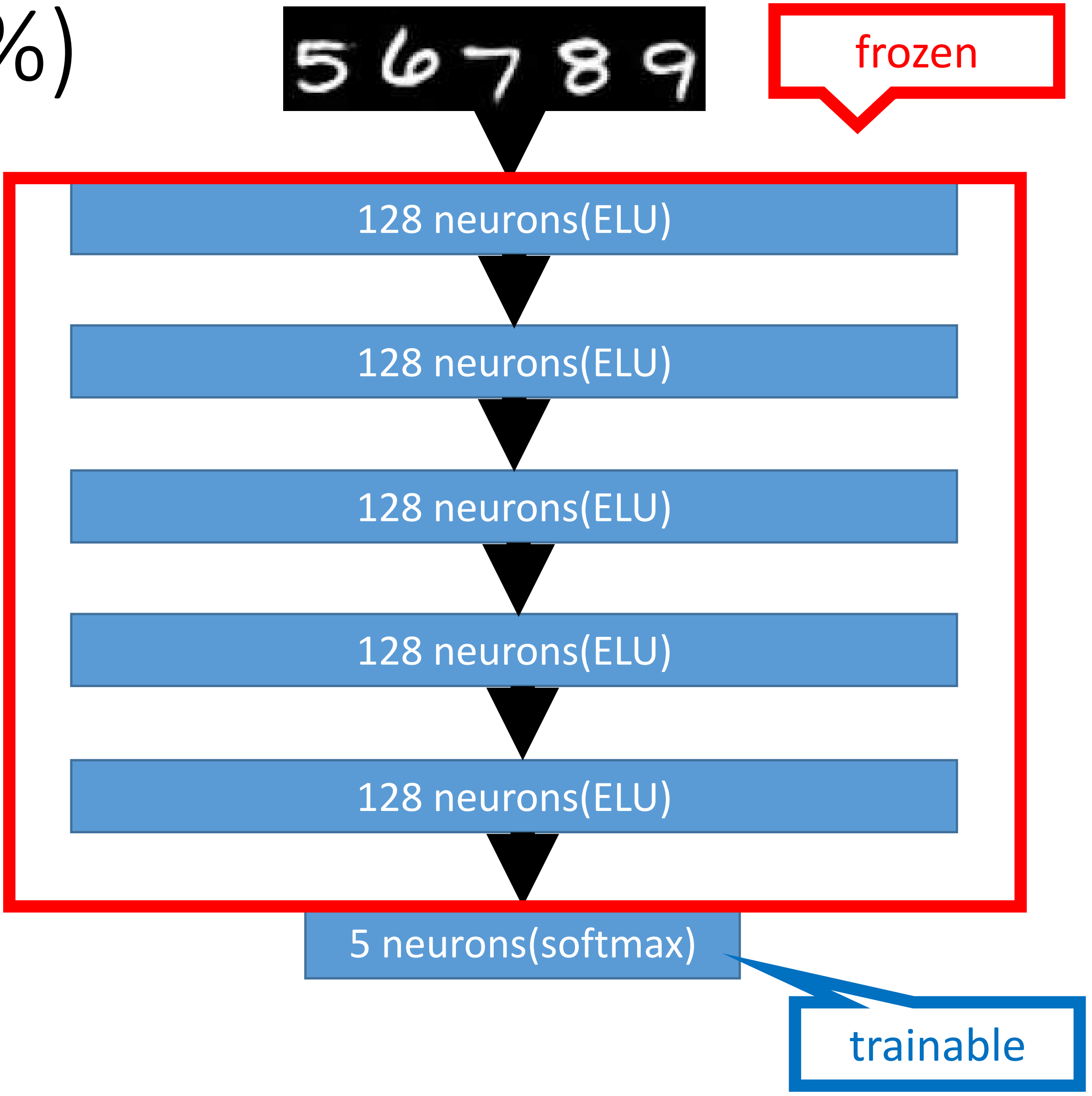
- Before you do HW3, check HW2 if you didn't give the "name" of each layer and function.
- If not, give the name and rerun HW2 again☺
  - The name of functions and layers are arbitrary for you
  - screenshot below is an example



```
X = tf.placeholder(tf.float32, shape=(None, n_inputs), name="X")
y = tf.placeholder(tf.int64, shape=(None), name="y")

dnn_outputs = dnn(X)

logits = tf.layers.dense(dnn_outputs, n_outputs, kernel_initializer=he_init, name="logits")
Y_proba = tf.nn.softmax(logits, name="Y_proba")
xentropy = tf.nn.sparse_softmax_cross_entropy_with_logits(labels=y, logits=logits)
loss = tf.reduce_mean(xentropy, name="loss")

optimizer = tf.train.AdamOptimizer(learning_rate)
training_op = optimizer.minimize(loss, name="training_op")

correct = tf.nn.in_top_k(logits, y, 1)
accuracy = tf.reduce_mean(tf.cast(correct, tf.float32), name="accuracy")
```

HW3.1 Softmax only(30%)

- Goal:
  - Load your model in HW2
  - Freeze the hidden layers 1~5
  - Only train softmax layer with digits 5~9

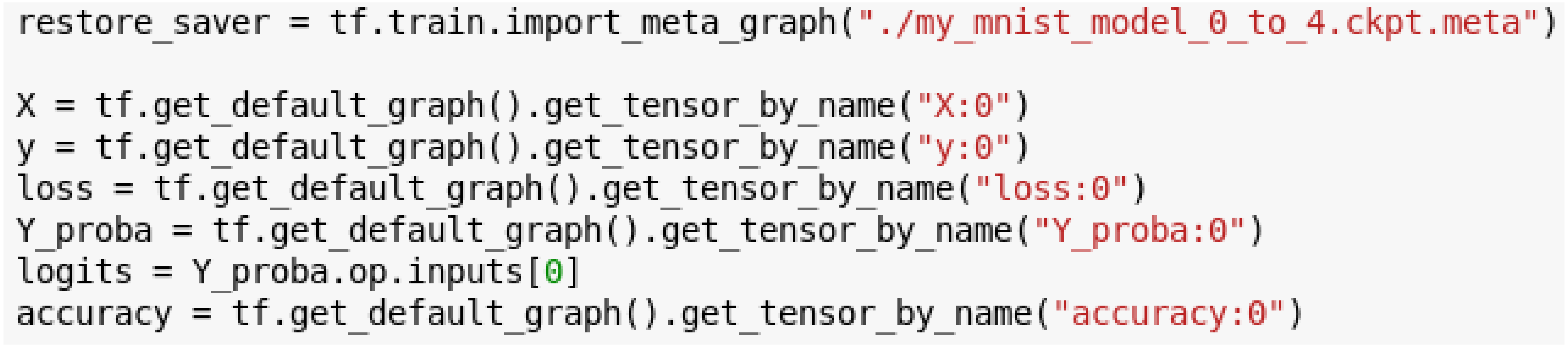


HW3.1 Softmax only(30%)

- Spec:
  - Keep only 100 instances per class in the training set (digits 5~9)
  - Keep only 30 instances per class in the validation set (digits 5~9)
  - Use full instances per class in testing set (digits 5~9)
    - TA already done three points above in sample code
  - Training 1000 epochs
  - Early stop if no progress in 20 epochs
  - Checkpoint name: Team01_HW3_1.ckpt
  - Get 0% in this part if violate any point above

HW3.1 Softmax only(30%)

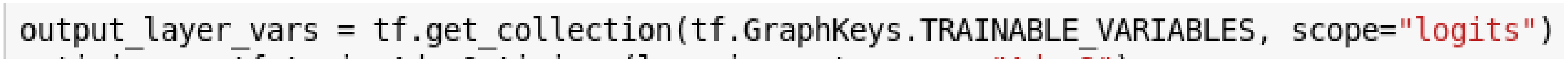
- Step 1:How to get tensor from HW2 model
    - Don’t just copy and paste because your naming may different



```
restore_saver = tf.train.import_meta_graph("./my_mnist_model_0_to_4.ckpt.meta")

X = tf.get_default_graph().get_tensor_by_name("X:0")
y = tf.get_default_graph().get_tensor_by_name("y:0")
loss = tf.get_default_graph().get_tensor_by_name("loss:0")
Y_proba = tf.get_default_graph().get_tensor_by_name("Y_proba:0")
logits = Y_proba.op.inputs[0]
accuracy = tf.get_default_graph().get_tensor_by_name("accuracy:0")
```

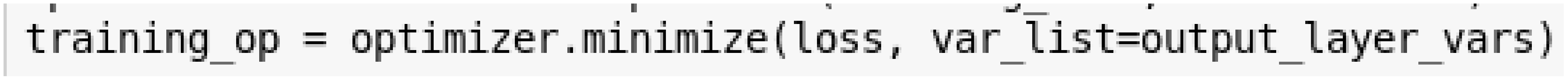
HW3.1 Softmax only(30%)

- Step 2:Get the softmax layer



```
output_layer_vars = tf.get_collection(tf.GraphKeys.TRAINABLE_VARIABLES, scope="logits")
```

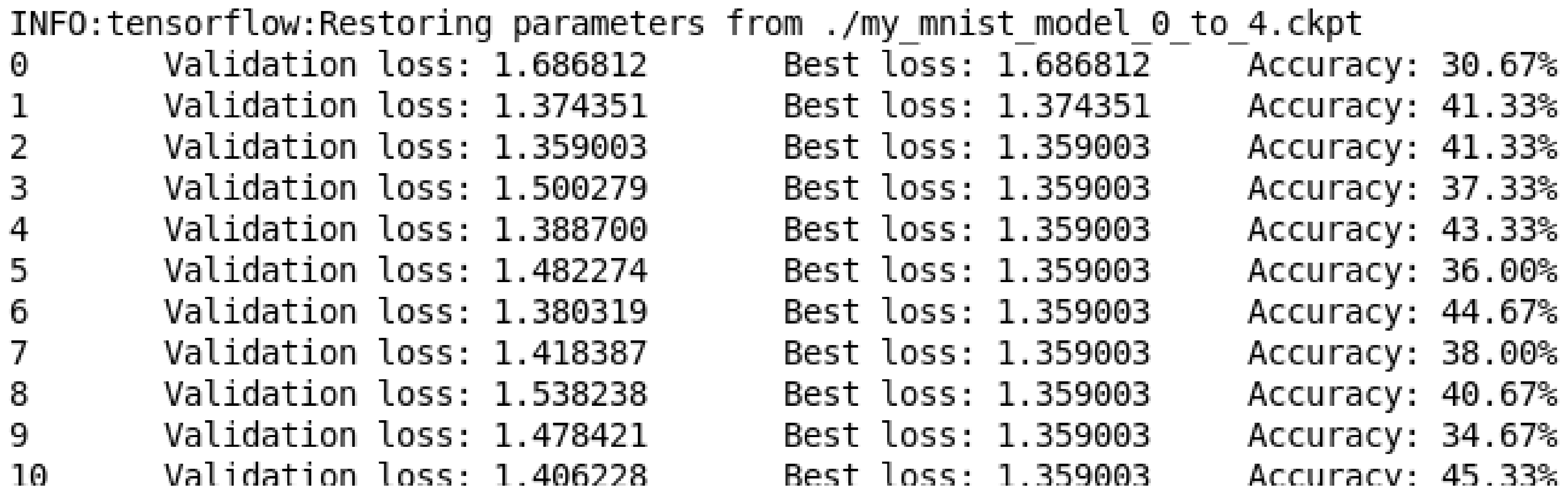
- Step 3
  - Exclude other layer’s variables from the optimizer's list of trainable variables
  - Keep only the softmax trainable variables



```
training_op = optimizer.minimize(loss, var_list=output_layer_vars)
```

HW3.1 Softmax only(30%)

- Step 4:Start training and print every epoch
  - Bad accuracy is normal☺



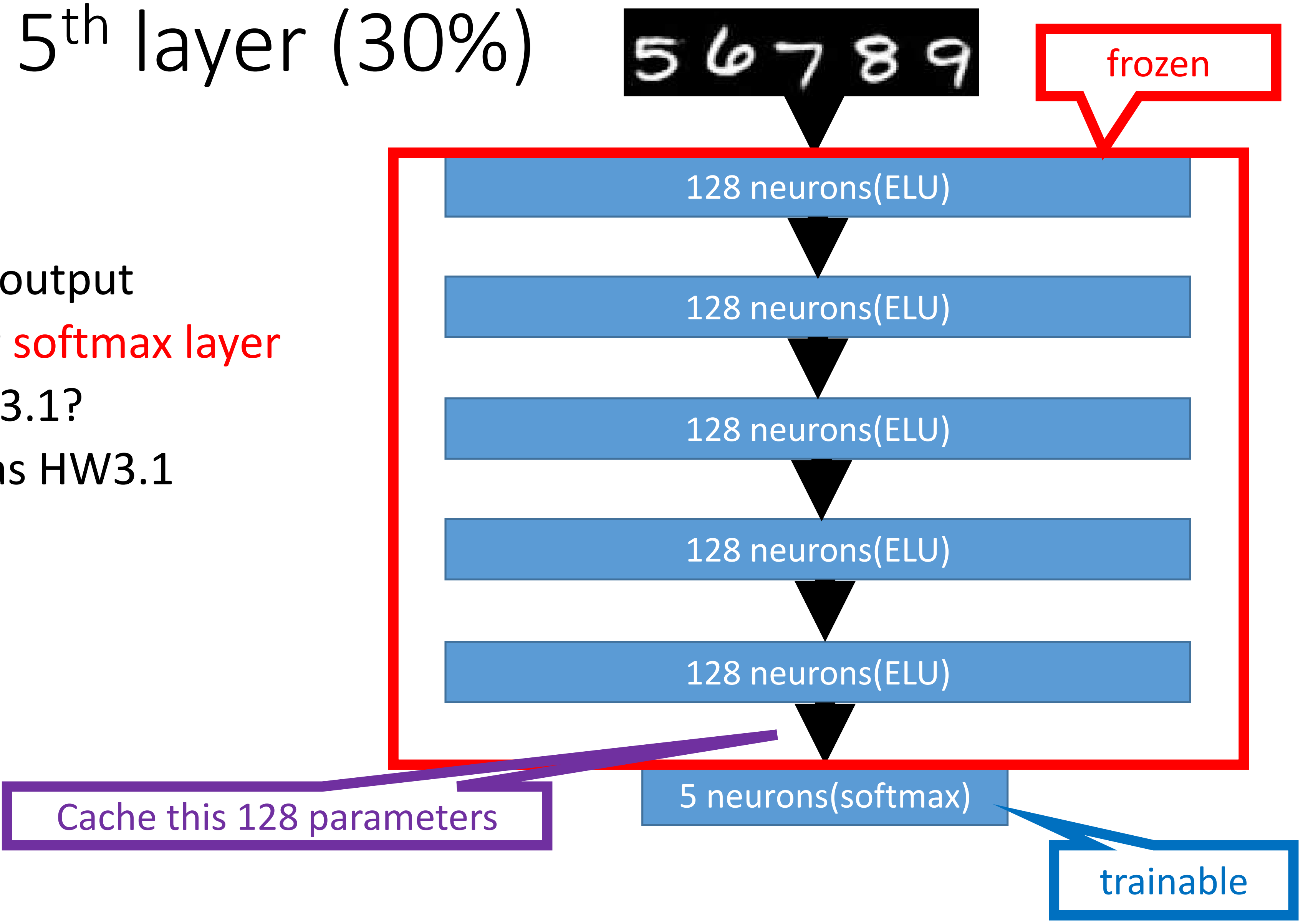
```
INFO:tensorflow:Restoring parameters from ./my_mnist_model_0_to_4.ckpt
0       Validation loss: 1.686812       Best loss: 1.686812     Accuracy: 30.67%
1       Validation loss: 1.374351       Best loss: 1.374351     Accuracy: 41.33%
2       Validation loss: 1.359003       Best loss: 1.359003     Accuracy: 41.33%
3       Validation loss: 1.500279       Best loss: 1.359003     Accuracy: 37.33%
4       Validation loss: 1.388700       Best loss: 1.359003     Accuracy: 43.33%
5       Validation loss: 1.482274       Best loss: 1.359003     Accuracy: 36.00%
6       Validation loss: 1.380319       Best loss: 1.359003     Accuracy: 44.67%
7       Validation loss: 1.418387       Best loss: 1.359003     Accuracy: 38.00%
8       Validation loss: 1.538238       Best loss: 1.359003     Accuracy: 40.67%
9       Validation loss: 1.478421       Best loss: 1.359003     Accuracy: 34.67%
10      Validation loss: 1.406228       Best loss: 1.359003     Accuracy: 45.33%
```

HW3.1 Softmax only(30%)

- Grading:
  - Commet:10%
  - Get tensor from HW2 and set softmax trainable: 10% (Step 1~3)
  - Training progress and print result per epoch: 10% (Step 4)

HW3.2 Cache 5th layer (30%)

- Goal:
  - Cache the 5th layer output
  - Reuse it on training softmax layer
  - Is it faster than HW3.1?
  - Code is 87% same as HW3.1



HW3.2 Cache 5th layer (30%)

- Spec:
  - Keep only 100 instances per class in the training set (digits 5~9)
  - Keep only 30 instances per class in the validation set (digits 5~9)
  - Use full instances per class in testing set (digits 5~9)
    - TA already done three points above in sample code
  - Training 1000 epochs
  - Early stop if no progress in 20 epochs
  - Checkpoint name: Team01_HW3_2.ckpt
  - Get 0% in this part if violate any point above

HW3.2 Cache 5th layer (30%)

- Step 1: Get tensor from HW2 model
- Step 2: Get the softmax layer
- Step 3: Exclude other layer’s variables and keep only the softmax trainable variables
- Step 3.5: Cache 5th layer output before training
- Step 4:Start training and print every epoch
    - Bad accuracy is normal☺

HW3.2 Cache 5th layer (30%)

- Step 3.5: Cache 5th layer output before training
  - Get 5th layer’s tensor
  - Feed training set and validation set into 5th layer (and 5th layer will recursive call 4th layer and recursive call 3rd layer …)
  - Save the parameters per sample
  - Feed this parameters as input when you train softmax layer

HW3.2 Cache 5th layer (30%)

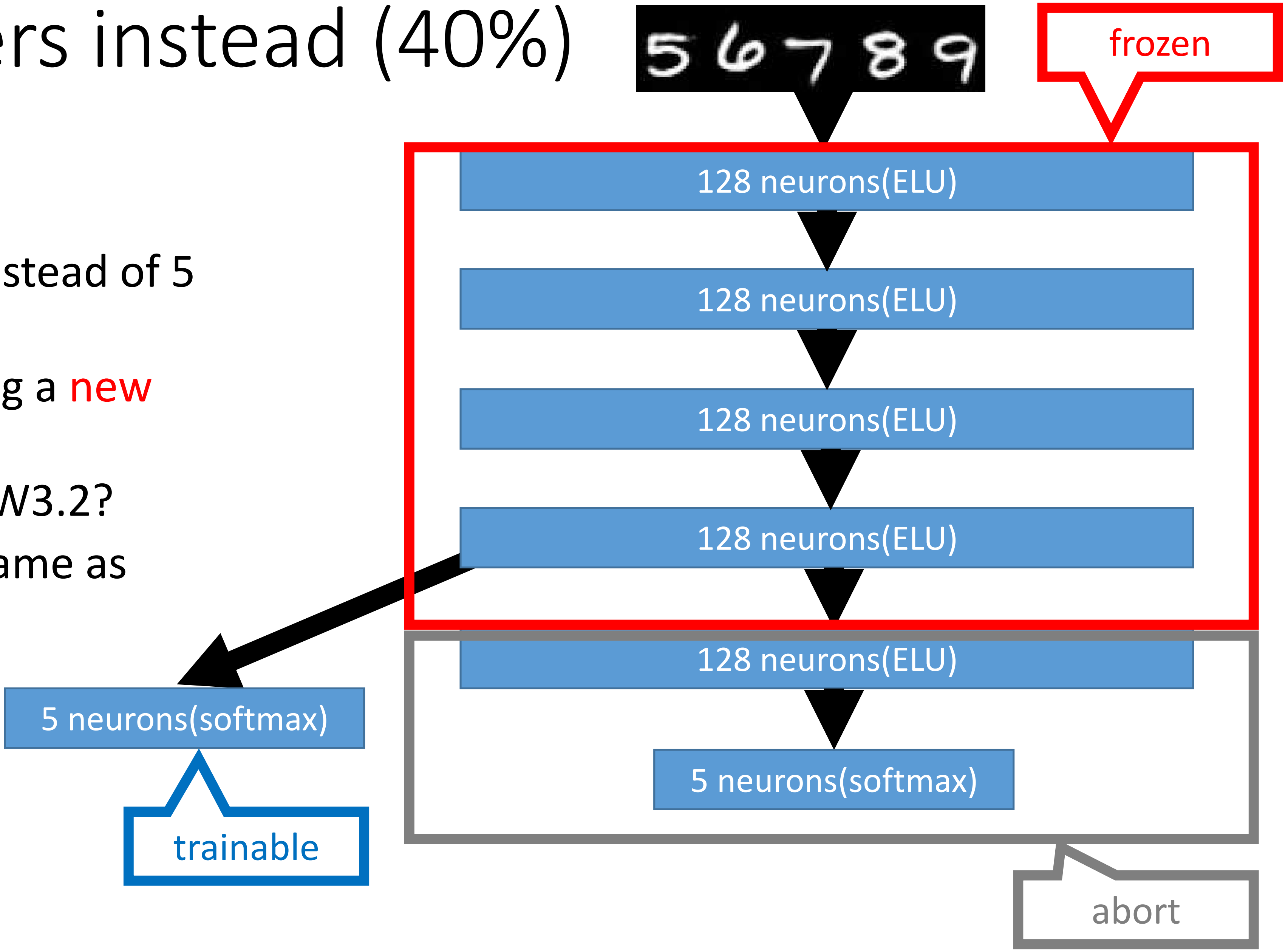
- Add time.time() at the top and the end of training processes in 3.1 and 3.2
    - t0 = time.time(), t1 = time.time()
    - t1 - t0 = your training time

HW3.2 Cache 5th layer (30%)

- Grading
  - Commet:10%
  - Get 5th layer tensor from HW2 and feed as input of softmax layer: 10% (Step 3.5)
  - Print training time in HW3.1&3.2, show HW3.2 faster than HW3.1: 10%

HW3.3 4 layers instead (40%)

- Goal:
  - Use the 4 layers instead of 5 frozen layer
  - Reuse it on training a new softmax layer
  - Is it better than HW3.2?
  - Code is still 87% same as HW3.1&3.2



HW3.3 4 layers instead (40%)

- Step 1: Get tensor from HW2 model
- Step 2: Get 4th layer output before training
- Step 3: Exclude other layer's variables and keep only the softmax trainable variables
- Step 4:Start training and print every epoch

HW3.3 4 layers instead (40%)

- Step 2: Get 4th layer output before training
  - Get 4th layer's tensor
  - Add a new softmax layer at the end

HW3.3 4 layers instead (40%)

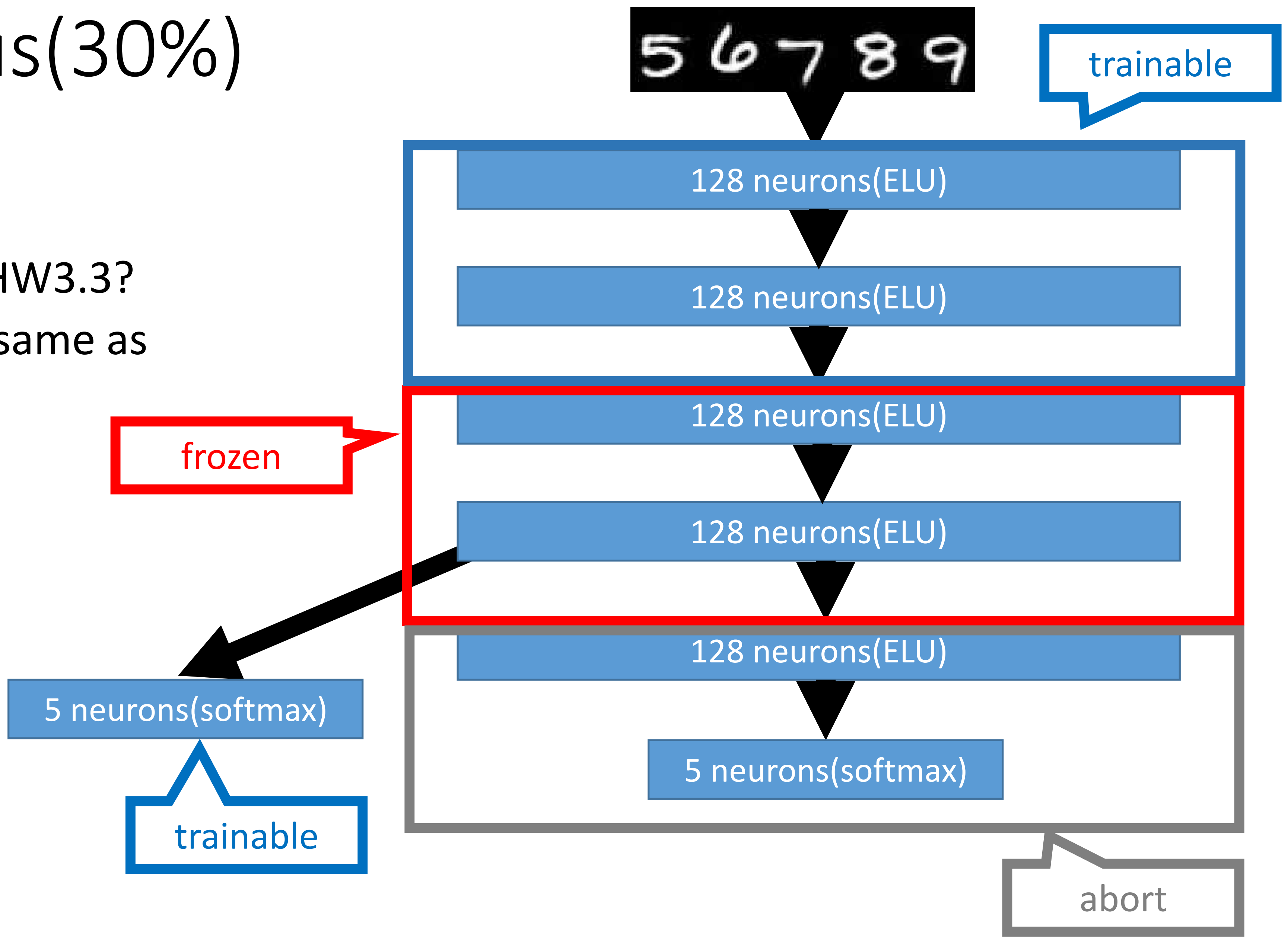
- Spec:
  - Keep only 100 instances per class in the training set (digits 5~9)
  - Keep only 30 instances per class in the validation set (digits 5~9)
  - Use full instances per class in testing set (digits 5~9)
    - TA already done three points above in sample code
  - Training 1000 epochs
  - Early stop if no progress in 20 epochs
  - Checkpoint name: Team01_HW3_3.ckpt
  - Get 0% in this part if violate any point above

HW3.3 4 layers instead (40%)

- Grading
  - Commet:10%
  - Get 4th layer tensor and add new softmax layer:10% (step 2)
  - Training progress and print result per epoch: 10% (step 4)
  - Accuracy better than HW3.1&3.2: 10%

HW3.4 Bonus(30%)

- Goal:
  - Is it better than HW3.3?
  - Code is still 87% same as HW3.1&3.2&3.3



HW3.4 Bonus(30%)

- Spec:
  - keep only 100 instances per class in the training set (digits 5~9)
  - keep only 30 instances per class in the validation set (digits 5~9)
  - Use full instances per class in testing set (digits 5~9)
    - TA already done three points above in sample code
  - Training 1000 epochs
  - Early stop if no progress in 20 epochs
  - Checkpoint name: Team01_HW3_4.ckpt
  - Get 0% in this part if violate any point above

HW3.4 Bonus(30%)

- Grading
  - Commet:10%
  - Training progress and print result per epoch: 10%
  - Accuracy better than HW3.3: 10%

Rule

- Deadline: 2018 01/05 23:59:59
- Naming: Team01_HW3.ipynb
- Copy will get 0 point
- Delay will get 0 point even 1 second
- Wrong naming will get 0 point