Deep Learning
Deep Neural Network

Yoon Joong Kim,
Hanbat National University

Deep Learning

CNN(2)
Convolutional Neural Networks

CNN(1): Background of Convolutional Neural Networks
CNN, Convolution Neural Network
Case study

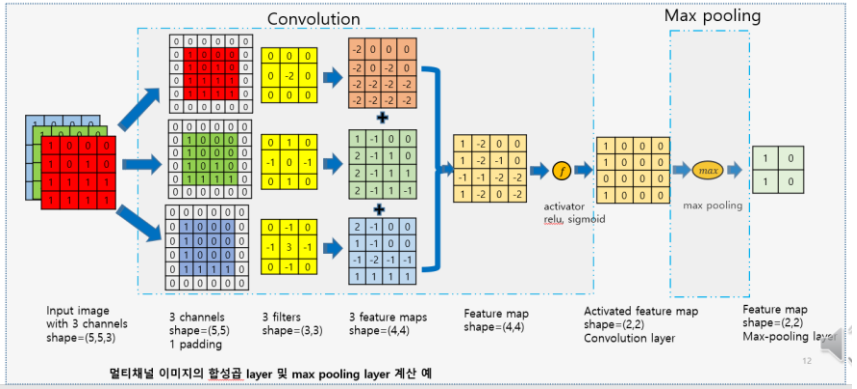
CNN(2): CNN examples

Yoon Joong Kim
Department of Computer Engineering, Hanbat National University
yjkim@hanbat.ac.kr

Contents

5. CNN examples

- 5.1 CNN 구성예



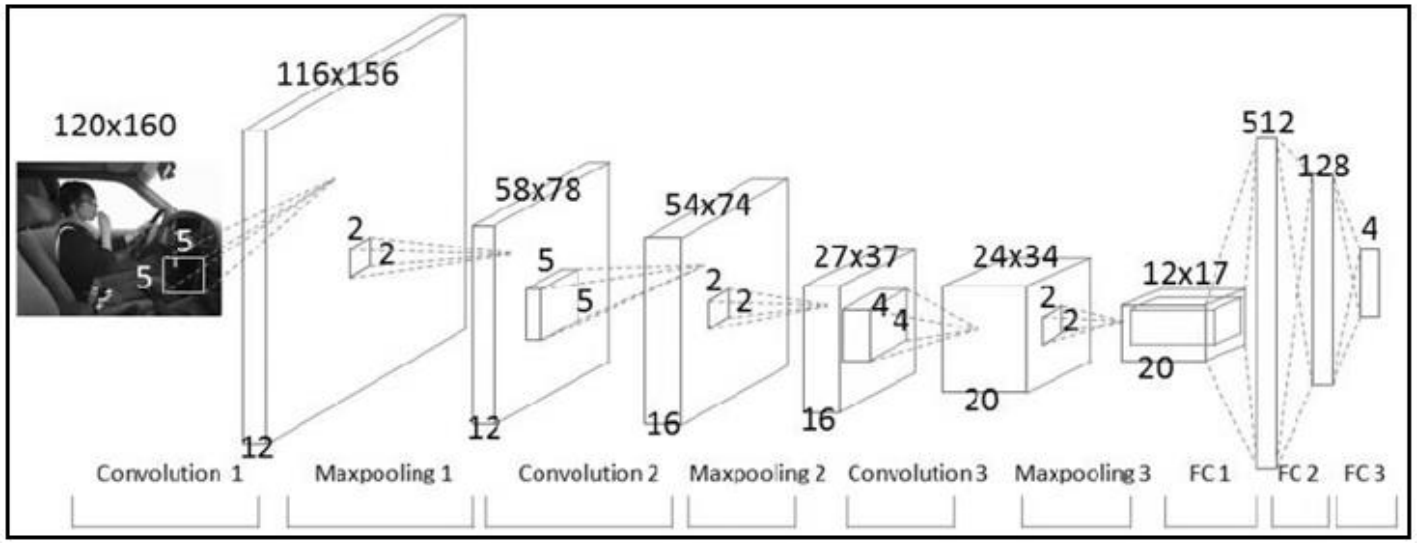
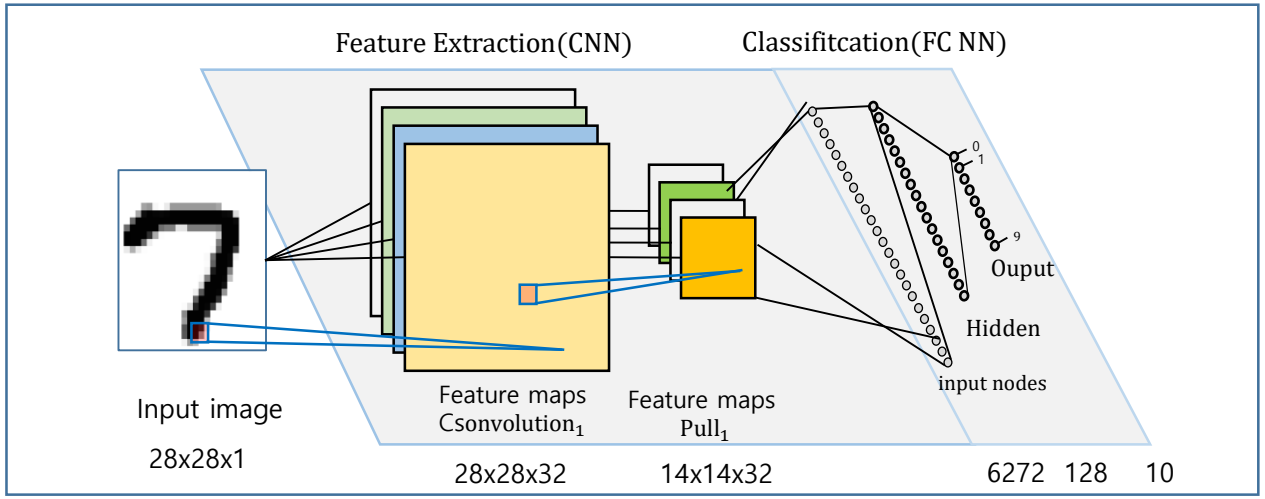


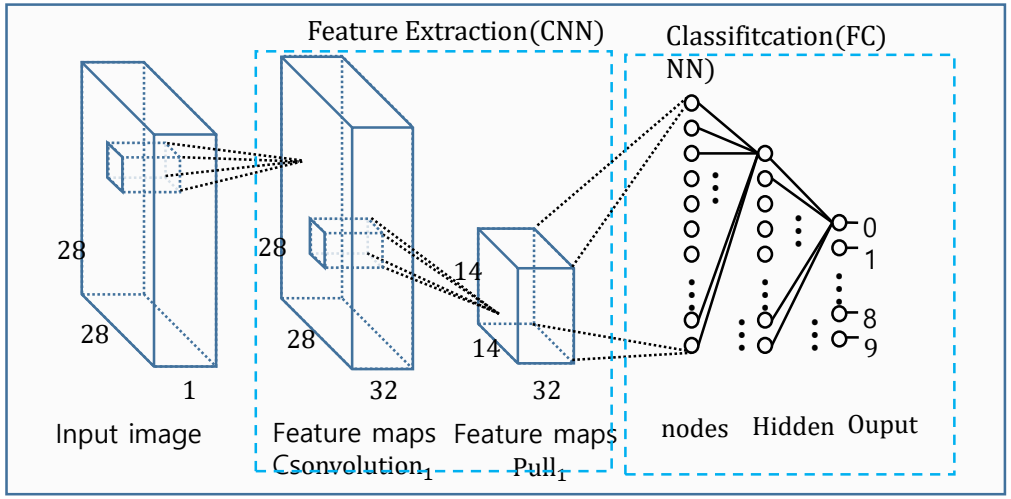
그림 8: 전형적인 CNN, 출처: https://www.researchgate.net/figure/Architecture-of-our-unsupervised-CNN-Network-contains-three-stages-each-of-which_283433254

```
model = Sequential()
model.add(Conv2D(12, kernel_size=(5, 5), activation='relu', input_shape=(120, 160, 1)))      #116x156
model.add(MaxPooling2D(pool_size=(2, 2)))                                                    #  58x78
model.add(Conv2D(16, kernel_size=(5, 5), activation='relu'))          #54x74
model.add(MaxPooling2D(pool_size=(2, 2)))                            #   27x37
model.add(Conv2D(20, kernel_size=(4, 4), activation='relu'))          # 24x34
model.add(MaxPooling2D(pool_size=(2, 2)))                            #   12x17
model.add(Flatten())                                                 #12x17=204
model.add(Dense(128, activation='relu'))
model.add(Dense(4, activation='softmax'))
```

5.2 Mnist digit classifier with 1 Convolution layer

- 1 Convolution NN



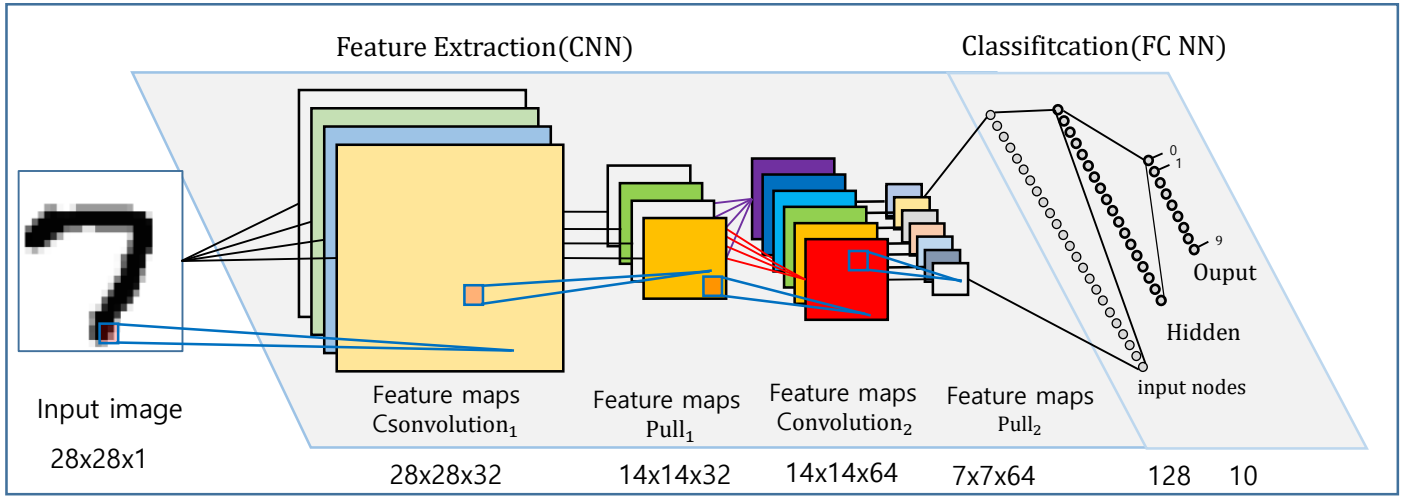


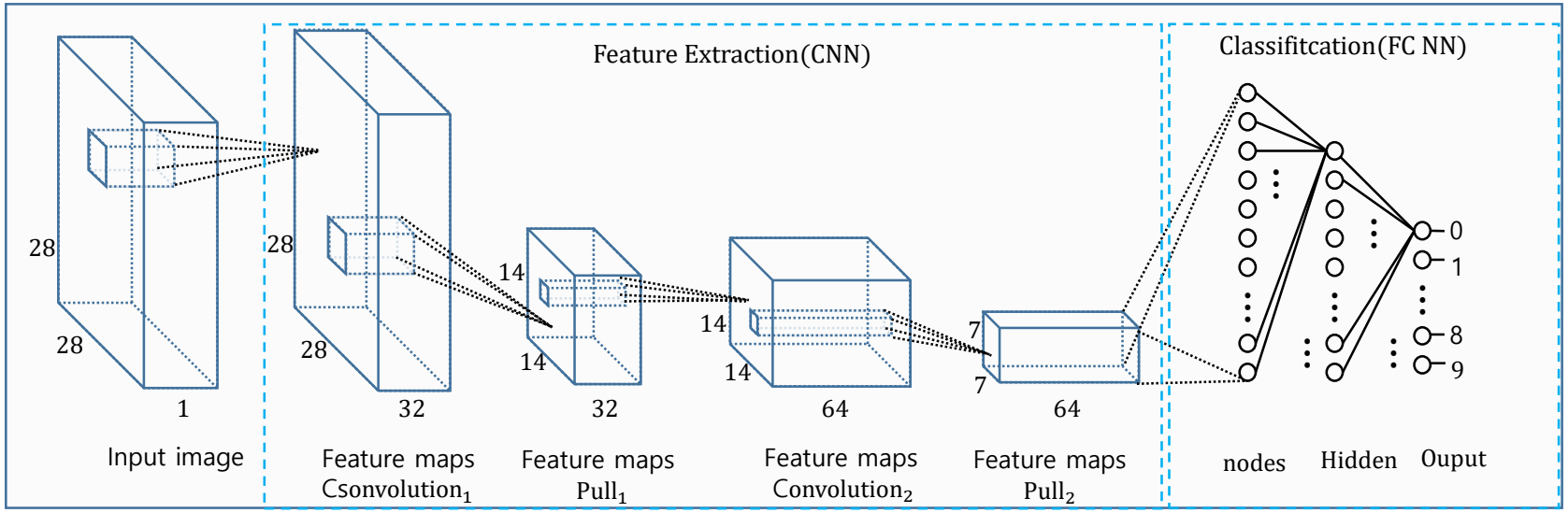

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',input_shape=input_shape)) #26,26,32
model.add(MaxPooling2D(pool_size=(2, 2)))                                             #14,14,32
model.add(Dropout(0.25))
model.add(Flatten())                                                                  #6272 = 14*14*32
model.add(Dense(128, activation='relu'))                                              #128
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))                                   #10
```

```
Epoch 10/12
60000/60000 [==============================] - 4s 73us/step - loss: 0.0459 - acc: 0.9852 - val_loss: 0.0402 - val_acc: 0.9865
Epoch 11/12
60000/60000 [==============================] - 5s 78us/step - loss: 0.0431 - acc: 0.9859 - val_loss: 0.0373 - val_acc: 0.9881
Epoch 12/12
60000/60000 [==============================] - 5s 75us/step - loss: 0.0389 - acc: 0.9873 - val_loss: 0.0357 - val_acc: 0.9880
Test loss: 0.035724134841622436
Test accuracy: 0.988
```

5.2 Mnist digit classifier with deep Convolution layers

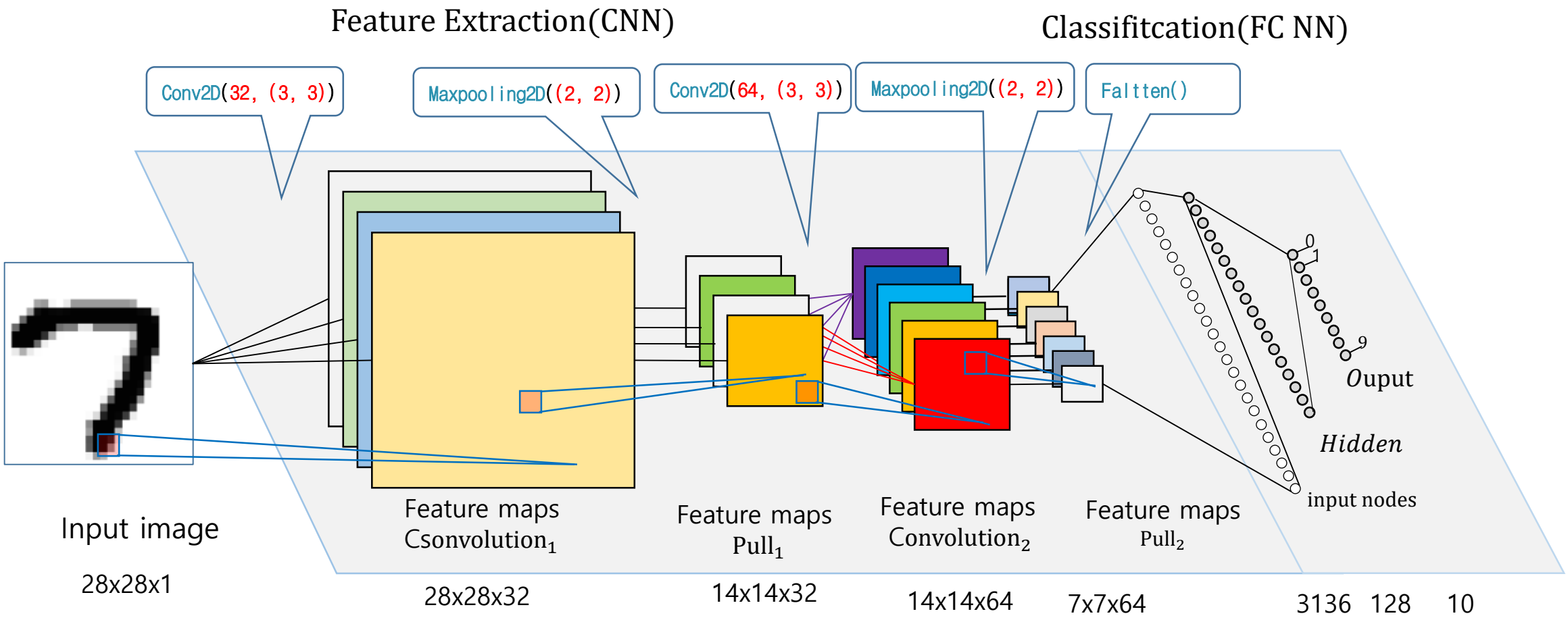
- 2 Convolution NN





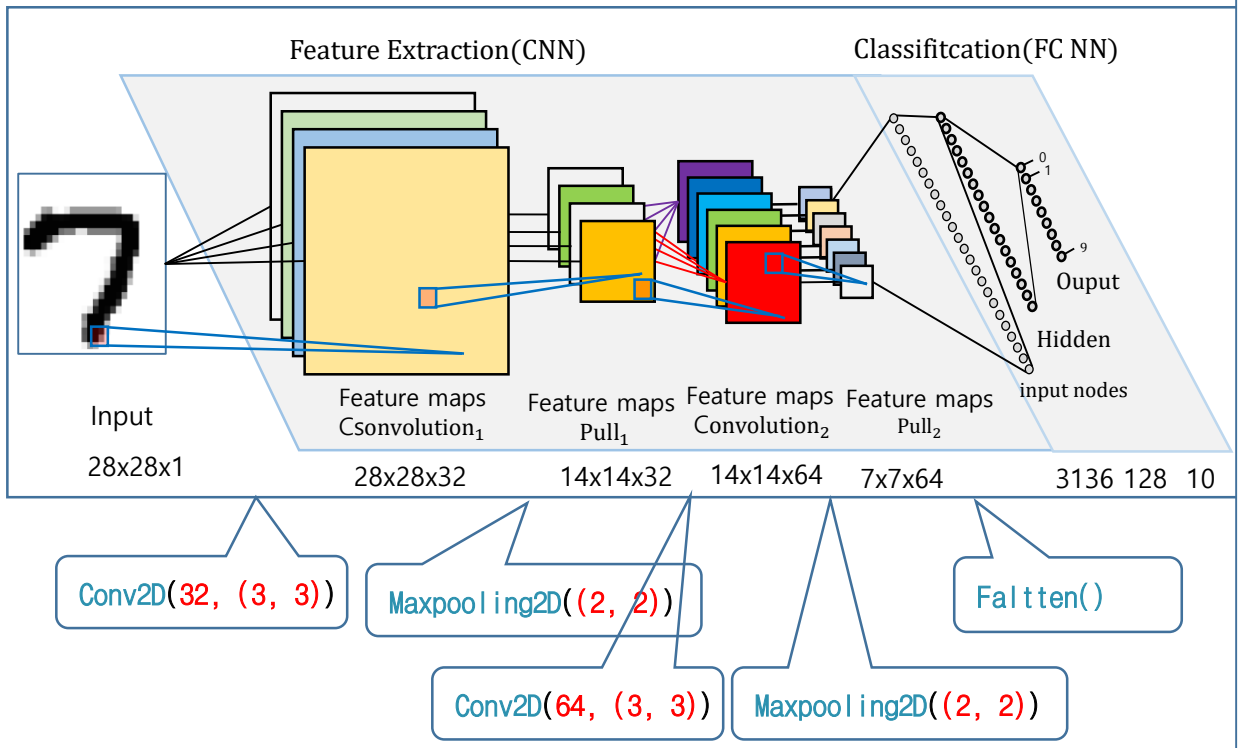
5.2 Mnist digit classifier with deep Convolution layers

- 2 Convolution NN



5.2 Mnist digit classifier with deep Convolution layers

- 2 Convolution NN




```
model = Sequential()
#convolution layer
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
                 input_shape=input_shape))                    #28,28,32
model.add(MaxPooling2D(pool_size=(2, 2)))                     #14,14,32
model.add(Dropout(0.25))
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu')) #14,14,64
model.add(MaxPooling2D(pool_size=(2, 2)))                     #7,7,64
model.add(Dropout(0.25))


model.add(Flatten())                                          #3136 = 7*7*64
model.add(Dense(128, activation='relu'))                      #128
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))           #10
```

5.2 Mnist digit classifier with deep Convolution layers

```
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.utils import to_categorical as ohe

batch_size = 128; num_classes = 10; epochs = 12

# input image dimensions
img_rows, img_cols = 28, 28
input_shape=(img_rows, img_cols,1)

# loead mnist image and train and test datasets
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols,
                  1).astype(float)/255
x_test  = x_test.reshape(x_test.shape[0],   img_rows, img_cols,
                  1).astype(float)/255
print('x_train shape:{}  y_train shape:{} ', x_train.shape,y_train.shape)
print('x_test shape :{}  y_test shape :{} ', x_test.shape,y_test.shape)

# convert label  to one_hot_encoding(label,10)
y_train_ohe = ohe(y_train, num_classes)
y_test_ohe  = ohe(y_test, num_classes)
```

```
model = Sequential()
#convolution layer
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
                  input_shape=input_shape))              #26,26,32
model.add(MaxPooling2D(pool_size=(2, 2)))                #14,14,32
model.add(Dropout(0.25))

model.add(Conv2D(64, kernel_size=(3, 3), activation='relu')) #14,14,64
model.add(MaxPooling2D(pool_size=(2, 2)))                #7,7,64
model.add(Dropout(0.25))

model.add(Flatten())                                     #3136 = 7*7*64
model.add(Dense(128, activation='relu'))                 #128
model.add(Dropout(0.25))
model.add(Dense(num_classes, activation='softmax')) #10

model.compile(
                  loss='categorical_crossentropy',
                  optimizer='adam
                  metrics=['accuracy'])

model.fit(x_train, y_train_ohe, validation_data=(x_test, y_test_ohe),
                  batch_size=batch_size, epochs=epochs, verbose=1)

score = model.evaluate(x_test, y_test_ohe, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

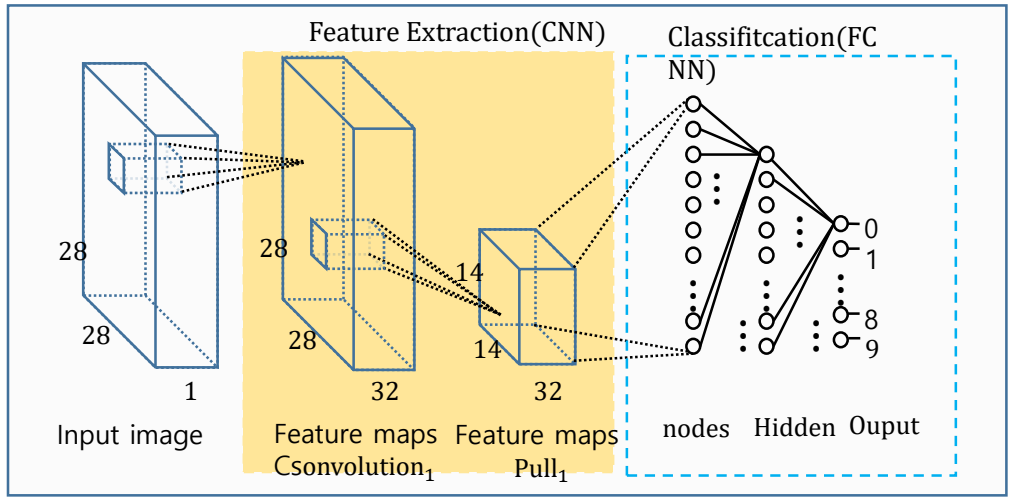
5.2 Mnist digit classifier with deep Convolution layers

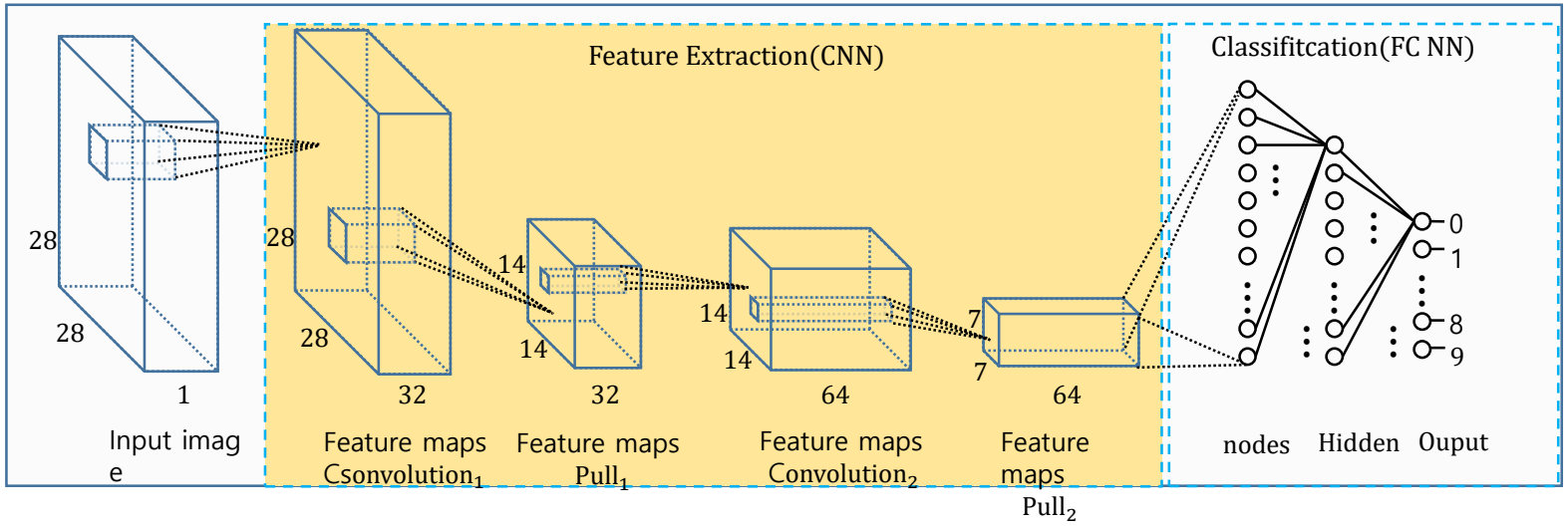
```
from keras.data
from keras.mod
from keras.laye
from keras.laye
from keras.utils

batch_size = 1

# input image
img_rows, img
input_shape=(i

# loead mnist
(x_train, y_train
x_train = x_trai
                1)
x_test  = x_test
                1)
print('x_train sh
print('x_test sha

# convert labe
y_train = ohe(y
y_test  = ohe(y_test, num_classes)
```

```
C:₩Program Files (x86)₩Microsoft Visual Studio₩Shared₩Python36_64₩python.exe
Using TensorFlow backend.
x_train shape:{}  y_train shape:{}  (60000, 28, 28, 1) (60000,)
x_test shape :{}  y_test shape :{}  (10000, 28, 28, 1) (10000,)
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 10s 166us/step - loss: 0.2960 - acc: 0.9077 - val_loss: 0.0724 - val_acc: 0.9766
Epoch 2/12
60000/60000 [==============================] - 5s 87us/step - loss: 0.0897 - acc: 0.9727 - val_loss: 0.0442 - val_acc: 0.9859
Epoch 3/12
60000/60000 [==============================] - 5s 88us/step - loss: 0.0655 - acc: 0.9799 - val_loss: 0.0331 - val_acc: 0.9882
Epoch 4/12
60000/60000 [==============================] - 5s 88us/step - loss: 0.0526 - acc: 0.9835 - val_loss: 0.0307 - val_acc: 0.9896
Epoch 5/12
60000/60000 [==============================] - 5s 89us/step - loss: 0.0474 - acc: 0.9855 - val_loss: 0.0278 - val_acc: 0.9906
Epoch 6/12
60000/60000 [==============================] - 5s 88us/step - loss: 0.0409 - acc: 0.9871 - val_loss: 0.0245 - val_acc: 0.9912
Epoch 7/12
60000/60000 [==============================] - 5s 90us/step - loss: 0.0359 - acc: 0.9886 - val_loss: 0.0249 - val_acc: 0.9916
Epoch 8/12
60000/60000 [==============================] - 5s 89us/step - loss: 0.0340 - acc: 0.9889 - val_loss: 0.0250 - val_acc: 0.9912
Epoch 9/12
60000/60000 [==============================] - 5s 91us/step - loss: 0.0299 - acc: 0.9904 - val_loss: 0.0235 - val_acc: 0.9917
Epoch 10/12
60000/60000 [==============================] - 5s 79us/step - loss: 0.0271 - acc: 0.9914 - val_loss: 0.0232 - val_acc: 0.9924
Epoch 11/12
60000/60000 [==============================] - 5s 91us/step - loss: 0.0254 - acc: 0.9915 - val_loss: 0.0207 - val_acc: 0.9932
Epoch 12/12
60000/60000 [==============================] - 5s 87us/step - loss: 0.0255 - acc: 0.9917 - val_loss: 0.0203 - val_acc: 0.9929
Test loss: 0.020317405389965 28
Test accuracy: 0.9929
```

```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

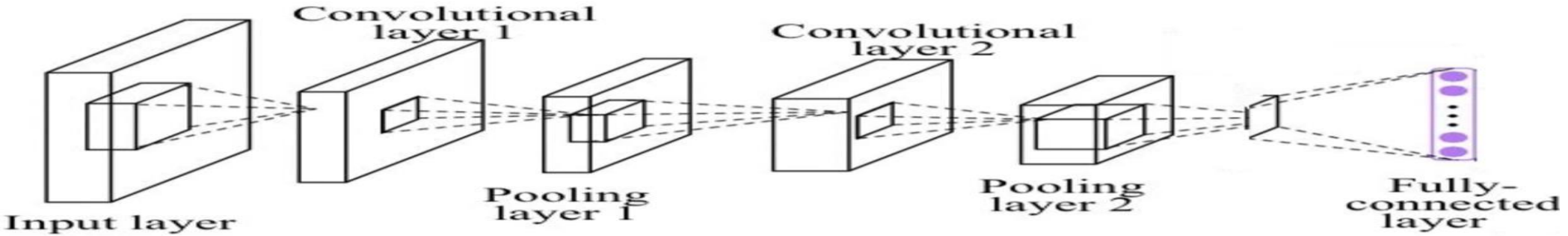
5.2 Mnist digit classifier with deep Convolution layers





5.3 exercise

- 다음 CNN구조로 minist image 인식 시스템을 구현화여 99.3%이상의 인식률을 얻을 수 있음을 확인 하시오.



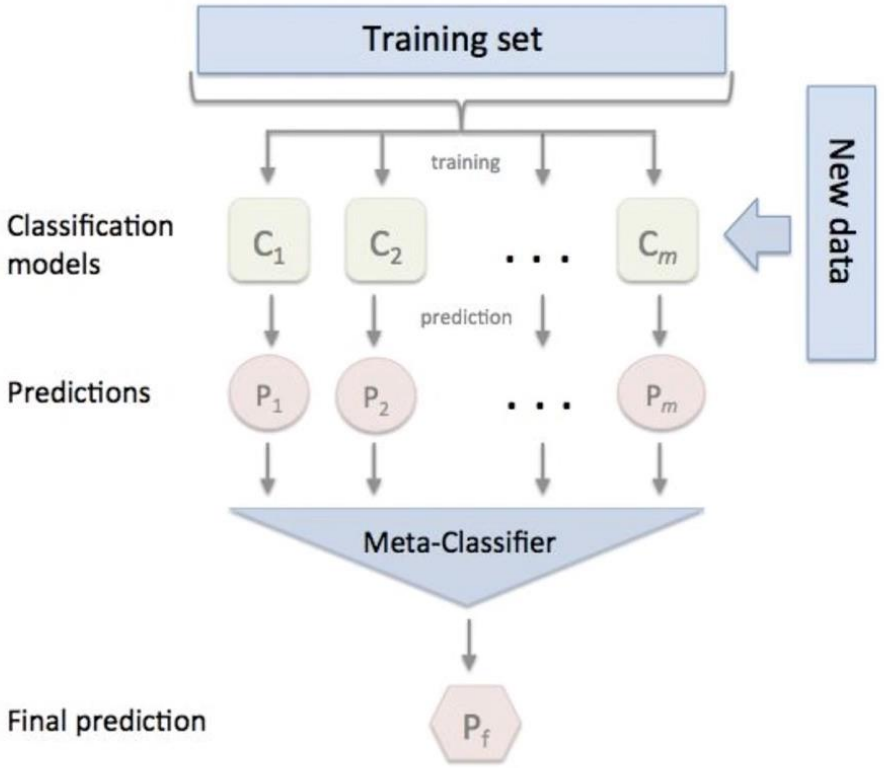
784: 28x28x1 28x28x32 14x14x32 14x14x64 7x7x64 : 3136

```
Epoch 12/12
60000/60000 [==============================
Test loss: 0.020794622257165 61
Test accuracy: 0.9935
Press any key to continue . . .
```

5.4 CIFAR-10

- CIFAR-10 데이터세트(Canadian Institute For Advanced Research)
  - 일반적으로 머신 러닝 및 컴퓨터 비전 알고리즘을 훈련시키는 데 사용되는 이미지 모음입니다.
  - 10 가지 클래스로 구성된 60,000 개의 32x32 컬러 이미지가 포함되어 있습니다.
  - 10 가지 다른 클래스는 비행기, 자동차, 새, 고양이, 사슴, 개, 개구리, 말, 배 및 트럭을 나타냅니다. 각 클래스마다 6,000 개의 이미지가 있습니다.
- [ConvNetJS demo: training on CIFAR-10]
  - http://cs.Stanford.edu/people/karpathy/convnetjs/demo/cifar10.html

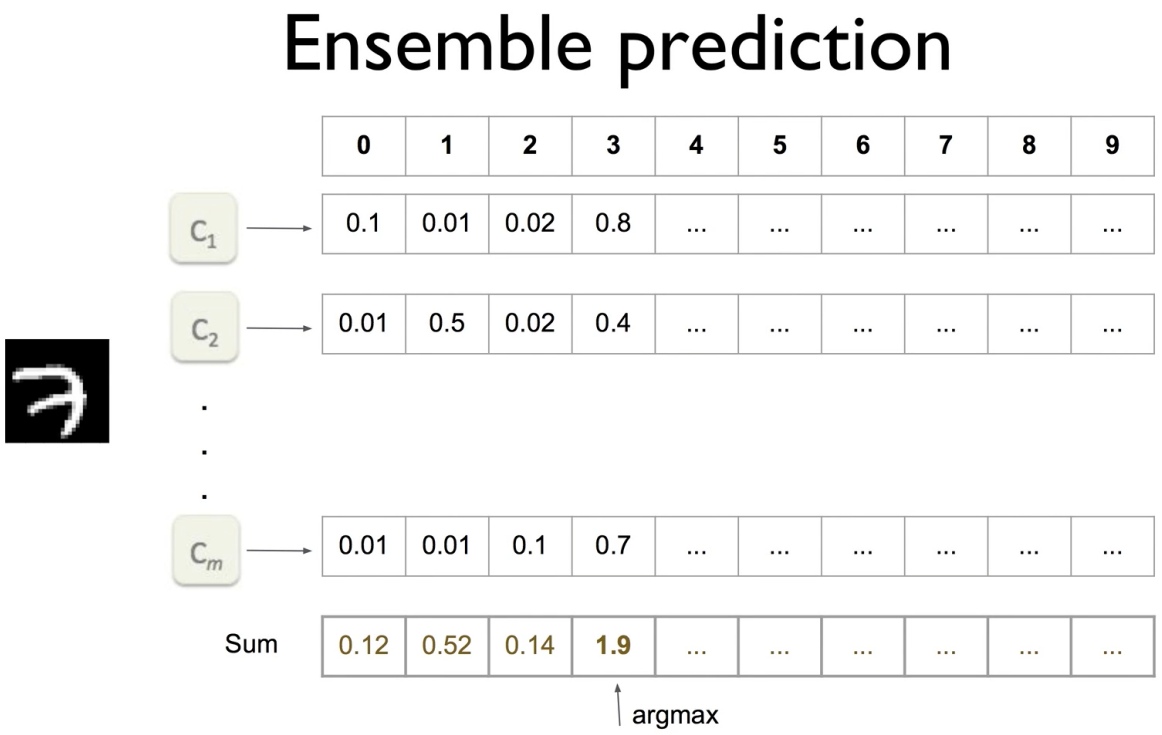
5.5 Ensemble



$$P_1 = C_1.predict(y_{train})$$
$$P_2 = C_2.predict(y_{train})$$
$$\vdots$$
$$P_n = C_n.predict(y_{train})$$

http://rasbt.github.io/mlxtend/user_guide/classifier/StackingClassifier/

5.5 Ensemble



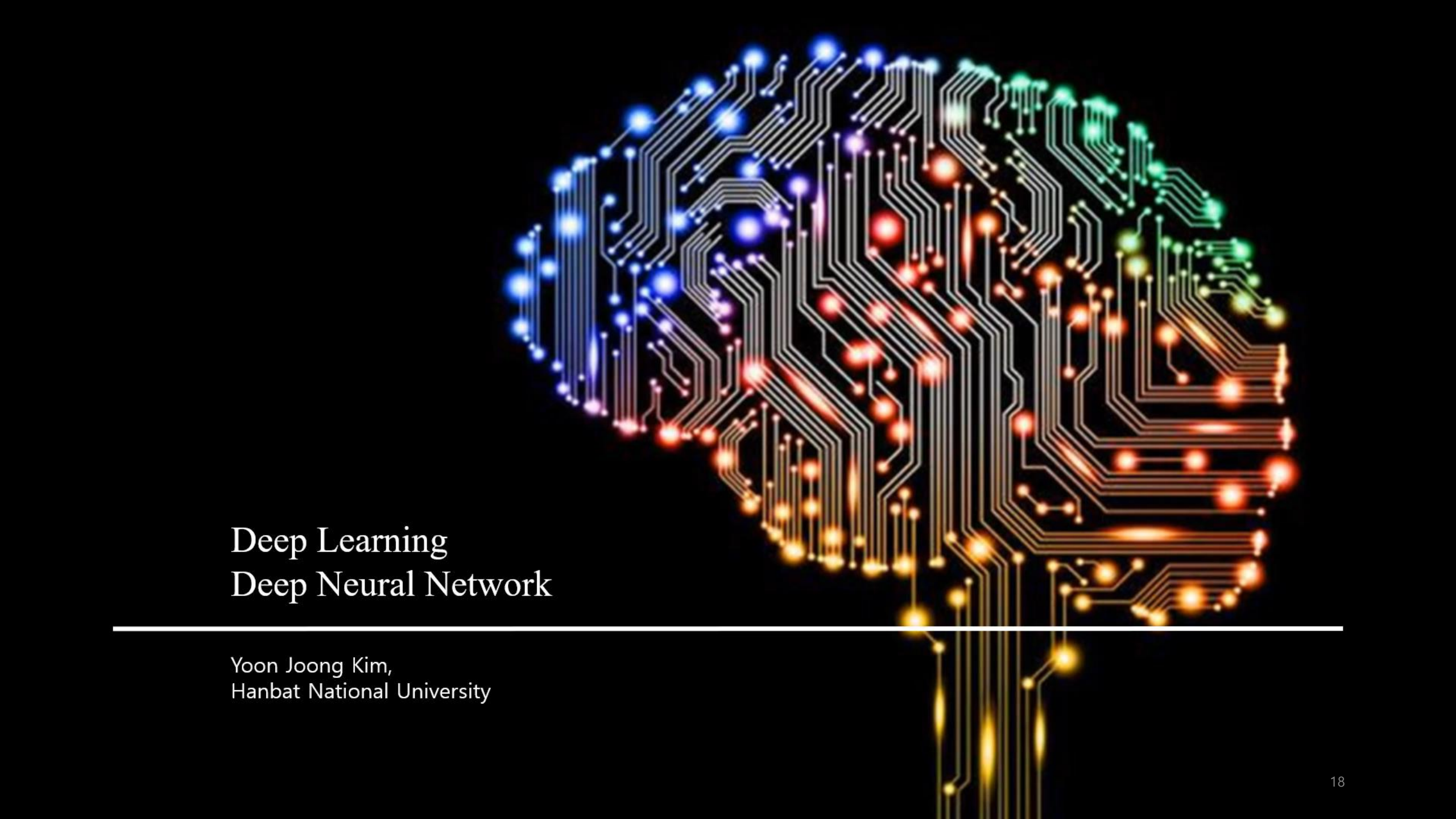

```
import numpy as np

predictions=np.zeros(10,dtype=float)
for i, model in enumerate(models):
            acc=model.evaluate(X_train,Y_train_ohe)
            print(' model[{}] acc:{}'.format(i,acc))
            p=mode.predict(X_train) #p=[0.1, 0.3, 0.2, .W0.5,,,]
            predictions =predictions+p  #[0..9] =[0..9]+[0..9]

ensenble_predicts=np.equal(np.argmax(predictions,1),
            np.argmax(Y_train_ohe,1))
ensenble_accuracy=ensenble_predicts.mean()
print('Ensenble accuracy : ',ensenble_accuracy)
```

Recap

1. Background of Convolutional Neural Networks
2. CNN, Convolution Neural Network
    1. Convolution
    2. Channel
    3. filter, kernel, stride, feature map and activation map
    4. Padding
    5. pooling layer
    6. 출력 레이어의 크기 계산
    7. Fully Connected Layer(FC layer)
4. Case study
    1. LeNet-5
    2. AlexNet
    3. GoogleNet
    4. ResNet
    5. Sentence Classification
    6. AlphaGo
5. CNN examples
    1. CNN 구성예
    2. Mnist digit classifier with CNN
    3. Exercise
    4. ConvNetJS demo: training on CIFAR-10
    5. Exnsenble

Recap

1. Background of Convolutional Neural Networks
2. CNN, Convolution Neural Network
    1. Convolution
    2. Channel
    3. filter, kernel, stride, feature map and activation map
    4. Padding
    5. pooling layer
    6. 출력 레이어의 크기 계산
    7. Fully Connected Layer(FC layer)
4. Case study
    1. LeNet-5
    2. AlexNet
    3. GoogleNet
    4. ResNet
    5. Sentence Classification
    6. AlphaGo
5. CNN examples
    1. CNN 구성예
    2. Mnist digit classifier with CNN
    3. Exercise
    4. ConvNetJS demo: training on CIFAR-10
    5. Exnsenble



Deep Learning
Deep Neural Network

Yoon Joong Kim,
Hanbat National University