In [1]:
```
# Loading the MNIST dataset
from keras.datasets import mnist
import tensorflow as tf
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

from keras import models
from keras import layers
# Network architecture
network = models.Sequential()
network.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))
network.add(layers.Dense(10, activation='softmax'))
# The compilation step
network.compile(optimizer='rmsprop',
    loss='categorical_crossentropy',
    metrics=['accuracy'])
# Preparing image data
train_images = train_images.reshape((60000, 28 * 28))
train_images = train_images.astype('float32') / 255
test_images = test_images.reshape((10000, 28 * 28))
test_images = test_images.astype('float32') / 255
# Preparing the labels
#from keras.utils import to_categorical
train_labels = tf.keras.utils.to_categorical(train_labels)
test_labels = tf.keras.utils.to_categorical(test_labels)
# Fit the model to training data
network.fit(train_images, train_labels, epochs=10, batch_size=128)

#정확도가 달라지는 이유
#훈련 세트는 각 배치마다 가중치가 변경되기 때문에
#정확도는 훈련 중 모든 배치에 대한 평균일 뿐이다.
#반복 학습을 통해 손실은 감소하고, 정확도는 증가한다.
```

```
Epoch 1/10
469/469 [==============================] - 3s 6ms/step - loss: 0.2599 - accuracy: 0.9249
Epoch 2/10
469/469 [==============================] - 2s 5ms/step - loss: 0.1055 - accuracy: 0.9688
Epoch 3/10
469/469 [==============================] - 2s 5ms/step - loss: 0.0682 - accuracy: 0.9790
Epoch 4/10
469/469 [==============================] - 2s 5ms/step - loss: 0.0502 - accuracy: 0.9849
Epoch 5/10
469/469 [==============================] - 2s 5ms/step - loss: 0.0373 - accuracy: 0.9888
Epoch 6/10
469/469 [==============================] - 2s 5ms/step - loss: 0.0280 - accuracy: 0.9916
Epoch 7/10
469/469 [==============================] - 2s 5ms/step - loss: 0.0219 - accuracy: 0.9934
Epoch 8/10
469/469 [==============================] - 3s 6ms/step - loss: 0.0166 - accuracy: 0.9954
Epoch 9/10
469/469 [==============================] - 3s 6ms/step - loss: 0.0132 - accuracy: 0.9962
Epoch 10/10
469/469 [==============================] - 3s 6ms/step - loss: 0.0101 - accuracy: 0.9972
```

Out[1]:
```
<keras.callbacks.History at 0x2632068d670>
```

In [2]:
```
from keras.datasets import imdb
(train_data, train_labels), (test_data, test_labels) =  imdb.load_data( num_words=10000)

import numpy as np
# 입력 텍스트 vectorization
def vectorize_sequences(sequences, dimension=10000):
        results = np.zeros((len(sequences), dimension))
        for i, sequence in enumerate(sequences):
                results[i, sequence] = 1.
        return results

x_train = vectorize_sequences(train_data)
x_test = vectorize_sequences(test_data)
y_train = np.asarray(train_labels).astype('float32')
y_test = np.asarray(test_labels).astype('float32')

from keras import models
from keras import layers

model = models.Sequential()
model.add(layers.Dense(16, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='rmsprop', loss='binary_crossentropy',
        metrics=['accuracy'])

model.fit(x_train, y_train, epochs=6, batch_size=512)
```

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

#epoch를 4번에서 6번으로 변경후 학습을 시킨 결과
#위에서 언급했듯 손실은 감소하고 정확도는 증가한다.
```

```
Epoch 1/6
49/49 [==============================] - 1s 8ms/step - loss: 0.4597 - accuracy: 0.8266
Epoch 2/6
49/49 [==============================] - 0s 9ms/step - loss: 0.2652 - accuracy: 0.9063
Epoch 3/6
49/49 [==============================] - 0s 10ms/step - loss: 0.2026 - accuracy: 0.9298
Epoch 4/6
49/49 [==============================] - 0s 9ms/step - loss: 0.1685 - accuracy: 0.9417
Epoch 5/6
49/49 [==============================] - 0s 9ms/step - loss: 0.1470 - accuracy: 0.9477
Epoch 6/6
49/49 [==============================] - 0s 9ms/step - loss: 0.1271 - accuracy: 0.9572
782/782 [==============================] - 1s 982us/step - loss: 0.3280 - accuracy: 0.8774
```