Convolutional neural networks

Practice Session 7

Changho Suh

January 24, 2024

Recap: Two building blocks of CNNs

1. Convolutional layer (Conv layer)

Role: Mimick neurons' behaviors:
Reacting only to receptive fields.

2. Pooling layer

Role: Downsample to reduce complexity
(# parameters & memory size).

Recap: Conv layer

1. A neuron reacts only to a **receptive field**.
2. A neighboring neuron concerns a receptive field shifted by **stride**.
3. Two types of **zero padding**: same, valid
4. Consists of a 2D **feature map**
5. Use of **multiple filters**
6. Has three **RGB** channels for colored images

Recap: Pooling layer

1. Two types of pooling:

   Max pooling, average pooling

2. A default choice: 2*2 pooling filter, stride=2

3. Works on each channel independently

Recap: Two popular CNNs

1. **AlexNet** (2012)

   Won the ImageNet competition.

Alex Krizhevsky

Ilya Sutskever

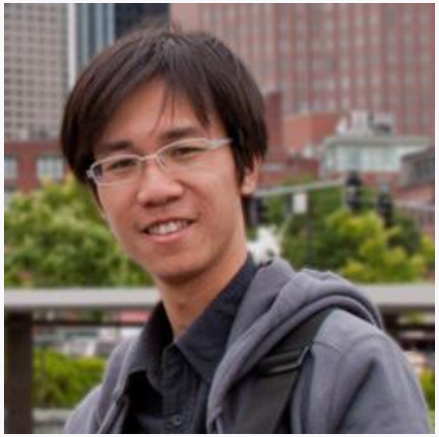
Geoffrey Hinton

   Anchored the deep learning revolution.

2. **ResNet** (2015)

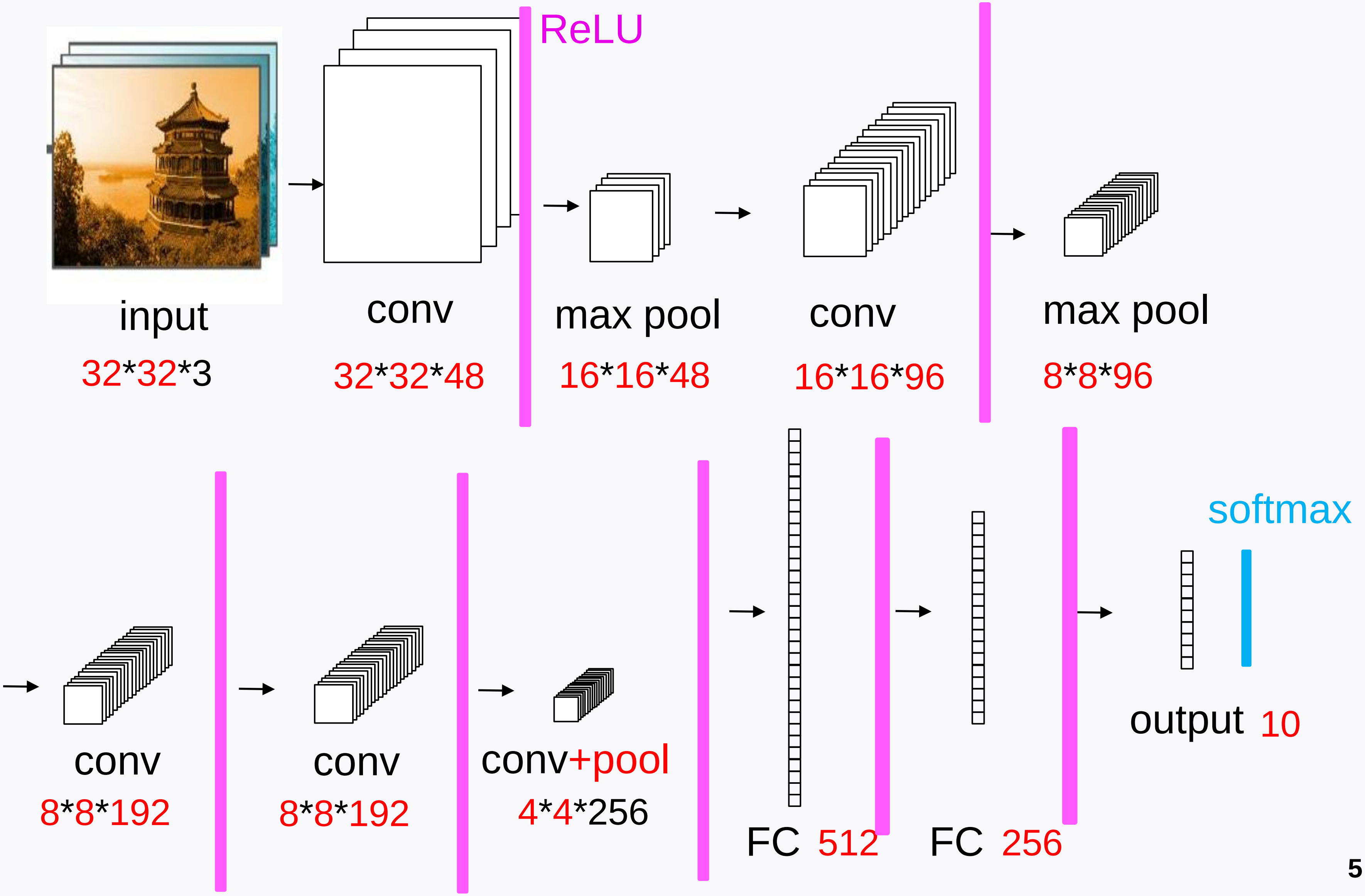
   Won the 2015 ImageNet competition.

   Currently the most powerful & arguably the simplest!

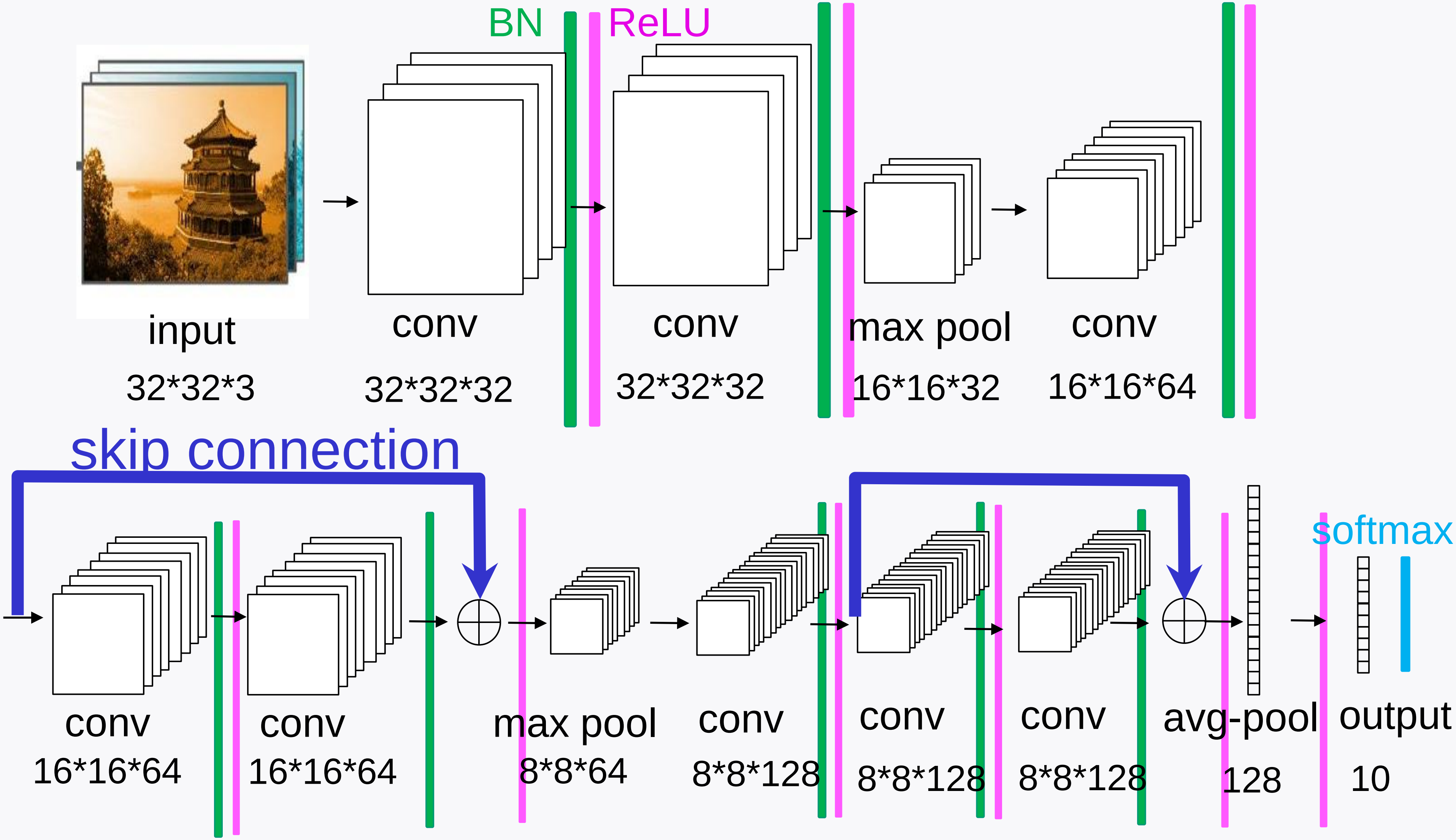


Kaiming He

Recap: Simplified AlexNet



Recap: Simplified ResNet



Outline

Will implement three prominent models:

1. LeNet5

   Task: Handwritten digit classification (MNIST)

2. Simplified AlexNet

   Task: Image recognition (CIFAR10)

3. Simplified ResNet

   Task: Image recognition (CIFAR10)

Handwritten digit classification

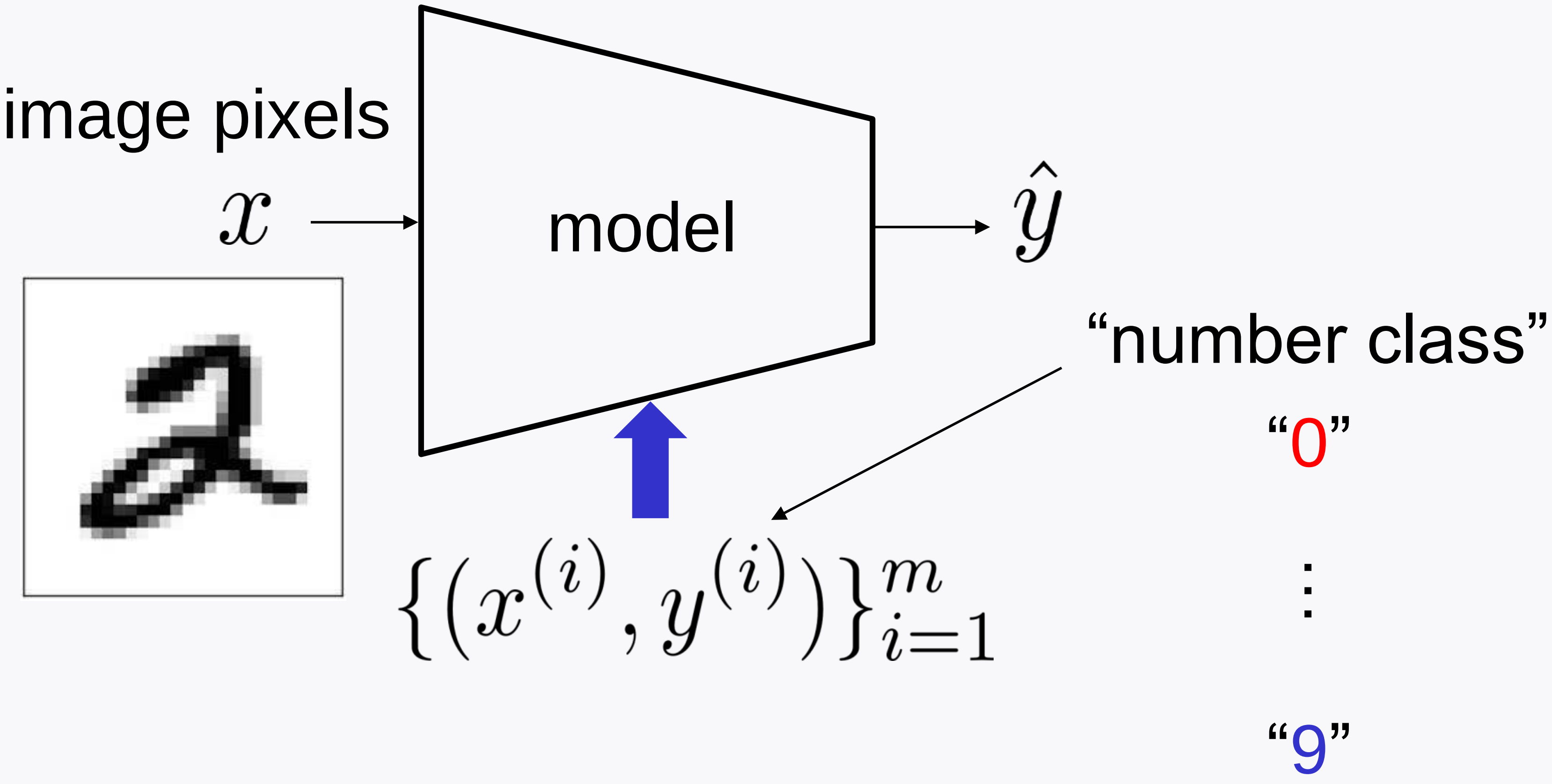


image pixels

x

model

$\hat{y}$

“number class”

“0”

⋮

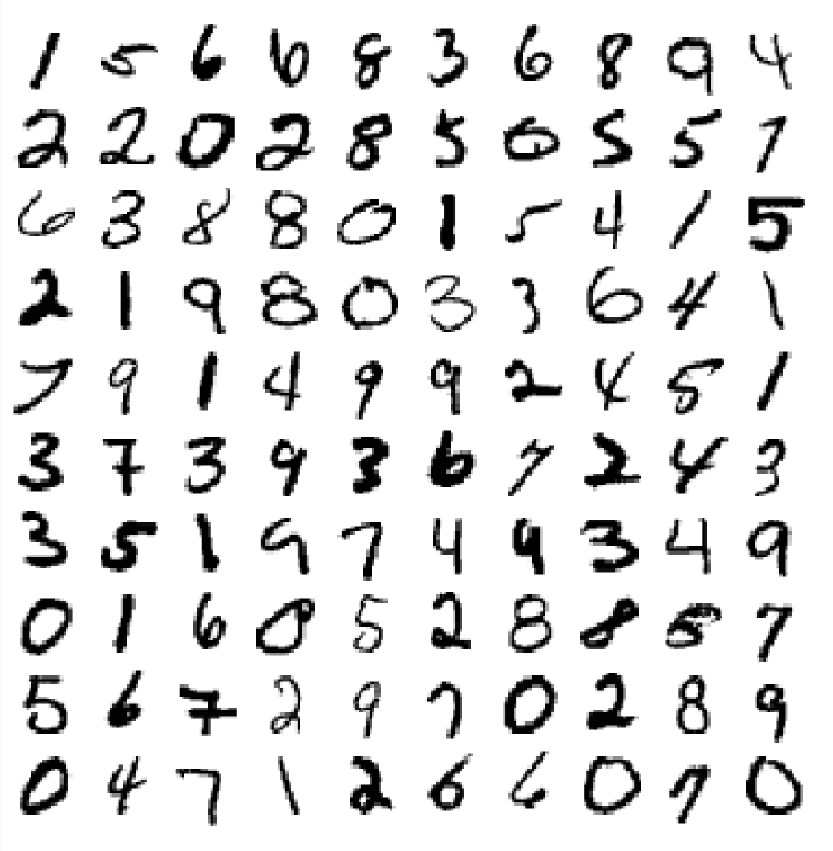
“9”

$$\{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$$

MNIST dataset

$$\{(x^{(i)}, y^{(i)})\}_{i=1}^{m} \qquad m = 60,000$$

Examples:

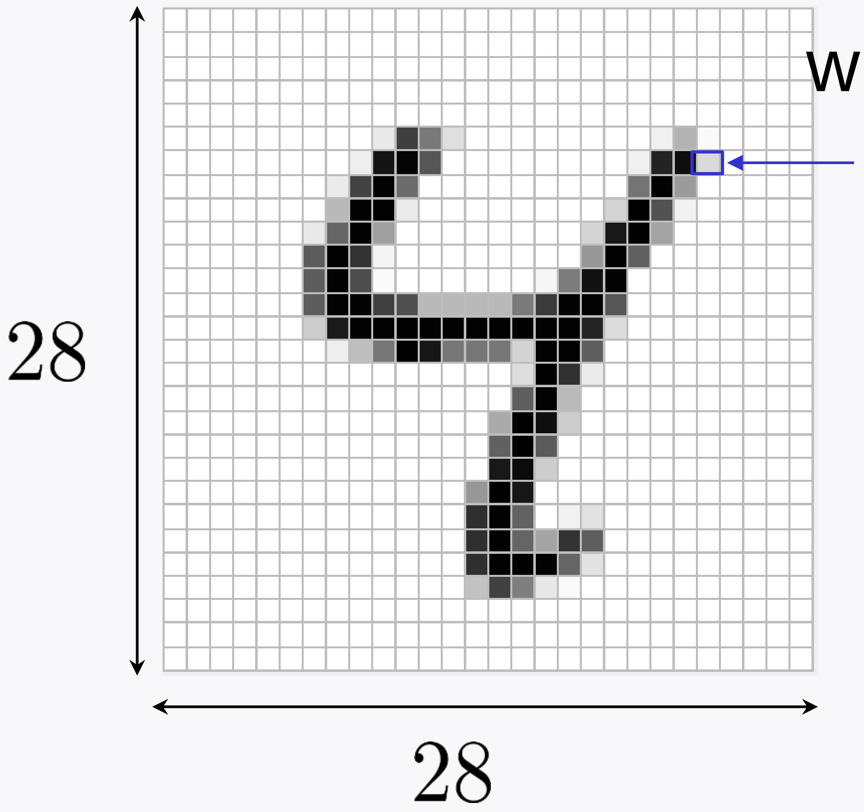


Yann LeCun 1998

Input image & label

$$\{(x^{(i)}, y^{(i)})\}_{i=1}^{m} \qquad m = 60,000$$

$$\in \{0, 1, \ldots, 9\}$$

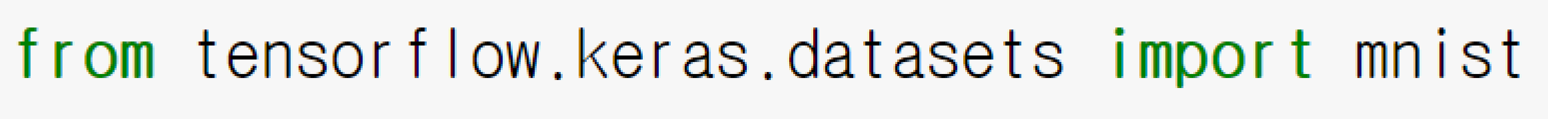


white

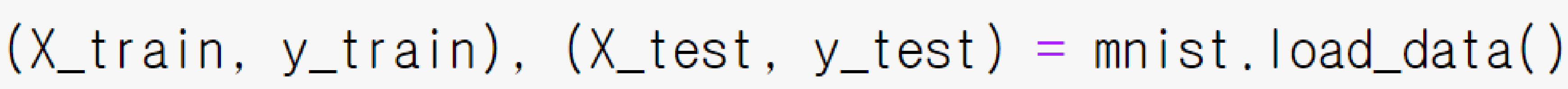
$[0, 1]$

black

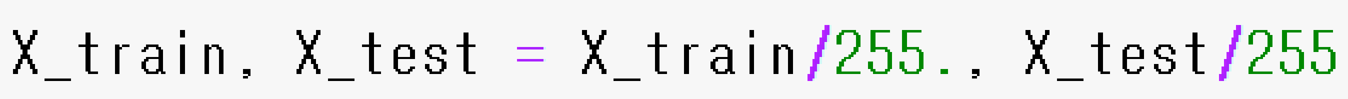
How to load MNIST dataset



```
from tensorflow.keras.datasets import mnist
```

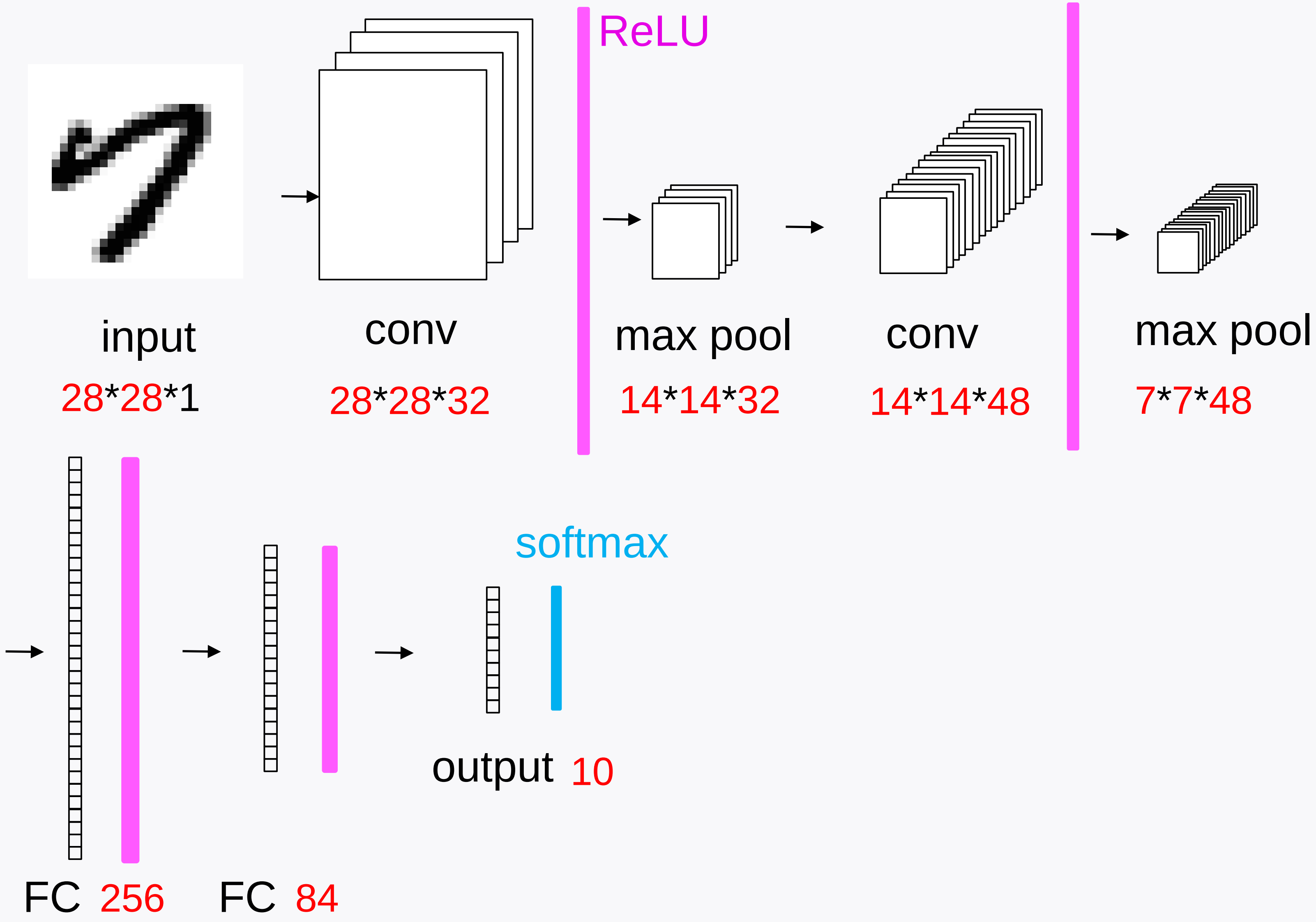


```
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```



```
X_train, X_test = X_train/255., X_test/255
```

LeNet-5: CNN model



ReLU

input 28*28*1

conv 28*28*32

max pool 14*14*32

conv 14*14*48

max pool 7*7*48

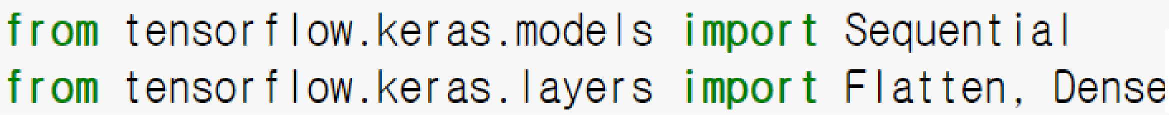
softmax

FC 256

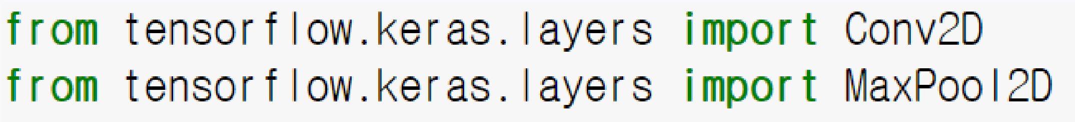
FC 84

output 10

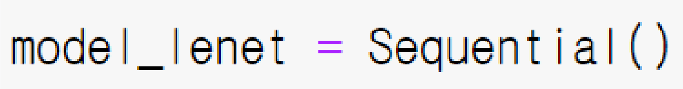
LeNet-5: Tensorflow coding



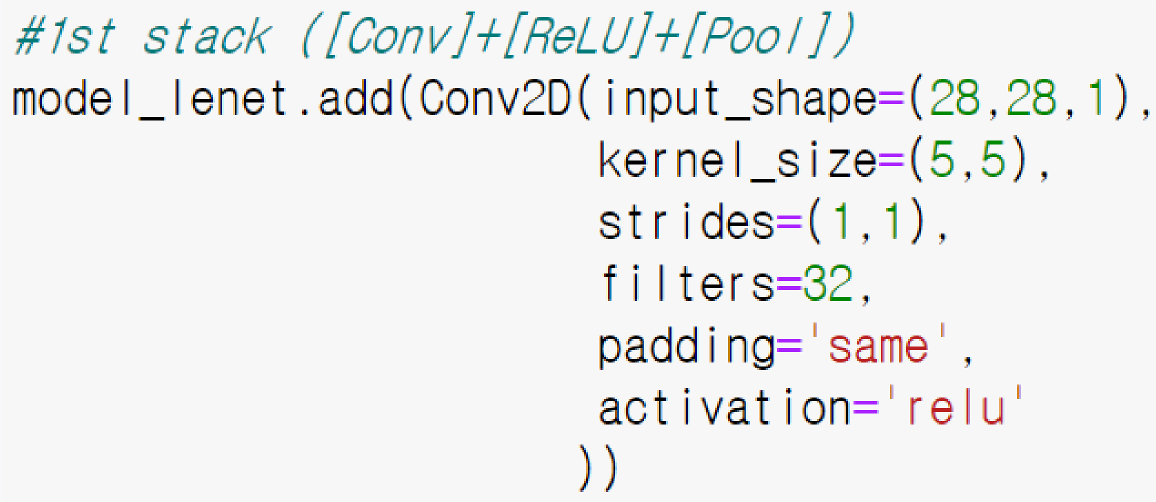
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten, Dense
```



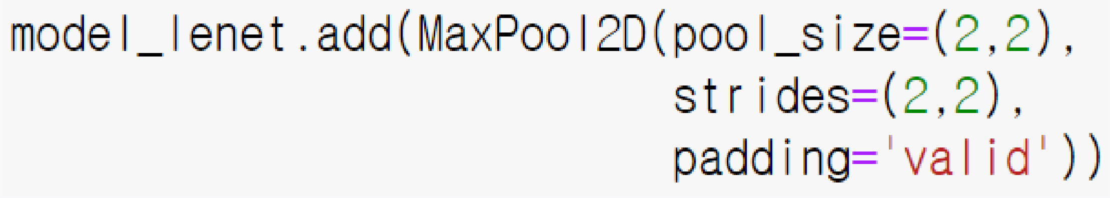
```
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import MaxPool2D
```



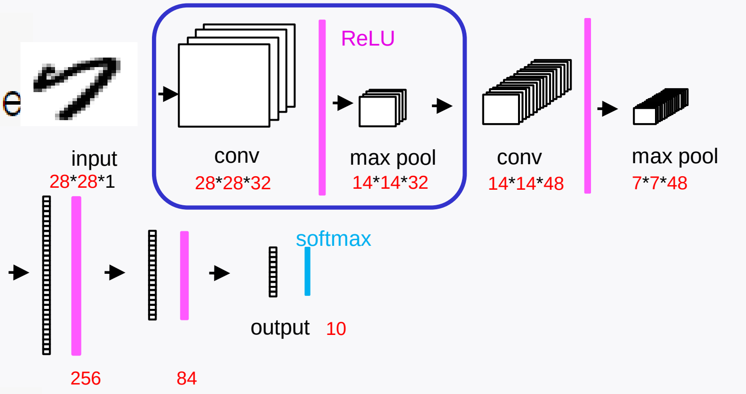
```
model_lenet = Sequential()
```



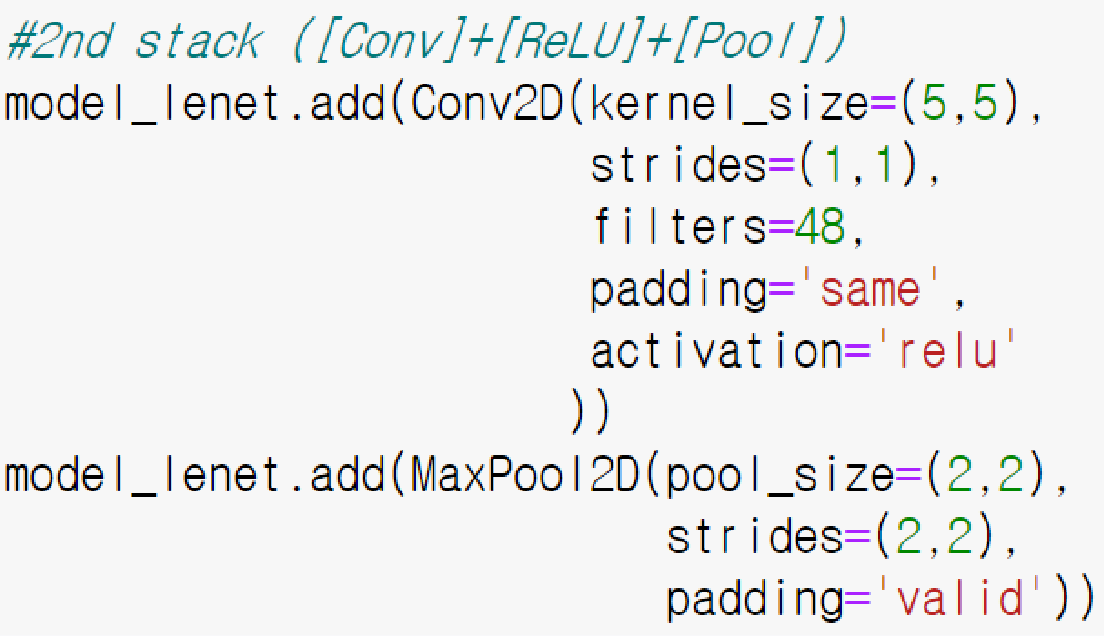
```
#1st stack ([Conv]+[ReLU]+[Pool])
model_lenet.add(Conv2D(input_shape=(28,28,1),
                       kernel_size=(5,5),
                       strides=(1,1),
                       filters=32,
                       padding='same',
                       activation='relu'
                      ))
```



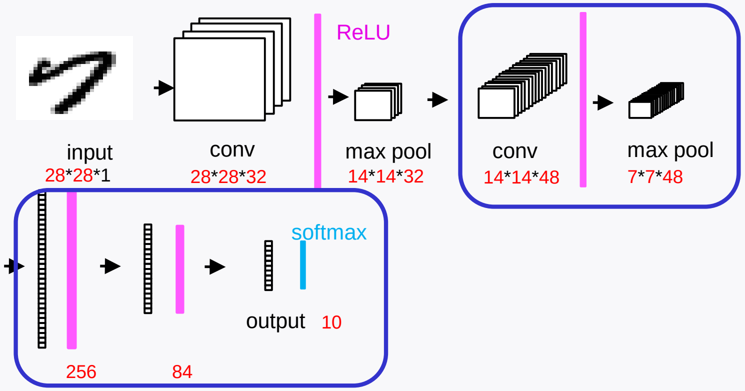
```
model_lenet.add(MaxPool2D(pool_size=(2,2),
                          strides=(2,2),
                          padding='valid'))
```

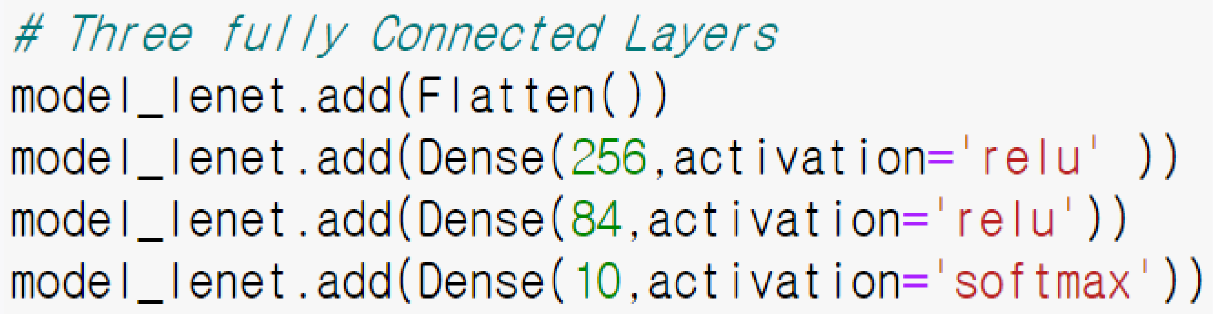


LeNet-5: Tensorflow coding



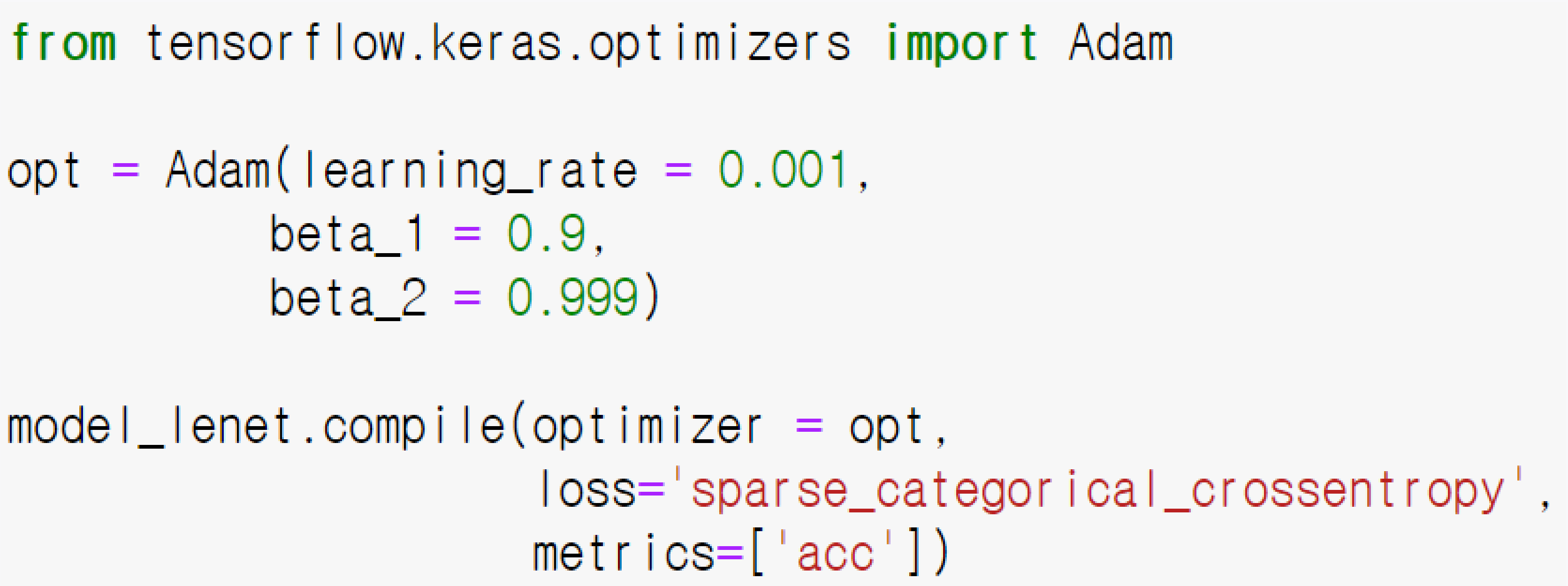
```
#2nd stack ([Conv]+[ReLU]+[Pool])
model_lenet.add(Conv2D(kernel_size=(5,5),
                       strides=(1,1),
                       filters=48,
                       padding='same',
                       activation='relu'
                      ))
model_lenet.add(MaxPool2D(pool_size=(2,2),
                          strides=(2,2),
                          padding='valid'))
```





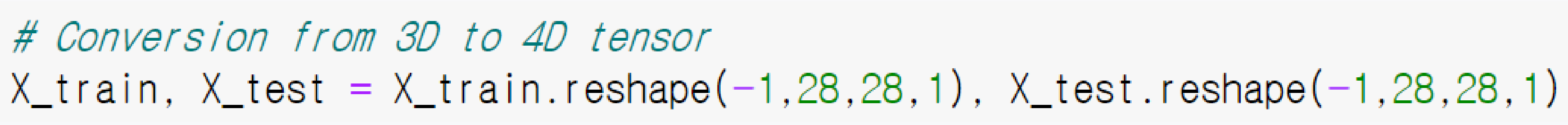
```
# Three fully Connected Layers
model_lenet.add(Flatten())
model_lenet.add(Dense(256,activation='relu' ))
model_lenet.add(Dense(84,activation='relu'))
model_lenet.add(Dense(10,activation='softmax'))
```

Compile

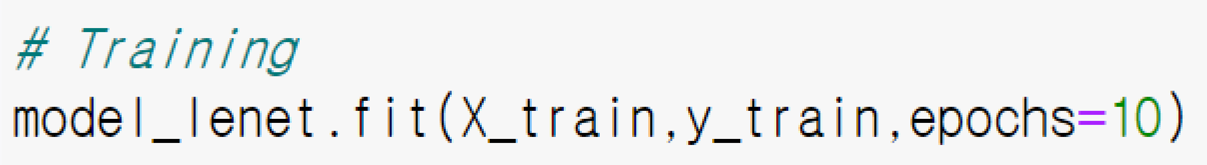


```python
from tensorflow.keras.optimizers import Adam

opt = Adam(learning_rate = 0.001,
           beta_1 = 0.9,
           beta_2 = 0.999)

model_lenet.compile(optimizer = opt,
                    loss='sparse_categorical_crossentropy',
                    metrics=['acc'])
```

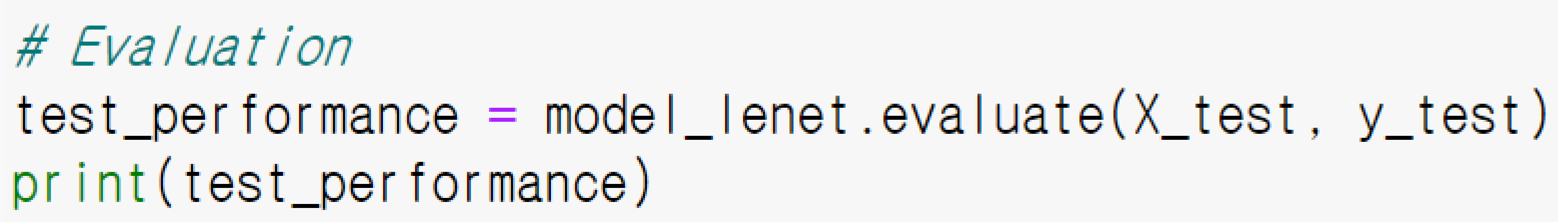
Training & evaluation



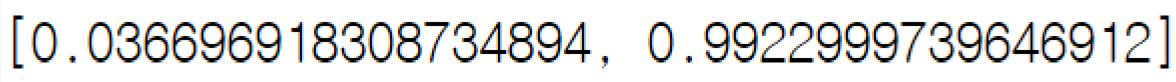
```
# Conversion from 3D to 4D tensor
X_train, X_test = X_train.reshape(-1,28,28,1), X_test.reshape(-1,28,28,1)
```



```
# Training
model_lenet.fit(X_train,y_train,epochs=10)
```



```
# Evaluation
test_performance = model_lenet.evaluate(X_test, y_test)
print(test_performance)
```



```
[0.03669691830873489, 0.9922999739646912]
```

Look ahead

Will implement simplified AlexNet in the context of image recognition (CIFAR10).