Convolution Neural Network

한성대학교

정보시스템공학과 하태준

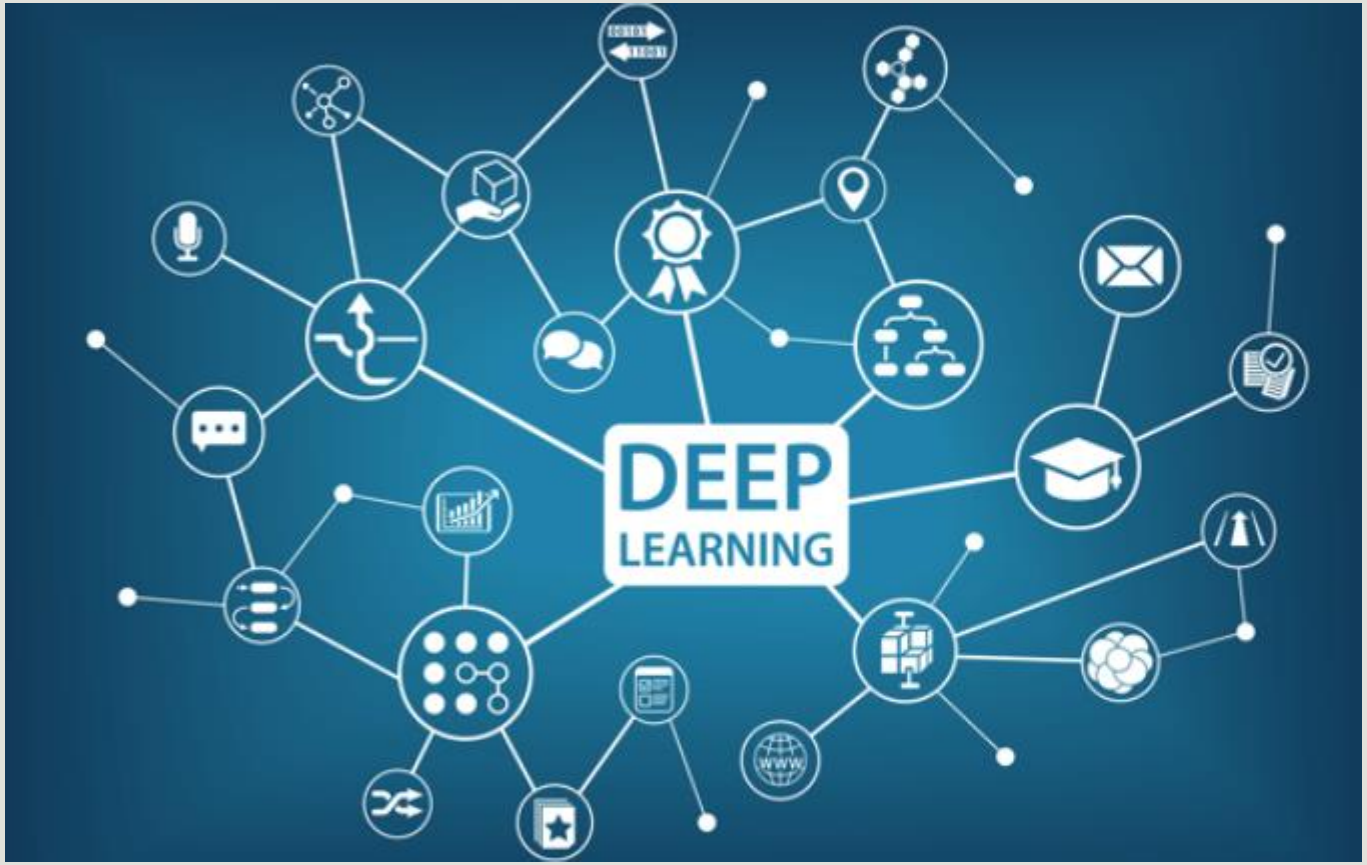
목차

1. CNN

2. Tensorflow 코드 설명

3. Keras 코드 설명

CNN(Convolution Neural Networks)



딥러닝의 중요 분야

- 이미지 검출
- 음성분석
- 자연어 검출

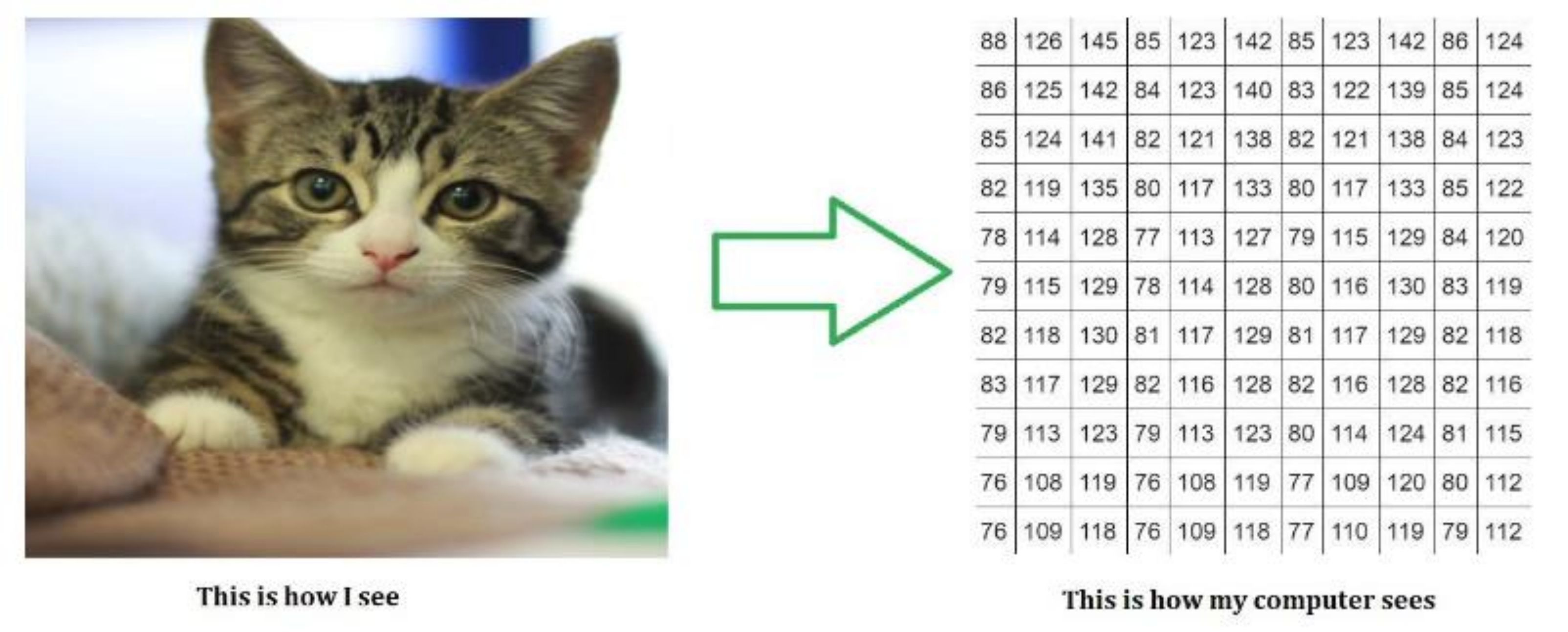
Convolution 의미

1. 번역기 : 대단히 복잡한 것? (나선형의)주름 ? 회선?

2. 의역 : 정보 2개를 특정한 룰에 따라서 섞는 과정.

2개의 정보가 서로 섞이는 순서 있는 절차

NN vs CNN



This is how I see

88	126	145	85	123	142	85	123	142	86	124
86	125	142	84	123	140	83	122	139	85	124
85	124	141	82	121	138	82	121	138	84	123
82	119	135	80	117	133	80	117	133	85	122
78	114	128	77	113	127	79	115	129	84	120
79	115	129	78	114	128	80	116	130	83	119
82	118	130	81	117	129	81	117	129	82	118
83	117	129	82	116	128	82	116	128	82	116
79	113	123	79	113	123	80	114	124	81	115
76	108	119	76	108	119	77	109	120	80	112
76	109	118	76	109	118	77	110	119	79	112

This is how my computer sees

Ex> 고양이 검출

if

고양이 전체가 나온 이미지

색상을 갖고 있는 이미지

NN vs CNN

NN vs CNN

1. Feature의 필요성
- 이미지의 픽셀 값으로 가로*세로*색상 만큼의 feature가 필요하다.

- 다른 feature를 갖는 데이터는 적용이 되지 않는다.

2. X와 Y의 매핑 패턴을 이해한다
- X의 패턴은 이해하지 못한다.

NN vs CNN

1. feature가 필요 없다.
- feature representation learning : 컴퓨터가 스스로 특징 표현을 만들어 내는 것

CNN 동작순서

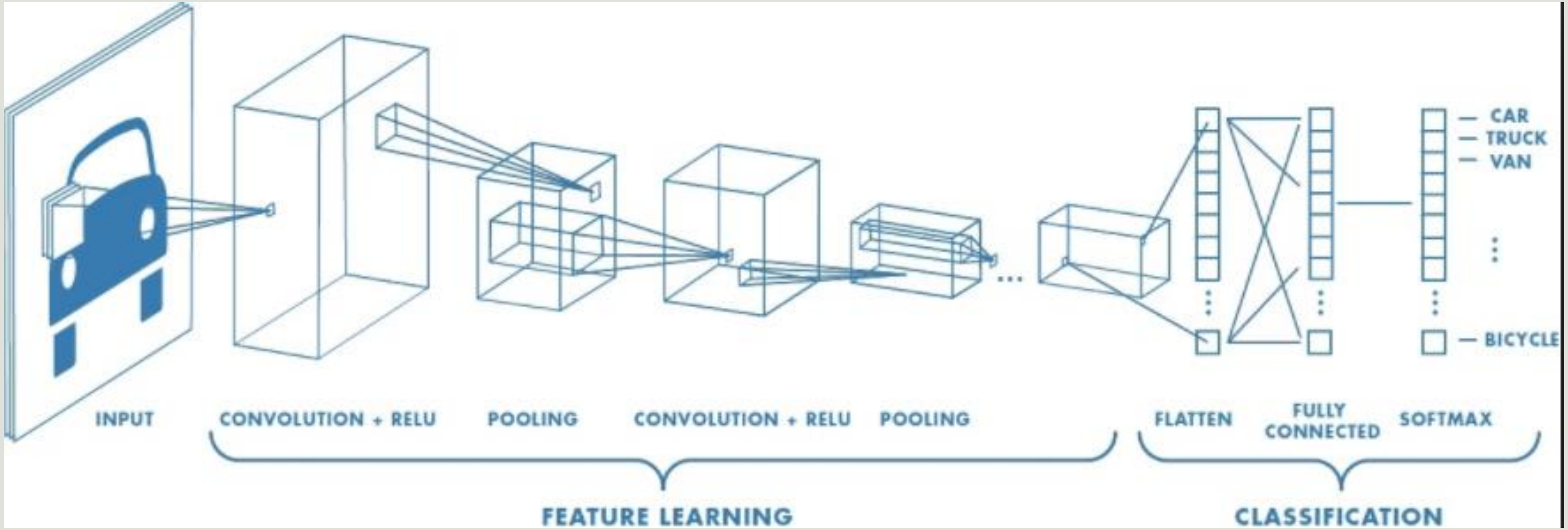
1. 모든 필터를 인풋데이터와 convolution한다.

2. 만들어진 이미지에 대해 pooling한다.

3. 정규화(ReLu).

4. 뉴럴 네트워크의 인풋으로 사용.

CNN 동작순서


INPUT
CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
CAR
TRUCK
VAN
BICYCLE
FEATURE LEARNING
CLASSIFICATION

1. Convolution

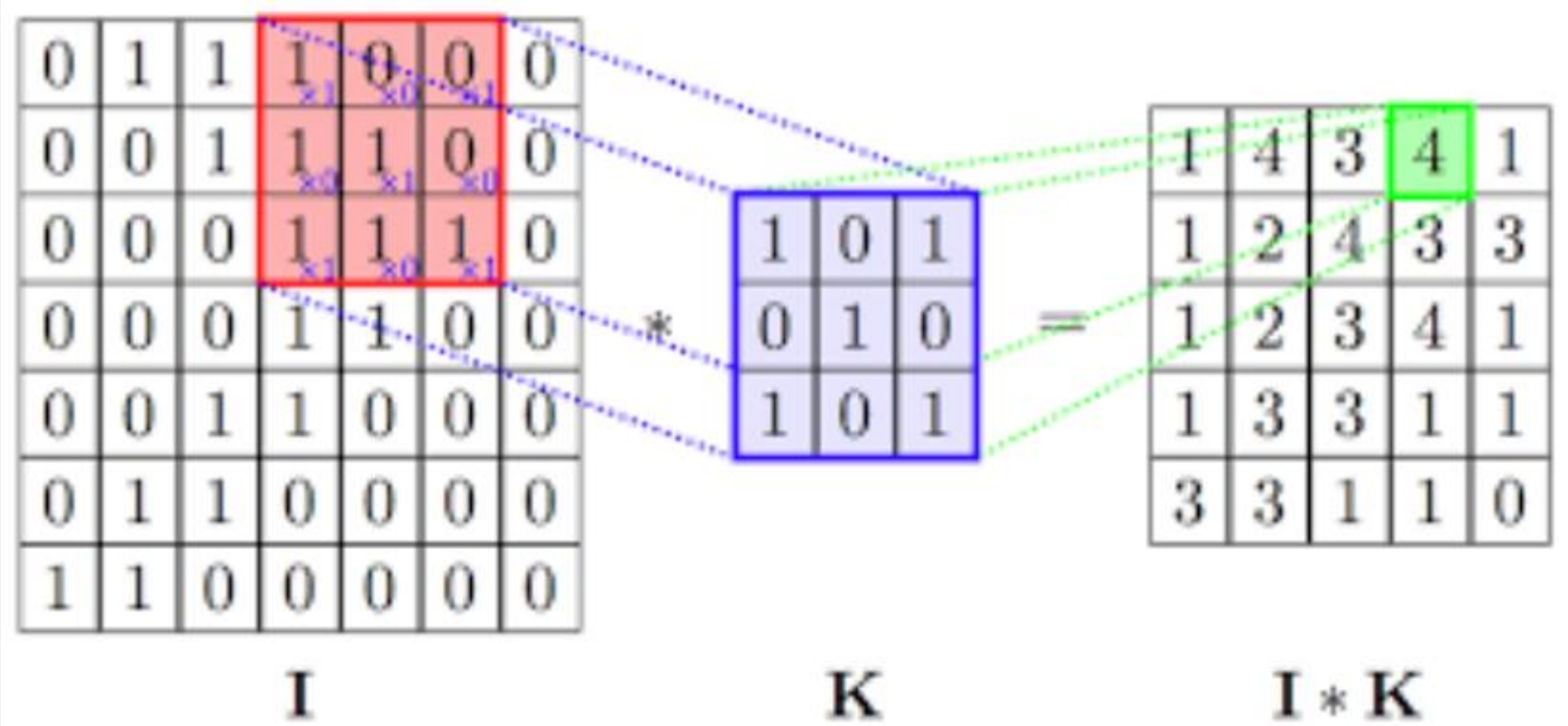
1. (필터가 적용될) 구역을 정하고

2. Dot연산을 한 후

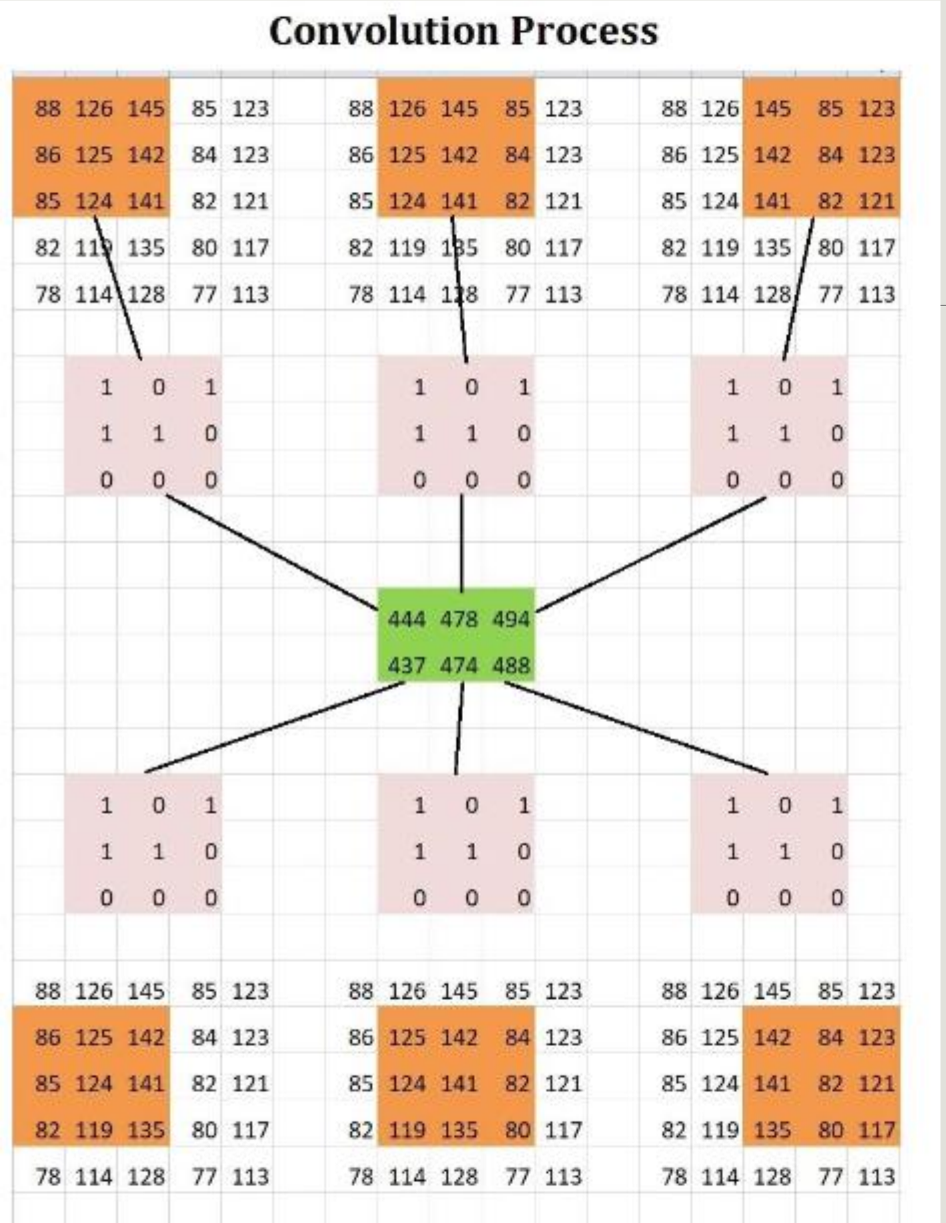
3. Stride만큼 옮긴다

4. 이미지가 끝날때 까지 반복

1. Convolution



$$\underset{\mathbf{I}}{\begin{bmatrix}0&1&1&1&0&0&0\\0&0&1&1&0&0&0\\0&0&0&1&1&1&0\\0&0&0&1&1&0&0\\0&0&1&1&0&0&0\\0&1&1&0&0&0&0\\1&1&0&0&0&0&0\end{bmatrix}} * \underset{\mathbf{K}}{\begin{bmatrix}1&0&1\\0&1&0\\1&0&1\end{bmatrix}} = \underset{\mathbf{I} * \mathbf{K}}{\begin{bmatrix}1&4&3&4&1\\1&2&4&3&3\\1&2&3&4&1\\1&3&3&1&1\\3&3&1&1&0\end{bmatrix}}$$



Convolution Process

88	126	145	85	123	88	126	145	85	123	88	126	145	85	123
86	125	142	84	123	86	125	142	84	123	86	125	142	84	123
85	124	141	82	121	85	124	141	82	121	85	124	141	82	121
82	119	135	80	117	82	119	135	80	117	82	119	135	80	117
78	114	128	77	113	78	114	128	77	113	78	114	128	77	113

1	0	1	1	0	1	1	0	1
1	1	0	1	1	0	1	1	0
0	0	0	0	0	0	0	0	0

444	478	494
437	474	488

1	0	1	1	0	1	1	0	1
1	1	0	1	1	0	1	1	0
0	0	0	0	0	0	0	0	0

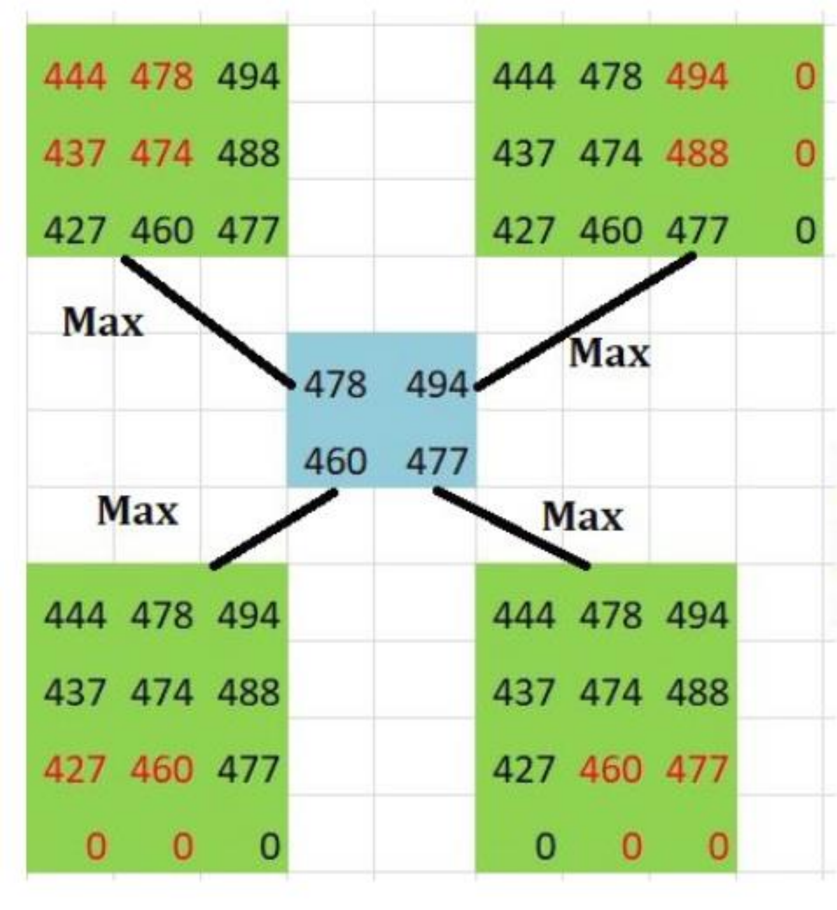
88	126	145	85	123	88	126	145	85	123	88	126	145	85	123
86	125	142	84	123	86	125	142	84	123	86	125	142	84	123
85	124	141	82	121	85	124	141	82	121	85	124	141	82	121
82	119	135	80	117	82	119	135	80	117	82	119	135	80	117
78	114	128	77	113	78	114	128	77	113	78	114	128	77	113

1. Convolution

주위 값들을 반영해 중앙의 값을 변화 시키는 것

어디로?
- 목적하는 작업의 성공률이 높은쪽으로 ! (Classfication이 잘되도록 2차이미지를 생성)

2. Pooling

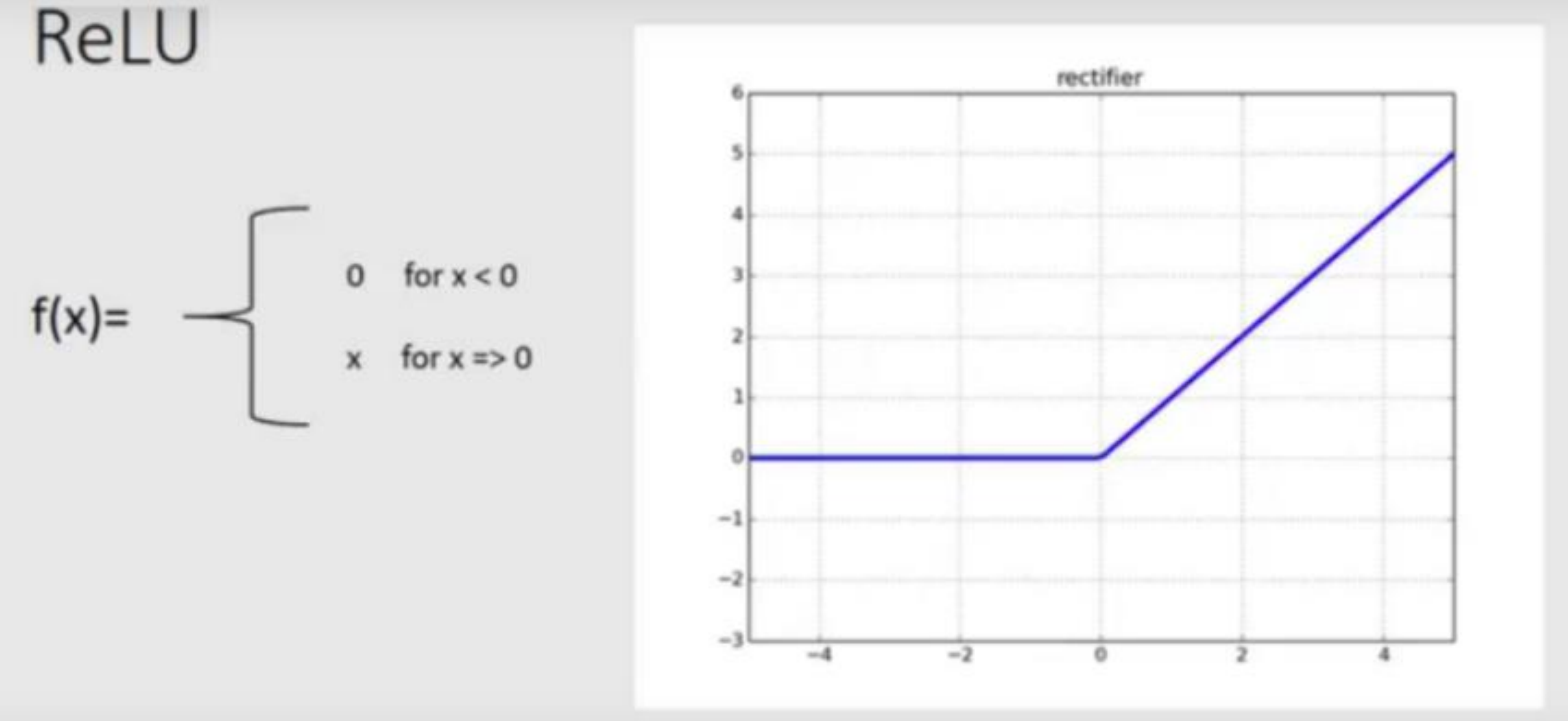


구역의 대표 값을 넣음으로써 이미지 사이즈를 줄이는 것

- Max, Average 등…

매우 큰 성능 효과

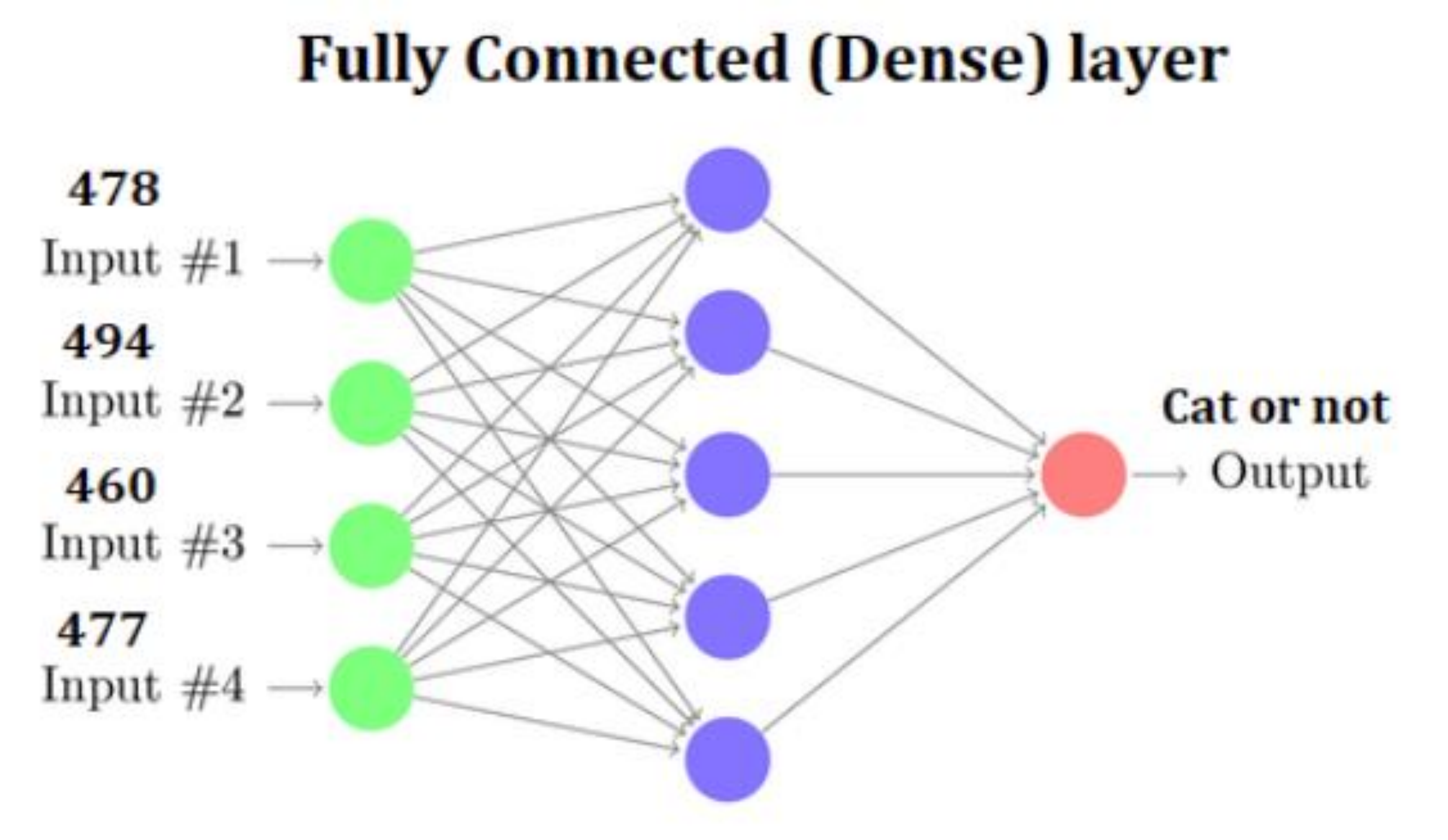
3. Normalize



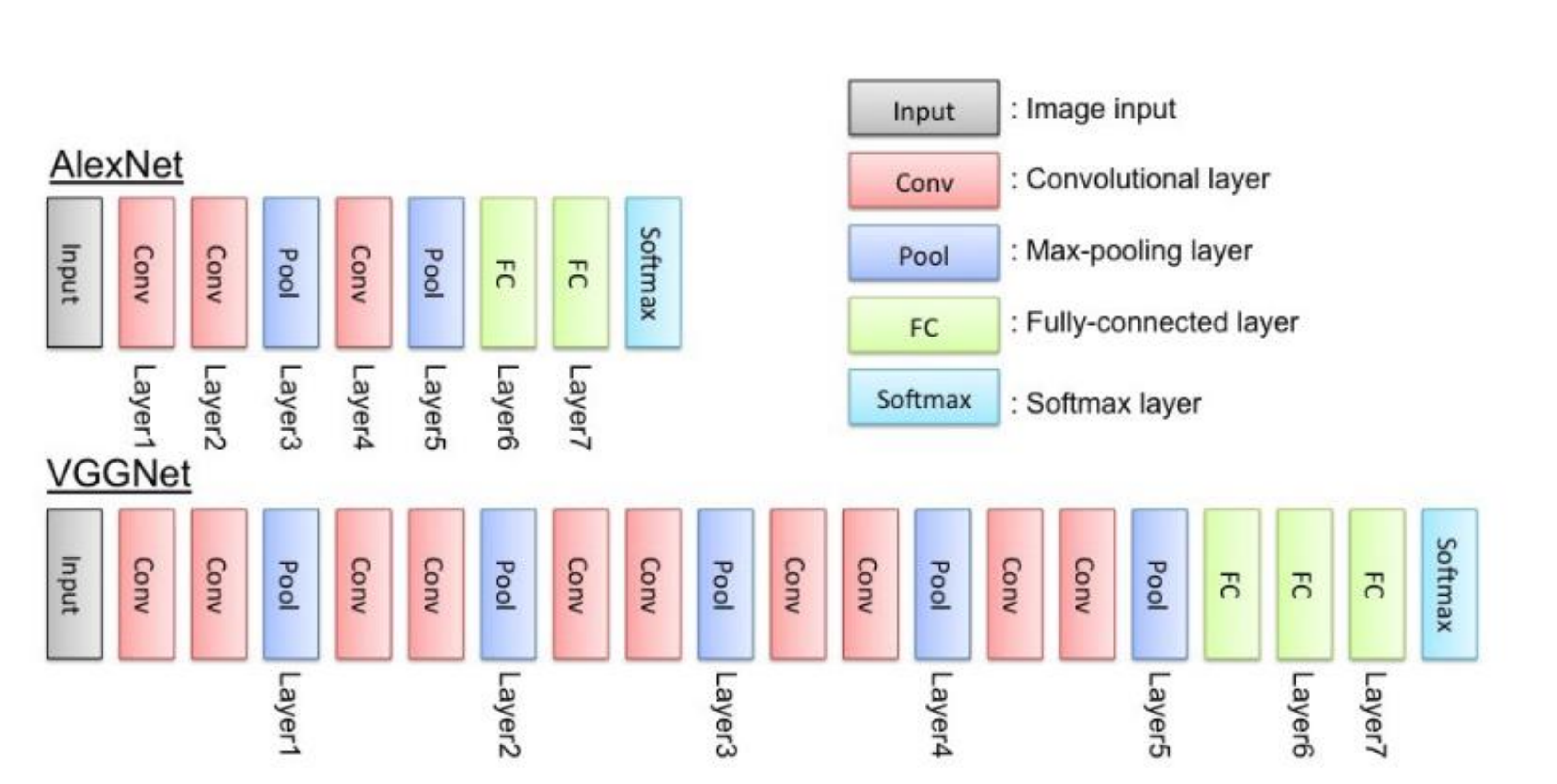
ReLU

$$f(x)= \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x => 0 \end{cases}$$

4. FC

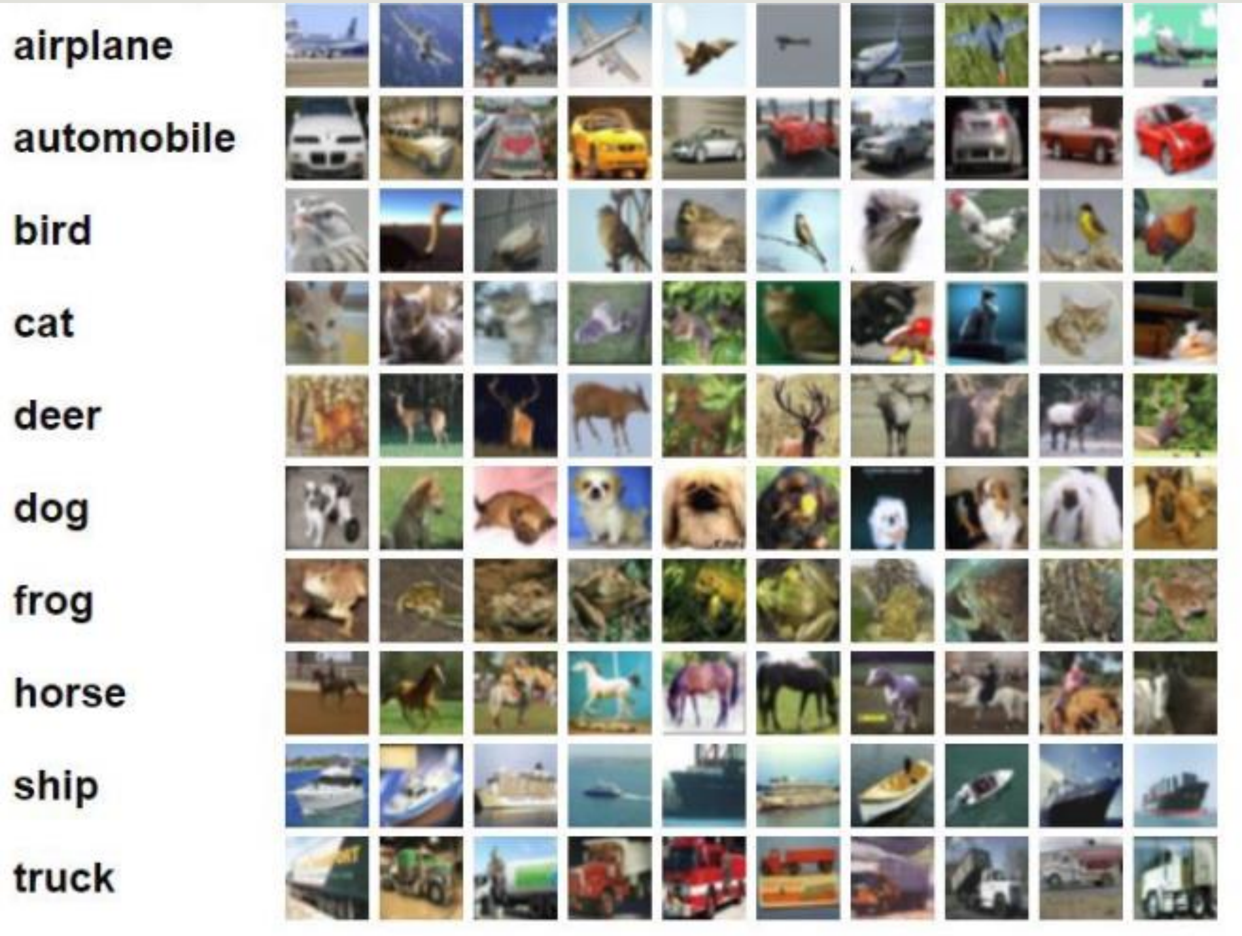

Fully Connected (Dense) layer
478
Input #1
494
Input #2
460
Input #3
477
Input #4
Cat or not
Output

CNN 예시

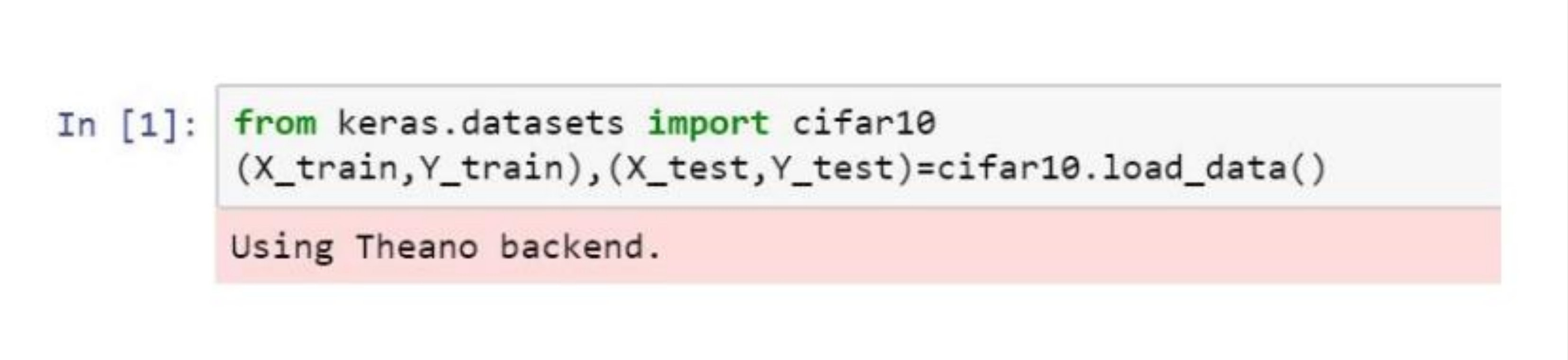


순서는 상관없다

Cifar10 image data set



데이터 가져오기

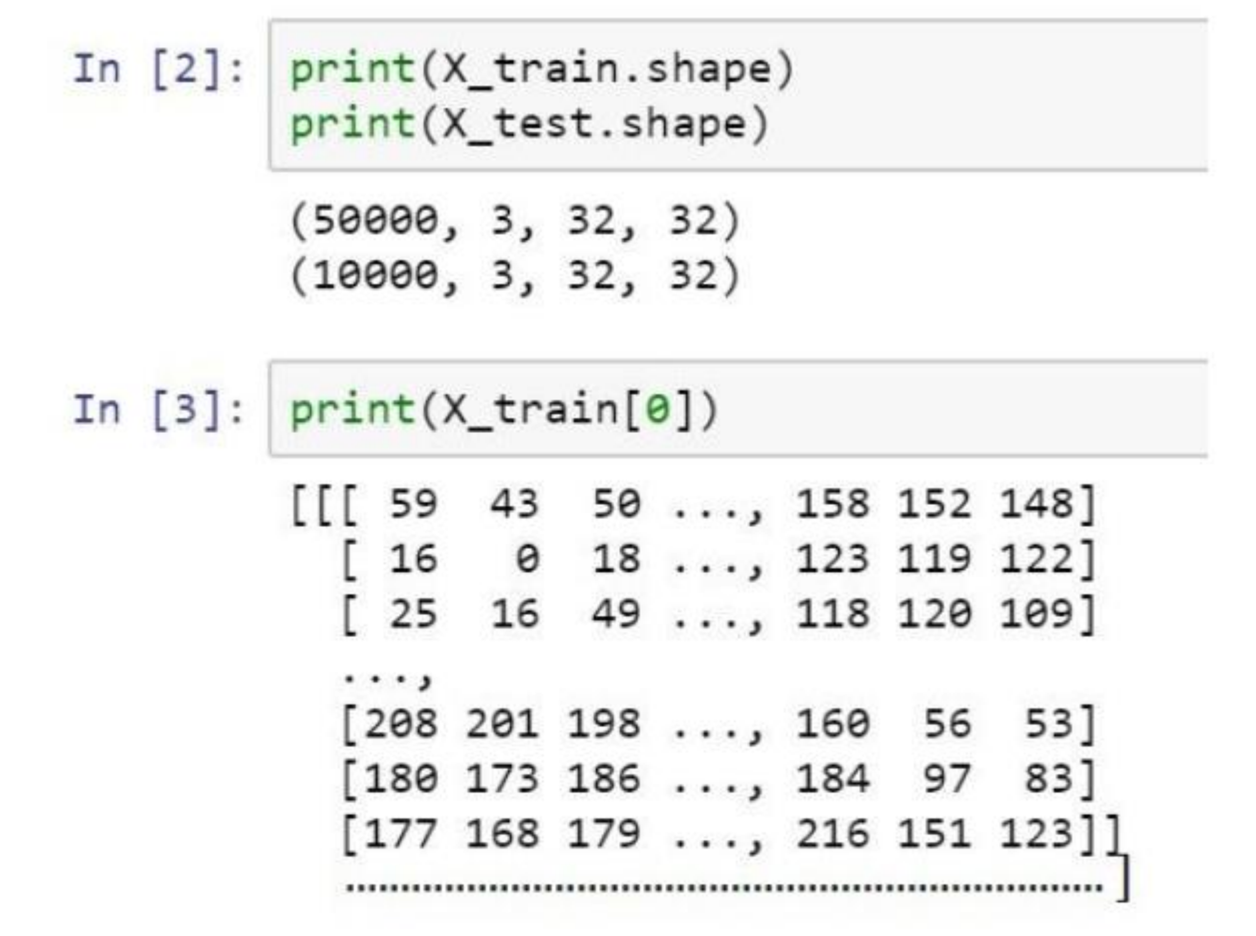


```
from keras.datasets import cifar10
(X_train,Y_train),(X_test,Y_test)=cifar10.load_data()
```

```
Using Theano backend.
```

from tensorflow.python.keras._impl.keras.datasets.cifar10 import load_data

데이터 가져오기



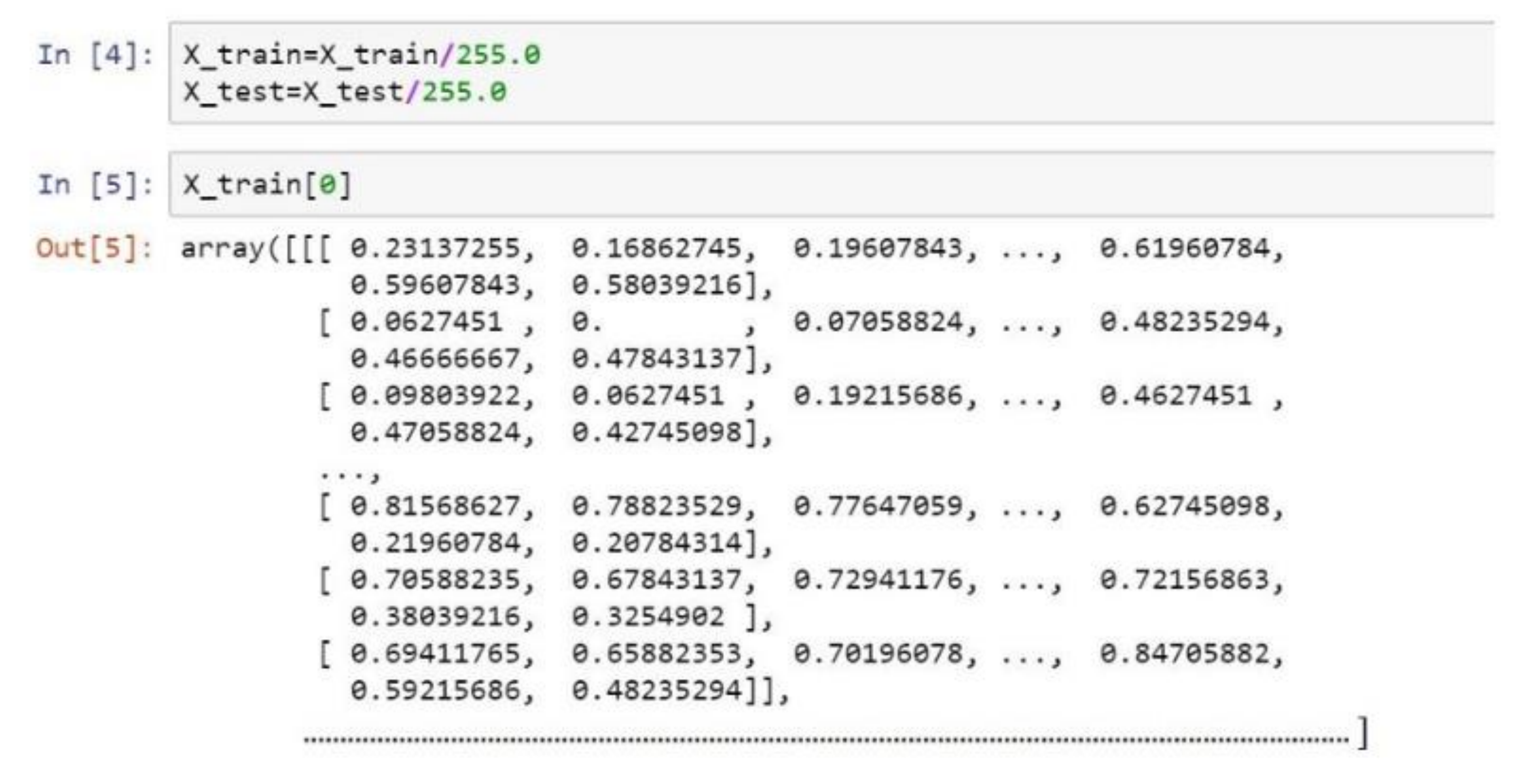
```
In [2]: print(X_train.shape)
        print(X_test.shape)

        (50000, 3, 32, 32)
        (10000, 3, 32, 32)

In [3]: print(X_train[0])

        [[[ 59  43  50 ..., 158 152 148]
          [ 16   0  18 ..., 123 119 122]
          [ 25  16  49 ..., 118 120 109]
          ...,
          [208 201 198 ..., 160  56  53]
          [180 173 186 ..., 184  97  83]
          [177 168 179 ..., 216 151 123]]
          ..............................]
```

- train 50000 / test 10000

- 3*32*32 (color*width*height)

- [0, 255]

데이터 전처리



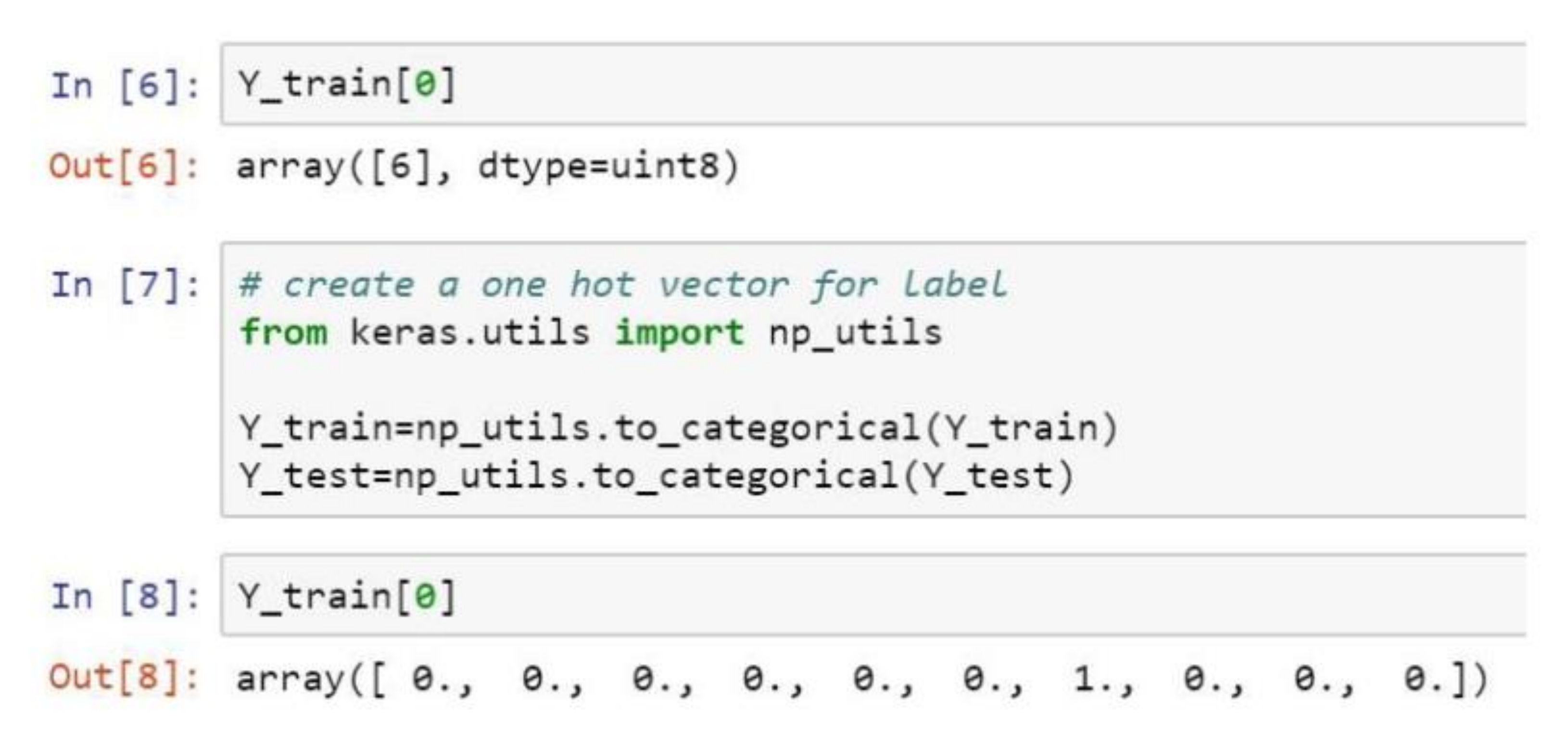
```
In [4]: X_train=X_train/255.0
        X_test=X_test/255.0
```

```
In [5]: X_train[0]
```

```
Out[5]: array([[[ 0.23137255,  0.16862745,  0.19607843, ...,  0.61960784,
                  0.59607843,  0.58039216],
                [ 0.0627451 ,  0.        ,  0.07058824, ...,  0.48235294,
                  0.46666667,  0.47843137],
                [ 0.09803922,  0.0627451 ,  0.19215686, ...,  0.4627451 ,
                  0.47058824,  0.42745098],
                ...,
                [ 0.81568627,  0.78823529,  0.77647059, ...,  0.62745098,
                  0.21960784,  0.20784314],
                [ 0.70588235,  0.67843137,  0.72941176, ...,  0.72156863,
                  0.38039216,  0.3254902 ],
                [ 0.69411765,  0.65882353,  0.70196078, ...,  0.84705882,
                  0.59215686,  0.48235294]],
               ..........................................................]
```

- normalize

데이터 전처리

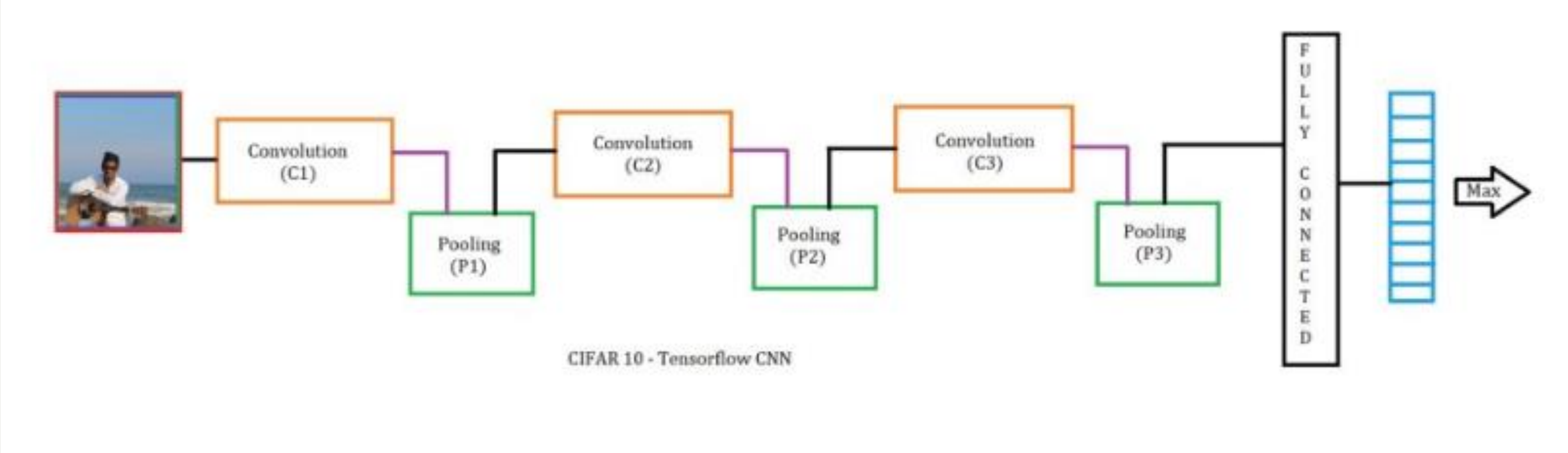


```
In [6]: Y_train[0]

Out[6]: array([6], dtype=uint8)

In [7]: # create a one hot vector for label
        from keras.utils import np_utils

        Y_train=np_utils.to_categorical(Y_train)
        Y_test=np_utils.to_categorical(Y_test)

In [8]: Y_train[0]

Out[8]: array([ 0.,  0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.])
```

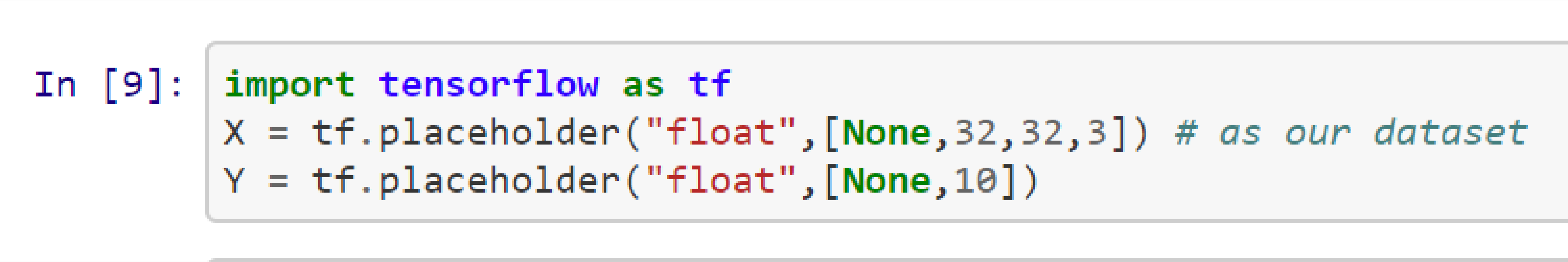
- 10 lable

Airplane, bird ..

TensorFlow : 구조



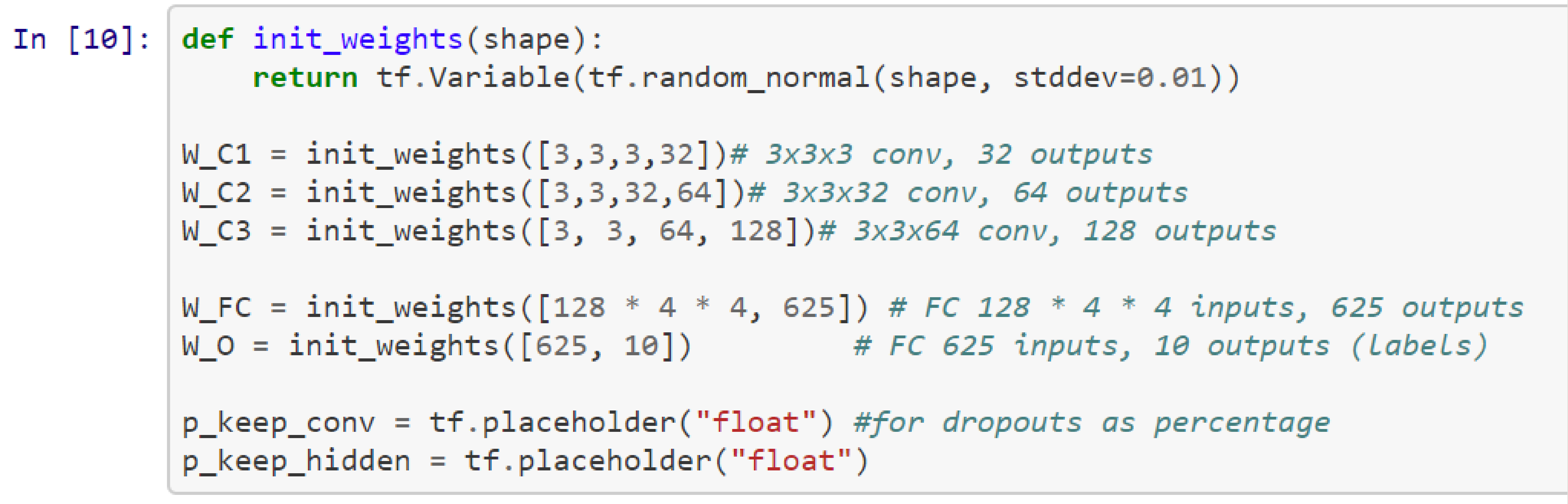
TensorFlow : placeholder



In [9]:

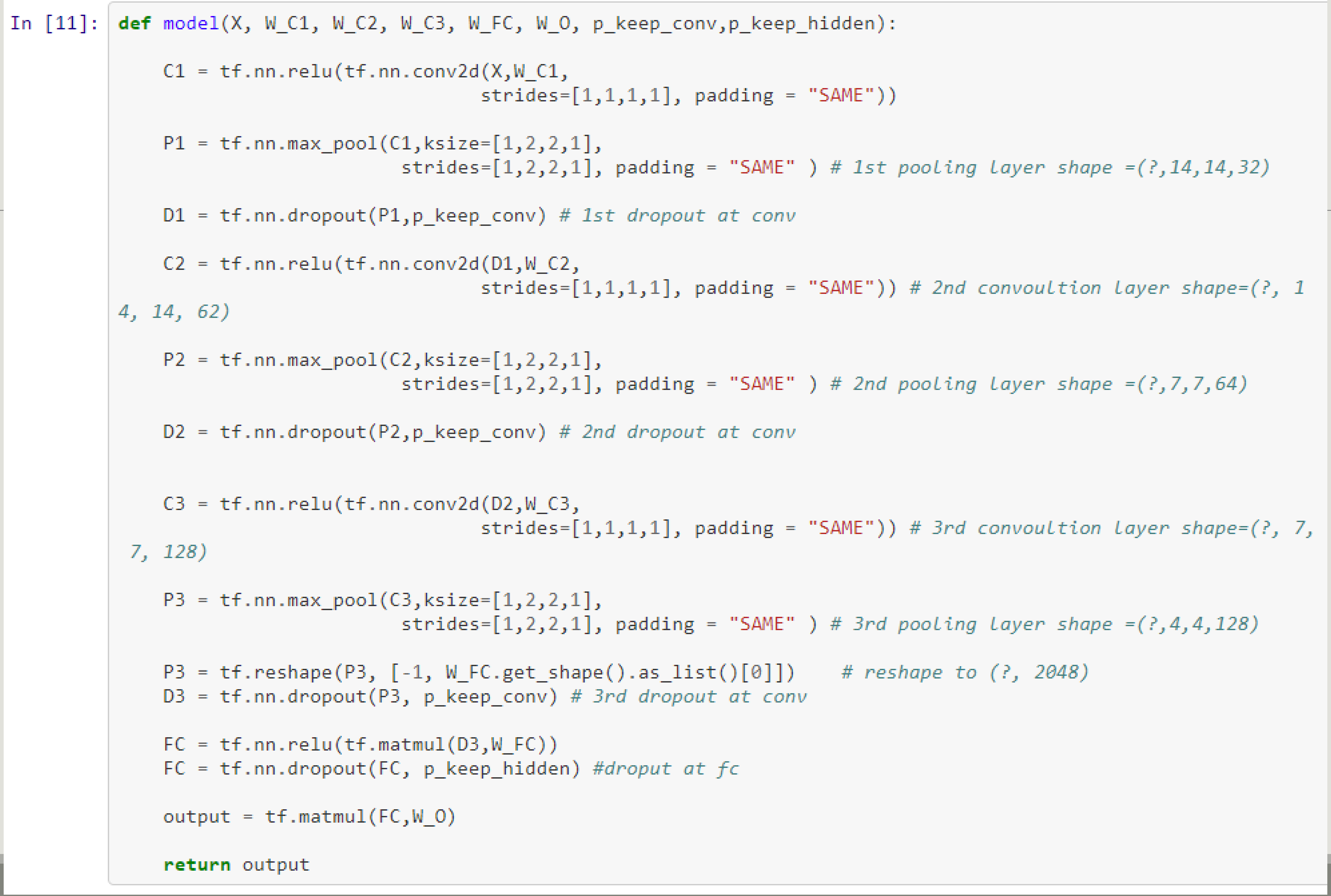
```
import tensorflow as tf
X = tf.placeholder("float",[None,32,32,3]) # as our dataset
Y = tf.placeholder("float",[None,10])
```

TensorFlow : 가중치 초기화



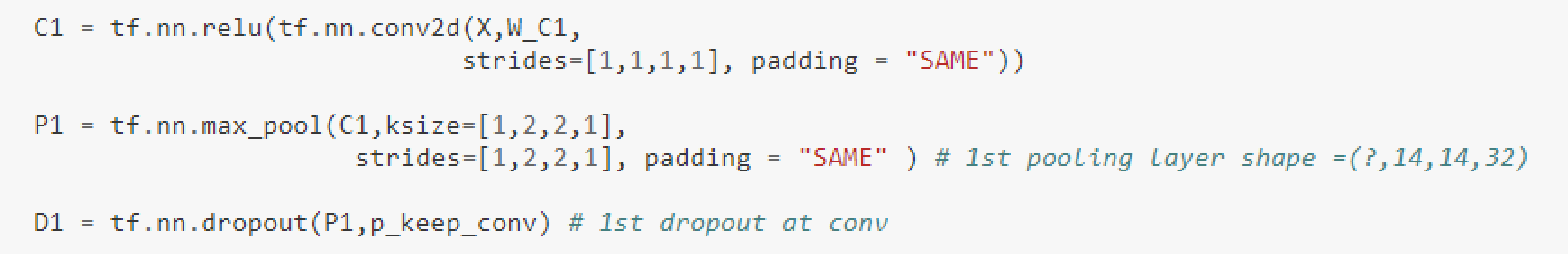
In [10]:

```
def init_weights(shape):
    return tf.Variable(tf.random_normal(shape, stddev=0.01))

W_C1 = init_weights([3,3,3,32])# 3x3x3 conv, 32 outputs
W_C2 = init_weights([3,3,32,64])# 3x3x32 conv, 64 outputs
W_C3 = init_weights([3, 3, 64, 128])# 3x3x64 conv, 128 outputs

W_FC = init_weights([128 * 4 * 4, 625]) # FC 128 * 4 * 4 inputs, 625 outputs
W_O = init_weights([625, 10])         # FC 625 inputs, 10 outputs (labels)

p_keep_conv = tf.placeholder("float") #for dropouts as percentage
p_keep_hidden = tf.placeholder("float")
```



```
In [11]:
def model(X, W_C1, W_C2, W_C3, W_FC, W_O, p_keep_conv,p_keep_hidden):

    C1 = tf.nn.relu(tf.nn.conv2d(X,W_C1,
                                strides=[1,1,1,1], padding = "SAME"))

    P1 = tf.nn.max_pool(C1,ksize=[1,2,2,1],
                         strides=[1,2,2,1], padding = "SAME" ) # 1st pooling layer shape =(?,14,14,32)

    D1 = tf.nn.dropout(P1,p_keep_conv) # 1st dropout at conv

    C2 = tf.nn.relu(tf.nn.conv2d(D1,W_C2,
                                strides=[1,1,1,1], padding = "SAME")) # 2nd convoultion layer shape=(?, 1
4, 14, 62)

    P2 = tf.nn.max_pool(C2,ksize=[1,2,2,1],
                         strides=[1,2,2,1], padding = "SAME" ) # 2nd pooling layer shape =(?,7,7,64)

    D2 = tf.nn.dropout(P2,p_keep_conv) # 2nd dropout at conv


    C3 = tf.nn.relu(tf.nn.conv2d(D2,W_C3,
                                strides=[1,1,1,1], padding = "SAME")) # 3rd convoultion layer shape=(?, 7,
 7, 128)

    P3 = tf.nn.max_pool(C3,ksize=[1,2,2,1],
                         strides=[1,2,2,1], padding = "SAME" ) # 3rd pooling layer shape =(?,4,4,128)

    P3 = tf.reshape(P3, [-1, W_FC.get_shape().as_list()[0]])    # reshape to (?, 2048)
    D3 = tf.nn.dropout(P3, p_keep_conv) # 3rd dropout at conv

    FC = tf.nn.relu(tf.matmul(D3,W_FC))
    FC = tf.nn.dropout(FC, p_keep_hidden) #droput at fc

    output = tf.matmul(FC,W_O)

    return output
```

모델 만들기 (일부)



```
C1 = tf.nn.relu(tf.nn.conv2d(X,W_C1,
                            strides=[1,1,1,1], padding = "SAME"))

P1 = tf.nn.max_pool(C1,ksize=[1,2,2,1],
                     strides=[1,2,2,1], padding = "SAME" ) # 1st pooling layer shape =(?,14,14,32)

D1 = tf.nn.dropout(P1,p_keep_conv) # 1st dropout at conv
```

C1(convolution layer 1), P1(pooling layer 1), D1(dropout)

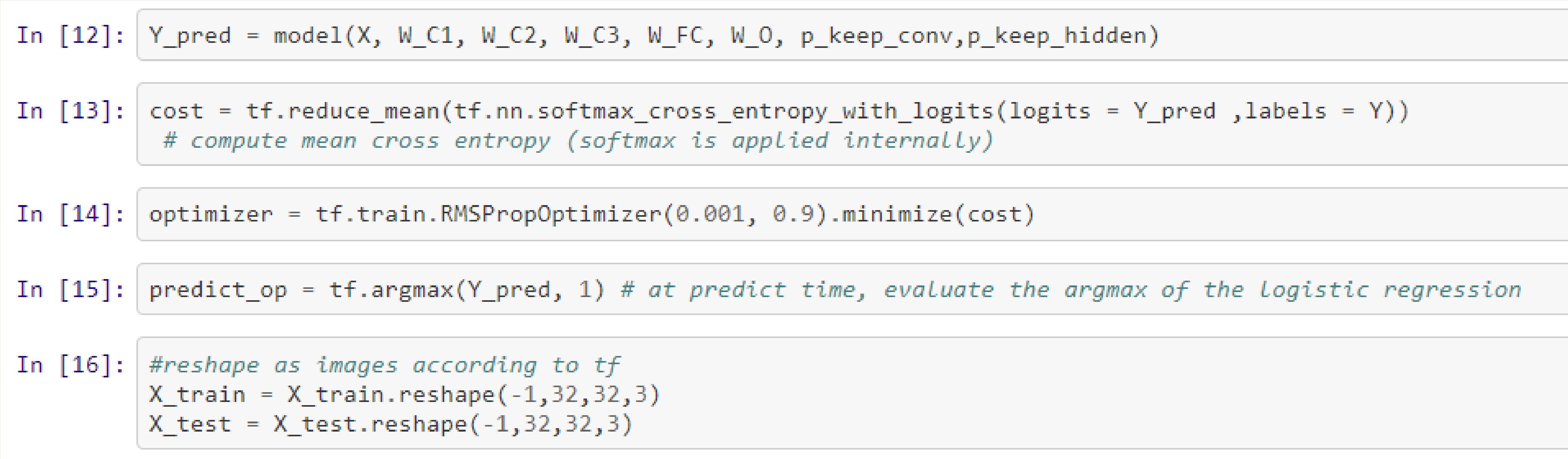
Tf.nn.relu(input)

Tf.nn.conv2d(input, weight, strides, padding)

Tf.nn.pool(input, size, strides, padding)

Tf.nn.dropout(input,keep_percentage)

TensorFlow : loss, optimizer



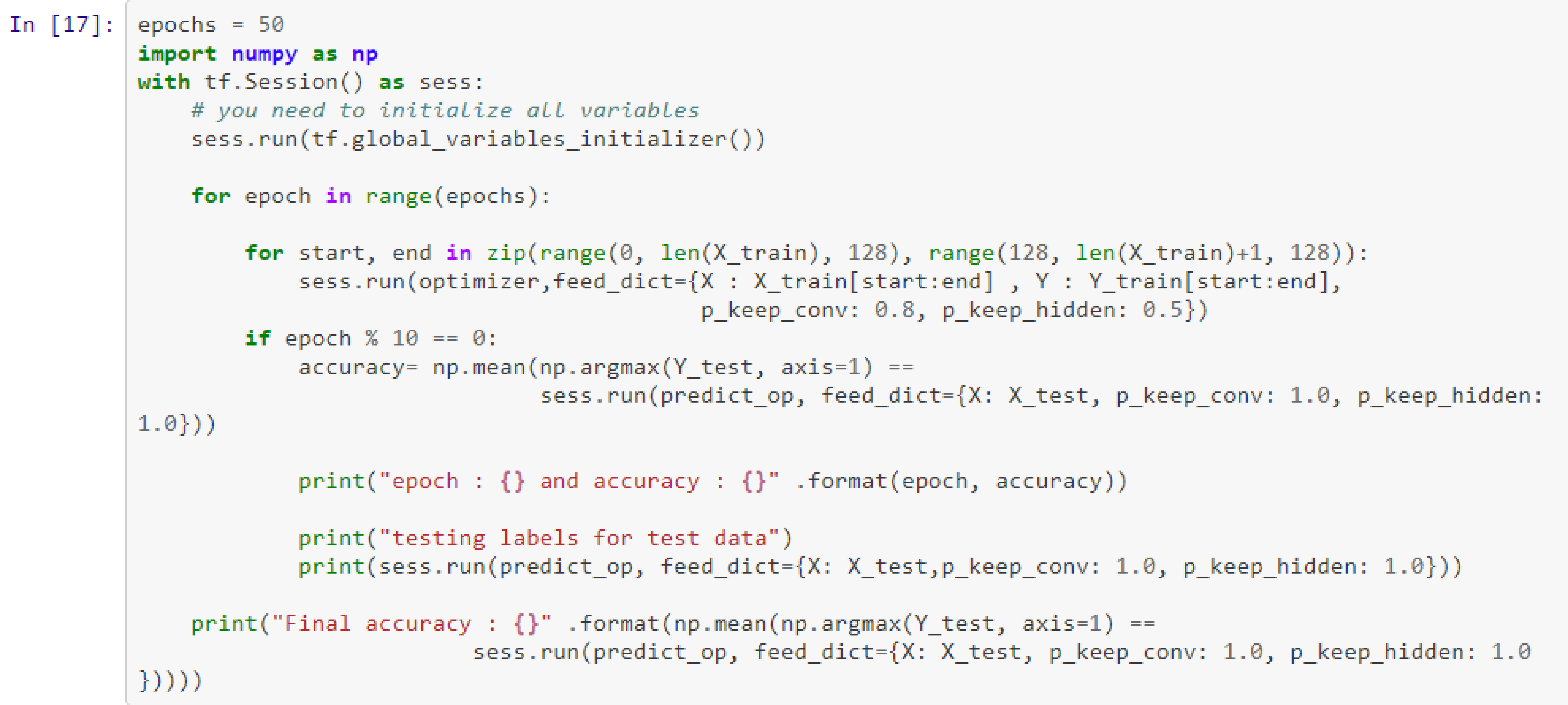
```
In [12]: Y_pred = model(X, W_C1, W_C2, W_C3, W_FC, W_O, p_keep_conv,p_keep_hidden)
```

```
In [13]: cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits = Y_pred ,labels = Y))
          # compute mean cross entropy (softmax is applied internally)
```

```
In [14]: optimizer = tf.train.RMSPropOptimizer(0.001, 0.9).minimize(cost)
```

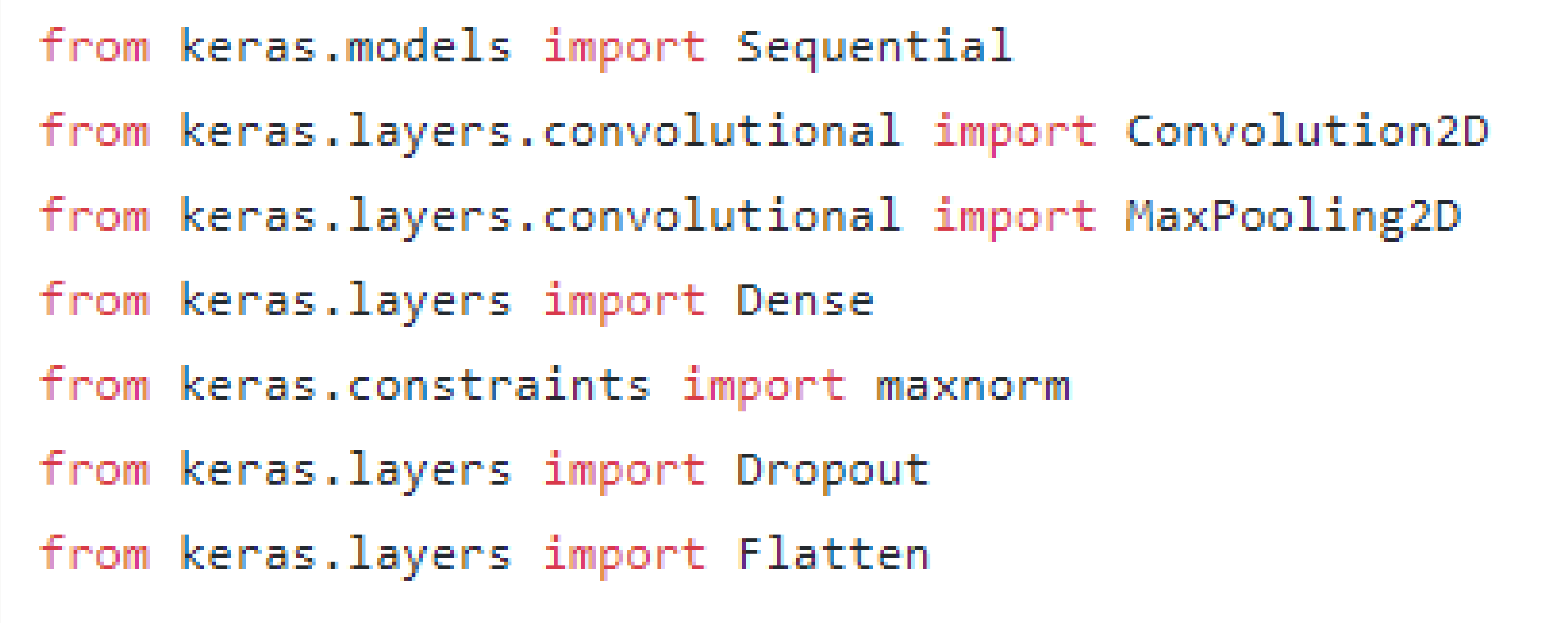
```
In [15]: predict_op = tf.argmax(Y_pred, 1) # at predict time, evaluate the argmax of the logistic regression
```

```
In [16]: #reshape as images according to tf
         X_train = X_train.reshape(-1,32,32,3)
         X_test = X_test.reshape(-1,32,32,3)
```



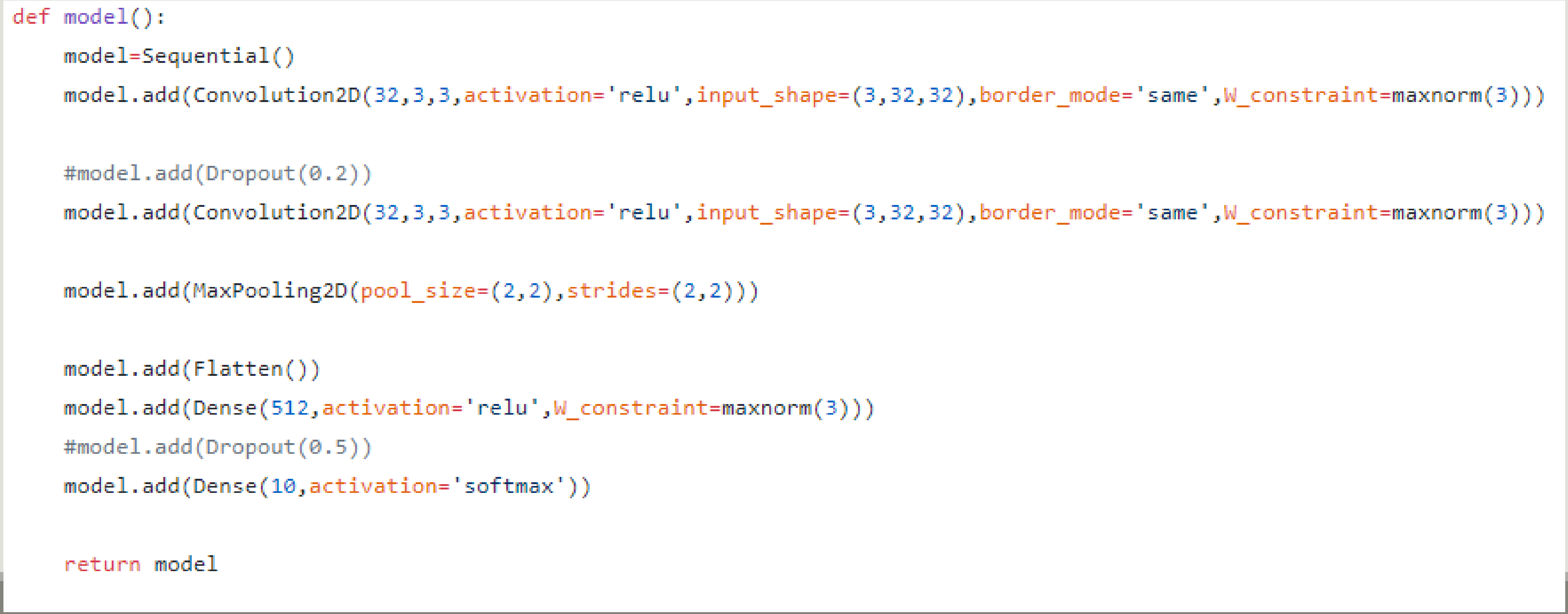
```
In [17]: epochs = 50
import numpy as np
with tf.Session() as sess:
    # you need to initialize all variables
    sess.run(tf.global_variables_initializer())

    for epoch in range(epochs):

        for start, end in zip(range(0, len(X_train), 128), range(128, len(X_train)+1, 128)):
            sess.run(optimizer,feed_dict={X : X_train[start:end] , Y : Y_train[start:end],
                                          p_keep_conv: 0.8, p_keep_hidden: 0.5})
        if epoch % 10 == 0:
            accuracy= np.mean(np.argmax(Y_test, axis=1) ==
                              sess.run(predict_op, feed_dict={X: X_test, p_keep_conv: 1.0, p_keep_hidden:
1.0}))

            print("epoch : {} and accuracy : {}" .format(epoch, accuracy))

            print("testing labels for test data")
            print(sess.run(predict_op, feed_dict={X: X_test,p_keep_conv: 1.0, p_keep_hidden: 1.0}))

    print("Final accuracy : {}" .format(np.mean(np.argmax(Y_test, axis=1) ==
                         sess.run(predict_op, feed_dict={X: X_test, p_keep_conv: 1.0, p_keep_hidden: 1.0
}))))
```

Keras : 임포트



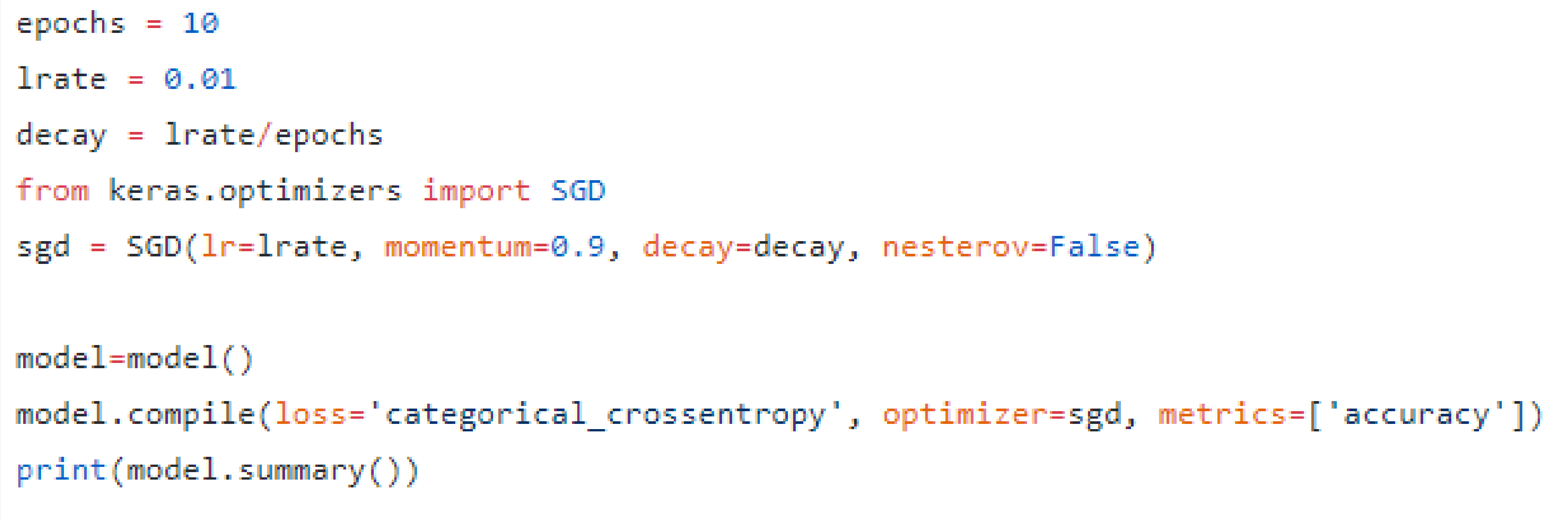
```
from keras.models import Sequential
from keras.layers.convolutional import Convolution2D
from keras.layers.convolutional import MaxPooling2D
from keras.layers import Dense
from keras.constraints import maxnorm
from keras.layers import Dropout
from keras.layers import Flatten
```

Keras : 모델 만들기



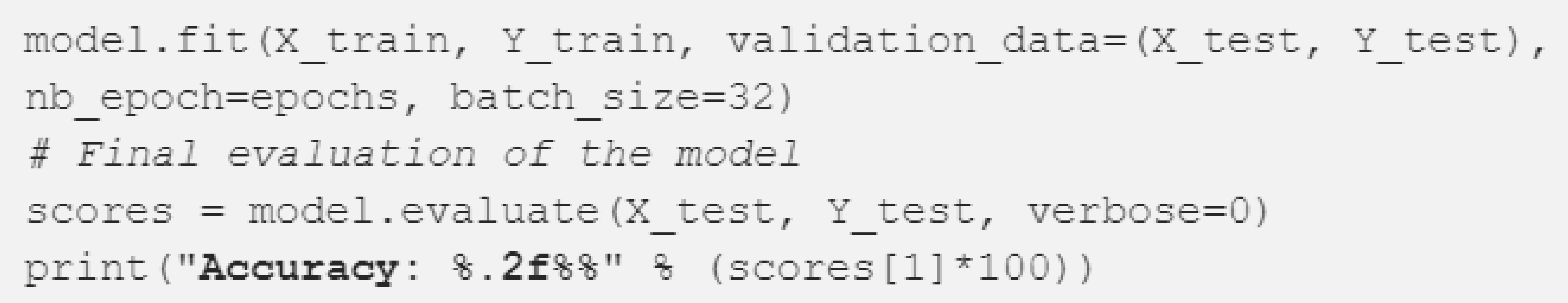
```
def model():
    model=Sequential()
    model.add(Convolution2D(32,3,3,activation='relu',input_shape=(3,32,32),border_mode='same',W_constraint=maxnorm(3)))

    #model.add(Dropout(0.2))
    model.add(Convolution2D(32,3,3,activation='relu',input_shape=(3,32,32),border_mode='same',W_constraint=maxnorm(3)))

    model.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))

    model.add(Flatten())
    model.add(Dense(512,activation='relu',W_constraint=maxnorm(3)))
    #model.add(Dropout(0.5))
    model.add(Dense(10,activation='softmax'))

    return model
```

Keras : optimizer, loss



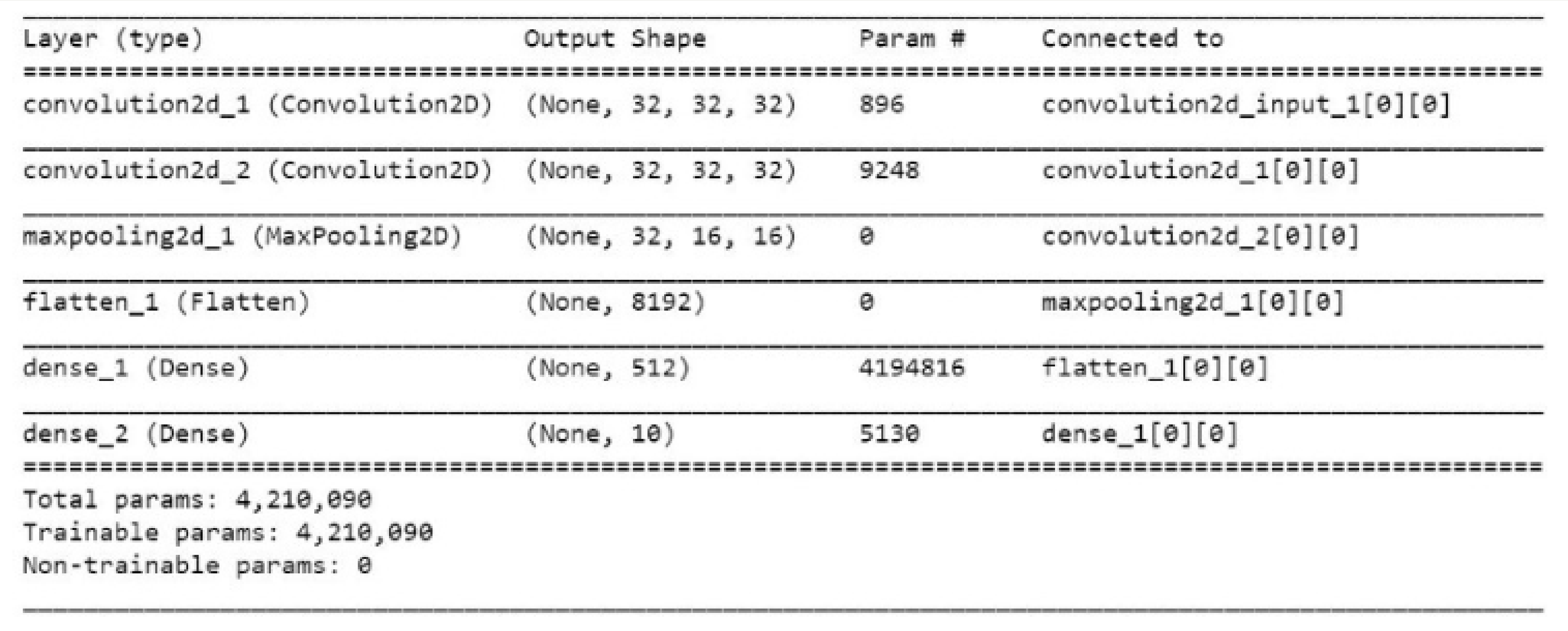
```
epochs = 10
lrate = 0.01
decay = lrate/epochs
from keras.optimizers import SGD
sgd = SGD(lr=lrate, momentum=0.9, decay=decay, nesterov=False)

model=model()
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
print(model.summary())
```

Keras :



```
model.fit(X_train, Y_train, validation_data=(X_test, Y_test),
nb_epoch=epochs, batch_size=32)
# Final evaluation of the model
scores = model.evaluate(X_test, Y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

Keras : Model.summary()



```
____________________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
convolution2d_1 (Convolution2D)  (None, 32, 32, 32)    896         convolution2d_input_1[0][0]

____________________________________________________________________________________________________
convolution2d_2 (Convolution2D)  (None, 32, 32, 32)    9248        convolution2d_1[0][0]

____________________________________________________________________________________________________
maxpooling2d_1 (MaxPooling2D)    (None, 32, 16, 16)    0           convolution2d_2[0][0]

____________________________________________________________________________________________________
flatten_1 (Flatten)              (None, 8192)          0           maxpooling2d_1[0][0]

____________________________________________________________________________________________________
dense_1 (Dense)                  (None, 512)           4194816     flatten_1[0][0]

____________________________________________________________________________________________________
dense_2 (Dense)                  (None, 10)            5130        dense_1[0][0]
====================================================================================================
Total params: 4,210,090
Trainable params: 4,210,090
Non-trainable params: 0

____________________________________________________________________________________________________
```

Keras vs Tensorflow

- Initialize

- Evaluate, summary

- Model.add()

- reshape