



# 합성곱신경망 (CNN)을 활용한 이미지 분류기 만들기

김효관

교육목표: 합성곱 신경망 개념을 이해하고 실습한다.

# CONTENTS

- 1 합성곱 신경망
- 2 합성곱 신경망의 이해
- 3 합성곱 신경망 실습
- 4 합성곱 신경망 실습



# 1. 합성곱 신경망

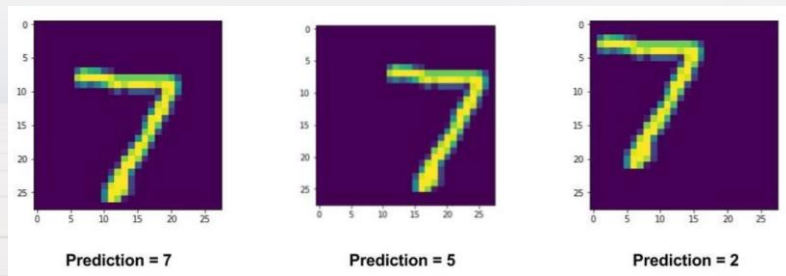
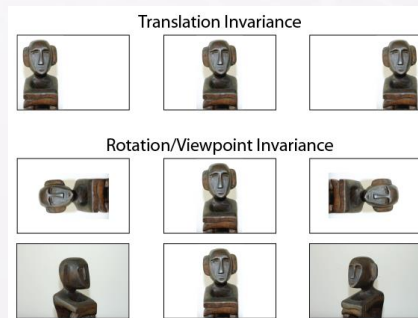
## 합성곱 신경망 등장배경

- 1 완전연결 신경망 으로는 전체 훈련데이터를 학습 후 예측
- 2 특정 특성만 활용하여 예측함이 필요함
- 3 완전연결 신경망 각 노드마다 연결되었기때문에 속도 저하 발생

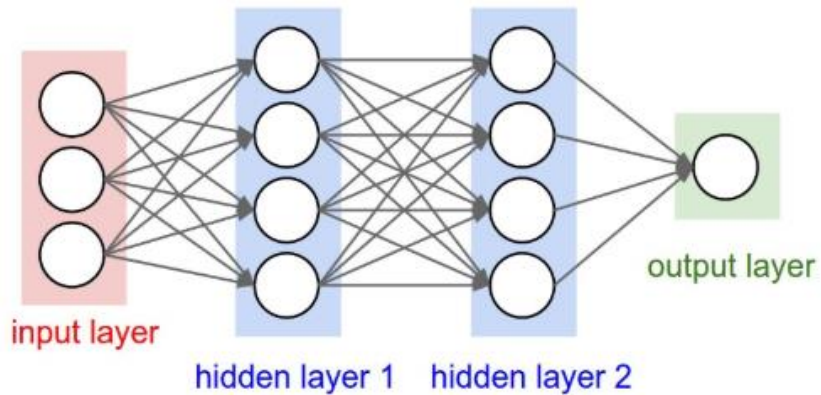
Fully Connected Layer

이미지 전체에 동일한 필터를 적용하여 유사 특성을 찾아냅니다.

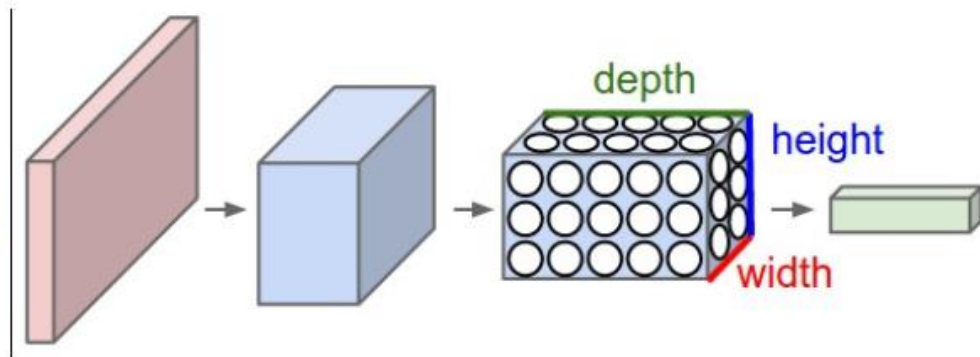
\* CNNs Translation Invariant를 뜻함



전체 결합 레이어 ( Fully Connected Layer )



합성곱 레이어 ( Convolutional Neural Network )



# 1. 합성곱 신경망

## 합성곱 신경망

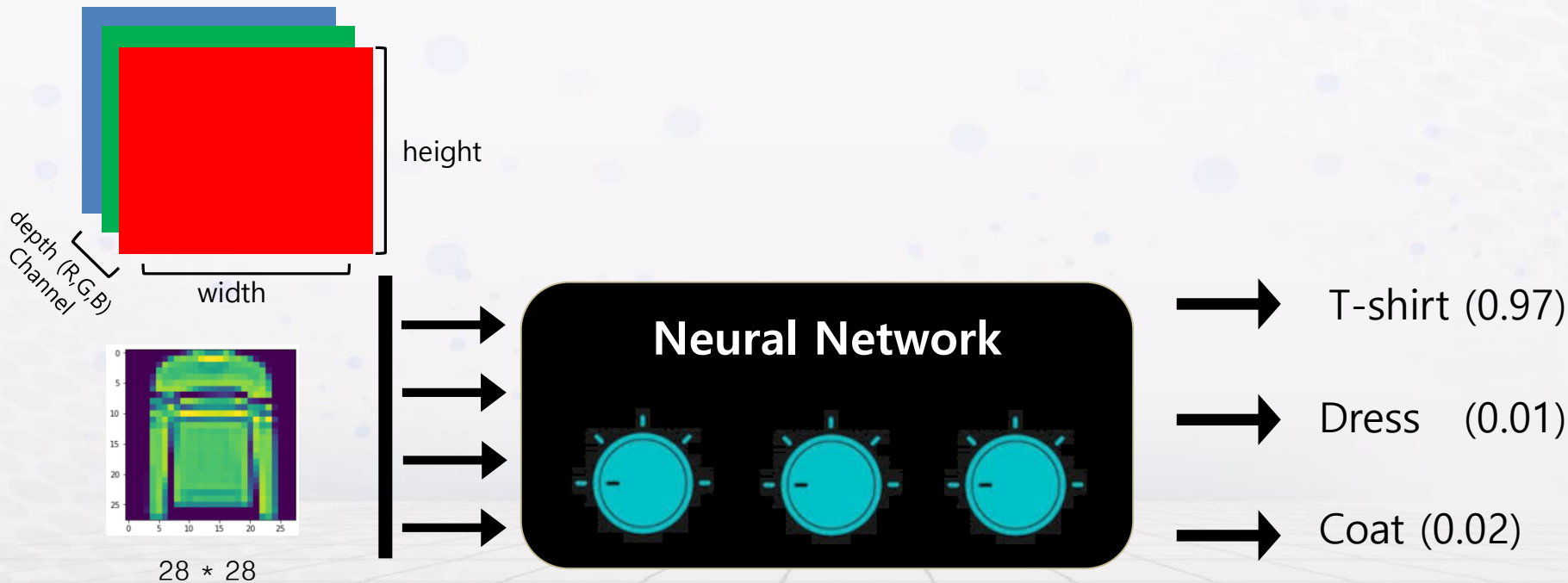
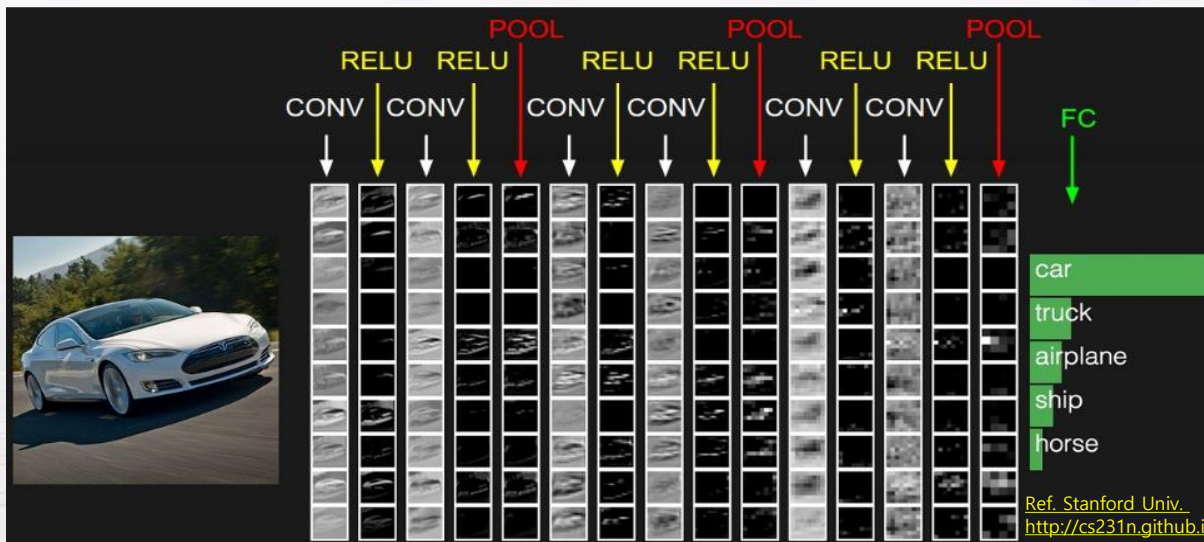
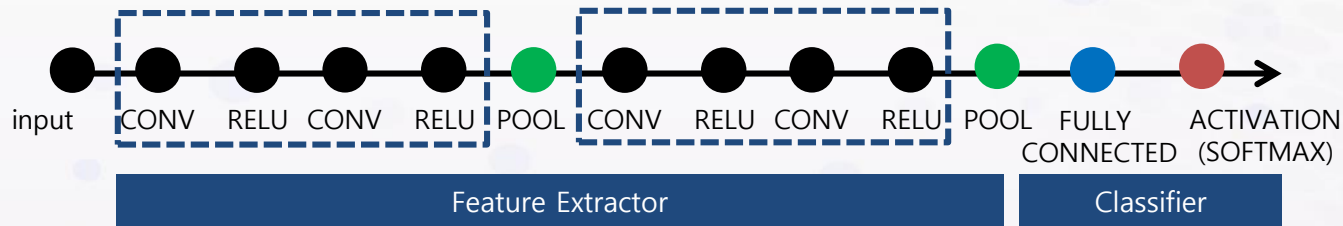


Image (28\*28, 3channel[RGB]) -> Array (28\*28\*3) 2352

# 1. 합성곱 신경망

## 합성곱 신경망 (Classical Architecture)



Ref. Stanford Univ.  
<http://cs231n.github.io/convolutional-networks/>

## 2. 합성곱 신경망의 이해

### 합성곱 레이어 (Conv Layer)

필터를 통해 Feature를 추출한다.



Input Neuron				
7	8	1	8	8
4	8	1	8	4
3	8	8	8	4
2	8	7	2	7
5	4	4	5	4

Filter		
1	0	1
1	0	1
1	1	1

Activation Map		
32		

×

=



kernel size = 3

7	8	1	8	8
4	8	1	8	4
3	8	8	8	4
2	8	7	2	7
5	4	4	5	4

×

1	0	1
1	0	1
1	1	1

=

32	56	34
33	49	33
33	39	39

$$7*1+4*1+3*1+8*0+8*0+8*1+1*1+1*1+8*1 = 32$$

$$8*1+8*1+8*1+1*0+1*0+8*1+8*1+8*1+8*1 = 56$$

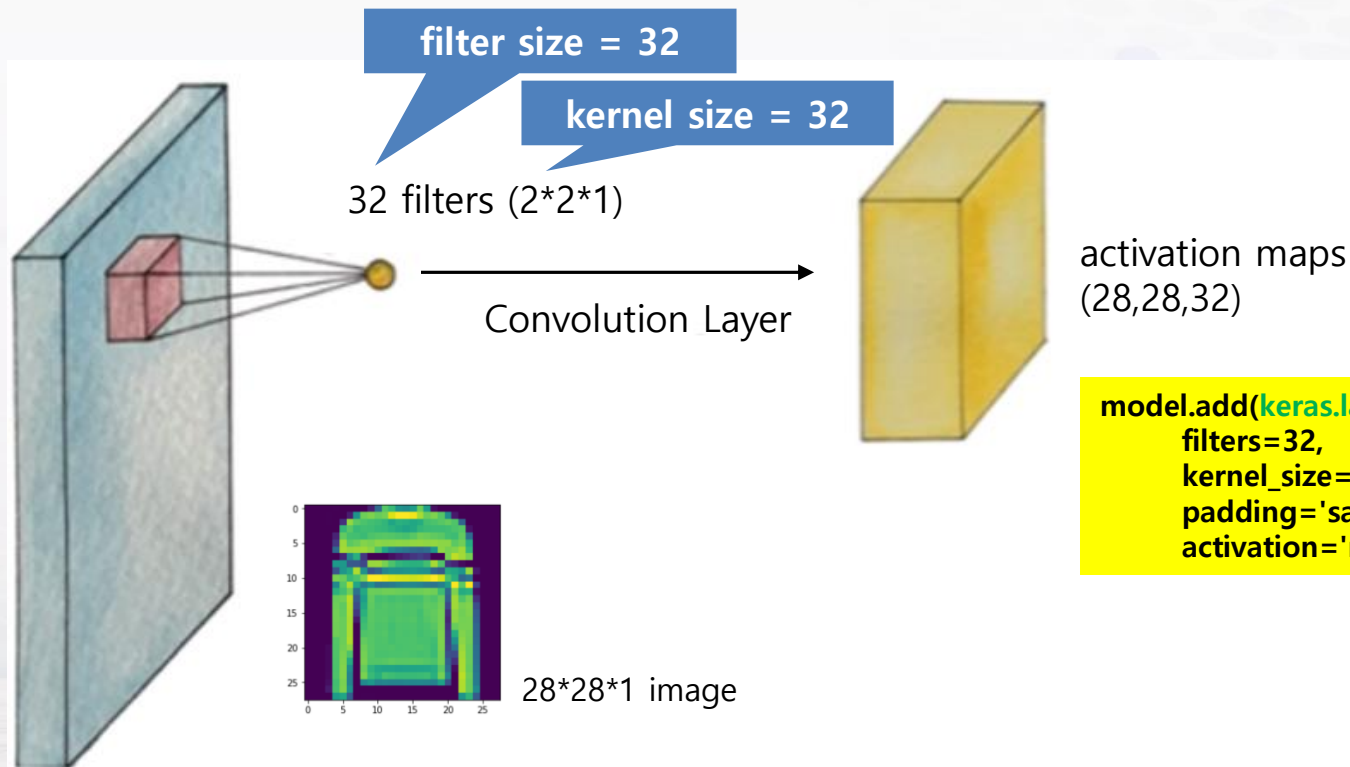
```
model.add(keras.layers.Conv2D(
    filters=32,
    kernel_size=2,
    padding='same',
    activation='relu'))
```

입력 사이즈 = 5  
필터 사이즈 = 3  
한칸씩 이동 : Stride = 1

출력 사이즈 =  
(입력사이즈-필터사이즈) / STRIDE + 1



## 2. 합성곱 신경망의 이해





## 2. 합성곱 신경망의 이해

### 합성곱 레이어 (Conv Layer)

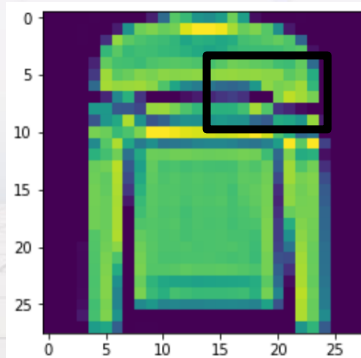
필터를 통해 Feature를 추출한다.



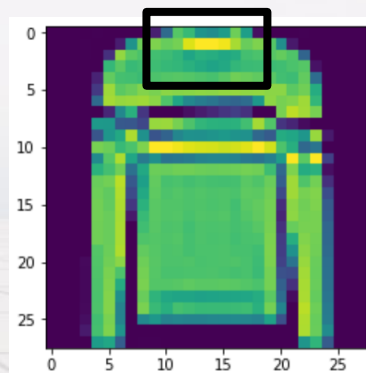
Filter는 detect 하고자 하는  
Feature를 담고 있음

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

Activation Map은  
찾고자 하는 Feature의  
유/무를 알려줌



×



## 2. 합성곱 신경망의 이해

합성곱 레이어 (Conv Layer)    필터를 통해 Feature를 추출한다.



Stride 를 몇칸씩 건너뛰면서  
필터를 이동할까요?

<https://arxiv.org/pdf/1409.1556.pdf>

논문에서는 Strider를 **한칸씩 이동하여**  
**최대한 겹치는 부분을 많이** 만들어서 Feature를 찾으라고 함.

1칸인 경우

7	8	1	8	8
4	8	1	8	4
3	8	8	8	4
2	8	7	2	7
5	4	4	5	4

7	8	1	8	8
4	8	1	8	4
3	8	8	8	4
2	8	7	2	7
5	4	4	5	4

7	8	1	8	8
4	8	1	8	4
3	8	8	8	4
2	8	7	2	7
5	4	4	5	4

2칸인 경우

7	8	1	8	8
4	8	1	8	4
3	8	8	8	4
2	8	7	2	7
5	4	4	5	4

7	8	1	8	8
4	8	1	8	4
3	8	8	8	4
2	8	7	2	7
5	4	4	5	4

## 2. 합성곱 신경망의 이해

### 합성곱 레이어 (Conv Layer)

필터를 통해 Feature를 추출한다.



Activation Map의 크기를  
줄이고 싶지 않은 경우?

**zero padding**을 사용합니다.

$\text{zero padding} = (\text{filtersize} - 1) / 2$ , Stride 1인 경우

Input Neuron				
7	8	1	8	8
4	8	1	8	4
3	8	8	8	4
2	8	7	2	7
5	4	4	5	4

×

Filter		
1	0	1
1	0	1
1	1	1

=

Activation Map				
0	0	0	0	0
0	32	56	34	0
0	33	49	33	0
0	33	39	39	0
0				

```
model.add(keras.layers.Conv2D(  
    filters=32,  
    kernel_size=2,  
    padding='same',  
    activation='relu'))
```



$$\text{Output Size} = \frac{(\text{Input Size} + 2 \times \text{Padding} - \text{Filter Size})}{\text{Stride}} + 1$$

Padding이 있을 경우 반영된 Input 크기

(Stride가 1일때) Input 크기에서 Filter 크기를 빼면, 남은 filter의 convolving 횟수가 나옴

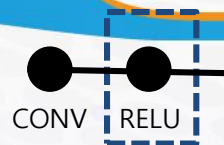
(Stride가 1이 아닐때) filter의 가능한 convolving 횟수를 stride값으로 나눔

Convolving하기 전, 첫번째 filter가 input과 inner product를 수행한 횟수

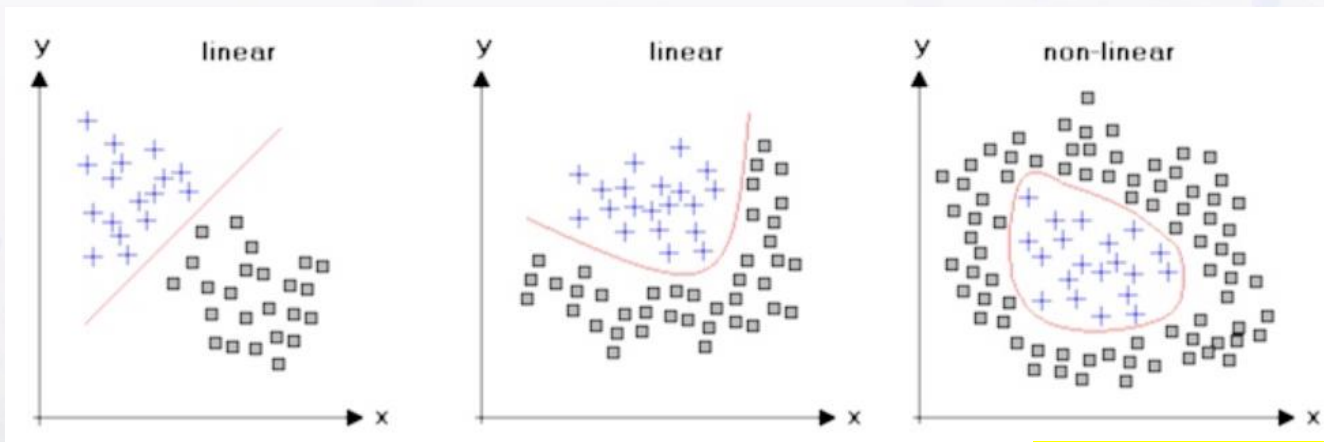
## 2. 합성곱 신경망의 이해

### Activation 레이어

Conv 레이어 다음에 사용하는 non linear 레이어  
Conv 출력을 Activation Map이라고 부르는 이유



Nonlinear Layer가 없다면 Matrix 연산만 하고 **Linear**한 단순 문제밖에 해결하지 못함



쉽게 말하면 정확도가 떨어짐

```
model.add(keras.layers.Conv2D(  
    filters=32,  
    kernel_size=2,  
    padding='same',  
    activation='relu'))
```

## 2. 합성곱 신경망의 이해

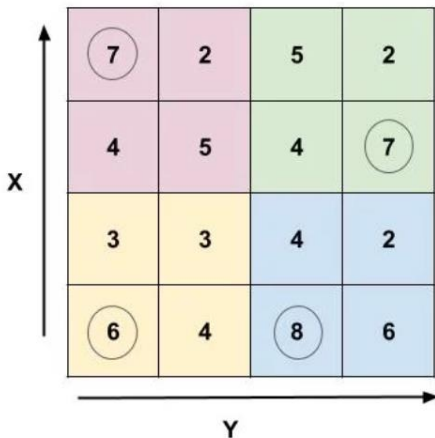
### 풀링 레이어 (Pooling Layer)

연속적인 Conv 레이어 사이에 데이터 압축위함



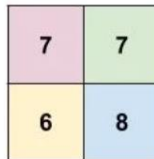
- 계산량이 줄어들어 **속도 개선에 영향을 주고**, 적은 파라미터는 오버피팅을 방지함 (dropout에서 뉴런을 일부 끄고 켜므로써 오버피팅 방지하는것과 비슷), max/sum/avg등 레이어 존재하나 max위주사용

Single Depth Slice



```
model.add(keras.layers.MaxPooling2D(pool_size=2))
```

Max Pool with  
2x2 Filters & Stride 2

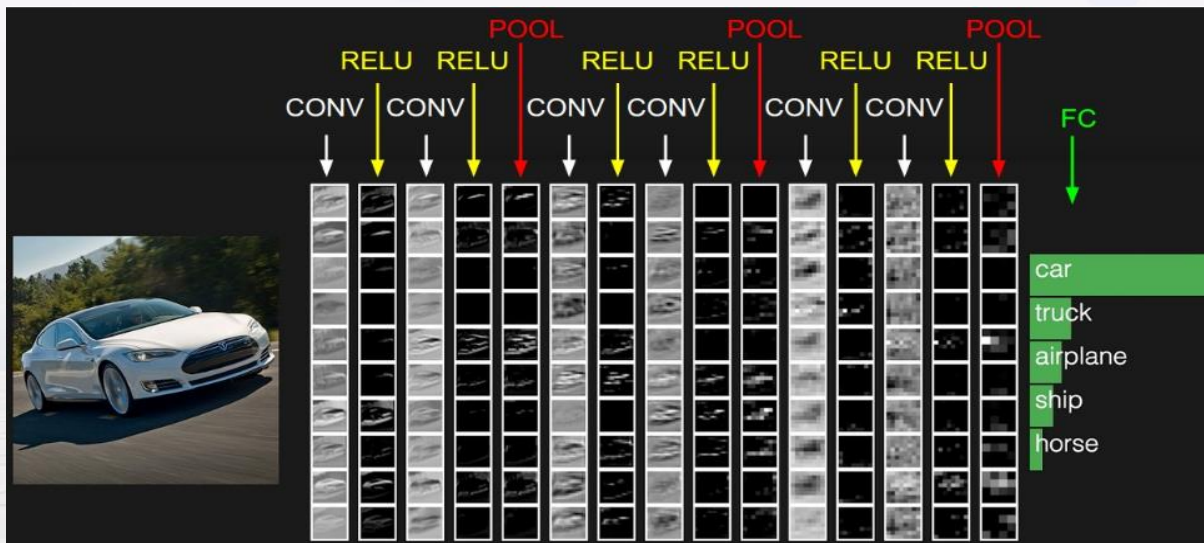
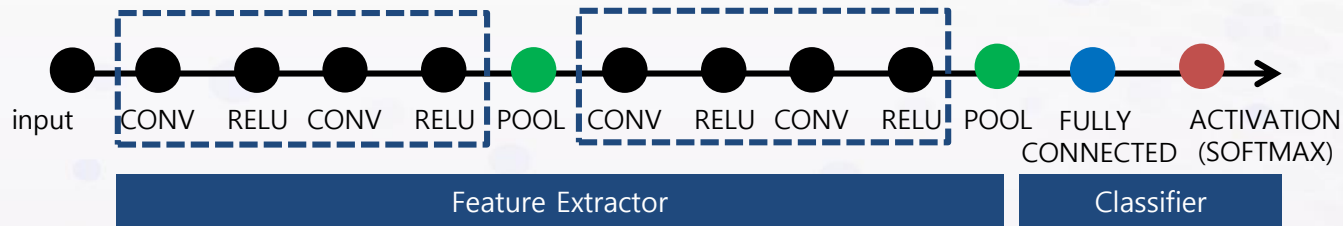


실제 주요 특징만 추출하여  
이미지가 약간 틀어져도  
인식하게 해줌

필수는 아님 (선택적 사용)

# 1. 합성곱 신경망

## 합성곱 신경망 (Classical Architecture)



Ref. Stanford Univ.

<http://cs231n.github.io/convolutional-networks/>



## 2. 합성곱 신경망의 이해

전연결 레이어 (Fully Conn Layer)

성능 향상을 위해 DROPOUT 활용



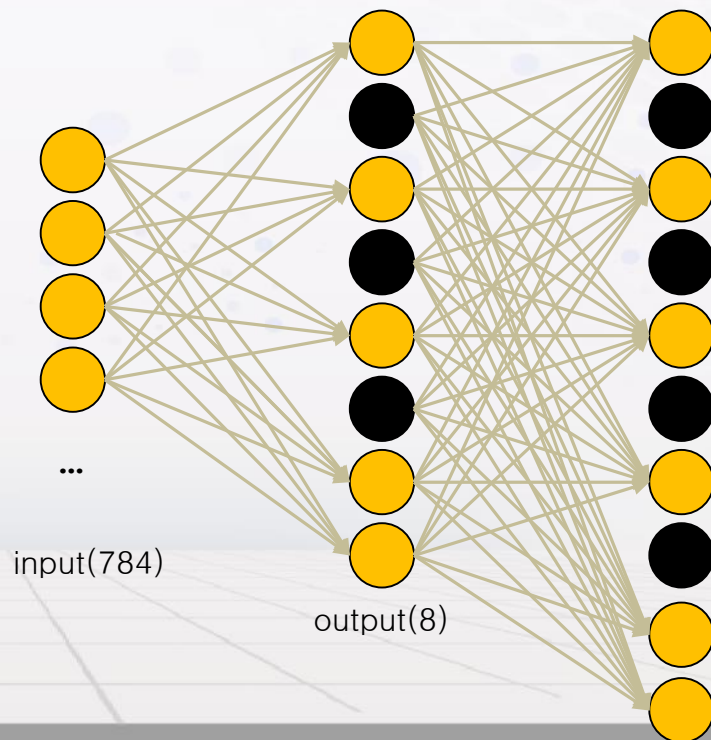
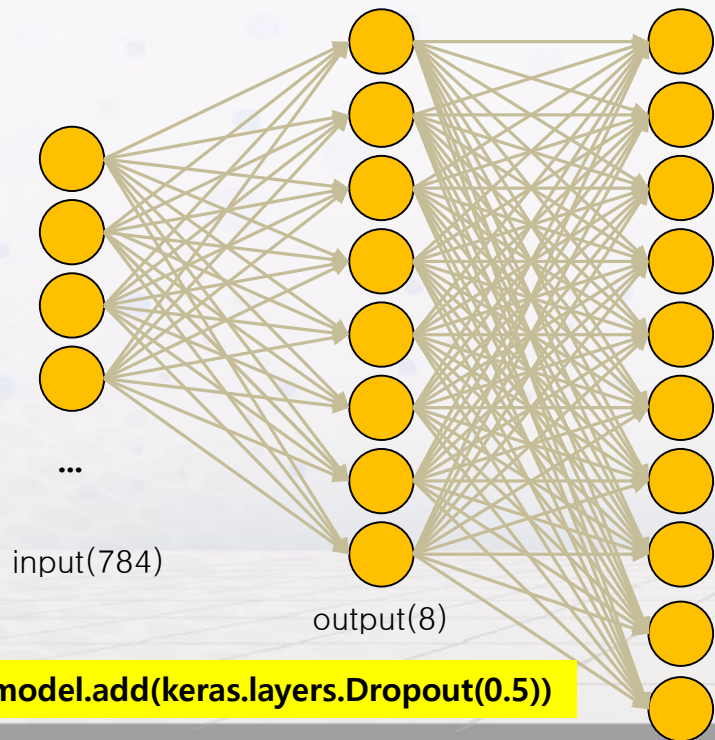
CNN 가장 마지막 레이어로 N개 클래스로 분류

## 2. 합성곱 신경망의 이해

Dropout 레이어 (dropout rate = weight 사용할 비율, 0.6은 60% 사용)



- overfitting을 줄이기 위해 전체 weight를 계산에 참여시키지 않고 layer에 포함된 일부 weight만 참여시킴



```
model.add(keras.layers.Dropout(0.5))
```

### 3. 합성곱 신경망 실습

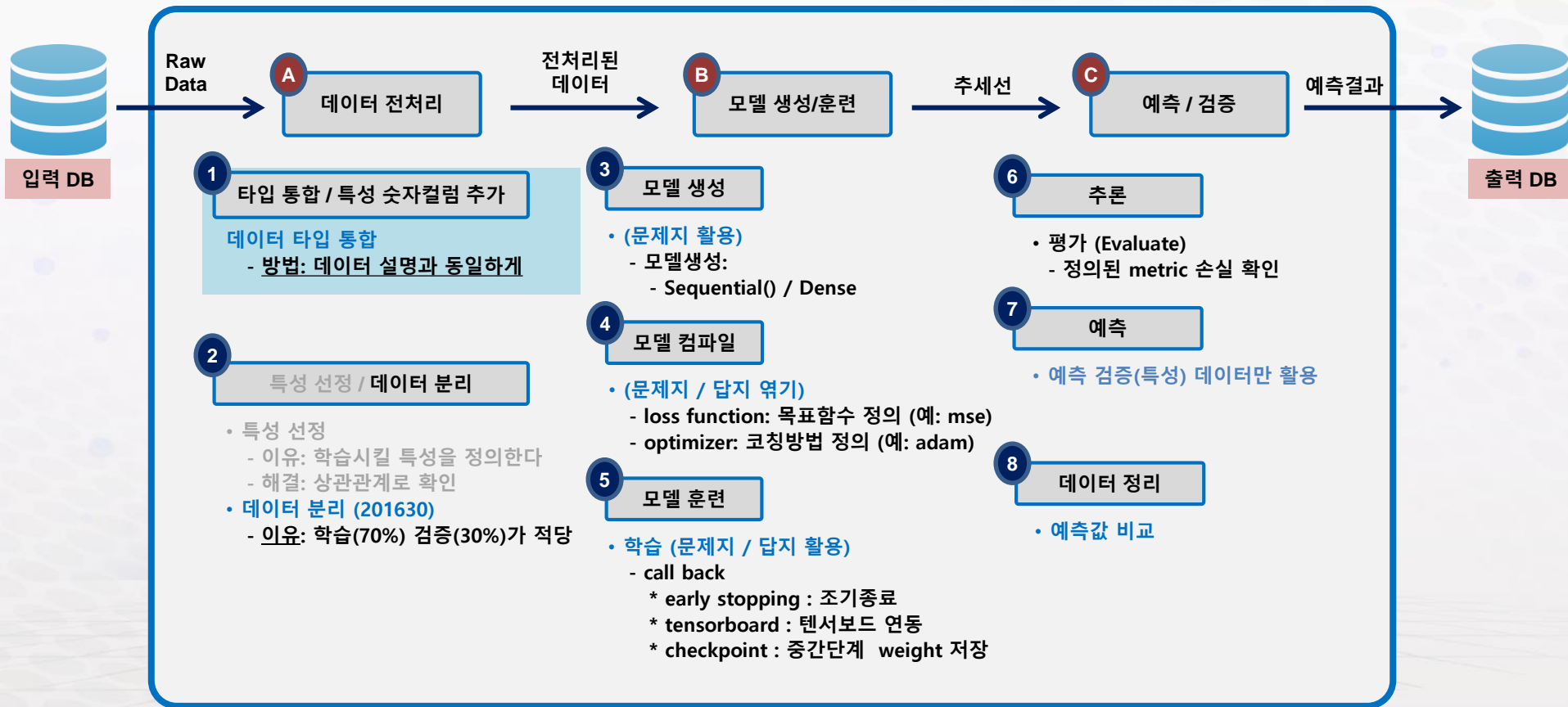
기본 응용 따라하기

### 3. 합성곱 신경망 실습

#### Image classification

순번	구분	내용	비고
1	문제	패션 MNIST Fashion	0~9 인식 (10개 범주, 티셔츠, 드레스등)
2	데이터	훈련데이터:6만 테스트데이터:1만	
3	해결방법	분류문제	

### 3. 합성곱 신경망 실습



### 3. 합성곱 신경망 실습

#### 라이브러리 선언

```
import os
import cv2
# 케라스 모델 생성 라이브러리
import keras
from keras import models
# 레이어 생성 라이브러리 (Dense: 입출력 연결)
from keras import layers

# numpy 라이브러리
import numpy as np
from numpy import array
# 케라스 카테고리 라이브러리
from keras.utils import to_categorical
# from sklearn.preprocessing import LabelEncoder
# from sklearn.preprocessing import OneHotEncoder
# 시각화 라이브러리
import matplotlib.pyplot as plt
%matplotlib inline
```

## 2. Keras 실습 (Image Classification)

### 1. 데이터 불러오기

# 미리 섞여진 fashion-mnist의 학습 데이터와 테스트 데이터 로드

```
(x_train, y_train), (x_test, y_test) = keras.datasets.fashion_mnist.load_data()
```

# 학습 셋과 테스트 셋의 데이터 개수

```
print(x_train.shape[0], 'train set')
```

```
print(x_test.shape[0], 'test set')
```

# 레이블 정의

```
fashion_mnist_labels = ["T-shirt/top", # 인덱스 0
```

```
    "Trouser",          # 인덱스 1
```

```
    "Pullover",         # 인덱스 2
```

```
    "Dress",            # 인덱스 3
```

```
    "Coat",             # 인덱스 4
```

```
    "Sandal",           # 인덱스 5
```

```
    "Shirt",            # 인덱스 6
```

```
    "Sneaker",          # 인덱스 7
```

```
    "Bag",              # 인덱스 8
```

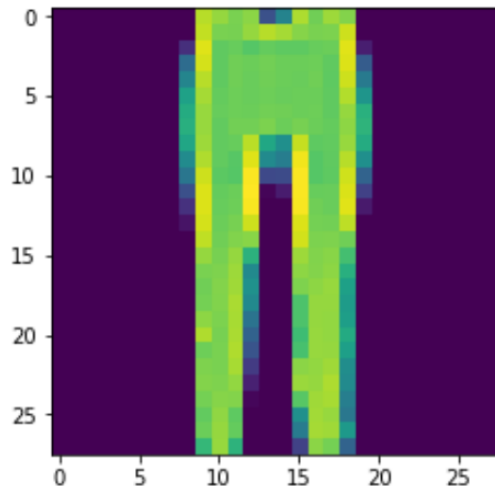
```
    "Ankle boot"]      # 인덱스 9
```

```
img_index = 106
```

```
label_index = y_train[img_index]
```

```
plt.imshow(x_train[img_index])
```

<matplotlib.image.AxesImage at 0x1b905e3cb38>





## 2. Keras 실습 (Image Classification)

### 2. 데이터 분리 및 정제

```
x_train = x_train.astype('float32') / 255  
x_test = x_test.astype('float32') / 255
```

# 입력 이미지의 크기를 (28, 28) 에서 (28, 28, 1) 로 배열 차원을 변경(reshape)

w, h = 28, 28

```
x_train = x_train.reshape(x_train.shape[0], w, h, 1)
```

```
x_test = x_test.reshape(x_test.shape[0], w, h, 1)
```

합성곱 신경망 (Conv)를 위해  
channel shape 추가 (depth)

# 레이블에 원-핫 인코딩 적용

# 원-핫 벡터는 단 하나의 차원에서만 1이고, 나머지 차원에서는 0인 벡터입니다.

```
y_train = keras.utils.to_categorical(y_train, 10)
```

```
y_test = keras.utils.to_categorical(y_test, 10)
```

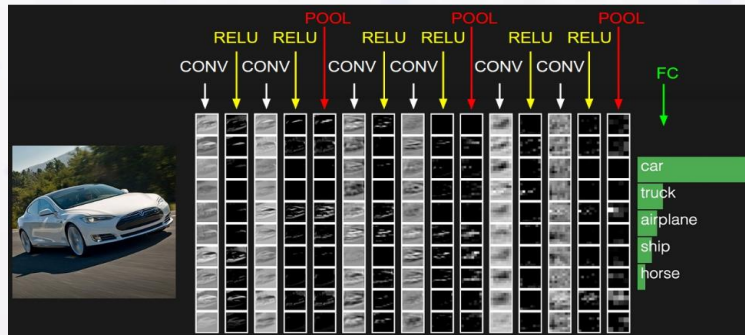
```
x_train shape: (55000, 28, 28, 1) y_train shape: (55000, 10)  
55000 train set  
5000 validation set  
10000 test set
```

## 2. Keras 실습 (Image Classification)

### 3. 모델 생성

```
from keras import Sequential
from keras.layers import MaxPooling2D, Conv2D, Dropout, Flatten, Dense
modelDim = x_train[0].shape
model = keras.Sequential()
# 신경망의 첫 번째 레이어에서 입력 데이터 크기를 정의해야 합니다.
model.add(Conv2D(filters=32, kernel_size=2,
                  padding='same',
                  activation='relu',
                  input_shape=modelDim))
model.add(MaxPooling2D(pool_size=2))
model.add(Dropout(0.3))
model.add(Conv2D(filters=32, kernel_size=2, padding='same',
                  activation='relu'))
model.add(MaxPooling2D(pool_size=2))
model.add(Dropout(0.3))

model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))
```



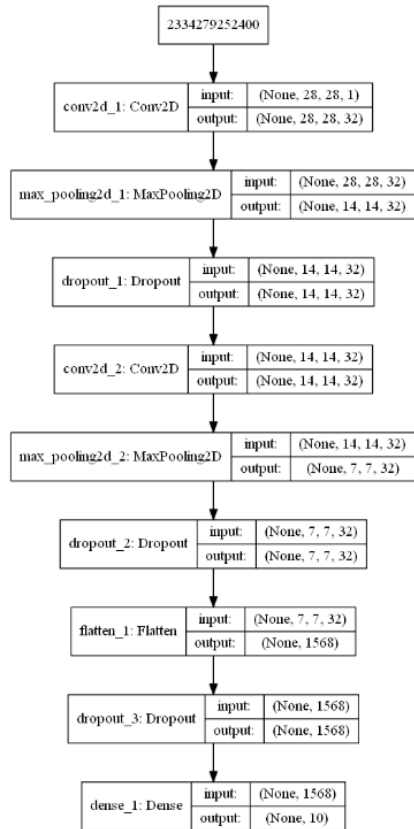
```
import warnings
warnings.filterwarnings(action="ignore")
```

## 2. Keras 실습 (Image Classification)

### 3. 모델 생성

```
from keras.utils import plot_model
from IPython.display import Image
plot_model(model, to_file='model_plot.png', show_shapes=True,
show_layer_names=True)
Image("model_plot.png")
```

실제 모델 시각화는  
Graphviz 설정 외에  
Site-package 내 graphviz 내  
Vis\_util 파일을 import pydotplus as pydot으로 변경 필요  
\* Conda install graphviz 필요



## 2. Keras 실습 (Image Classification)

### 4. 모델 컴파일

```
# Sequential 방식 케라스모델  
# 손실함수(LOSS): 훈련동안 최소화될 값 지표 (mse, categorical_crossentropy)  
# 손실함수를 기반으로 Neural Net 업데이터 결정 (mse, mae, accuracy)  
model.compile(optimizer='adam',  
              loss='categorical_crossentropy',  
              metrics=['accuracy'])
```

### 5. 케라스모델 훈련방법 설정

```
1 # Sequential 방식 케라스모델  
2 # 손실함수(LOSS): 훈련동안 최소화될 값 지표 (mse, categorical_crossentropy)  
3 # 손실함수를 기반으로 Neural Net 업데이터 결정 (mse, mae, accuracy)  
4 model.compile(optimizer='adam',  
5               loss='categorical_crossentropy',  
6               metrics=['accuracy'])
```

## 2. Keras 실습 (Image Classification)

### 5. 모델 훈련

```
from datetime import datetime
```

```
now = datetime.now()
```

```
currdate = now.strftime("%Y_%m_%d_%H_%M")
```

```
callbacks = [
```

```
    keras.callbacks.TensorBoard(
```

```
        log_dir = save_dir,
```

```
        write_graph=True,
```

```
        write_images=True),
```

```
    keras.callbacks.EarlyStopping(
```

```
        monitor = 'val_acc', patience=10,
```

```
    )
```

```
]
```

```
model.fit(x_train,  
          y_train,  
          batch_size=32,  
          epochs=5,  
          validation_split=0.2,  
          callbacks=callbacks  
          )
```

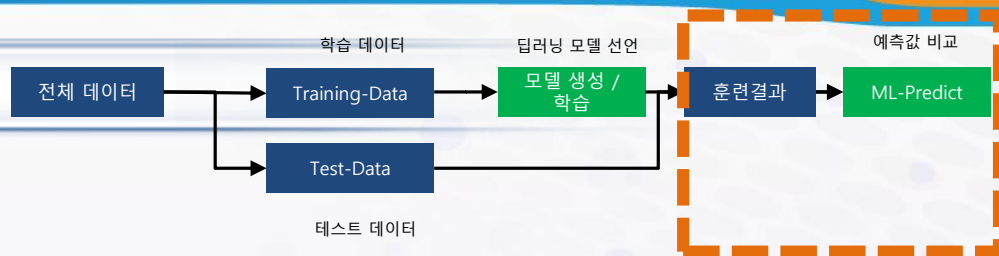
## 2. Keras 실습 (Image Classification)

### 6. 모델 추론

# verbose: 정보표시 레벨 (0,1)

```
test_loss, test_acc = model.evaluate(x_test,  
                                     y_test,  
                                     verbose=1)
```

```
print(test_loss, test_acc)
```



### 모델 추론

```
1 # verbose: 정보표시 레벨 (0,1)
2 test_loss, test_acc = model.evaluate(x_test,
3                                     y_test,
4                                     verbose=1)
5 print(test_loss, test_acc)
6
```

```
10000/10000 [=====] - 2s 186us/step
0.3653928399562836 0.8735
```

## 2. Keras 실습 (Image Classification)

### 7. 모델 예측

# 테스트데이터 1개 추출

```
testimg = x_test[190]
```

# 테스트데이터 1개 답지 확인 (바지)

```
y_test[190]
```

# 테스트데이터 이미지로 변환

```
testimg2 = testimg * 255
```

```
testimg3 = testimg2.reshape(28,28)
```

```
plt.imshow(testimg3, cmap="gray")
```

정규화한 내용 적용 후 jpg저장  
cv2.imwrite("./test.jpg", testimg2)

# 데이터 정제작업 수행

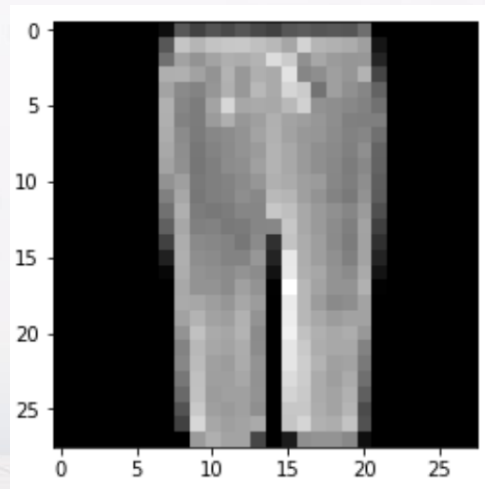
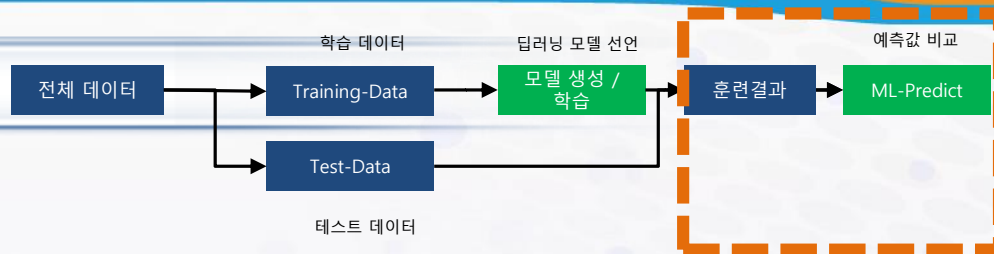
```
testimg4 = testimg3/255
```

```
testimg5 = testimg4.reshape(1,28,28,1)
```

# 예측 수행

```
test_predictions = model.predict(testimg5)
```

```
test_predictions
```



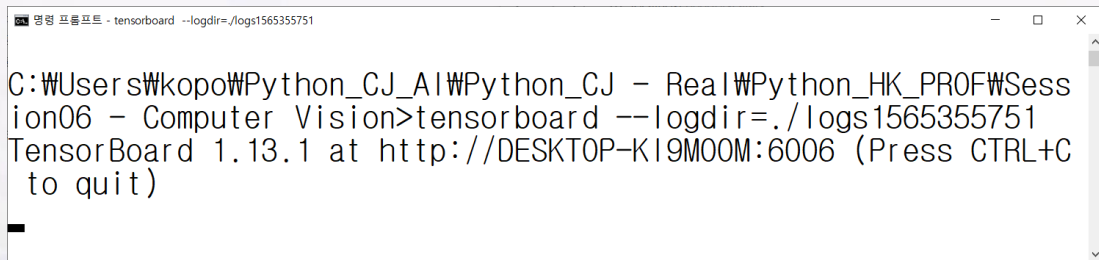


## 2. Keras 실습 (Image Classification)

### 8. 훈련내용 확인하기

1. 명령프롬프트(cmd) 켜고 python 작업 위치로 이동

2. 명령어 실행: `tensorboard --logdir=./logs...`



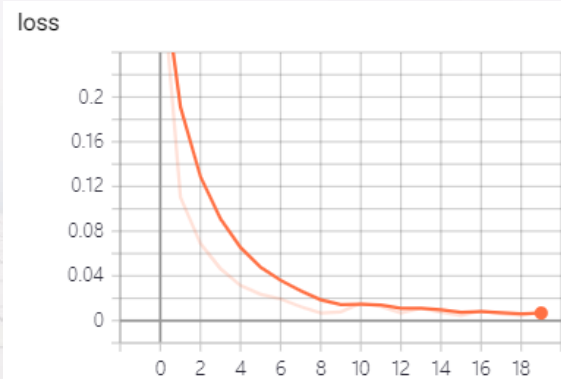
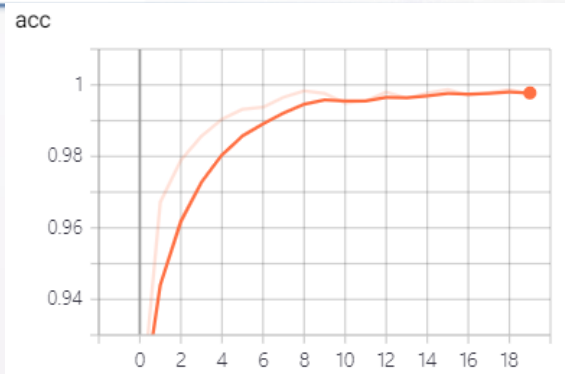
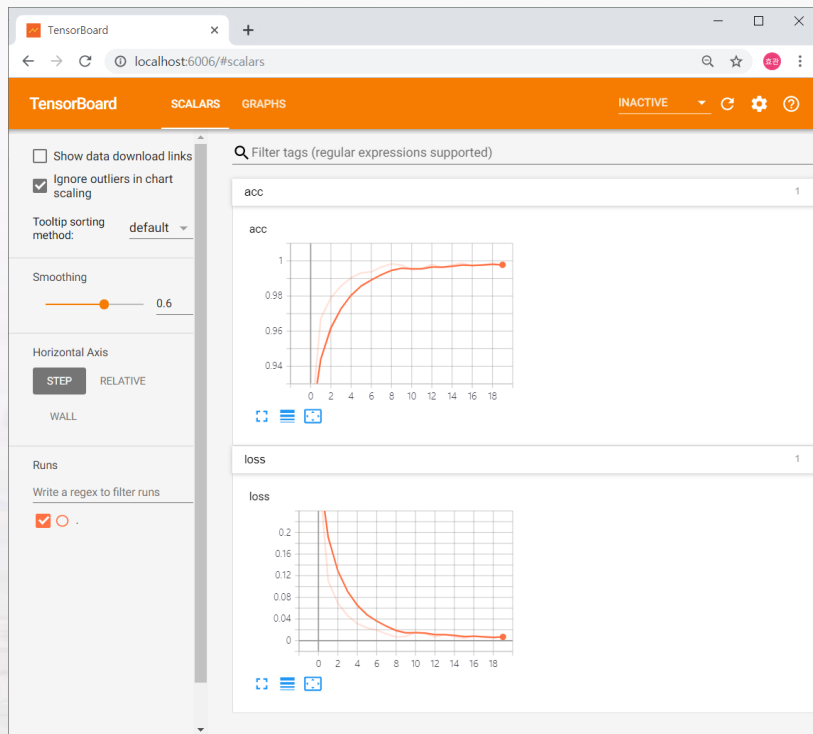
```
명령 프롬프트 - tensorboard --logdir=./logs1565355751
C:\Users\Wkopo\Python_CJ_AI\Python_CJ - Real\Python_HK_PRO\FSession06 - Computer Vision>tensorboard --logdir=./logs1565355751
TensorBoard 1.13.1 at http://DESKTOP-KI9M00M:6006 (Press CTRL+C to quit)
```

3. 웹 페이지 오픈 후 `localhost:6006` 실행

<http://localhost:6006>

## 2. Keras 실습 (Image Classification)

### 8. 훈련내용 확인하기

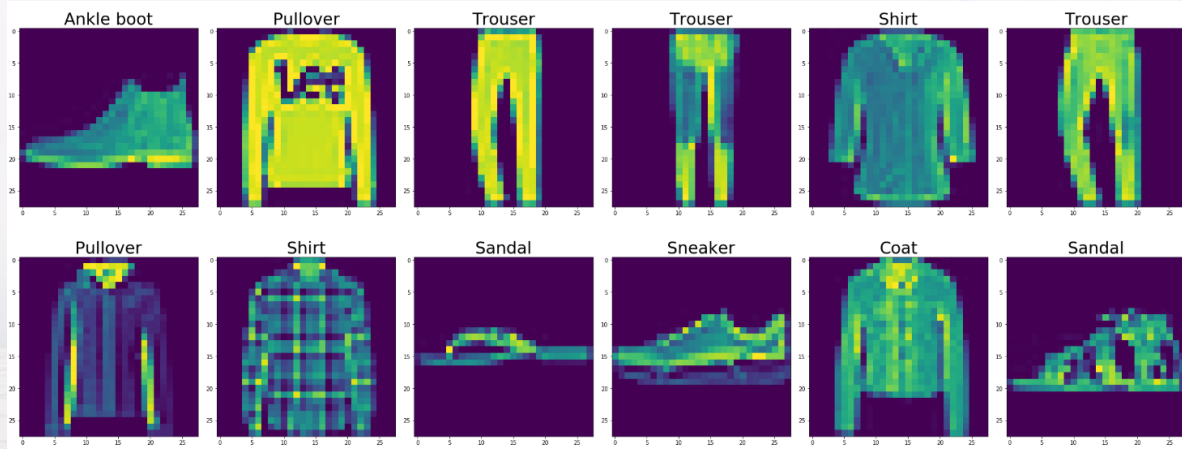


### 3. 합성곱 신경망 실습

이해한 내용을 활용하여  
결과를 도출하고 실제 plt를 응용하여 결과비교를 해보세요.

```
1 def plot_data(X, Y, num_figures):
2     plt.figure(figsize=(30, 20))
3     for i in range(num_figures):
4         plt.subplot(2, num_figures, i+1)
5         plt.imshow(X[i])
6         contents = fashion_mnist_labels[np.argmax(Y[i])]
7         plt.title(contents, fontsize=30)
8         plt.tight_layout()
```

```
1 for i in range(0, 30, 6):
2     plot_data(inverse_test[i:], test_predictions[i:], 6)
```



## 2. Keras 실습 (Image Classification)

## 재학습 (참조)

## 모델 선언 구조 저장

## # 모델 저장

```
model_json = model.to_json()
```

```
with open("model.json", "w") as json_file:
    json_file.write(model_json)
```

```
model.save_weights(" keras_images.h5 ")
```

## 역전파를 통한 업데이트된 가중치 저장

## # 모델 불러오기

```
from keras.models import model_from_json
json_file = open("model.json", "r")
loaded_model_json = json_file.read()
json_file.close()
```

```
loaded_model = model_from_json(saved_model_json)
loaded_model.load_weights("keras_images.h5")
```

## # 모델 재 컴파일

```
optimizer = keras.optimizers.Adam()
loaded_model.compile(loss='accuracy',
                    optimizer=optimizer,
                    metrics=['accuracy'])
```

## loaded\_model.summary()

## # 모델 재 학습

```
from keras.callbacks import EarlyStopping
early_stopping_monitor = EarlyStopping(patience=50)
EPOCHS = 100
```

## #모델 훈련 (훈련/검증을 80%, 20%로 나눔)

[illegible]

### 3. 합성곱 신경망 실습

Step1. 데이터 변경 후 모델 실습

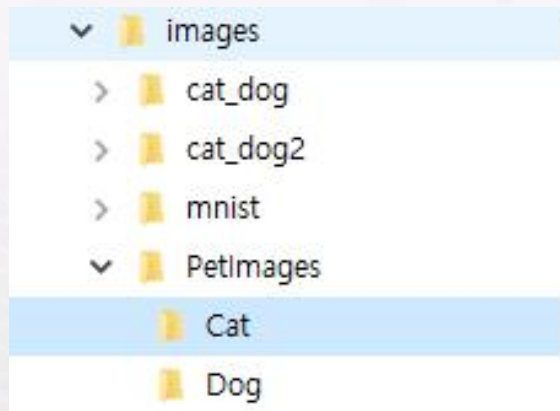
심화 응용 따라하기  
하단의 cnn 따라하기 내용을 실습한 후  
결과를 도출 후 [haiteam@kopo.ac.kr](mailto:haiteam@kopo.ac.kr) 메일로 전송  
( categorical\_crossentropy 활용)

### 3. 합성곱 신경망 실습

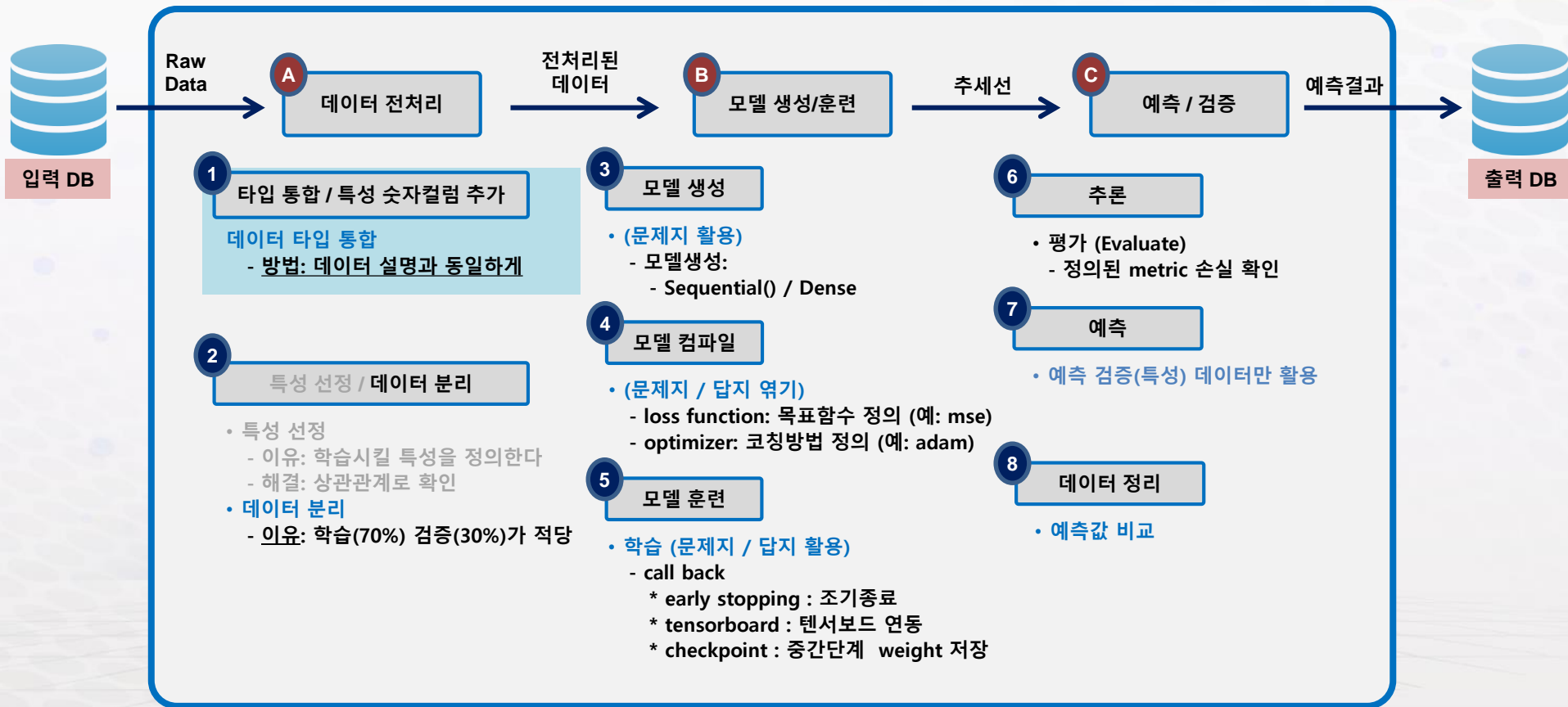
#### 데이터 준비

데이터를 다운받은 후 아래 폴더에 저장하세요.

<https://www.microsoft.com/en-us/download/details.aspx?id=54765>



### 3. 합성곱 신경망 실습





## 2. Keras 실습 (Image Classification)

### 2. 데이터 불러오기

[훈련/테스트 세트] : 이미지/답지 별도 저장

폴더 리스트 반복

cat, dog

이미지 리스트 반복

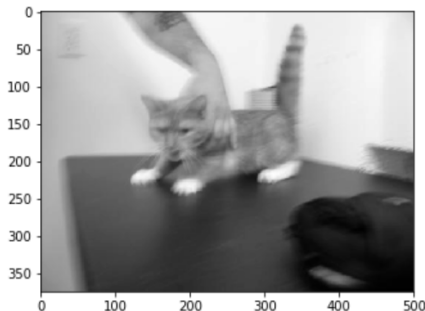
img\_4.jpg, ..

데이터 저장

- 컬러: 그레이
- 사이즈: 50 \* 50
- 이미지 저장
- 폴더 인덱스-> 답지

```
for category in categories:  
    path = os.path.join(basedir, category)  
    for img in os.listdir(path):  
        img_array = cv2.imread(os.path.join(path, img), cv2.IMREAD_GRAYSCALE)  
        plt.imshow(img_array, cmap="gray")  
        plt.show()  
        break  
    break
```

```
1 for category in categories:  
2     path = os.path.join(basedir, category)  
3     for img in os.listdir(path):  
4         img_array = cv2.imread(os.path.join(path, img), cv2.IMREAD_GRAYSCALE)  
5         plt.imshow(img_array, cmap="gray")  
6         plt.show()  
7         break  
8     break
```



## 2. Keras 실습 (Image Classification)

### 2. 데이터 불러오기

```
basedir = "../images/PetImages/"  
categories = os.listdir(basedir)
```

```
train_images=[]  
train_labels=[]
```

```
for category in categories:
```

```
    path = os.path.join(basedir, category)
```

```
    class_num = categories.index(category)
```

```
    for img in os.listdir(path):
```

```
        try:
```

```
            img_array = cv2.imread(os.path.join(path, img), cv2.IMREAD_GRAYSCALE)
```

```
            new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
```

```
            # 2차원 배열로 앞에는 이미지, 2번째는 답지 대입
```

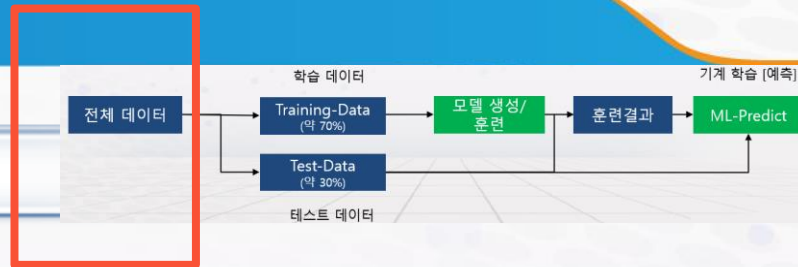
```
            train_images.append(new_array)
```

```
            train_labels.append(category)
```

```
except Exception as e:
```

```
    pass
```

Exception 처리



### 3. 합성곱 신경망 실습

Step2. 데이터 변경 및 loss\_function 변경 실습

심화 응용 따라하기  
하단의 cnn 따라하기 내용을 실습한 후  
결과를 도출 후 [haiteam@kopo.ac.kr](mailto:haiteam@kopo.ac.kr) 메일로 전송  
( `binary_crossentropy` 활용)

## 4. 핵심정리 및 Q&A

### 기억합시다

1

합성곱 신경망 등장배경을 이해한다 (주요 피처만 학습하여 속도개선/정확도 개선)

2

합성곱 신경망에 대해서 이해한다. (필터 및 레이어의 이해)

3

합성곱 신경망 실습을 통해 작동원리를 이해한다.

The input to the policy network is a  $19 \times 19 \times 48$  image stack consisting of 48 feature planes. The first hidden layer zero pads the input into a  $23 \times 23$  image, then convolves  $k$  filters of kernel size  $5 \times 5$  with stride 1 with the input image and applies a rectifier nonlinearity. Each of the subsequent hidden layers 2 to 12 zero pads the respective previous hidden layer into a  $21 \times 21$  image, then convolves  $k$  filters of kernel size  $3 \times 3$  with stride 1, again followed by a rectifier nonlinearity. The final layer convolves 1 filter of kernel size  $1 \times 1$  with stride 1, with a different bias for each position, and applies a softmax function. The match version of AlphaGo used  $k = 192$  filters; [Fig. 2b](#) and [Extended Data Table 3](#) additionally show the results of training with  $k = 128, 256$  and  $384$  filters.

감사합니다.

