참조. 딥러닝 작동 원리

https://github.com/tensorflow/tensorflow/releases/tag/v2.0.0

TensorFlow 2.0.0

goldiegadde released this Sep 30, 2019

Release 2.0.0

Major Features and Improvements

TensorFlow 2.0 focuses on **simplicity** and **ease of use**, featuring updates like:

- Easy model building with Keras and eager execution.
- Robust model deployment in production on any platform.
- Powerful experimentation for research.
- API simplification by reducing duplication and removing deprecated endpoints.

For details on best practices with 2.0, see the Effective 2.0 guide

For information on upgrading your existing TensorFlow 1.x models, please refer to our Upgrade and Migration guides. We have also released a collection of tutorials and getting started guides.

Highlights

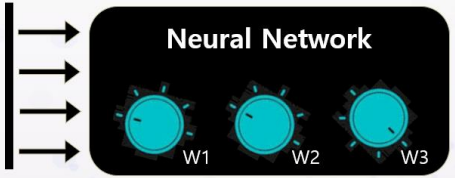
- TF 2.0 delivers Keras as the central high level API used to build and train models. Keras provides several model-building APIs such as Sequential, Functional, and Subclassing along with eager execution, for immediate iteration and intuitive debugging, and `tf.data`, for building scalable input pipelines. Checkout guide for additional details.

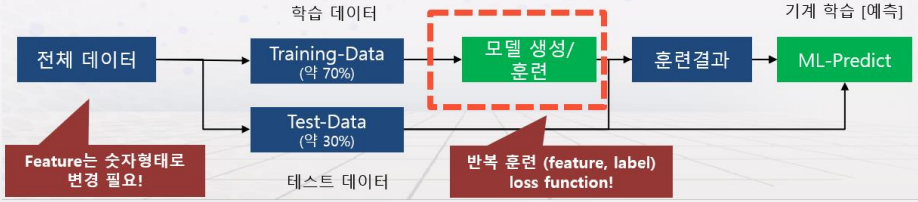
참조. 딥러닝 작동 원리

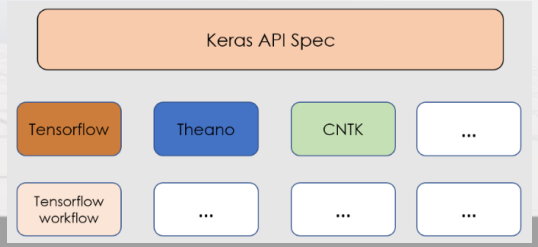
케라스 특징

TensorFlow 2.0 (2019년 9월30일 릴리즈)
* 텐서보드연동 문제로 인해 아래 버전 설치
pip install tensorflow==1.14
pip install keras

1. 2015년 3월 출시된 TensorFlow등 딥러닝 엔진 API
2. 구글, MS 아마존 등 우량 기업들이 지원
3. 고수준 신경망 API (백엔드: Tesnsor flow 2.0 정식)
4. CPU/GPU 실행통한 속도 가속화기능 탑재
5. 합성곱, 순환 신경망 지원
6. MIT 라이센스로 상업적인 프로젝트도 자유롭게 사용가능

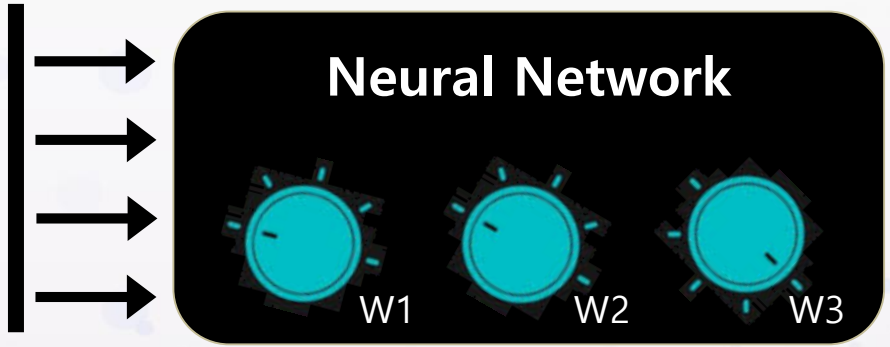






참조. 딥러닝 작동 원리

케라스 모델 프로세스



학습 데이터

기계 학습 [예측]

전체 데이터

Training-Data
(약 70%)

Test-Data
(약 30%)

모델 생성/
훈련

훈련결과

ML-Predict

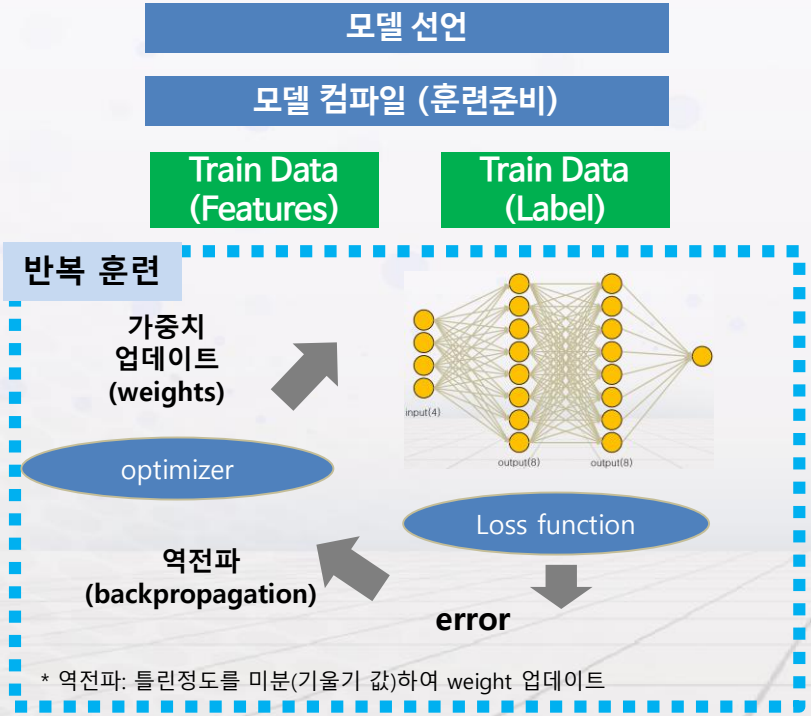
Feature는 숫자형태로
변경 필요!

테스트 데이터

반복 훈련 (feature, label)
loss function!

참조. 딥러닝 작동 원리

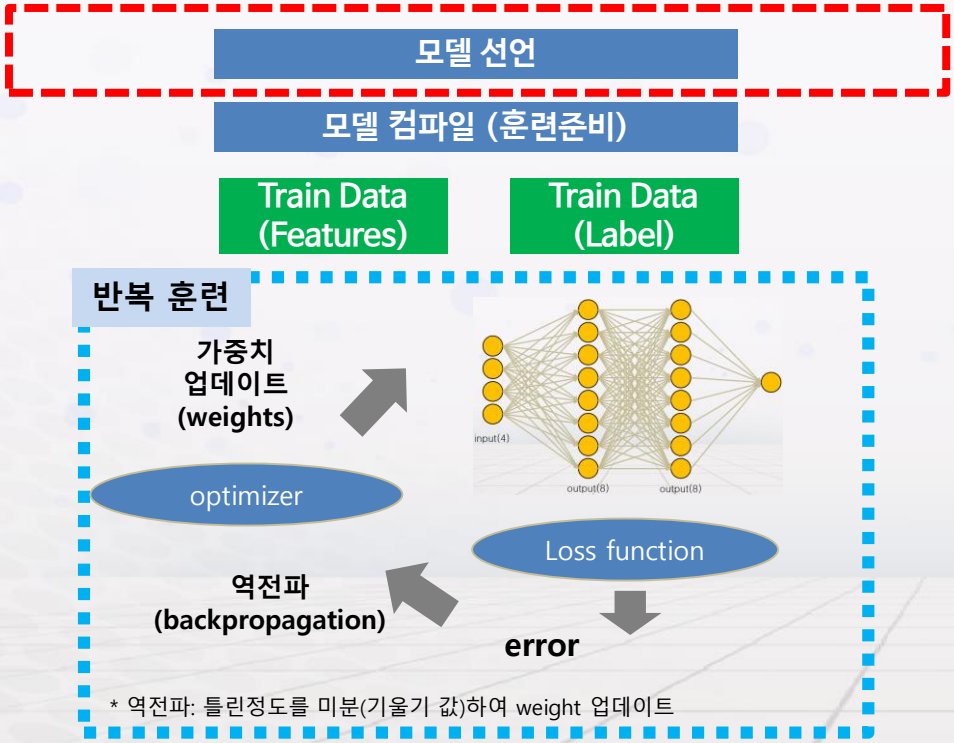
케라스 모델 프로세스



주요용어	내용
모델선언	Kears에서 Layer 적용 모델 선언
컴파일	손실함수 및 최적화방법 설정 - 손실함수: 훈련동안 최적화될 지표 - 최적화방법: 답이 틀렸을 경우 코칭방법 설정 (단 훈련셋에 검증구간을 별도 분리)
훈련	훈련데이터의 Feature와 Label을 활용하여 손실함수 지표를 최적화하기위하여 역전파(손실함수 결과 개선을 위해 가중치 수정) 반복 수행

참조. 딥러닝 작동 원리

케라스 모델 프로세스

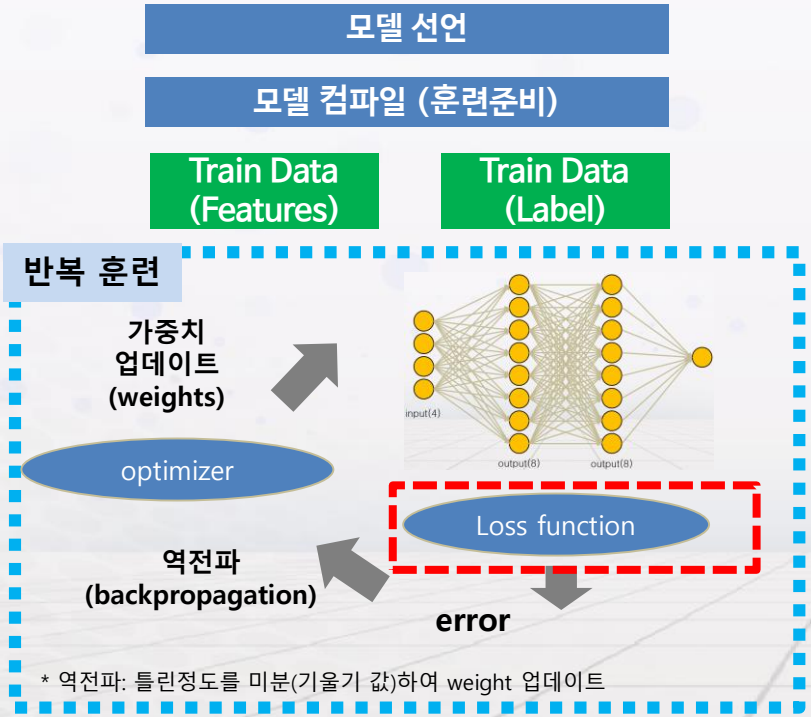


Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	98 MB	0.749	0.921	25,636,712	-
ResNet101	171 MB	0.764	0.928	44,707,176	-
ResNet152	232 MB	0.766	0.931	60,419,944	-
ResNet50V2	98 MB	0.760	0.930	25,613,800	-
ResNet101V2	171 MB	0.772	0.938	44,675,560	-
ResNet152V2	232 MB	0.780	0.942	60,380,648	-
ResNeXt50	96 MB	0.777	0.938	25,097,128	-
ResNeXt101	170 MB	0.787	0.943	44,315,560	-
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88
DenseNet121	33 MB	0.750	0.923	8,062,504	121
DenseNet169	57 MB	0.762	0.932	14,307,880	169
DenseNet201	80 MB	0.773	0.936	20,242,984	201
NASNetMobile	23 MB	0.744	0.919	5,326,716	-
NASNetLarge	343 MB	0.825	0.960	88,949,818	-

https://keras.io/applications/

참조. 딥러닝 작동 원리

케라스 모델 프로세스

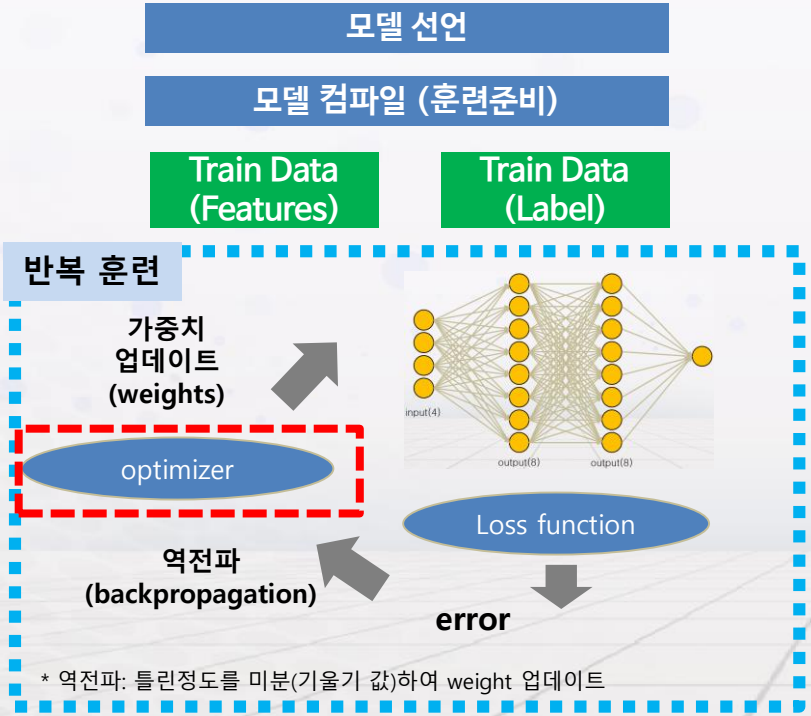


Loss Function (손실함수)	내용
Mean_squared_error	연속 숫자 예측
Mean_absolute_error	
Mean_absolute_percentage_error	
Mean_squared_logarithmic_error	
Squared_hinge	
Hinge	
Categorical_hinge	
Logcosh	
Categorical_crossentropy	멀티 카테고리 예측
Sparse_categorical_crossentropy	
Binary_corssentropy	2개 카테고리 예측

https://keras.io/losses/

참조. 딥러닝 작동 원리

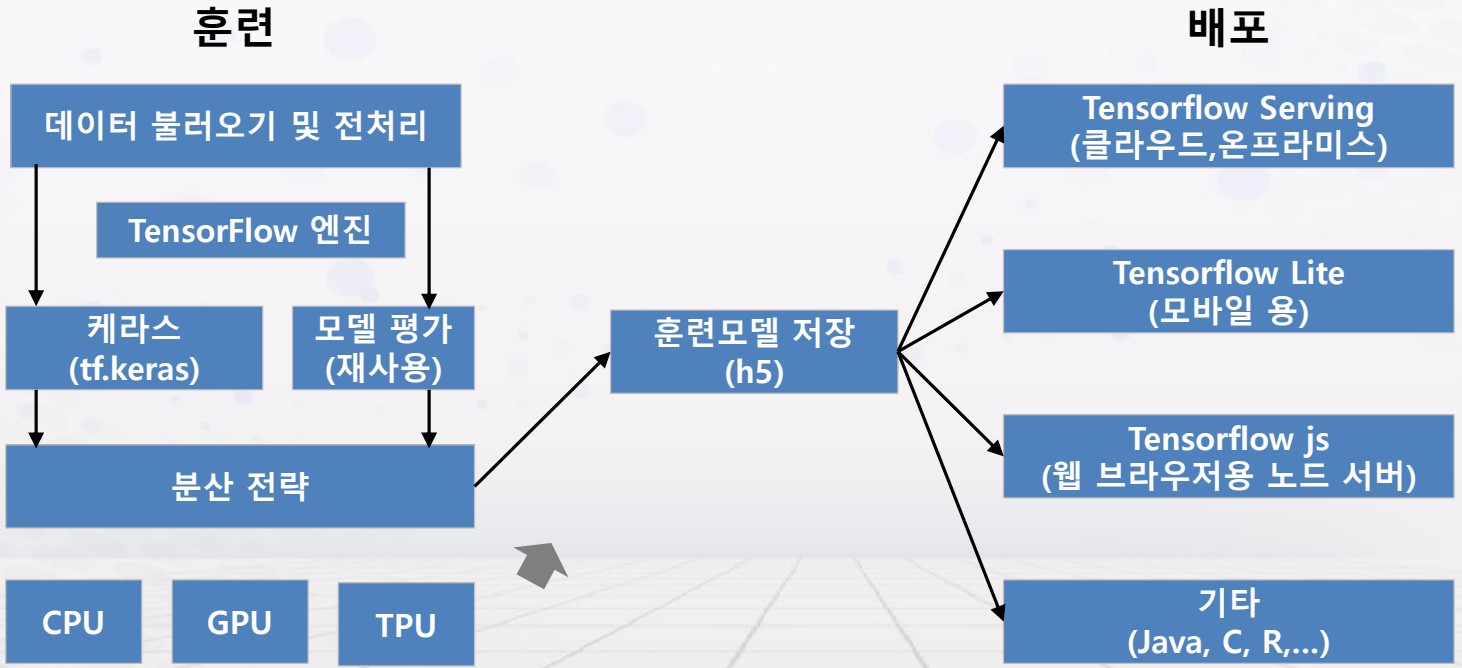
케라스 모델 프로세스



Optimizer (최적화)	비고
GD (Gradient Descent)	정확하지만 느리다
SGD (Stochastic Gradient Descent)	빠르지만 찾는 방향 뒤죽박죽
RMSprop	스탭줄일 시 이전 맥락 확인
Adagrad	안가본곳은 빠르게 가본곳 천천히
Adadelta	스탭너무 작아져서 정지 안되게
Adam	RMSprop + Momentom 방향/스탭사이즈 적절하게

참조. 딥러닝 작동 원리

전체 프로세스



* TPU: Tensor Processing Unit

참조. 딥러닝 작동 원리

주요 수식

머신러닝은 특성을 선별하여 훈련 시켰다면
딥러닝은 **가능한 특성은 전부 포함** 시킨다.

$$Y = w * X + b$$

판매량

휴일정보

할인정보

참조. 딥러닝 작동 원리

주요 수식

실측 값 $Y = w * X + b$

예측 값 $\hat{Y} = w * \hat{X} + b$

참조. 딥러닝 작동 원리

주요 수식

실측 값

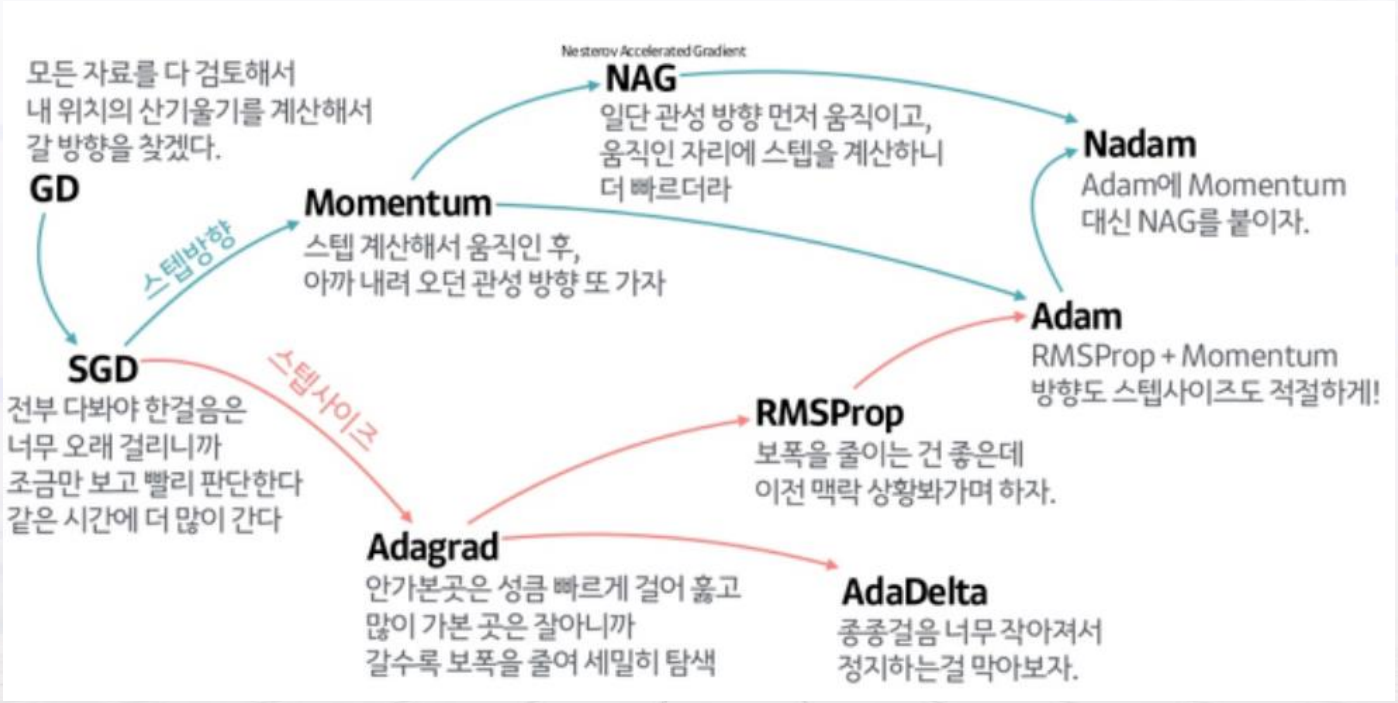
$$60 = w * 30\% + b$$

$$80 = w * 50\% + b$$

예측 값

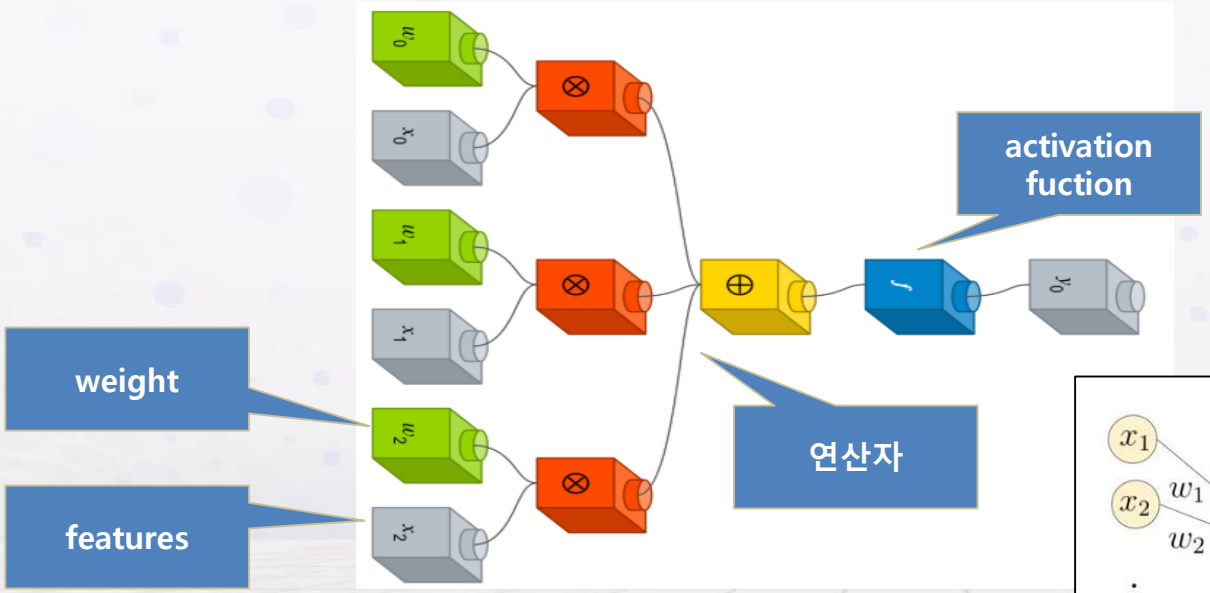
$$\hat{Y} = w * 40\% + b$$

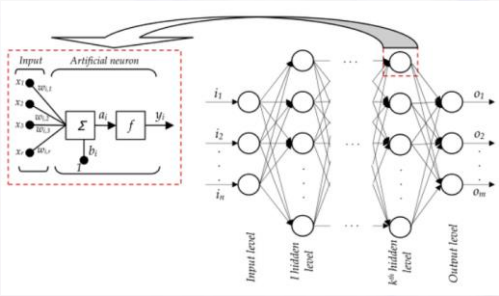
참조. 딥러닝 작동 원리

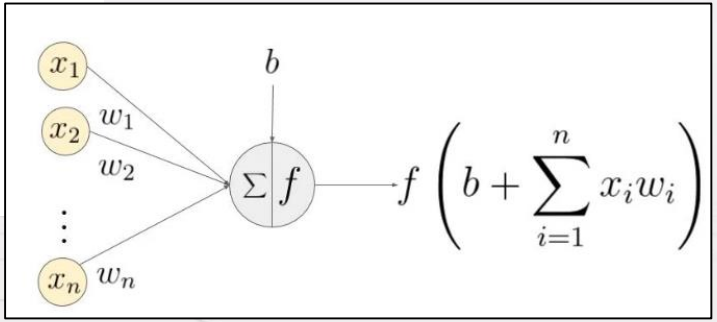

모든 자료를 다 검토해서
내 위치의 산기울기를 계산해서
갈 방향을 찾겠다.
GD
스텝방향
SGD
전부 다봐야 한걸음은
너무 오래 걸리니까
조금만 보고 빨리 판단한다
같은 시간에 더 많이 간다
Momentum
스텝 계산해서 움직인 후,
아까 내려 오던 관성 방향 또 가자
Nesterov Accelerated Gradient
NAG
일단 관성 방향 먼저 움직이고,
움직인 자리에 스텝을 계산하니
더 빠르더라
Nadam
Adam에 Momentum
대신 NAG를 붙이자.
스텝사이즈
Adagrad
안가본곳은 성큼 빠르게 걸어 훑고
많이 가본 곳은 잘아니까
갈수록 보폭을 줄여 세밀히 탐색
RMSProp
보폭을 줄이는 건 좋은데
이전 맥락 상황봐가며 하자.
Adam
RMSProp + Momentum
방향도 스텝사이즈도 적절하게!
AdaDelta
종종걸음 너무 작아져서
정지하는걸 막아보자.

참조. 딥러닝 작동 원리

Neuron 상세 내용







https://tykimos.github.io/2017/01/27/MLP_Layer_Talk/

참조. 딥러닝 작동 원리

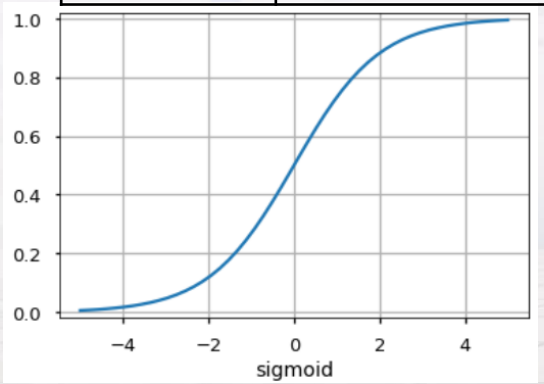
activation 함수

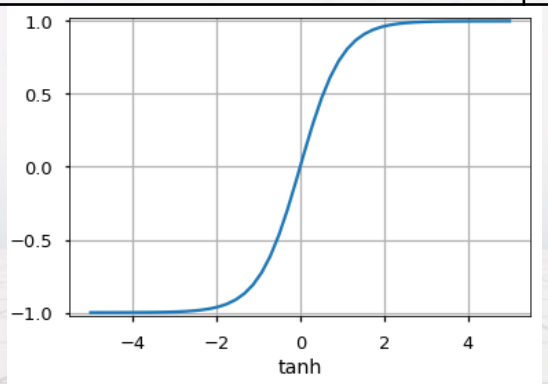
Activation functions on CIFAR-10

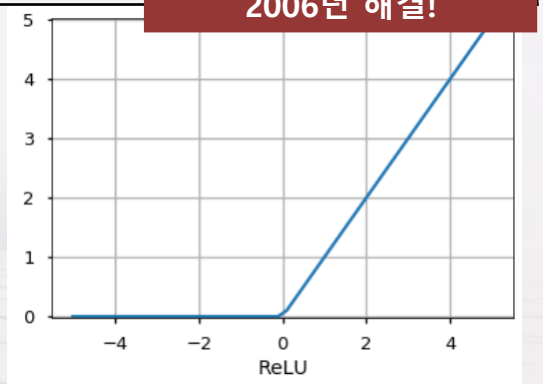
maxout	ReLU	VLReLU	tanh	Sigmoid
93.94	**92.11**	92.97	89.28	n/c
93.78	91.74	92.40	89.48	n/c
–	91.93	**93.09**	–	n/c
91.75	90.63	92.27	**89.82**	n/c
n/c†	90.91	92.43	89.54	n/c

함수종류	설명	비고
linear	디폴트, 입력뉴런과 가중치로 계산된 결과값 그대로 출력	
sigmoid	시그모이드함수, 이진분류 문제에서 출력층(output)에 주로 사용	0~1 사이 실수 값 출력
relu	음수에 대해 0으로 처리하는 함수, 기존의 sigmoid를 개선함	max(0, x)
softmax	소프트맥스, 다중 클래스분류문제 출력층(output)에 주로 사용	이미지 분류 시 유용
tanh	sigmoid 함수를 재활용하기 위한 함수. sigmoid의 범위를 -1에서 1	

역전파 시 0~1 사이로 전파하여 손실 부분을 2006년 해결!



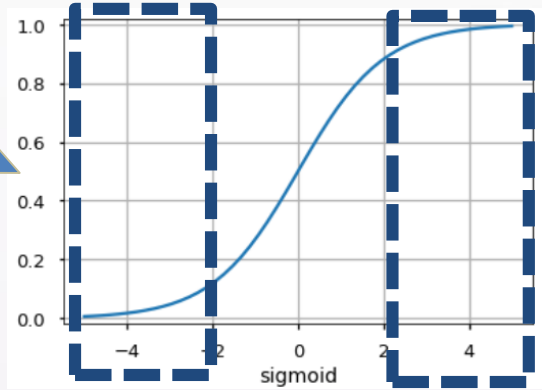




참조. 딥러닝 작동 원리

Sigmoid 활용 시
미분 값 0에 가까워져
역전파로 전달 시
Weight 업데이트 거의 안됨

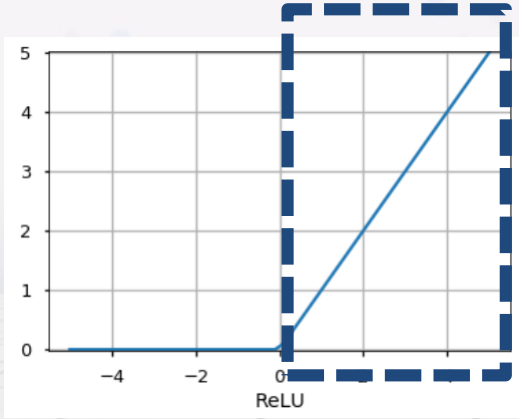
-> Vanishing Gradient
로 인해 학습이 안되는 현상을
Under fitting이라고 함



0~1 사이 값

Rectified Linear Unit

미분 값이 전부 0 이여서
역전파 시
Weight 업데이트 안되는
경우 없음!



F(x) = max(0, x)

https://en.wikipedia.org/wiki/Rectifier_(neural_networks)

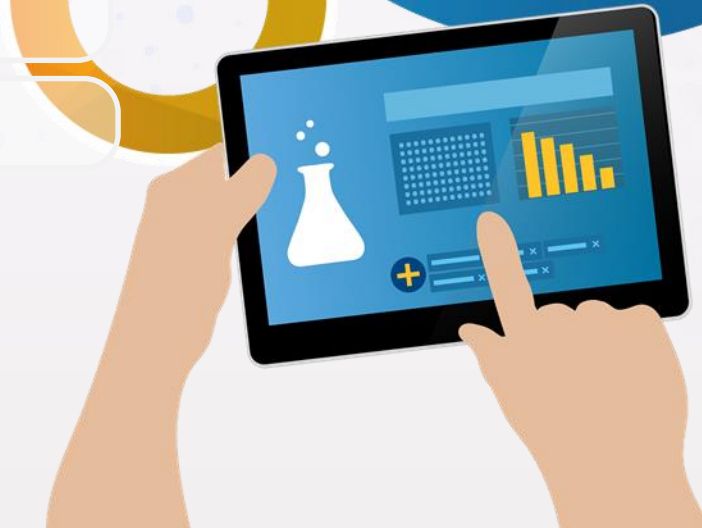
이미지 분류를 통한 딥러닝 이해하기 (Image Classification)

김 효 관

교육목표: Neural Network를 이해하고 keras 실습

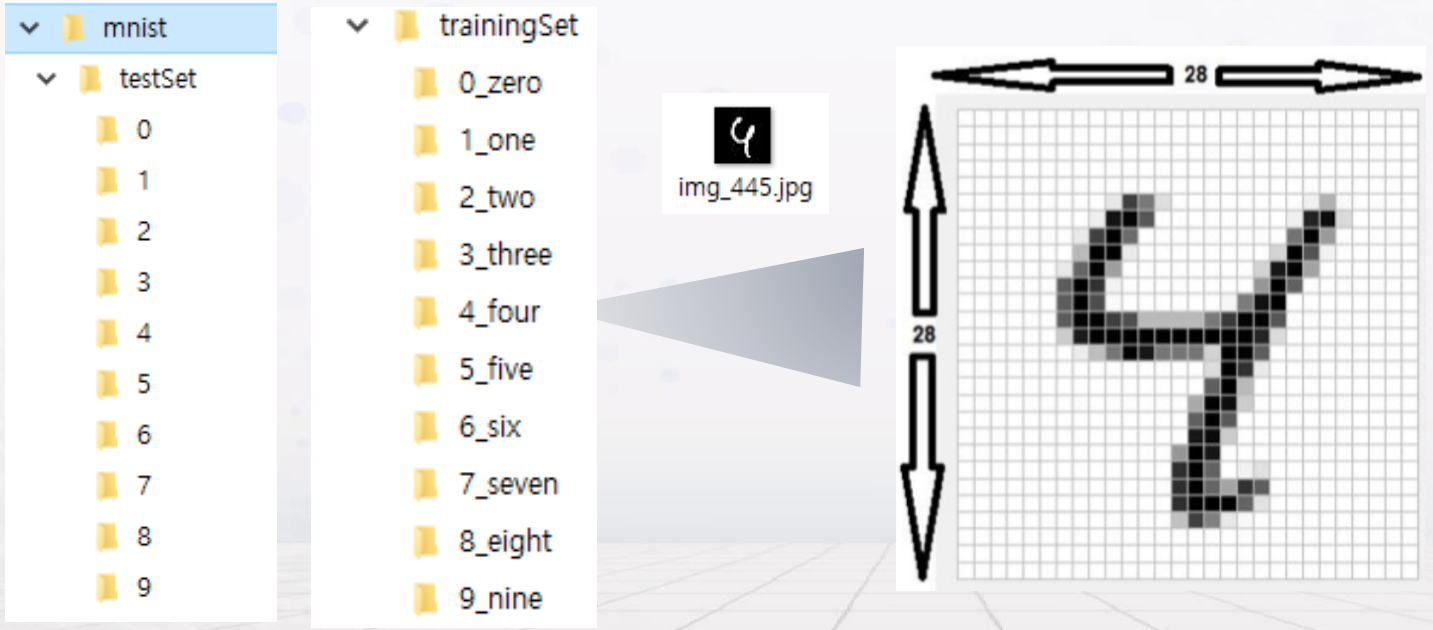
CONTENTS

1. Image Classfication 문제
2. Keras 실습 (Image Classification)
3. 핵심정리 및 Q&A



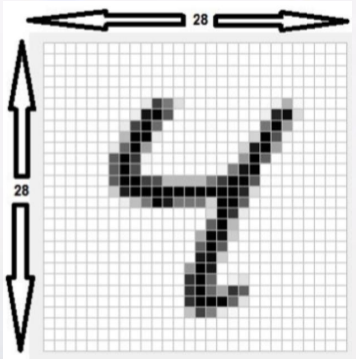
1. Image Classificatino 문제

데이터 설명



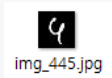
1. Image Classificatino 문제

Image classification



1. Image Classificatino 문제

Image classification

순번	구분	내용	비고
1	문제	손글씨 숫자인식  img_445.jpg ➡ 4	0~9 인식 (10개 범주, 클래스)
2	데이터	훈련데이터:6만 테스트데이터:1만	1980년대 미국 국립표준기술연구소(NIST) 수집
3	해결방법	분류문제	

2. Keras 실습 (Image Classification)

1. 라이브러리 선언

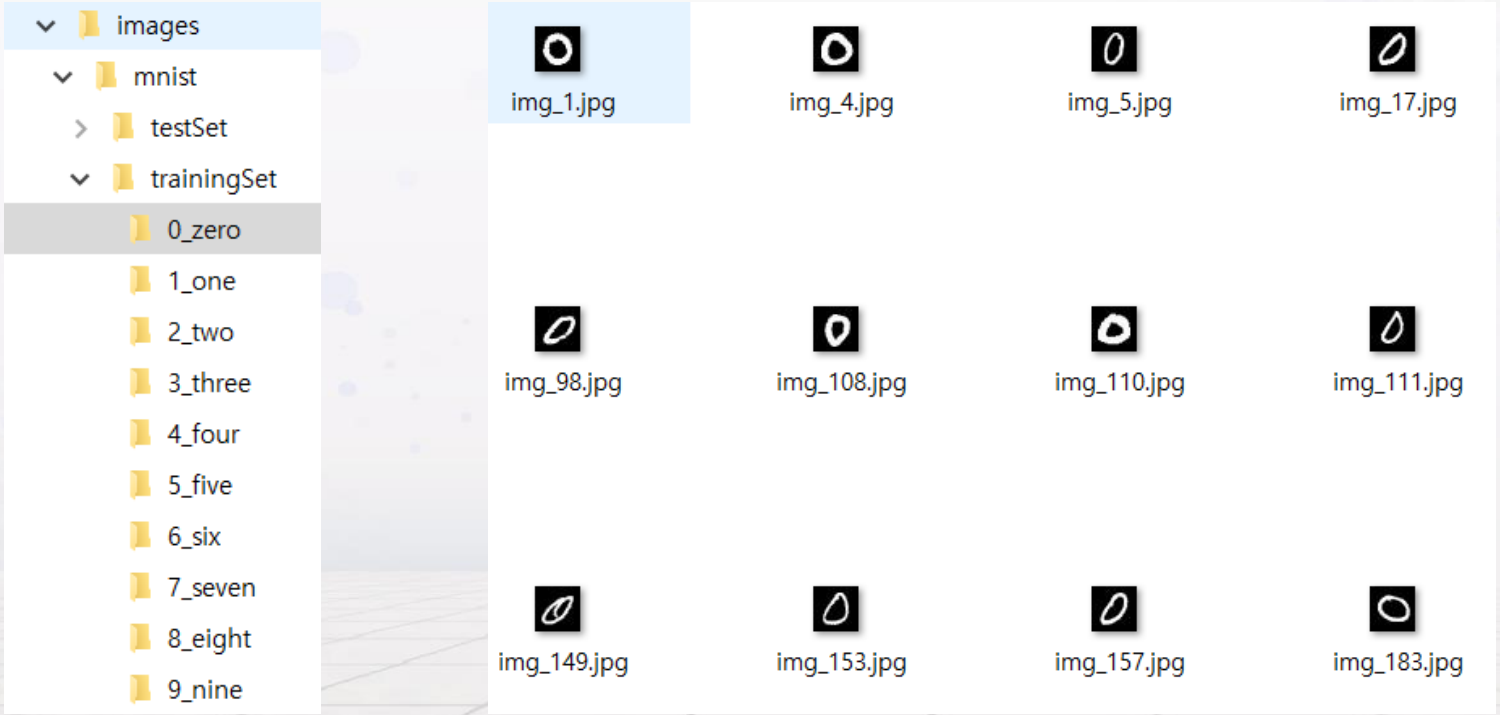
```
import os
import cv2
import keras
# 케라스 모델 생성 라이브러리
from keras import models
# 레이어 생성 라이브러리 (Dense: 입출력 연결)
from keras import layers
# 케라스 샘플데이터[mnist] 라이브러리 불러오기
from keras.datasets import mnist
# numpy 라이브러리
import numpy as np
from numpy import array
# 케라스 카테고리 라이브러리
from keras.utils import to_categorical
from sklearn.preprocessing import OneHotEncoder
# 시각화 라이브러리
import matplotlib.pyplot as plt
%matplotlib inline
```


1. 라이브러리 선언


```
1  import os
2  import cv2
3  # 케라스 모델 생성 라이브러리
4  import keras
5  from keras import models
6  # 레이어 생성 라이브러리 (Dense: 입출력 연결)
7  from keras import layers
8  # 케라스 샘플데이터[mnist] 라이브러리 불러오기
9  from keras.datasets import mnist
10 # numpy 라이브러리
11 import numpy as np
12 from numpy import array
13 # 케라스 카테고리 라이브러리
14 from keras.utils import to_categorical
15 #from sklearn.preprocessing import LabelEncoder
16 #from sklearn.preprocessing import OneHotEncoder
17 # 시각화 라이브러리
18 import matplotlib.pyplot as plt
19 %matplotlib inline
```

2. Keras 실습 (Image Classification)

2. 데이터 불러오기



2. Keras 실습 (Image Classification)

2. 데이터 불러오기

[훈련/테스트 세트] : 이미지/답지 별도 저장

폴더 리스트 반복

0_zero, 1_one, ...

이미지 리스트 반복

img_4.jpg, ..

데이터 저장

- 컬러: 그레이
- 사이즈: 28 * 28
- 이미지 저장
- 폴더 인덱스-> 답지

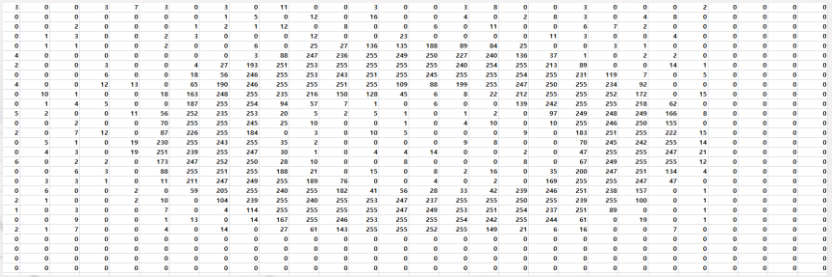
```
1  img
```

```
array([[ 3,  0,  0,  3,  7,  3,  0,  3,  0, 11,  0,  0,  3,
         0,  0,  3,  8,  0,  0,  3,  0,  0,  0,  2,  0,  0,
         0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  1,  5,  0, 12,  0, 16,
         0,  0,  4,  0,  2,  8,  3,  0,  4,  8,  0,  0,  0,
         0,  0],
```

1행

2행

```
img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
import pandas as pd

pd.DataFrame(img).to_csv("d:/zero.csv")
```



2. Keras 실습 (Image Classification)

2. 데이터 불러오기 #1

```
TRAIN_DIR = '../images/mnist/trainingSet/'
train_folder_list = array(os.listdir(TRAIN_DIR))
train_folder_list

IMG_SIZE = 28

train_images=[]
train_labels=[]

# 각 폴더 0, 1 순 탐색
for index in range(len(train_folder_list)):
    path = os.path.join(TRAIN_DIR, train_folder_list[index])
    path = path + ' / '
    img_list = os.listdir(path)
    # 폴더 및 img 파일명 찾아 경로 생성
    for img in img_list:
        img_path = os.path.join(path, img)
        img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
        new_img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
        train_images.append(new_img)
        train_labels.append(index)
```

폴더 리스트: 0_zero, 1_one,...

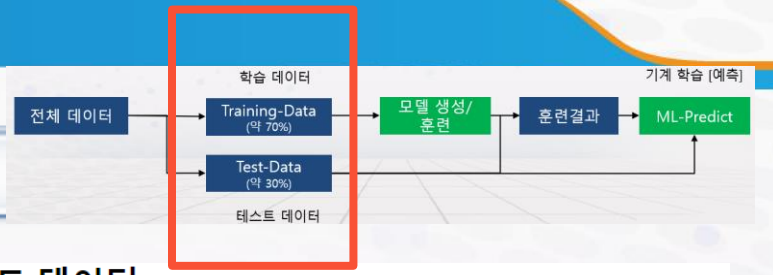
imread 후 try pass 필요

```
try:
    img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
    new_img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
    train_images.append(new_img)
    train_labels.append(index)
except:
    pass
```

```
try:
    img = cv2.imread(img_path, cv2.IMREAD_GRAYSCA
    new_img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
    train_images.append(new_img)
    train_labels.append(index)
except Exception as e:
    print(e)
    pass
```

2. Keras 실습 (Image Classification)

2. 데이터 불러오기 #2




```
TEST_DIR = '../images/mnist/testSet/'
test_folder_list = array(os.listdir(TEST_DIR))
test_folder_list

test_images=[]
test_labels=[]

# 각 폴더 0, 1 순 탐색
for index in range(len(test_folder_list)):
    path = os.path.join(TEST_DIR, test_folder_list[index])
    path = path + '/'
    img_list = os.listdir(path)
    # 폴더 및 img 파일명 찾아 경로 생성
    for img in img_list:
        img_path = os.path.join(path, img)
        img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
        new_img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
        test_images.append(new_img)
        test_labels.append(index)
```

2. 테스트 데이터

```
1  TEST_DIR = '../images/mnist/testSet/'
2  test_folder_list = array(os.listdir(TEST_DIR))
3  test_folder_list
4
5  test_images=[]
6  test_labels=[]
7  for index in range(len(test_folder_list)):
8      path = os.path.join(TEST_DIR, test_folder_list[index])
9      path = path + '/'
10     img_list = os.listdir(path)
11     for img in img_list:
12         img_path = os.path.join(path, img)
13         img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
14         new_img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
15         test_images.append(new_img)
16         test_labels.append(index)
```

2. Keras 실습 (Image Classification)

2. 데이터 불러오기 #3

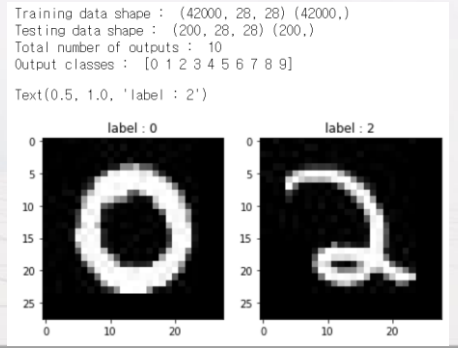
```
train_images = array(train_images)
train_labels = array(train_labels)
test_images = array(test_images)
test_labels = array(test_labels)
print('Training data shape : ', train_images.shape, train_labels.shape)
print('Testing data shape : ', test_images.shape, test_labels.shape)
```

list -> array 형태 변환

```
Training data shape :  (42000, 28, 28) (42000,)
Testing data shape :  (200, 28, 28) (200,)
```

```
# 훈련데이터의 답지분류 범위 정의
classes = np.unique(train_labels)
nClasses = len(classes)
print('Total number of outputs : ', nClasses)
print('Output classes : ', classes)
```

```
# 훈련/데이터 데이터 시각화
plt.figure(figsize=[7,5])
plt.subplot(121)
plt.imshow(train_images[0], cmap='gray')
plt.title("label : {}".format(train_labels[0]))
plt.subplot(122)
plt.imshow(test_images[10000], cmap='gray')
plt.title("label : {}".format(test_labels[42]))
```

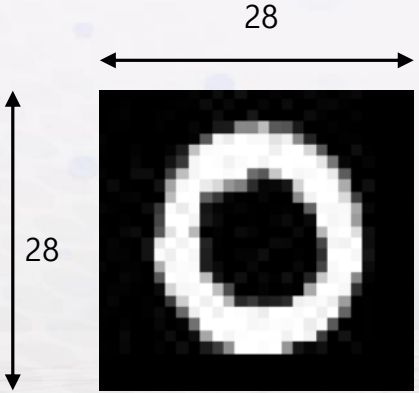


2. Keras 실습 (Image Classification)

3. 데이터 정제


```
# Change from matrix to array of dimension 28x28 to array of dimention width(28), height(28), depth(1)
train_data = train_images.reshape(len(train_images), IMG_SIZE,IMG_SIZE,1)
test_data = test_images.reshape(len(test_images), IMG_SIZE,IMG_SIZE,1)
```

color channel
현재는 그레이 1차원이기
때문에 정제 필요없음




```
1 train_images.shape
```

```
(42000, 28, 28)
```

```
1 train_data.shape
```

```
(42000, 28, 28, 1)
```

2. Keras 실습 (Image Classification)

3. 데이터 정제

```
train_data = train_data.astype('float')
test_data = test_data.astype('float')

train_data /= 255
test_data /= 255
```

정규화는 필수임

3-2. float 타입변환 및 정규화

```
1  train_data = train_data.astype('float')
2  test_data = test_data.astype('float')
3
4  train_data /= 255
5  test_data /= 255
```

2. Keras 실습 (Image Classification)

3. 데이터 정제

```
# Change the labels from integer to categorical data
train_labels_one_hot = to_categorical(train_labels)
test_labels_one_hot = to_categorical(test_labels)
```

향후 softmax activation 활용하기 위함

```
# Display the change for category label using one-hot encoding
print('Original label 0 : ', train_labels[2])
print('After conversion to categorical ( one-hot ) : ', train_labels_one_hot[2])
```

0 ➡ [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

2 ➡ [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]

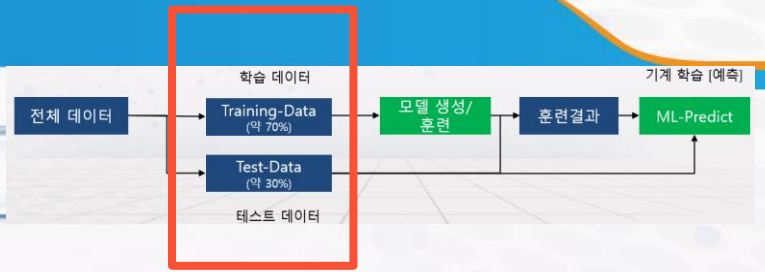
reverse 시
np.argmax(train_labels_one_hot[10000]

[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

0.80
0.05
0.05
0.05
0.05
0.05
0.00
0.00
0.00
0.00

이미지 분류 시
사용되는 softmax 함수와
맵핑됨!

2. Keras 실습 (Image Classification)



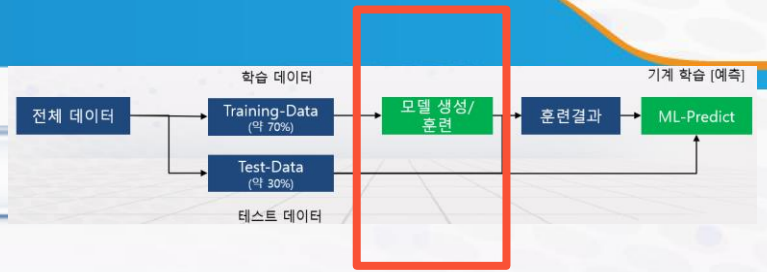
3. 데이터 정제

```
from sklearn.preprocessing import OneHotEncoder
oe_label = OneHotEncoder()

train_labels_t = train_labels.reshape(-1,1)
test_labels_t = test_labels.reshape(-1,1)
oe_label.fit(train_labels_t)
train_labels_one_hot = oe_label.transform(train_labels_t).toarray()
test_labels_one_hot = oe_label.transform(test_labels_t).toarray()
```

```
oe_label.inverse_transform(test_labels_one_hot).reshape(-1)
```

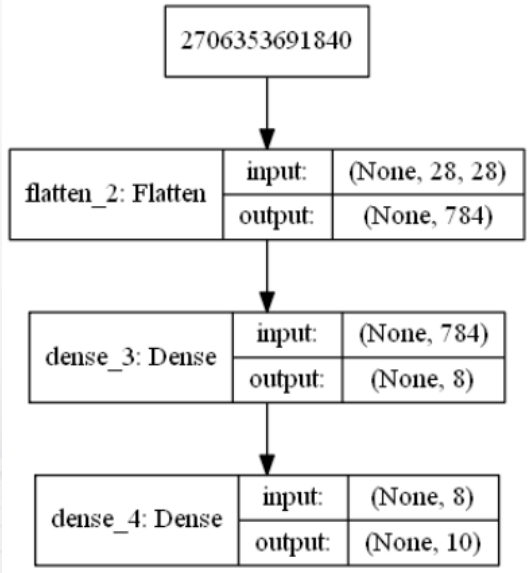
기존 형태를 초기화 시킨 후 열컬럼(1) 변환

2. Keras 실습 (Image Classification)



4. 케라스 모델 정의

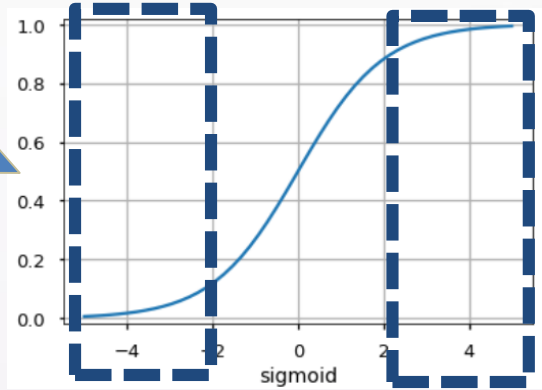
```
imprt keras
from keras import Sequential
from keras.layers import Flatten, Dense

# 입력데이터 형태
modelDim = train_data[0].shape

# 층 누적 기본형태
model = Sequential()

# 신경망의 첫 번째 레이어에서 입력 데이터 크기 정의
model.add(Flatten(input_shape=modelDim))
model.add(Dense(8, activation='relu'))
model.add(Dense(nClasses, activation='softmax'))

# model.summary()를 통해 모델을 살펴보세요.
model.summary()
```



참조. 딥러닝 작동 원리

Sigmoid 활용 시
미분 값 0에 가까워져
역전파로 전달 시
Weight 업데이트 거의 안됨

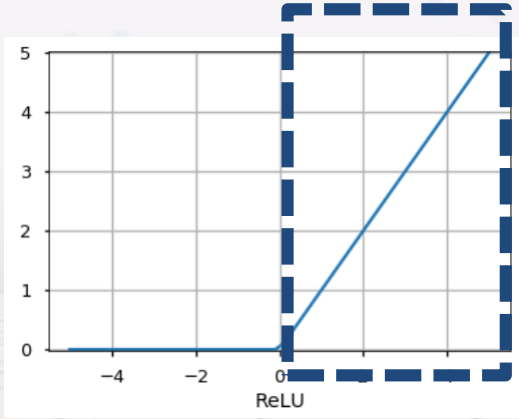
-> Vanishing Gradient
로 인해 학습이 안되는 현상을
Under fitting이라고 함



0~1 사이 값

Rectified Linear Unit

미분 값이 전부 0 이여서
역전파 시
Weight 업데이트 안되는
경우 없음!

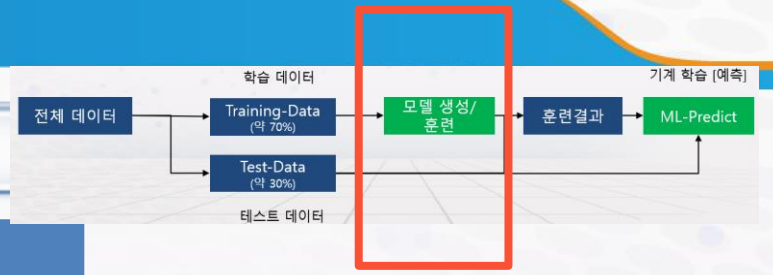


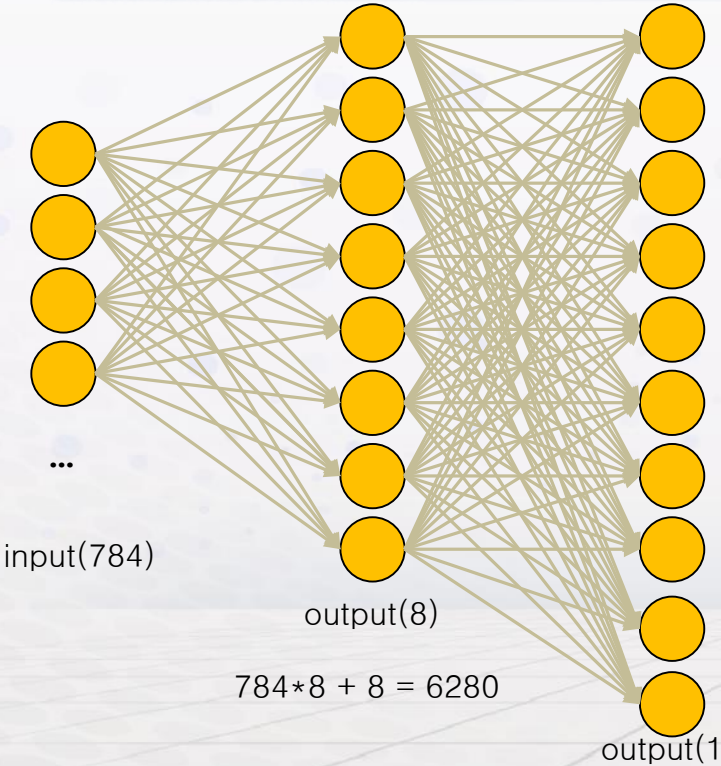
F(x) = max(0, x)

https://en.wikipedia.org/wiki/Rectifier_(neural_networks)

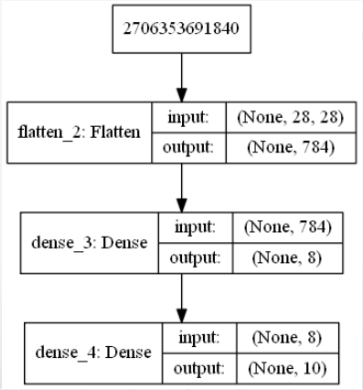
2. Keras 실습 (Image Classification)

4. 케라스 모델 정의





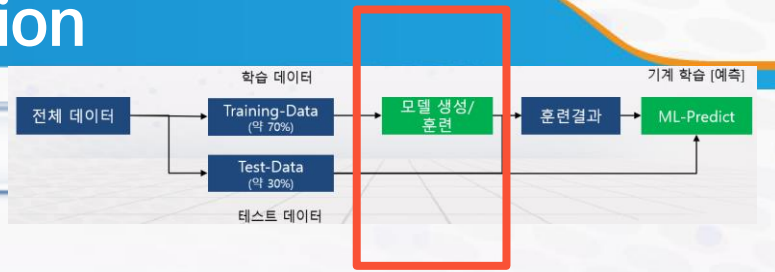
Dense:
전결합 모드



Layer (type)	Output Shape	Param #
flatten_1 (Flatten)	(None, 784)	0
dense_1 (Dense)	(None, 8)	6280
dense_2 (Dense)	(None, 10)	90

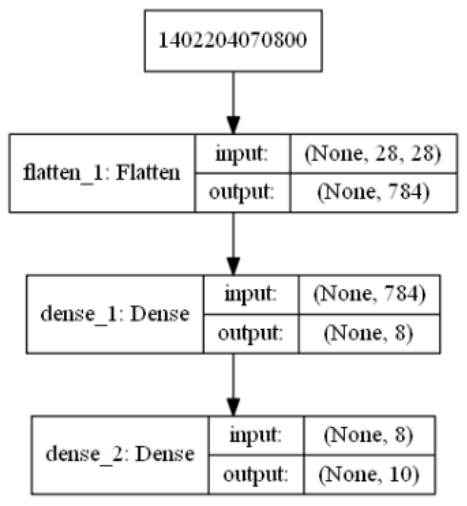
Total params: 6,370
Trainable params: 6,370
Non-trainable params: 0

4-4. 딥러닝 모형 실습 (keras)-Regression



4. 케라스 모델 정의 (Skip!, summary 대체)

```
# 모델 시각화
from keras.utils import plot_model
plot_model(model, to_file='model_plot.png',
           show_shapes=True,
           show_layer_names=True)
from IPython.display import Image
Image(retina=True, filename='model_plot.png')
```

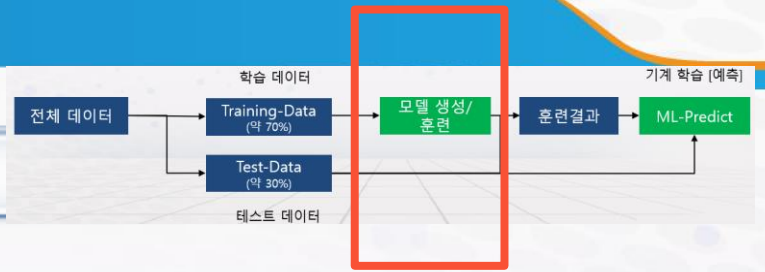


실제 모델 시각화는
Graphviz 설정 외에 (안될경우에!!!)
Site-package 내 graphviz 들어가서
Vis_util 파일을 import pydotplus as pydot으로 변경 필요
* Conda install graphviz 필요

C:₩Users₩hk₩Anaconda3₩Lib₩site-packages

```
FileNotFoundError: [WinError 2] 지정된 파일을 찾을 수 없습니다

During handling of the above exception, another exception occurred:

FileNotFoundError                         Traceback (most recent call last)
~₩Anaconda3₩lib₩site-packages₩keras₩utils₩vis_utils.py in _check_pydot()
     27         # to check the pydot/graphviz installation.
---> 28         pydot.Dot.create(pydot.Dot())
     29     except OSError:
```

2. Keras 실습 (Image Classification)



5. 케라스 모델 훈련방법 설정

```
# Sequatial 방싱 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

```
# Sequatial 방싱 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'])
```

2. Keras 실습 (Image Classification)

6. 모델 훈련

```
from datetime import datetime
now = datetime.now()
date = now.strftime("%Y-%m-%d_%H%M")
save_dir = "./logs_{}".format(date)
callbacks = [
    keras.callbacks.TensorBoard(
    log_dir = save_dir,
    write_graph=True,
    write_images=True),

    keras.callbacks.EarlyStopping(
    monitor = 'val_acc', patience=10,
    )
]

# 모델을 32개의 샘플씩 미니 배치를 만들어 20번의 에포크 동안 훈련
history = model.fit(train_data, train_labels_one_hot,
                    batch_size=32,
                    epochs=20,
callbacks = callbacks)
```

텐서보드 활용

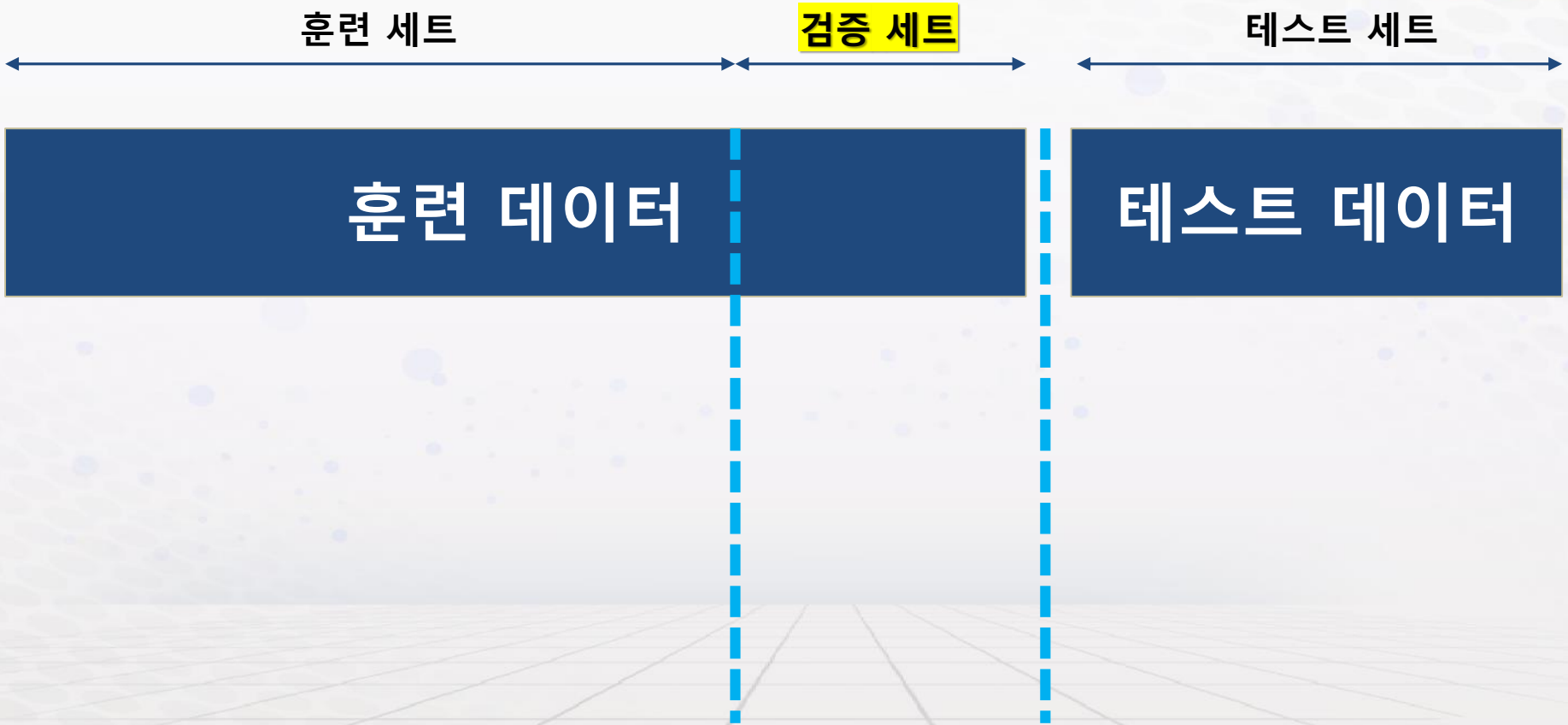
과적합 피하기 조기종료
(patience: 몇회 연속 val 손실 참기)

배치: 몇문제를 풀고 신경망 업데이할지 정함

에포크: 반복훈련 횟수

validation_split=0.2 사용여부 확인

[참조] 훈련데이터 분리 / 데이터 많은 경우



[참조] 훈련데이터 분리 / 데이터 많은 경우

참조

```
# 훈련데이터 분리 (훈련/검증)
trainingData_features,₩
validData_features,₩
trainingData_label,₩
validata_label=₩
train_test_split(train_data,
                 train_labels_one_hot,
                 test_size = 0.2,
                 random_state=20)

# 훈련 시 검증데이터 설정
history = model.fit(trainingData_features, trainingData_label,
validation_data=(validData_features,validData_label),
                 batch_size=32,
                 epochs=10,
                 callbacks = callbacks)
```

4-4. 딥러닝 모형 실습 (keras)-Regression

재학습 (참조)

모델 선언 구조 저장

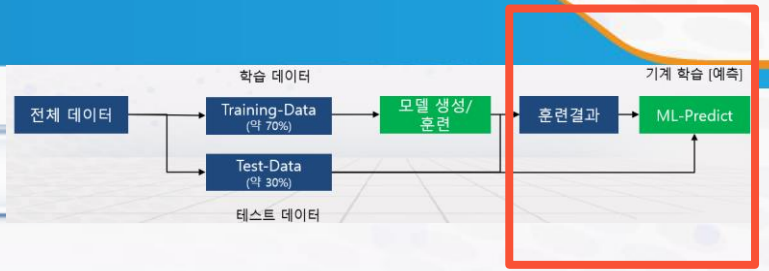
```
# 모델 저장
model_json = model.to_json()

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

model.save_weights("linear_keras_sellout.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(loaded_model_json)
loaded_model.load_weights("linear_keras_sellout.h5")
```

```
# 모델 재 컴파일
loaded_model.compile(optimizer='adam',
            loss='categorical_crossentropy',
            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)
history =loaded_model.fit(trainingData_features,
                trainingData_label,
                validation_split=0.2, epochs= EPOCHS,
callbacks=[early_stopping_monitor])
```

2. Keras 실습 (Image Classification)



7. 모델 추론

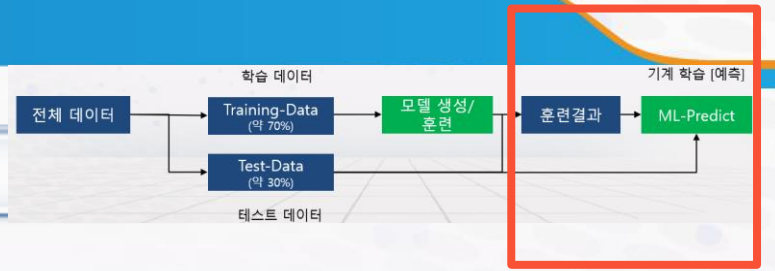
```
# verbose: 정보표시 레벨 (0,1)
test_loss, test_acc = model.evaluate(test_data,
                                     test_labels_one_hot)
print(test_loss, test_acc)
```

7. 모델 추론

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

```
200/200 [==============================] - 0s 95us/step
3.32854268014431 0.775
```

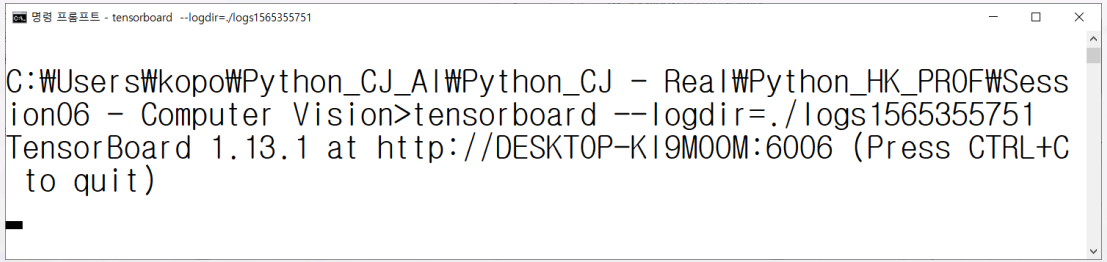
2. Keras 실습 (Image Classification)



8. 훈련내용 확인하기

1. 명령프롬프트(cmd) 켜고 python 작업 위치로 이동

2. 명령어 실행: tensorboard --logdir=./logs15...

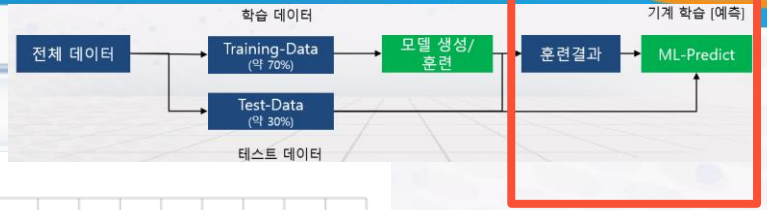


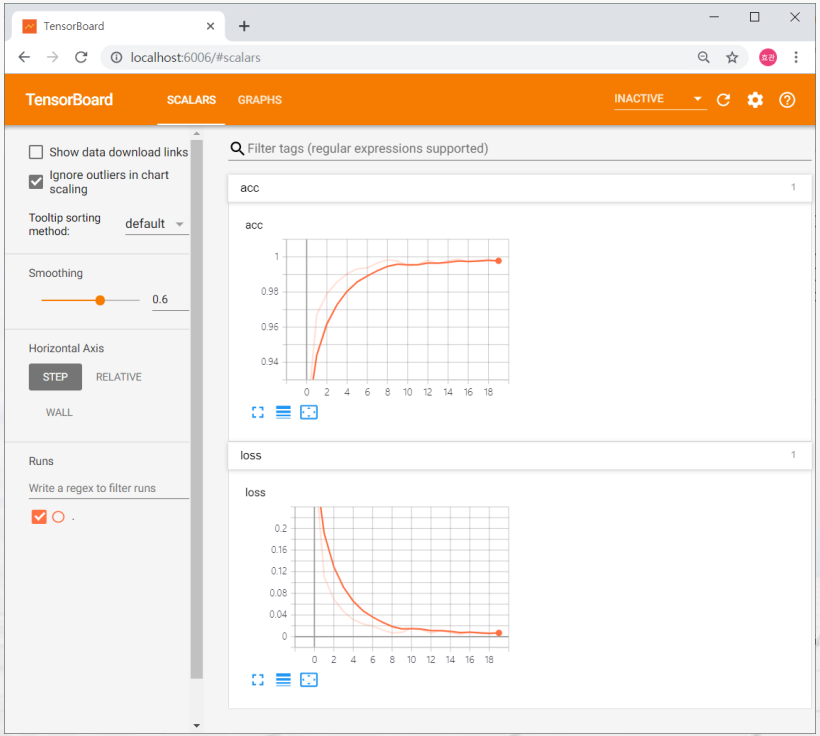
3. 웹열고 localhost:6006 실행

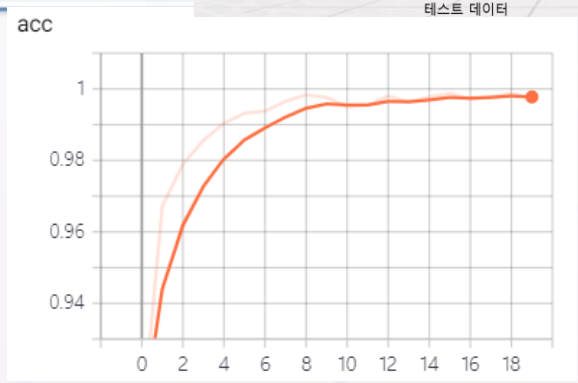
http://localhost:6006

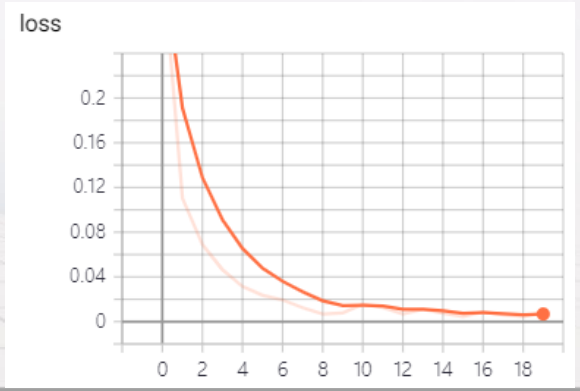
2. Keras 실습 (Image Classification)

8. 훈련내용 확인하기

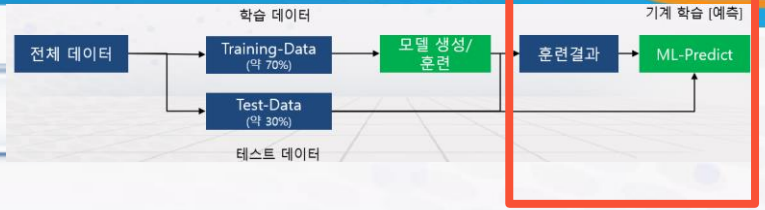








2. Keras 실습 (Image Classification)



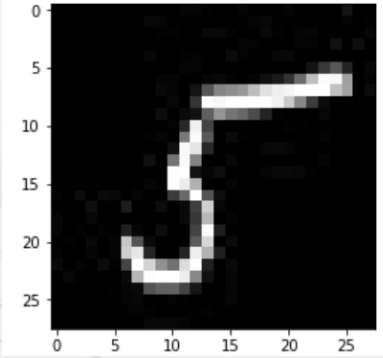
9. 예측

```
prd_images = []
imgpah="../images/mnist/testSet/5/img_24.jpg"
img = cv2.imread(imgpah, cv2.IMREAD_GRAYSCALE)
prd_images.append(img)
prd_images=array(prd_images)

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

9. 예측 및 비교

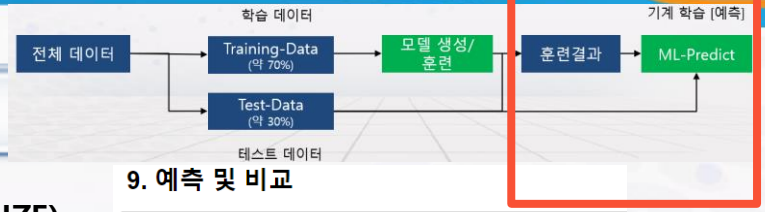
```
1 prd_images = []
2 imgpah="../images/mnist/testSet/5/img_24.jpg"
3 img = cv2.imread(imgpah, cv2.IMREAD_GRAYSCALE)
4 prd_images.append(img)
5 prd_images=array(prd_images)
6
7 plt.imshow(img, cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x236f1148908>
```



2. Keras 실습 (Image Classification)

9. 예측

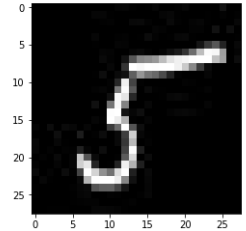



```
test_data = prd_images.reshape(len(prd_images) , IMG_SIZE, IMG_SIZE)
test_data = test_data.astype('float')
test_data /= 255
# Predict the most likely class
label_pred = model.predict(test_data[[0],:])

label_pred

np.argmax(label_pred)
```

9. 예측 및 비교

```
prd_images = []
imgpah="../images/mnist/testSet/5/img_24.jpg"
img = cv2.imread(imgpah, cv2.IMREAD_GRAYSCALE)
prd_images.append(img)
prd_images=array(prd_images)

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

```
<matplotlib.image.AxesImage at 0x236f17e8908>
```




```
test_data = prd_images.reshape(len(prd_images) , IMG_SIZE, IMG_SIZE)
test_data = test_data.astype('float')
test_data /= 255
```

```
# Predict the most likely class
label_pred = model.predict(test_data[[0],:])
```

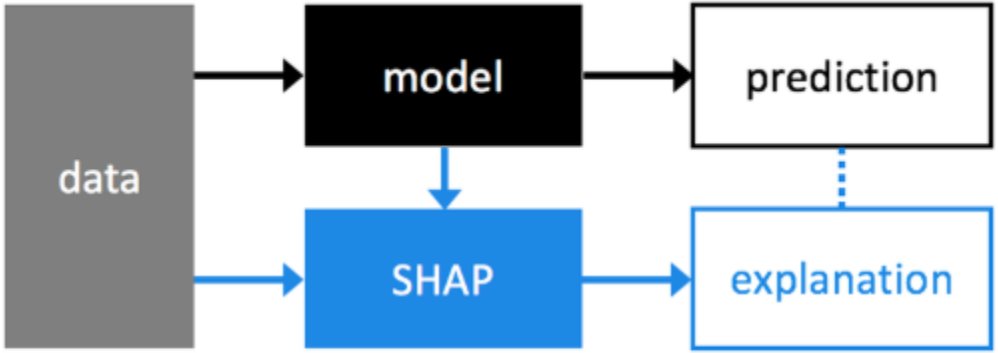
```
label_pred
```

```
array([[6.2127179e-04, 1.4492736e-09, 1.3350871e-06, 1.8702475e-06,
        1.4132684e-08, 9.9937457e-01, 1.4239636e-09, 9.4714380e-07,
        1.8930112e-14, 2.1316687e-14]], dtype=float32)
```

```
np.argmax(label_pred)
```

```
5
```

참조 (Explainable AI, SHAP)

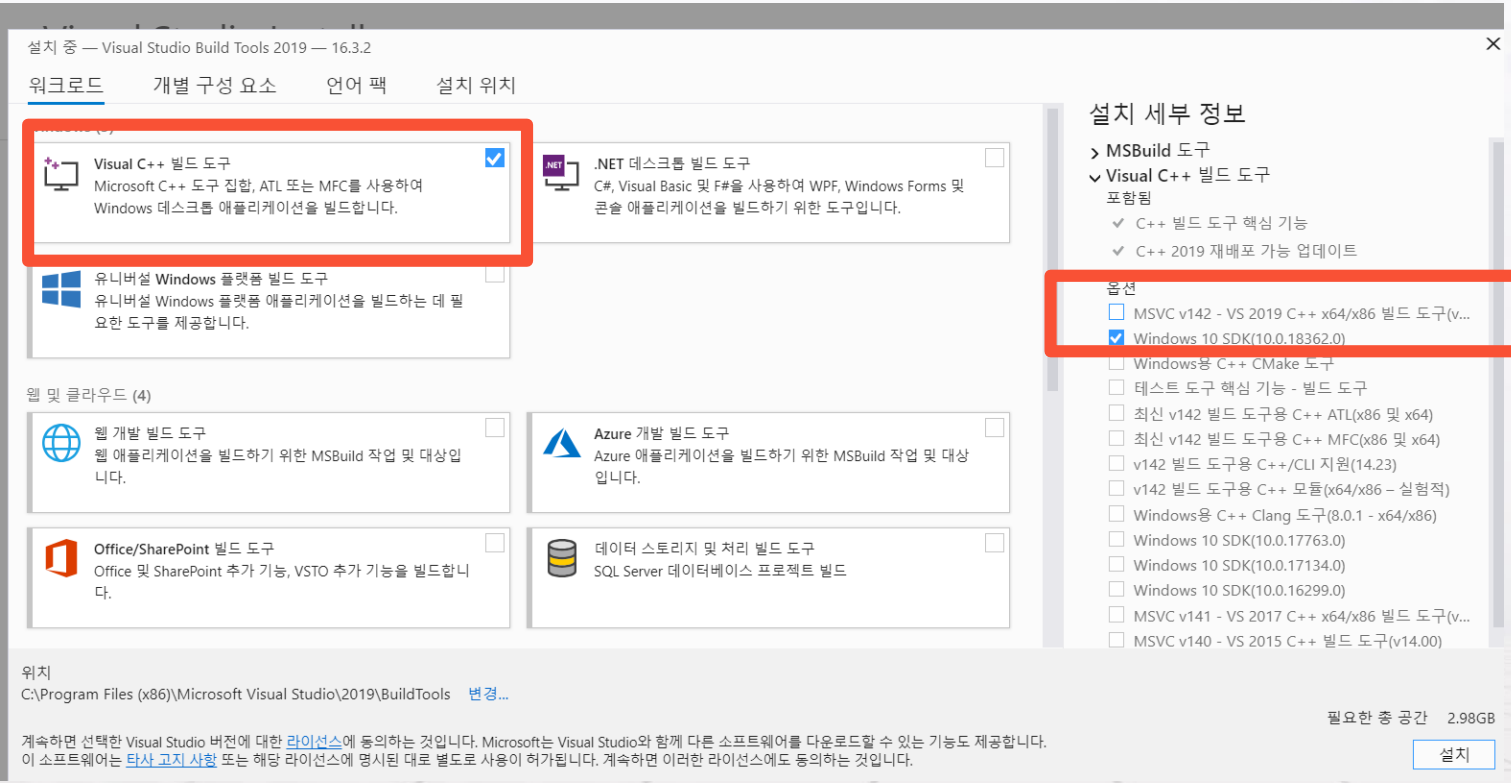


SHAP (SHapley Additive exPlanations) is a unified approach to explain the output of any machine learning model. SHAP connects game theory with local explanations, uniting several previous methods and representing the only possible consistent and locally accurate additive feature attribution method based on expectations (see the SHAP NIPS paper for details).

출처: https://shap.readthedocs.io/en/latest/

참조 (Explainable AI, SHAP)

SHAP 설치하기 : sparkkorea.com 프로그램(XAI_SHAP 폴더자료 다운) 이후 vsBuildTools.exe 설치 후 재시작!



2개 모두 선택

참조 (Explainable AI, SHAP)

SHAP 계산방식 : 내가 빠졌을때의 기여도는?
(코드 100줄을에 대한)

V(X)	Line of codes
L	10
M	30
N	5
L,M	50
L,N	40
M,N	35
L,M,N	100

순서	L 기여도	M 기여도	N 기여도
L,M,N	V(L)=10	V(L,M)-VL) = 50-10	V(L,M,N)=V(L,M) = 100-50
L,N,M	V(L)=10	V(L,M,N)-V(L,N)	V(L,N)-V(L)
M,L,N	V(L,M) – V(M)	V(M)	V(L,M,N)-V(L,M)
M,N,L	V(L,M,N)-V(M,N)	V(M)	V((M,N)-V(M)
N,L,M	V(L,N)-V(L)	V(L,M,N)-V(L,N)	V(N)
N,M,L	V(L,M,N)-V(M,N)	V(M,N)	V(N)

참조 (Explainable AI, SHAP)

SHAP 계산방식 : 내가 빠졌을때의 기여도는?
(코드 100줄을에 대한)

순서	L 기여도	M 기여도	N 기여도
L,M,N	V(L)=10	V(L,M)-VL) = 50-10	V(L,M,N)=V(L,M) = 100-50
L,N,M	V(L)=10	V(L,M,N)-V(L,N)	V(L,N)-V(L)
M,L,N	V(L,M) – V(M)	V(M)	V(L,M,N)-V(L,M)
M,N,L	V(L,M,N)-V(M,N)	V(M)	V((M,N)-V(M)
N,L,M	V(L,N)-V(L)	V(L,M,N)-V(L,N)	V(N)
N,M,L	V(L,M,N)-V(M,N)	V(M,N)	V(N)

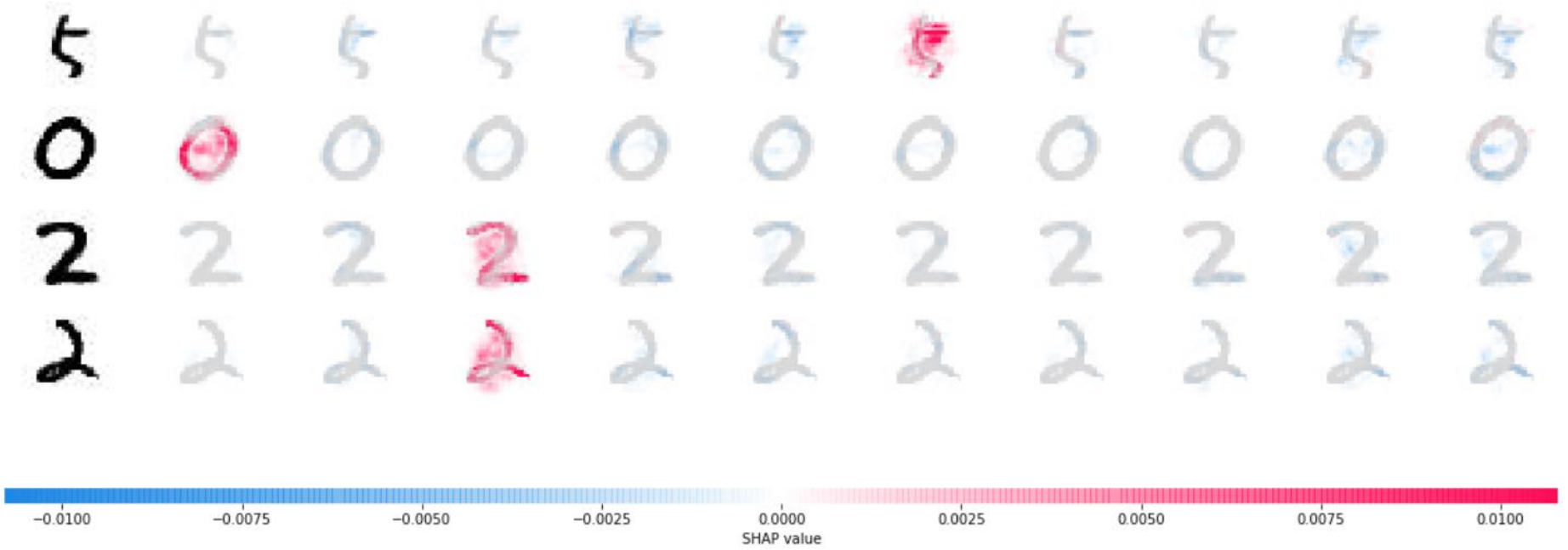
기여자	SHAP 계산 공식	SHAP 값
L	1/6(10+10+20+65+35+65)	34.17
M	1/6(40+60+30+30+60+30)	41.7
N	1/6(50+30+50+5+5+5)	24.17

참조 (Explainable AI, SHAP)

5,0,2,2 값을 5,0,2,2라고 분류하게 한 과거 데이터 중 10개의 공헌도
중 6번째 이미지가 가장 기여도가 컸으며 특히 윗쪽 끝 꺽이는 부분이 가장 의미있는 요소임

```
6
7  # plot the feature attributions
8  shap.image_plot(shap_values, -trainingData_features[1:5])
```

빨간색은 긍정 영향 +



참조 (Explainable AI, SHAP)

```
import shap
import numpy as np

# 설명할 백데이터 샘플을 생성함 (100 개 무작위 선택)
background = train_data[np.random.choice(train_data.shape[0], 100, replace=False)]

# 백그라운드 이미지들에대해서 설명값 생성
e = shap.DeepExplainer(model, background)

# shap 값 생성
#e = shap.DeepExplainer((model.layers[0].input, model.layers[-1].output), background)
test_choice = test_data[np.random.choice(test_data.shape[0], 5, replace=False)]

shap_values = e.shap_values(test_choice[0:5])

# plot the feature attributions
shap.image_plot(shap_values, -test_choice[0:5])
```

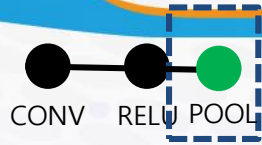
참조. (과적합 방지 Dropout)

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

```
model_reg = models.Sequential()
model_reg.add(layers.Dense(512, activation='relu', input_shape=(dimData,)))
model_reg.add(layers.Dropout(0.5))
model_reg.add(layers.Dense(512, activation='relu'))
model_reg.add(layers.Dropout(0.5))
model_reg.add(layers.Dense(nClasses, activation='softmax'))
```

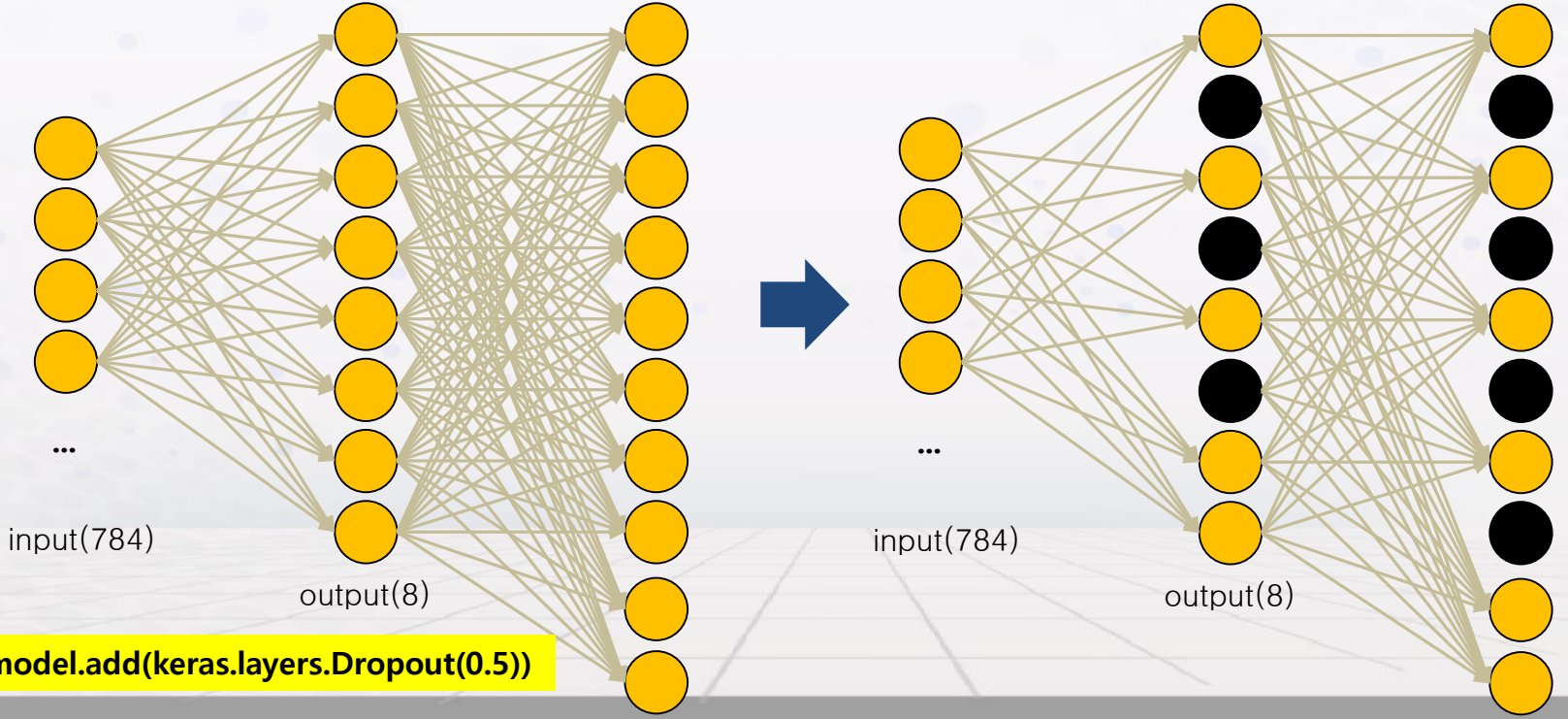
Dropout은 훈련중 neurons을 랜덤하게 끄게 함으로써 과적합 (overfitting 문제를 해결한다)

참조. (과적합 방지 Dropout)



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. 핵심정리 및 Q&A

기억합시다

1. Neural Network에 대해서 이해한다.
2. 일반 이미지가 어떻게 데이터로 표현되는지를 이해한다.
3. 일반 이미지가 어떠한 전처리과정을 거치는지 이해한다.
4. Keras 실습을 통해 작동원리를 이해한다.

감사합니다.