```
import tensorflow as tf
mnist = tf.keras.datasets.mnist


(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train = x_train / 255.0
x_test = x_test / 255.0
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.np
11493376/11490434 [==============================] - 0s 0us/step
```

```
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(512, activation=tf.nn.relu), # 784개의 뉴런에서 바로 10개의 확률을 추출하

  tf.keras.layers.Dense(10, activation=tf.nn.softmax) # 위에서 펼친 784개의 뉴런을 촘촘하게 거미줄
])
model.compile(optimizer='adam',# 이 최적화머신은 절반은 모멘텀(관성)으로 이루어져 있다.
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/ops/resou
Instructions for updating:
Colocations handled automatically by placer.
```

optimizer 즉, 최적화머신은 다양한 머신들이 있는데, Gradient Descent(경사하강법), adam(Adaptive Moment Estimation) 등 최적의 답을 찾아내는데 필요한 머신들로 구성되어있다. 결과에 따라서 최적화 머신을 다르게 하면 결과가 달라질 수 있다. adam이 optimizer중에는 제일 준수한 성능을 내는 것으로 알려져 있다.

adam이 어떻게 구성되어있는지 알아볼 필요가 있다.

http://shuuki4.github.io/deep%20learning/2016/05/20/Gradient-Descent-Algorithm-Overview.html

```
model.fit(x_train, y_train, epochs=5) # x_train 데이터와 y_train 데이터를 5번 반복하며 model에 fi
```

```
Epoch 1/5
60000/60000 [==============================] - 13s 216us/sample - loss: 0.2043 - acc: 0.93
Epoch 2/5
60000/60000 [==============================] - 12s 208us/sample - loss: 0.0816 - acc: 0.97
Epoch 3/5
60000/60000 [==============================] - 12s 208us/sample - loss: 0.0528 - acc: 0.98
Epoch 4/5
60000/60000 [==============================] - 13s 211us/sample - loss: 0.0367 - acc: 0.98
Epoch 5/5
60000/60000 [==============================] - 13s 212us/sample - loss: 0.0277 - acc: 0.99
<tensorflow.python.keras.callbacks.History at 0x7f436a865208>
```

```
model.summary() # W * X + B
# 입력값에 bias를 더한값과 가중치를 곱한 값이 파라미터가 된다.
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 784)               0
_________________________________________________________________
dense (Dense)                (None, 512)               401920
_________________________________________________________________
dense_1 (Dense)              (None, 10)                5130
=================================================================
Total params: 407,050
```

```
print("=======================================")
print("Layer 1 : ", (784 + 1) * 512)
print("Layer 2 : ", (512 + 1) * 10)
print("=======================================")
```

```
=======================================
Layer 1 :  401920
Layer 2 :  5130
=======================================
```

Training Accuracy

```
model.evaluate(x_train, y_train)
```

```
60000/60000 [==============================] - 3s 52us/sample - loss: 0.0236 - acc: 0.9922
[0.023623107492068085, 0.99223334]
```

Test Accuracy

```
model.evaluate(x_test, y_test)
```

```
10000/10000 [==============================] - 1s 55us/sample - loss: 0.0750 - acc: 0.9786
[0.07497045551972696, 0.9786]
```

Real World Challenge: Large difference between training and testing set accuracy