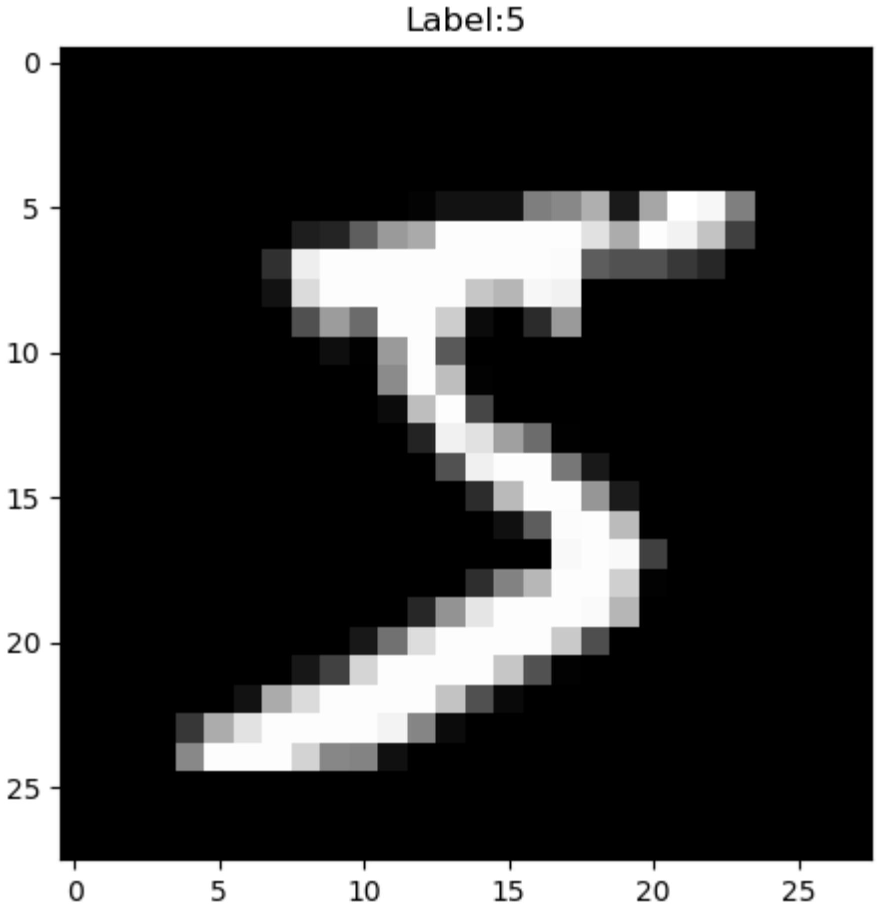
MNIST



- Shape of each data : [28, 28]
- Range : 0.0 to 1.0
- You can see the image of each data.

Code review

[Objective]

Your model should classifiy of the images into 10 classes (0~9).

[PyTorch Code structure]

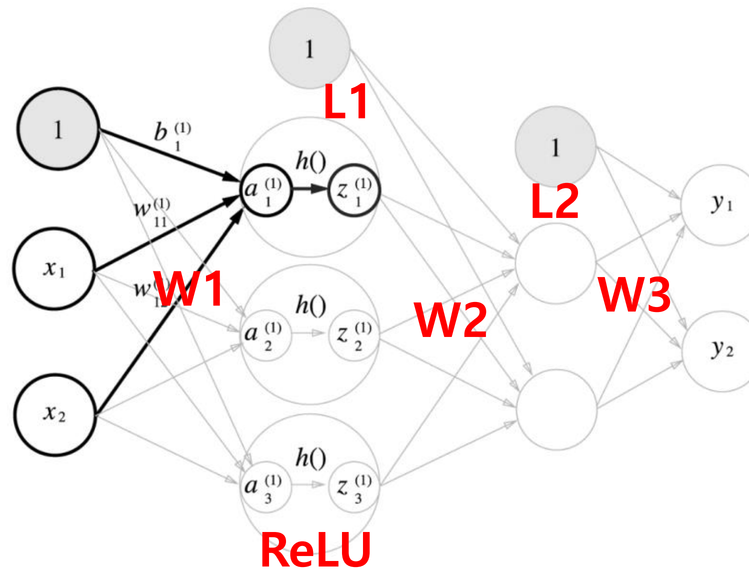
- **MNIST_train.py**
- **MNIST_model.py**
- **MNIST_evaluation.py**

[TensorFLow Code structure]

- **MNIST_train.py**
- **MNIST_eval.py**

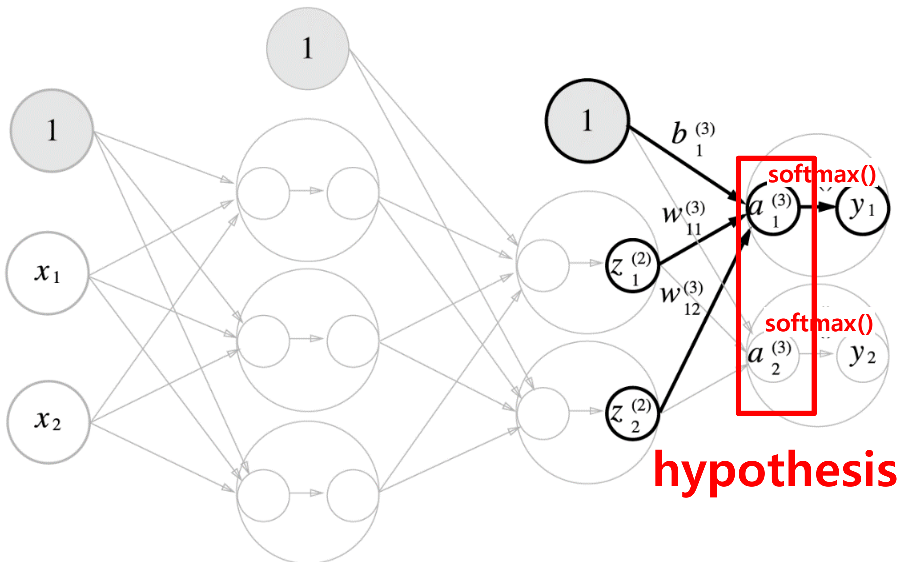
MLP model (Affine, Activation) - TensorFlow

```python
X = tf.placeholder(tf.float32, [None, 784], name="X")
Y = tf.placeholder(tf.float32, [None, 10], name="Y") # [0 0 0 0 0 0 0 0 1 0]

W1 = tf.get_variable("W1", shape=[784, 300])
b1 = tf.Variable(tf.random_normal([300]))
L1 = tf.nn.relu(tf.matmul(X, W1) + b1)

W2 = tf.get_variable("W2", shape=[300, 200])
b2 = tf.Variable(tf.random_normal([200]))
L2 = tf.nn.relu(tf.matmul(L1, W2) + b2)

W3 = tf.get_variable("W3", shape=[200, 10])
b3 = tf.Variable(tf.random_normal([10]))

hypothesis = tf.nn.xw_plus_b(L2, W3, b3, name="hypothesis")
```



MLP model (Softmax, Loss Function) - TensorFlow

```
cost =
tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=hypothesis, labels=Y))
```



MLP model (backpropagation, optimizer) - TensorFlow

```
learning_rate = 0.001

optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(cost)
```

- **AdaDelta**

This is an another upgraded version of Adagrad.

$$G = \gamma G + (1-\gamma)(\nabla_\theta J(\theta_t))^2$$

$$\Delta_\theta = \frac{\sqrt{s+\epsilon}}{\sqrt{G+\epsilon}} \cdot \nabla_\theta J(\theta_t)$$

$$\theta = \theta - \Delta_\theta$$

$$s = \gamma s + (1-\gamma)\Delta_\theta^2$$

s: step size (instead of learning rate)

- **Adam**

This is mixture of RMSProp and momentum. This is one of the **most popular** gradient descent optimization algorithms.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla_\theta J(\theta)$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)(\nabla_\theta J(\theta))^2$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

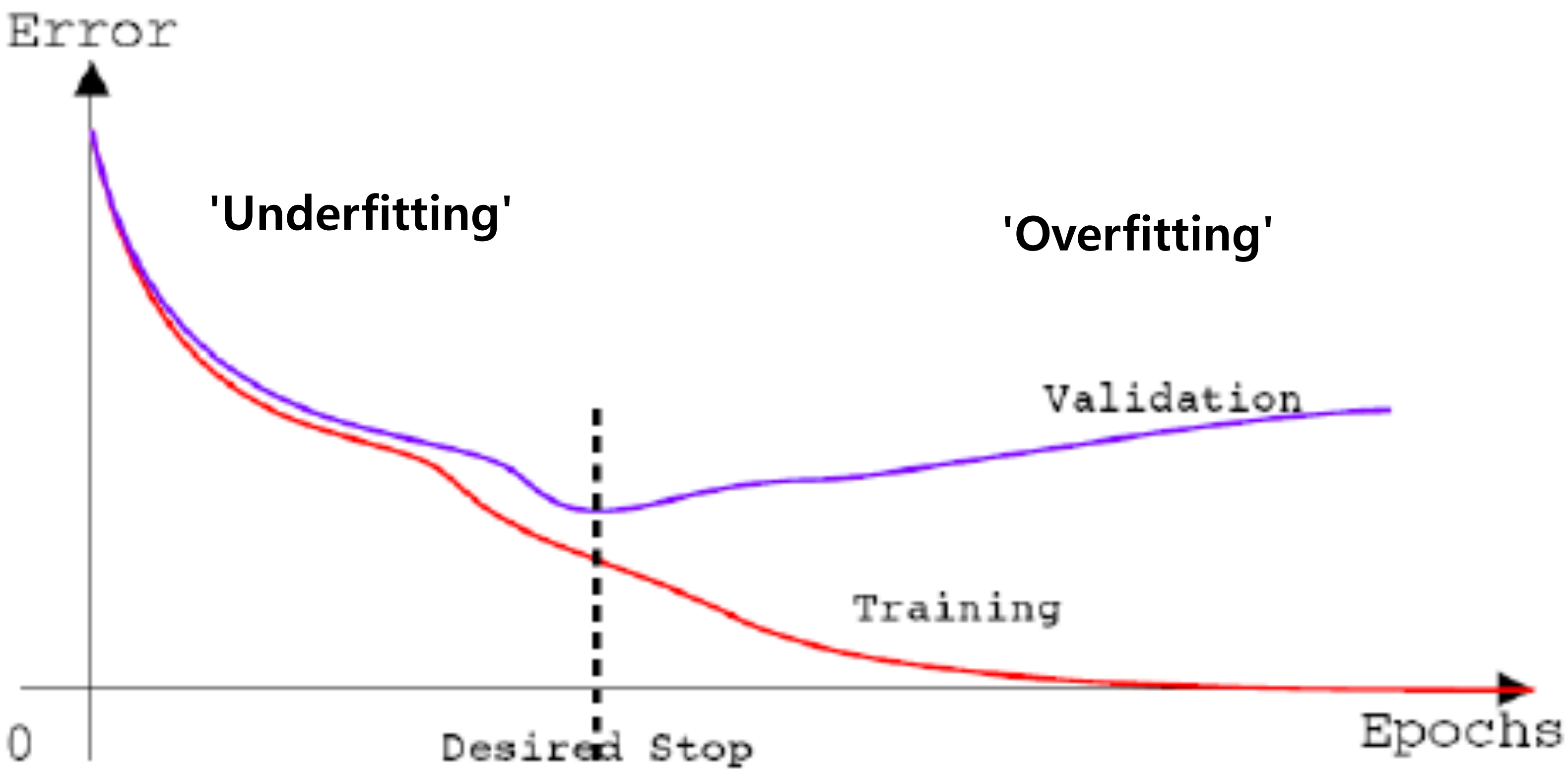
$$\theta = \theta - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}}\hat{m}_t$$

MLP training - TensorFlow

```
sess = tf.Session()
sess.run(tf.global_variables_initializer())

training_epochs = 30
batch_size = 100

max = 0
early_stopped = 0
for epoch in range(training_epochs):
    avg_cost = 0
    total_batch = int(mnist.train.num_examples / batch_size) #iteration 55000/ 100 = 550

    for i in range(total_batch):
        batch_xs, batch_ys = mnist.train.next_batch(batch_size) # (100, 784), (100, 10)
        feed_dict = {X: batch_xs, Y: batch_ys}
        c, _, a = sess.run([cost, optimizer, summary_op], feed_dict=feed_dict)
        avg_cost += c / total_batch
```

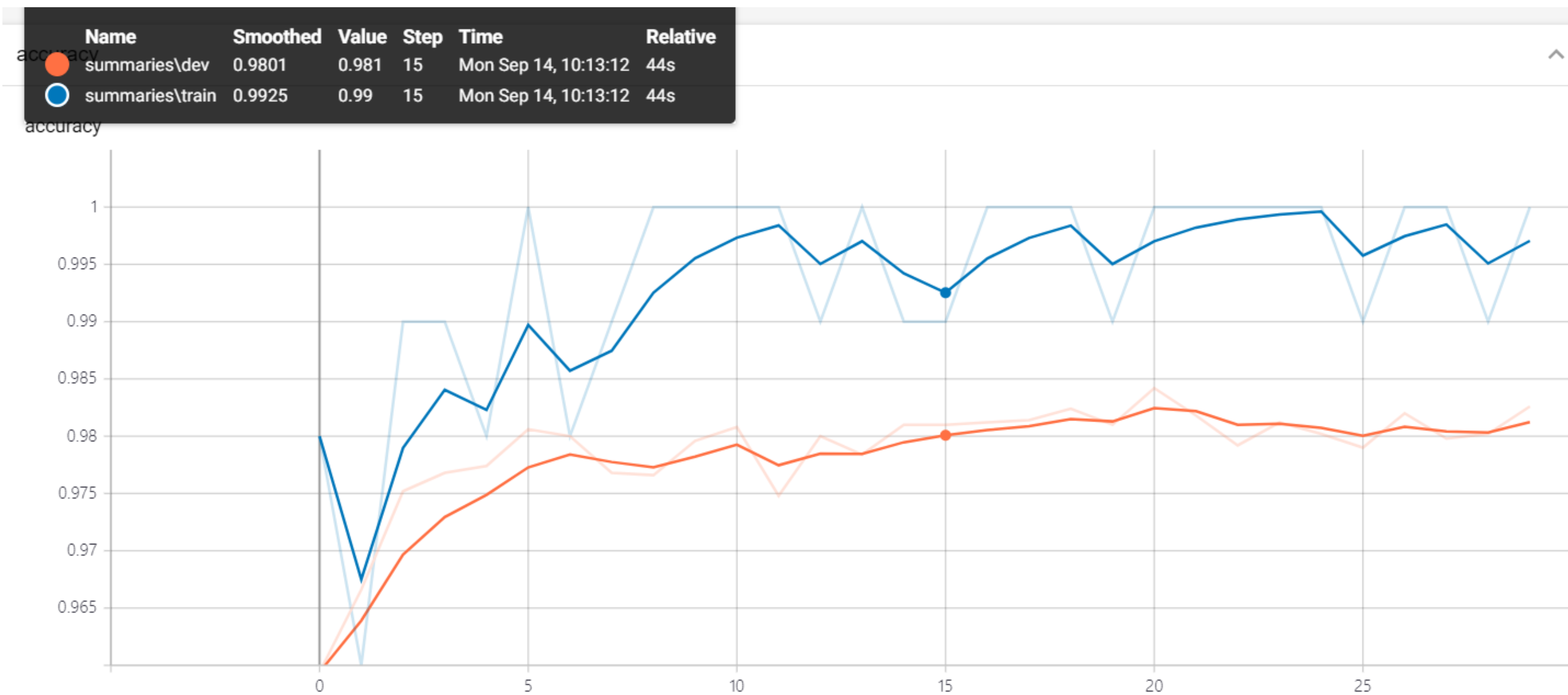
MLP training (early stopping) - TensorFlow

```
print('Validation Accuracy:', val_accuracy)
if val_accuracy > max:
    max = val_accuracy
    early_stopped = epoch + 1
    saver.save(sess, checkpoint_prefix, global_step=early_stopped)
```



Accuracy plot (Tensor board)- TensorFlow

- Terminal에서 가상환경 activate 확인
- tensorboard --logdir=
  C:₩Users₩82102₩PycharmProjects₩cose474₩MNIST₩runs₩1600045942 입력



MLP evaluation - TensorFlow

MNIST_eval.py

```
tf.flags.DEFINE_string("checkpoint_dir", "./runs/1600039587/checkpoints", "Checkpoint
directory from training run")

FLAGS = tf.flags.FLAGS
checkpoint_file = tf.train.latest_checkpoint(FLAGS.checkpoint_dir)
graph = tf.Graph()
with graph.as_default():
    sess = tf.Session()
    with sess.as_default():
        saver = tf.train.import_meta_graph("{}.meta".format(checkpoint_file))
        saver.restore(sess, checkpoint_file)

        X = graph.get_operation_by_name("X").outputs[0]
        Y = graph.get_operation_by_name("Y").outputs[0]
        hypothesis = graph.get_operation_by_name("hypothesis").outputs[0]
        correct_prediction = tf.equal(tf.argmax(hypothesis, 1), tf.argmax(Y, 1))
        accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
        test_accuracy = sess.run(accuracy, feed_dict={X: mnist.test.images, Y:
mnist.test.labels})
        print('Test Max Accuracy:', test_accuracy)
```

Assignment 2: MNIST classification

[Objective]

Your model should classifiy of the images into 10 classes (0~9).

[Requirements]

1. Implement 4-layer perceptron with Pytorch or Tensorflow.

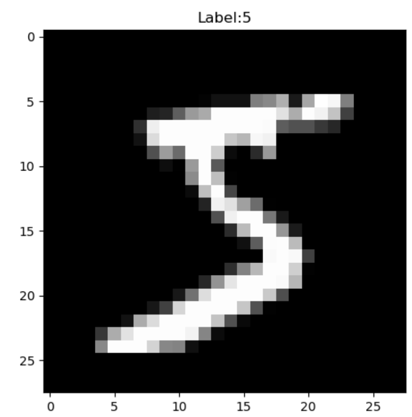
(Basic code is provided)

2. You should experiment with settings stated in the evaluation report, and report the result of each settings.

3. You should attach the plot of the validation dataset accuracy plot.

4. You should report the experimental results.

(all kinds of additional experiments are recommended)



model

"5!"

Assignment 2: MNIST classification

[Evaluation report]

MNIST Evaluation Report													
	Batch_size	Activation function	# of layers	Layer size	Epoch	Weight initialization	Optimizer	Learning rate	Weight decay	Dropout	training time	Early stopping epoch	Accuracy
Setting #1	200	ReLU	3	300, 200	30	x	Adam	0.001	X	0			
Setting #2	200	ReLU	4	200, 200, 200	100	x	Adam	0.001	X	0			
Setting #3	200	ReLU	4	600, 600, 800	100	x	Adam	0.001	X	0			
Setting #4	200	ReLU	4	200, 200, 200	100	He	Adam	0.001	X	0			
Setting #5	200	ReLU	4	200, 200, 200	100	He	Adadelta	0.001	X	0			
Setting #6	200	ReLU	4	200, 200, 200	100	He	Adam	0.001	O(lambda=0.01)	0			
Setting #7	200	ReLU	4	200, 200, 200	100	He	Adam	0.01	O(lambda=0.01)	0			
Setting #8	200	ReLU	4	200, 200, 200	100	He	Adam	0.01	O(lambda=0.01)	0.2			
...additional setting													
Validation dataset accuracy plot													

Setting #1 | Setting #2 | Setting #3 | Setting #4

[결과 정리]

Assignment 2: MNIST classification

- **Evaluation Criteria**

Simplicity	How concisely did you write the code? - 배점 6점 4 Layer: 4점 Weight initializer: 1점 Dropout: 1점 Weight decay: 1점
Performance	How well did the results of the code perform? - 배점 2점 - acc 97.5%이상 달성 시 만점
Brevity and Clarity	How concisely and clearly did you explain the results? - 배점 2점

Assignment 2: MNIST classification

- Due to : **~ 9.20(Sun)**

- Submission : Online submission on blackboard

- Your submission should contain
  1) The whole code of your implementation
  2) The evaluation report

- You must implement the components yourself!
- File name : StudentID_Name.zip