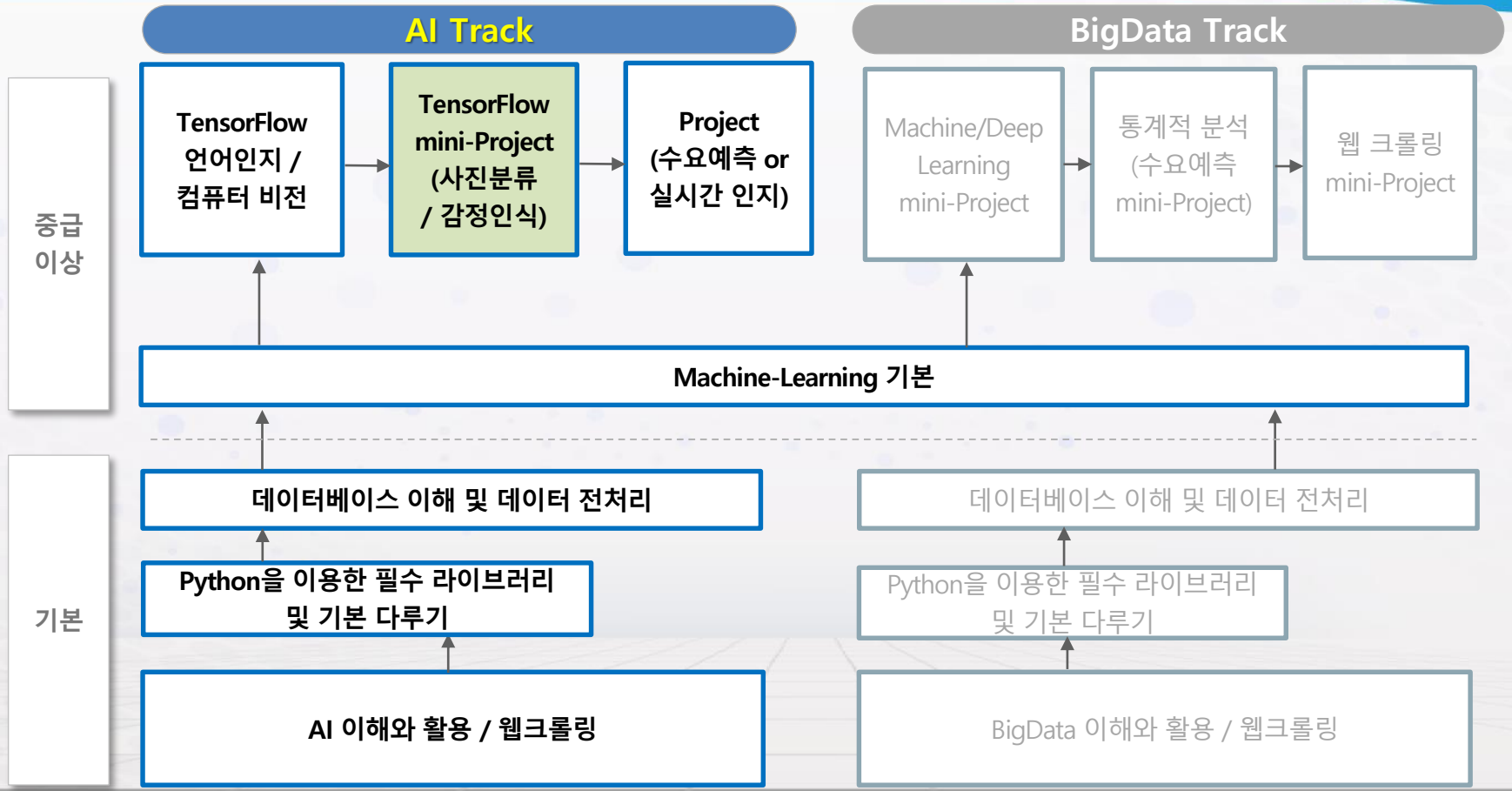
바탕화면 내 " 5. 딥러닝 프로젝트 " 을 다운로드 받은 후

코드는 코드 폴더 내
Images 은 dataset과 동일한 레벨에 저장해주세요.

주요 시나리오


AI Track
BigData Track
중급 이상
기본
TensorFlow 언어인지 / 컴퓨터 비전
TensorFlow mini-Project (사진분류 / 감정인식)
Project (수요예측 or 실시간 인지)
Machine/Deep Learning mini-Project
통계적 분석 (수요예측 mini-Project)
웹 크롤링 mini-Project
Machine-Learning 기본
데이터베이스 이해 및 데이터 전처리
데이터베이스 이해 및 데이터 전처리
Python을 이용한 필수 라이브러리 및 기본 다루기
Python을 이용한 필수 라이브러리 및 기본 다루기
AI 이해와 활용 / 웹크롤링
BigData 이해와 활용 / 웹크롤링

기본과정 시나리오

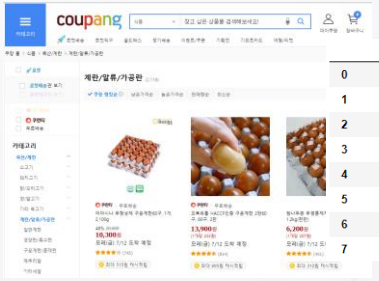
부족한 데이터를 외부에서 가져오는 방법 학습

웹 크롤링 (공공데이터 포털 / 웹데이터 스크랩)

공공데이터포털 내 실시간 대기질 나쁨 관측소 수집

	대기오염 나쁨 위치	관측소위치
0	중구	서울특별시 중구 덕수궁길 15시청서소문별관 3동
1	청계천로	서울 중구 청계천로 184(청계천4가사거리 남강빌딩 앞)
2	용산구	서울 용산구 한남대로 136서울특별시중부기술교육원
3	강변북로	서울 성동구 강변북로 257한강사업본부 옆
4	홍릉로	서울 동대문구 홍릉로 1(청량리전철역 사거리 SC제일은행 앞)

실시간 쿠팡 내 특정 카테고리 상품가격 수집

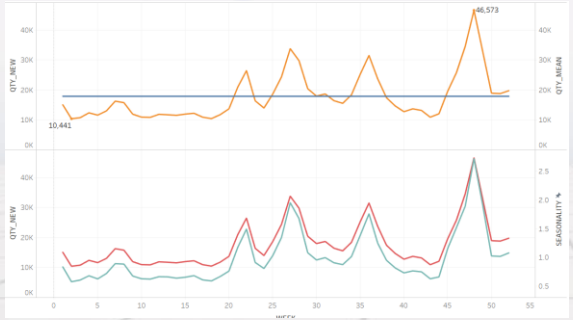


	0	1
0	더마시나 무항생제 구운계란60구, 1개, 2,100g	10,300
1	오복유통 HACCP인증 구운계란 2판60구, 60구, 2판	13,900
2	참나무촌 무염훈제계란, 30개입, 1.2kg(한판)	6,200
3	꾼란 맥반석 구운계란 30구 1판, 30개입, 1.2kg	7,900
4	맛군 촉촉 쫄깃 구운 계란, 30알, 1박스	7,900
5	감동란 간이베어 있는 촉촉한 반숙계란, 50g, 30개입	16,900
6	[계란사랑] 맥반석 구운계란 구운란 60구 (2판), 2700g	11,900
7	진주햄 오마이 포켓 메추리알 5p, 25g, 10개입	9,440

통계적 분석 (통계적 로직 작성)

데이터 조작 방법 학습 (조인/그룹바이!)

아이템별 계절성 지수(주차효과) 산출



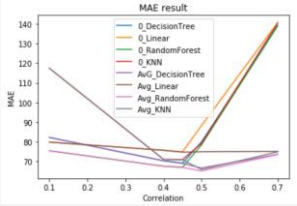
regionid	productgroup	yearweek	volume	avg_volume
A01	ST0001	201624	772706	672850.4211
A01	ST0001	201625	786218	672850.4211
A01	ST0001	201626	871247	672850.4211
A01	ST0001	201641	746061	672850.4211
A01	ST0002	201401	155729	189489.9242
A01	ST0002	201402	123875	189489.9242
A01	ST0002	201403	114744	189489.9242
A01	ST0002	201404	137162	189489.9242

심화과정 Step1 시나리오

머신러닝 / 딥러닝 (기계 학습)

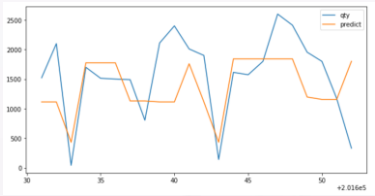
통계 외 기계를
학습 시키는 방법 학습

수요예측 및
아이템별 베스트 머신러닝 모델 추천



	ITEM	MIN_MAE	MODEL	COR_STD	FEATURES
0	ITEM043	123.73	DecisionTreeRegressor	0.4	HCLUS, PRO_PERCENT, HOLI_YN
1	ITEM044	4.29	DecisionTreeRegressor	0.4	YEARWEEK, YEAR, PRO_PERCENT, PROM_YN
2	ITEM051	5.98	LinearRegression	0.4	HCLUS, PRO_PERCENT, HOLI_YN

loss function 별 ANN (Artifical Neural Network) 실습
(수요예측, 당뇨병 여부, 손글씨 예측)



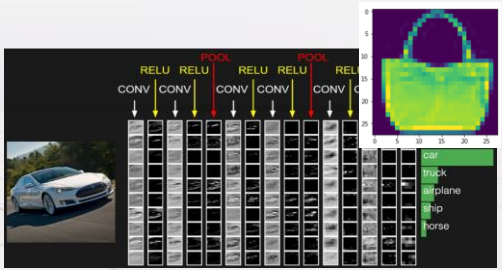
	diastolic	triceps	insulin	bmi	dpf	age	diabetes
	72	35	0	33.6	0.627	50	1
	66	29	0	26.6	0.351	31	0
	64	0	0	23.3	0.672	32	1
	66	23	94	28.1	0.167	21	0
	40	35	168	43.1	2.288	33	1

언어인지 / 컴퓨터 비전

딥러닝 응용

딥러닝 연동을 위한 자연어 처리
딥러닝 및 RNN 이해

딥러닝 연동을 위한 opencv 주요 핵심
딥러닝 및 CNN 이해

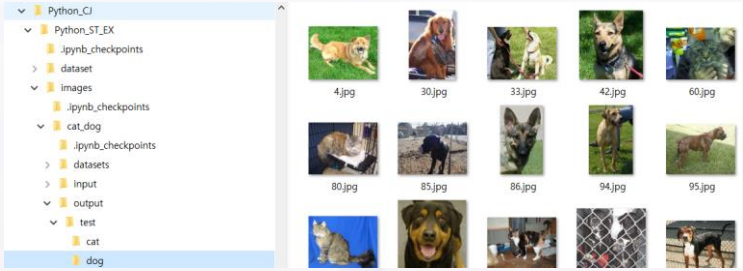


심화과정 Step2 시나리오

딥러닝 프로젝트

딥러닝 / 컴퓨터비전 융합 실습

사진 분류기 (CNN)

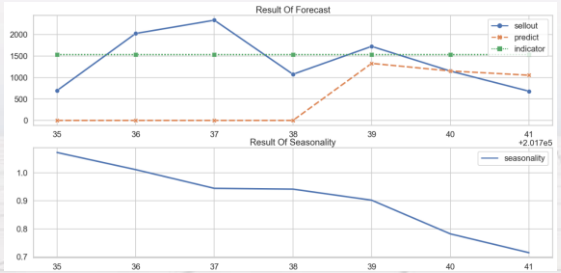


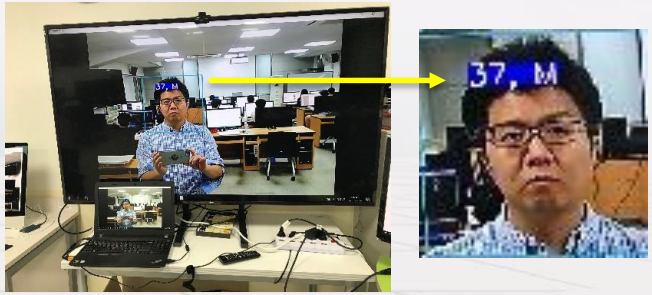
얼굴/감정 인식모델 (Haar, CNN)

프로젝트

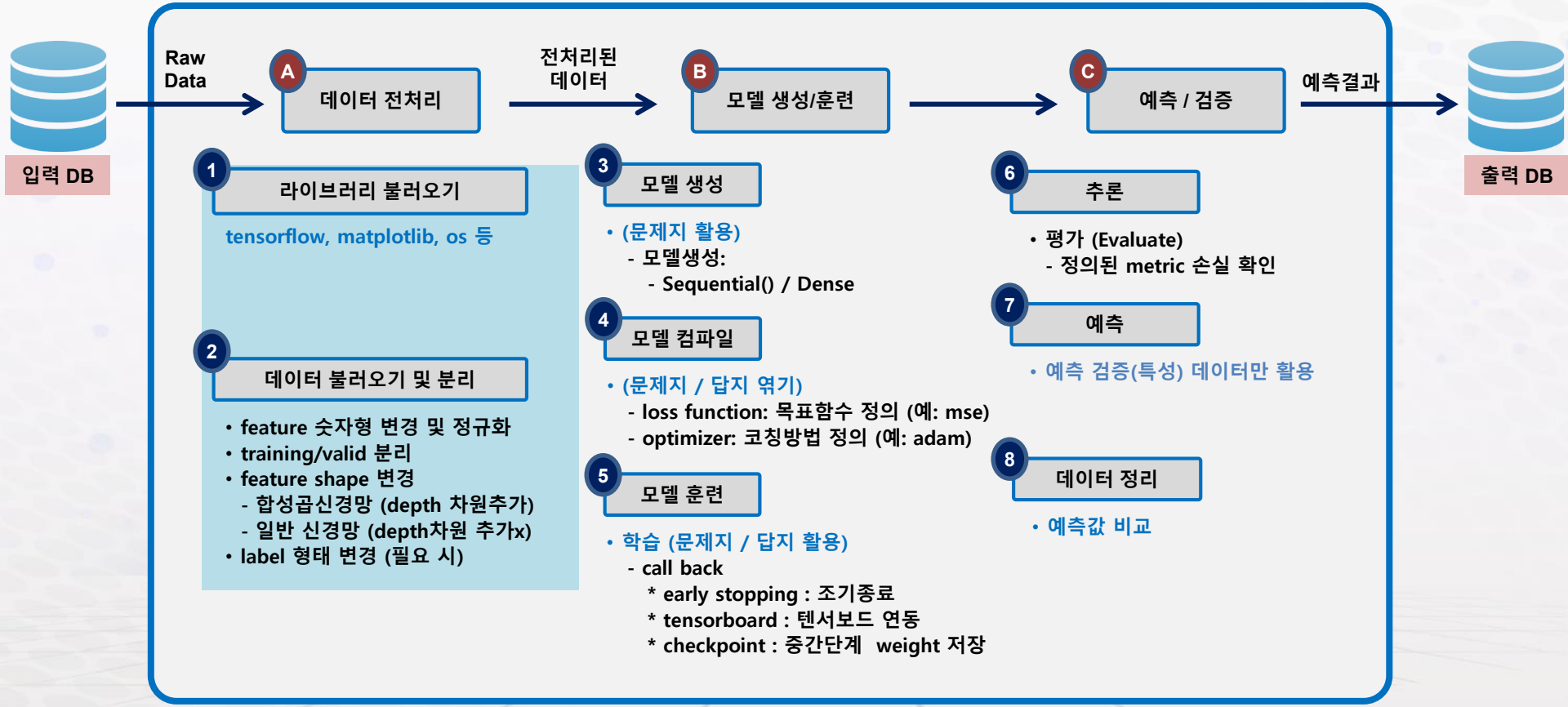
수요예측 (하이브리드) or
딥러닝 / 컴퓨터 비전 응용 실습

수요예측 or 나이/성별 인식 (데이터 처리, 통계, CNN)



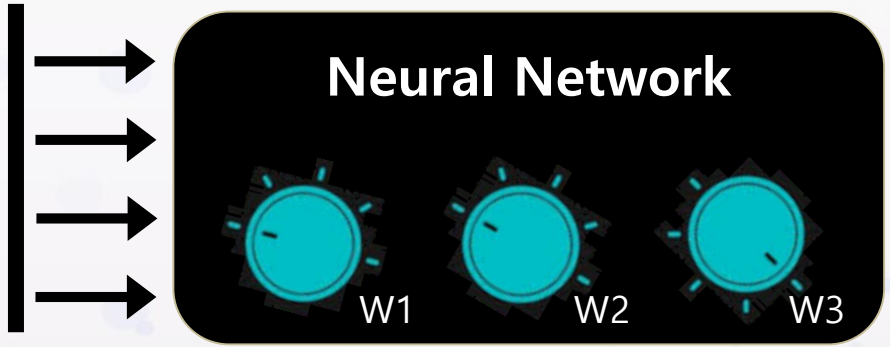


참조. 딥러닝 작동 원리


입력 DB
Raw Data
A
데이터 전처리
전처리된 데이터
B
모델 생성/훈련
C
예측 / 검증
예측결과
출력 DB
1
라이브러리 불러오기
tensorflow, matplotlib, os 등
2
데이터 불러오기 및 분리
• feature 숫자형 변경 및 정규화
• training/valid 분리
• feature shape 변경
- 합성곱신경망 (depth 차원추가)
- 일반 신경망 (depth차원 추가x)
• label 형태 변경 (필요 시)
3
모델 생성
• (문제지 활용)
- 모델생성:
- Sequential() / Dense
4
모델 컴파일
• (문제지 / 답지 엮기)
- loss function: 목표함수 정의 (예: mse)
- optimizer: 코칭방법 정의 (예: adam)
5
모델 훈련
• 학습 (문제지 / 답지 활용)
- call back
* early stopping : 조기종료
* tensorboard : 텐서보드 연동
* checkpoint : 중간단계 weight 저장
6
추론
• 평가 (Evaluate)
- 정의된 metric 손실 확인
7
예측
• 예측 검증(특성) 데이터만 활용
8
데이터 정리
• 예측값 비교

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

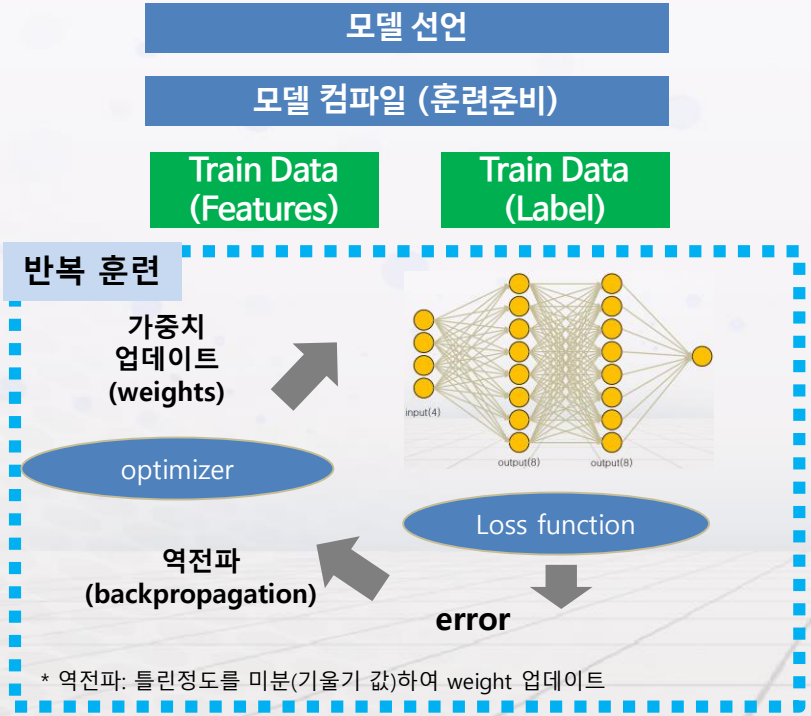
Feature는 숫자형태로
변경 필요!

테스트 데이터

반복 훈련 (feature, label)
loss function!

참조. 딥러닝 작동 원리

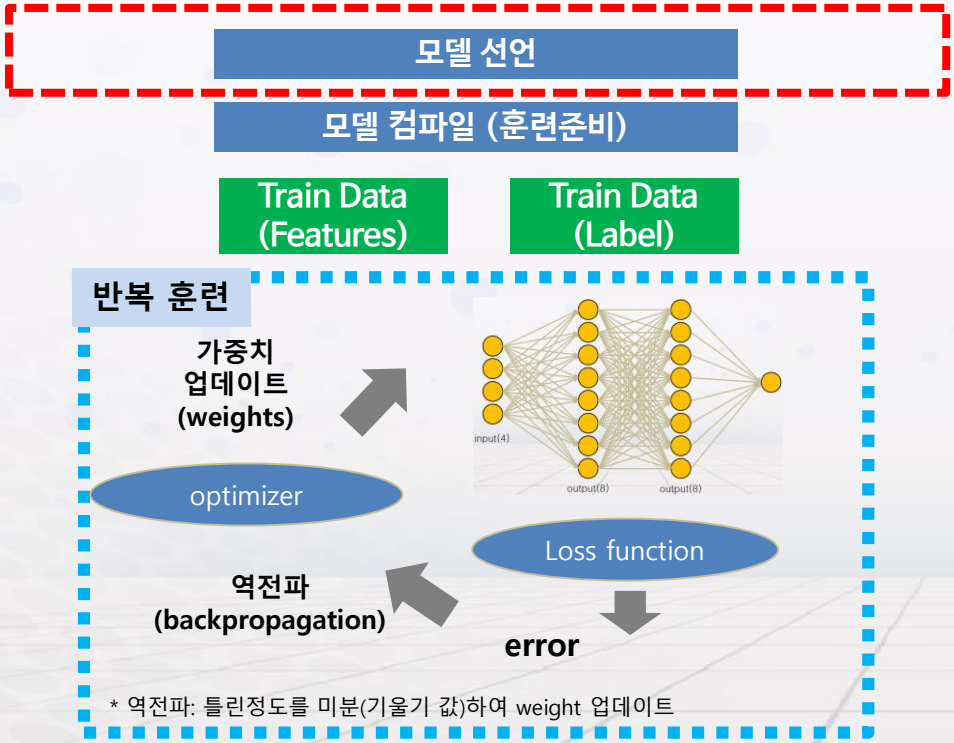
케라스 모델 프로세스



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

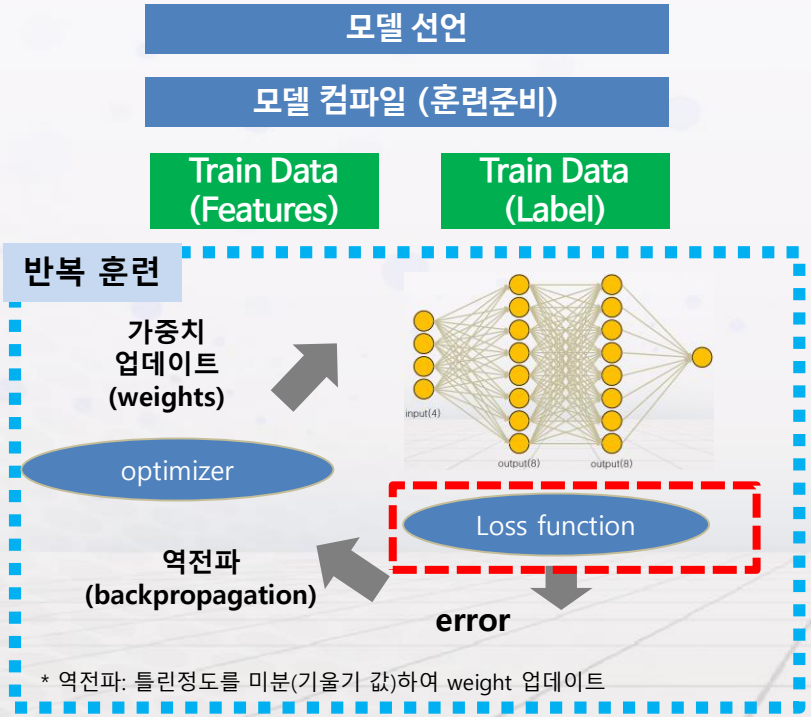


Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	98 MB	0.749	0.921	25,636,712	-
ResNet101	171 MB	0.764	0.928	44,707,176	-
ResNet152	232 MB	0.766	0.931	60,419,944	-
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

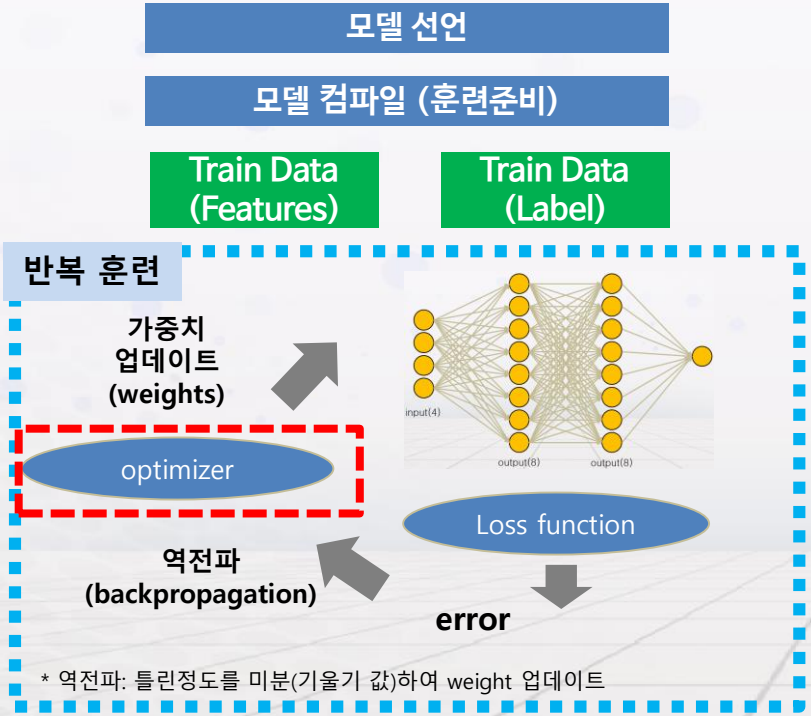


Loss Function (손실함수)	내용
Mean_squared_error	연속 숫자 예측
Mean_absolute_error	
Mean_absolute_percentage_error	
Mean_squared_logarithmic_error	
Squared_hinge	
Hinge	
Categorical_hinge	
Logcosh	
Categorical_crossentropy	멀티 카테고리 예측
Sparse_categorical_crossentropy	
Binary_corssentropy	2개 카테고리 예측

https://keras.io/losses/

참조. 딥러닝 작동 원리

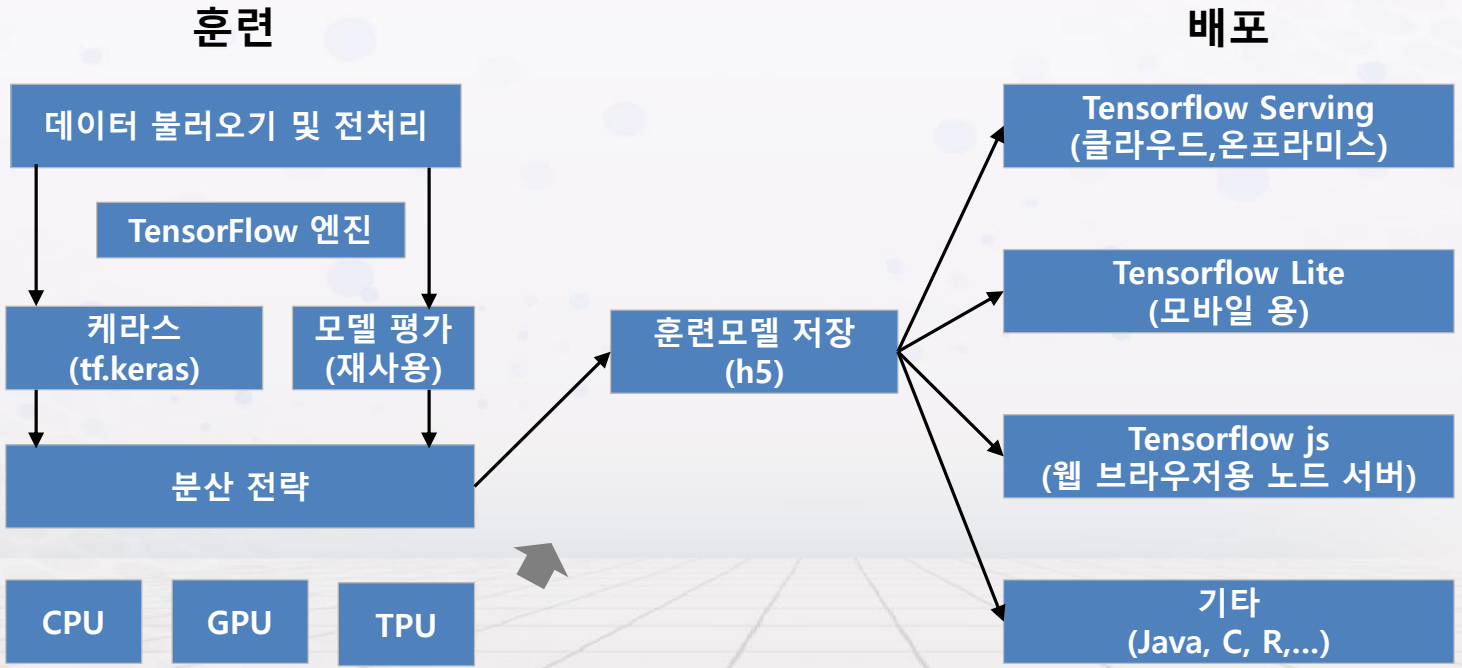
케라스 모델 프로세스



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

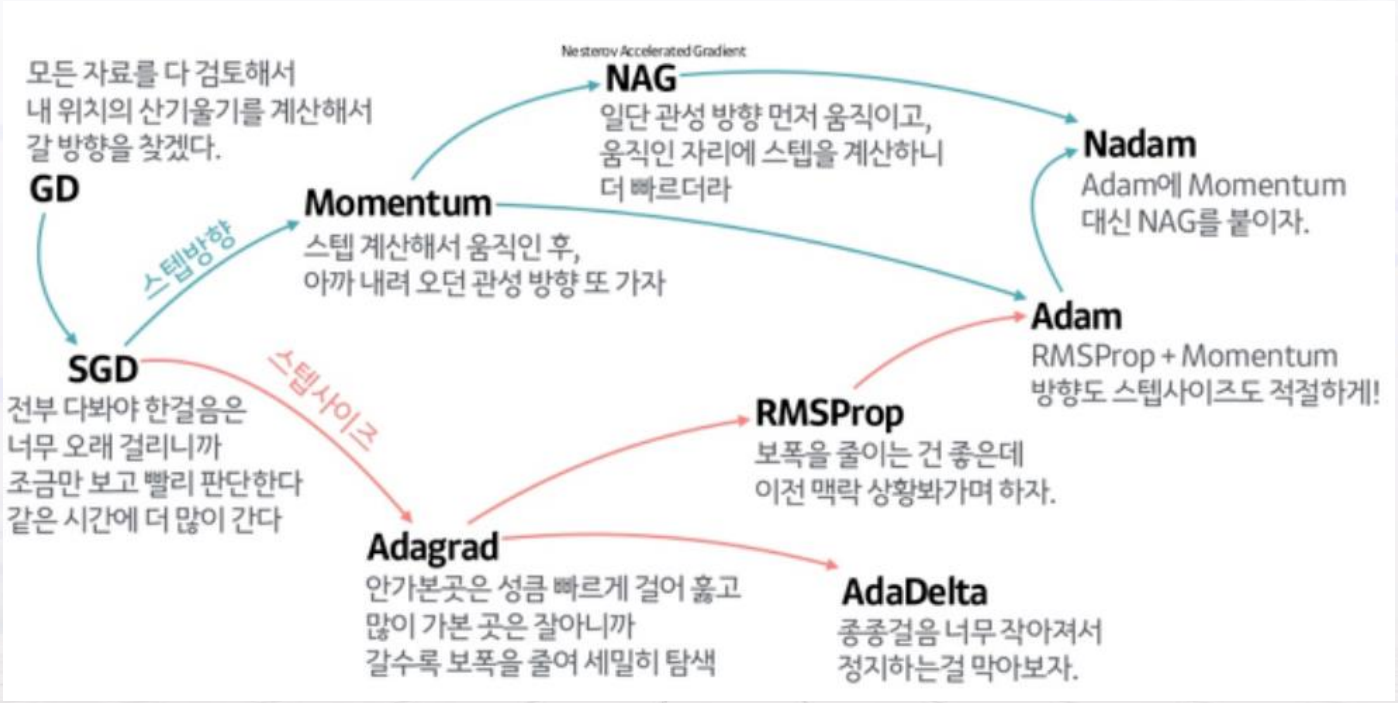
$$60 = w * 30\% + b$$

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

예측 값

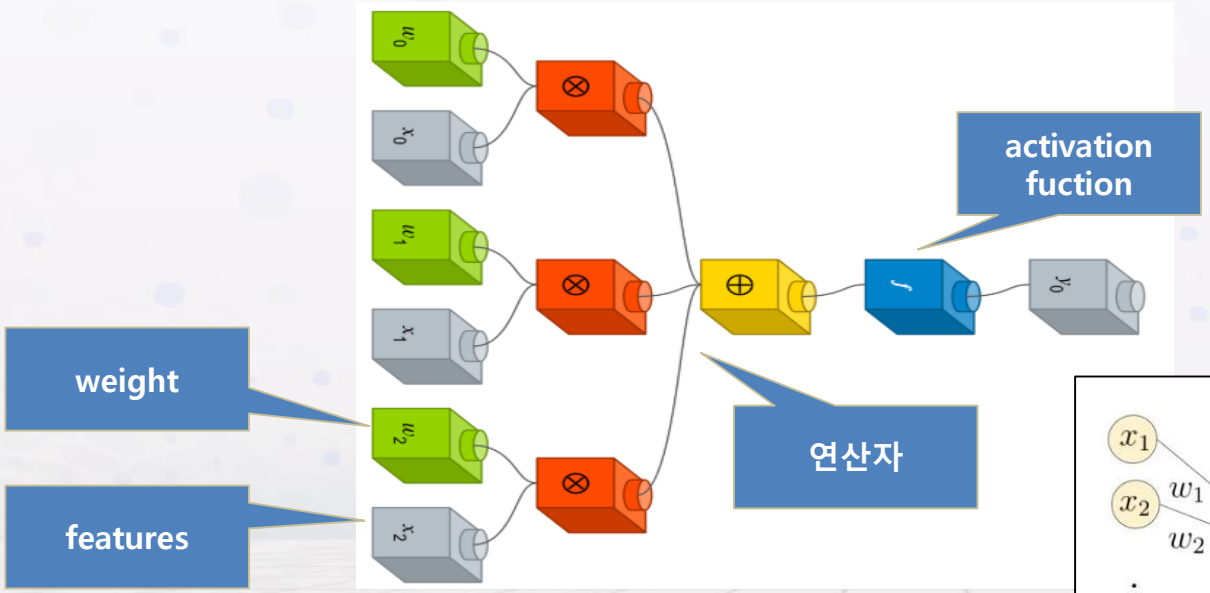
$$\hat{Y} = w * 40\% + b$$

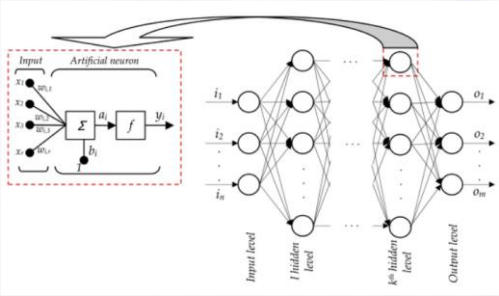
참조. 딥러닝 작동 원리

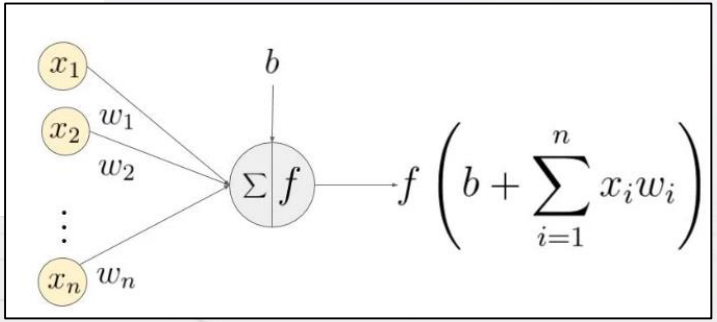

모든 자료를 다 검토해서
내 위치의 산기울기를 계산해서
갈 방향을 찾겠다.
GD
스텝방향
SGD
전부 다봐야 한걸음은
너무 오래 걸리니까
조금만 보고 빨리 판단한다
같은 시간에 더 많이 간다
Momentum
스텝 계산해서 움직인 후,
아까 내려 오던 관성 방향 또 가자
Nesterov Accelerated Gradient
NAG
일단 관성 방향 먼저 움직이고,
움직인 자리에 스텝을 계산하니
더 빠르더라
Nadam
Adam에 Momentum
대신 NAG를 붙이자.
스텝사이즈
Adagrad
안가본곳은 성큼 빠르게 걸어 훑고
많이 가본 곳은 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

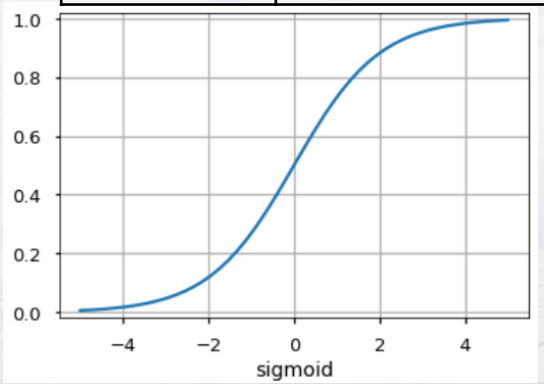
activation 함수

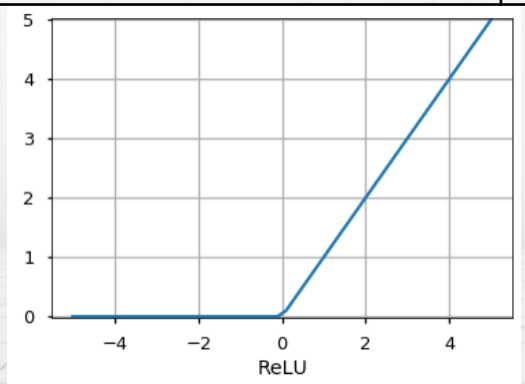
Activation functions on CIFAR-10

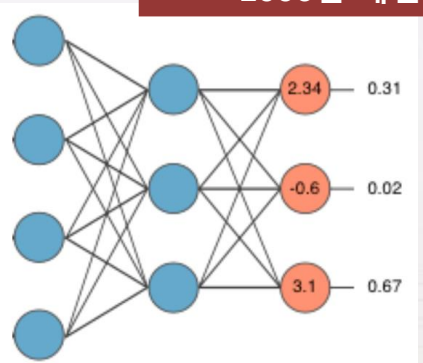
maxout	ReLU	VLReLU	tanh	Sigmoid
93.94	**92.11**	92.97	89.28	n/c
93.78	91.74	92.40	89.48	n/c
–	91.93	**93.09**	–	n/c
91.75	90.63	92.27	**89.82**	n/c
n/c†	90.91	92.43	89.54	n/c

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

Regression 예측 (loss: “mean squared error”)

미래 판매량/거래량이 어떻게 되지?

RELU -> 0~실제값을 출력값으로!

참조. 딥러닝 작동 원리

Binary crossentropy 예측 (loss: “binary crossentropy”)

당뇨병이 걸릴 확률은?

80%

sigmoid -> 0~1 사이 출력

참조. 딥러닝 작동 원리

categorical crossentropy 예측 (loss: “categorical crossentropy”)

카테고리 중에 어떤 카테고리가 맞는지 예측

행복한 표정 70%

화난 표정 20%

슬픈 표정 10%

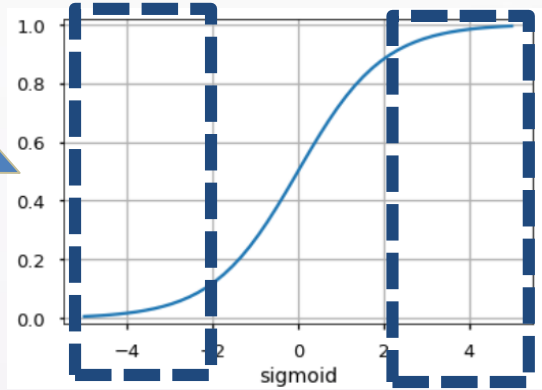
softmax

softmax -> 여러 카테고리의 합이 1이 되도록 0~1 값 출력

참조. 딥러닝 작동 원리

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

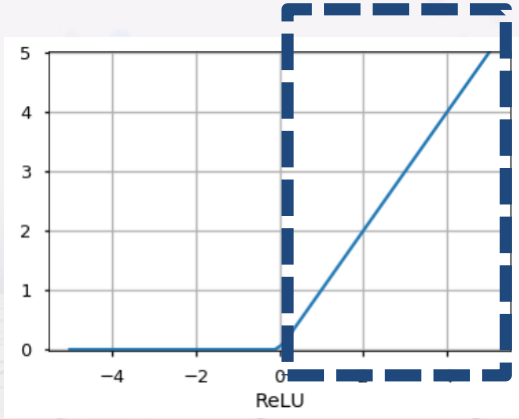
-> Vanishing Gradient
로 인해 학습이 안되는 현상을
Under fitting이라고 함



0~1 사이 값

Rectified Linear Unit

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



F(x) = max(0, x)

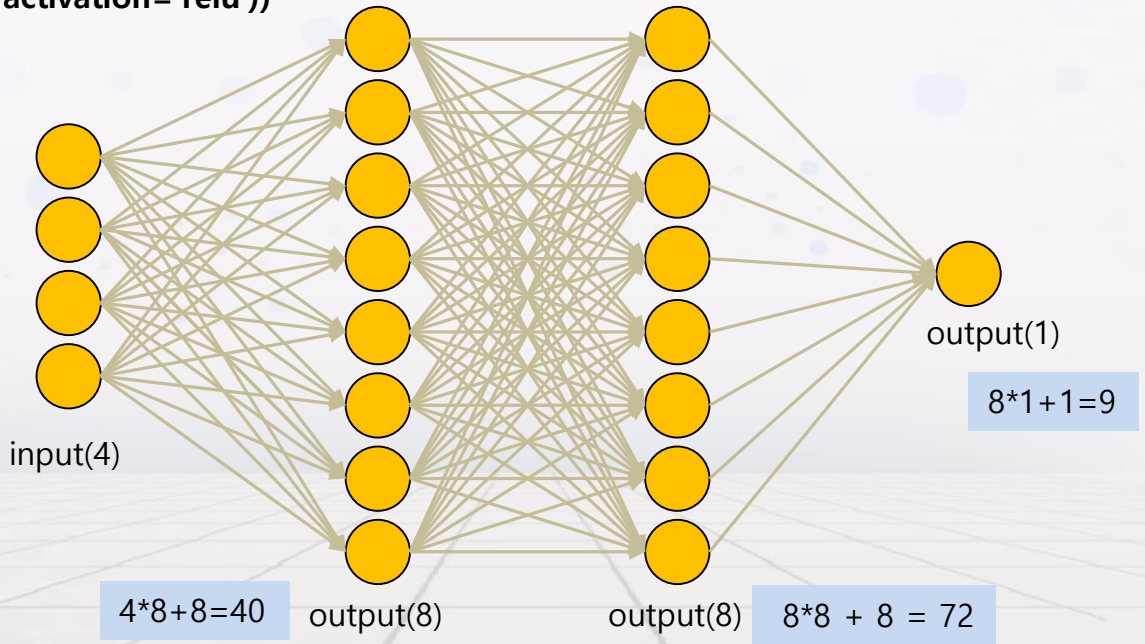
https://en.wikipedia.org/wiki/Rectifier_(neural_networks)

참조. 딥러닝 작동 원리

모델 생성

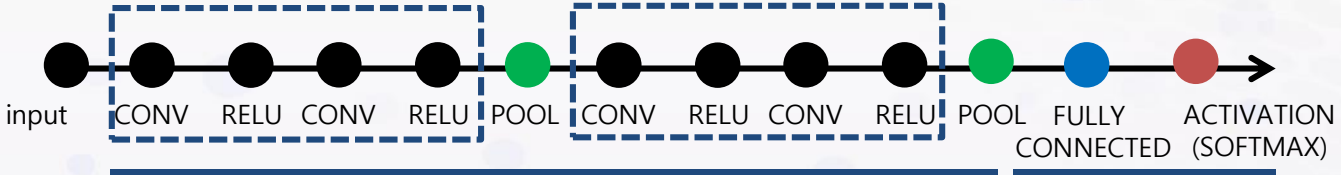
```
model = Sequential()
model.add(Dense(8, activation='relu', input_shape=(len(features),)))
model.add(Dense(8, activation='relu'))
momodel.add(Dense(1, activation='relu'))
```

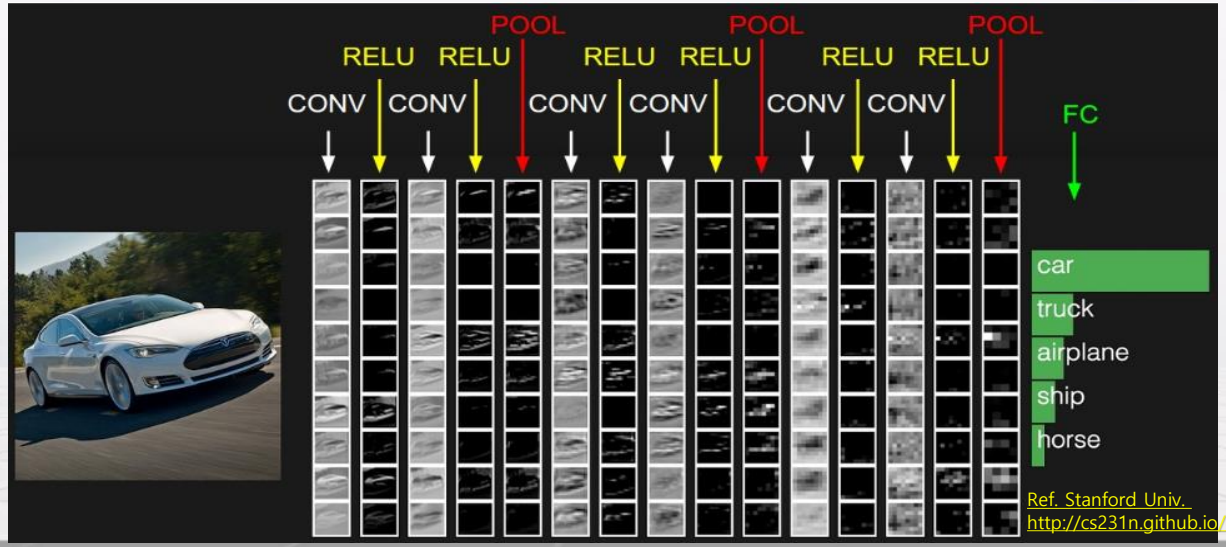
Dense:
Fully Connected Layer



참조. 합성곱 신경망 원리

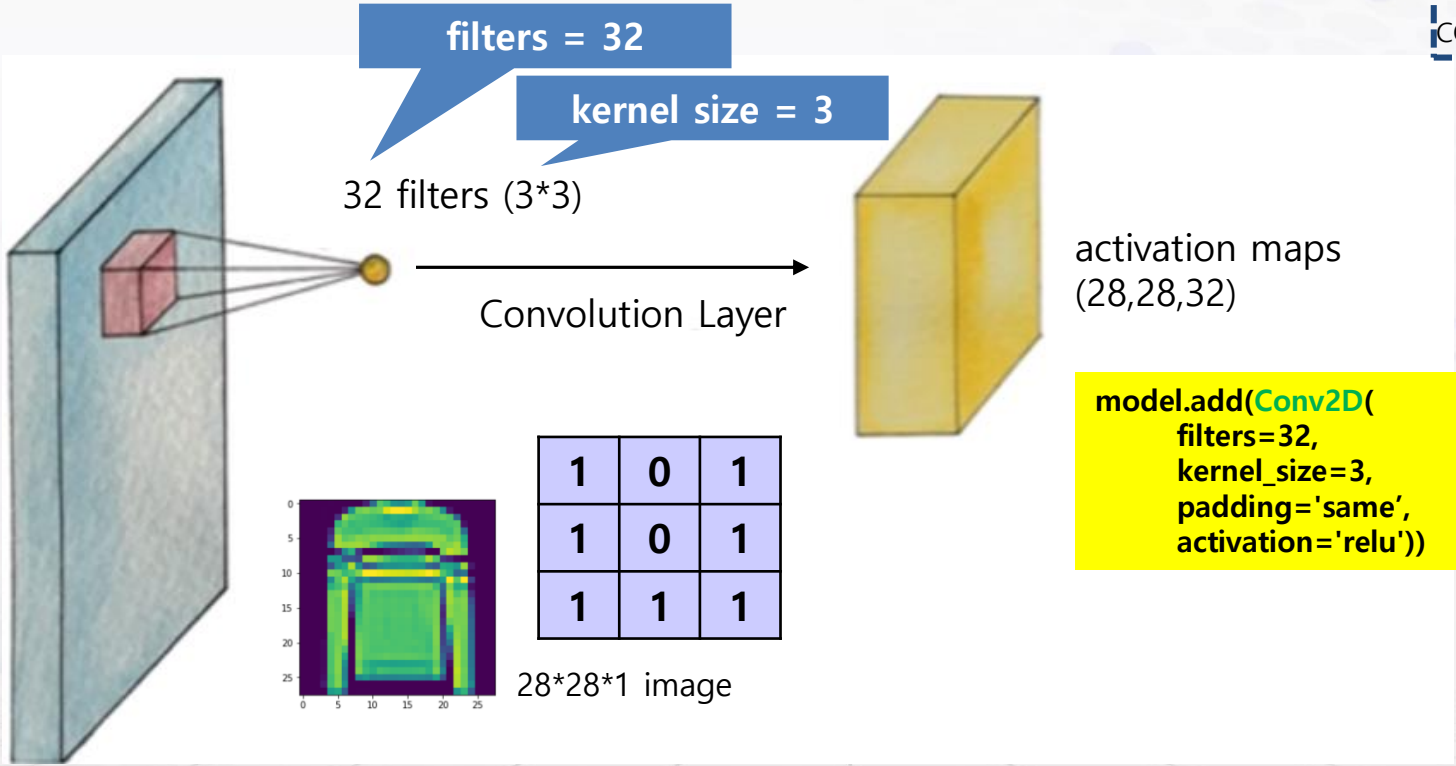
합성곱 신경망 (Classical Architechture)





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

참조. 합성곱 신경망 원리




```
model.add(Conv2D(
        filters=32,
        kernel_size=3,
        padding='same',
        activation='relu'))
```

참조. 합성곱 신경망 원리

합성곱 레이어 (Conv Layer)

필터를 통해 다양한 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

kernel size = 3

Activation Map

32		

```
model.add(Conv2D(
        filters=32,
        kernel_size=3,
        padding='same',
        activation='relu'))
```

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

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

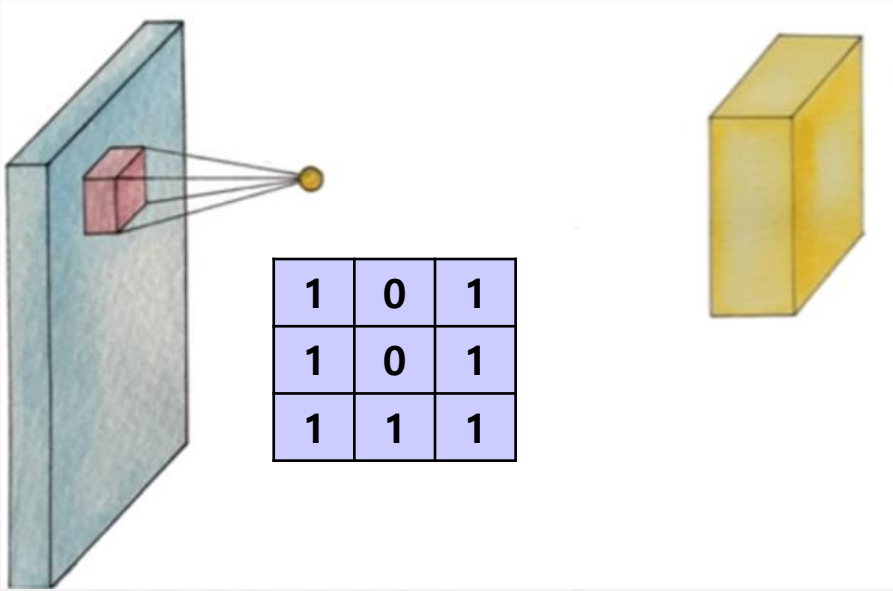
1	0	1
1	0	1
1	1	1

32	56	34
33	49	33
33	39	39

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

출력 사이즈 =
(입력사이즈-필터사이즈) / STRIDE + 1

참조. 합성곱 신경망 원리




```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 14, 14, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 32)          0
_________________________________________________________________
flatten (Flatten)            (None, 1568)              0
_________________________________________________________________
dense (Dense)                (None, 10)                15690
=================================================================
Total params: 25,258
Trainable params: 25,258
Non-trainable params: 0
```

parameters #1 = depth size(1) * kernel size(3) * kernel size(3) * filters(32) + b (32) = 320
parameters #2 = depth size(32) * kernel size(3) * kernel size(3) * filters(32) + b (32) = 9248

참조. 합성곱 신경망 원리

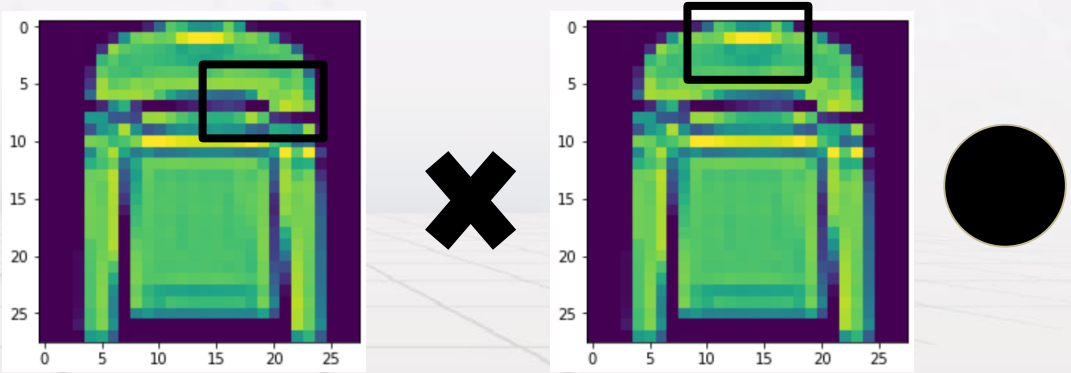
합성곱 레이어 (Conv Layer)

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**



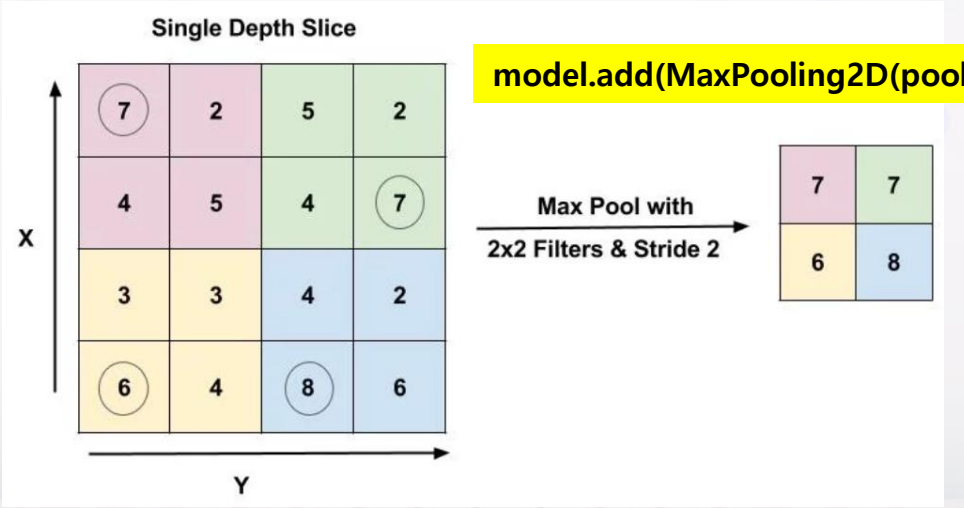
참조. 합성곱 신경망 원리

풀링 레이어 (Pooling Layer)

연속적인 Conv 레이어 사이에 데이터 압축위함

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

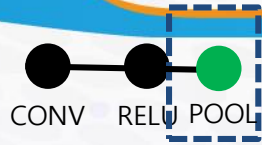
실제 주요 특징만 추출하여
이미지가 약간 틀어져도
인식하게 해줌




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

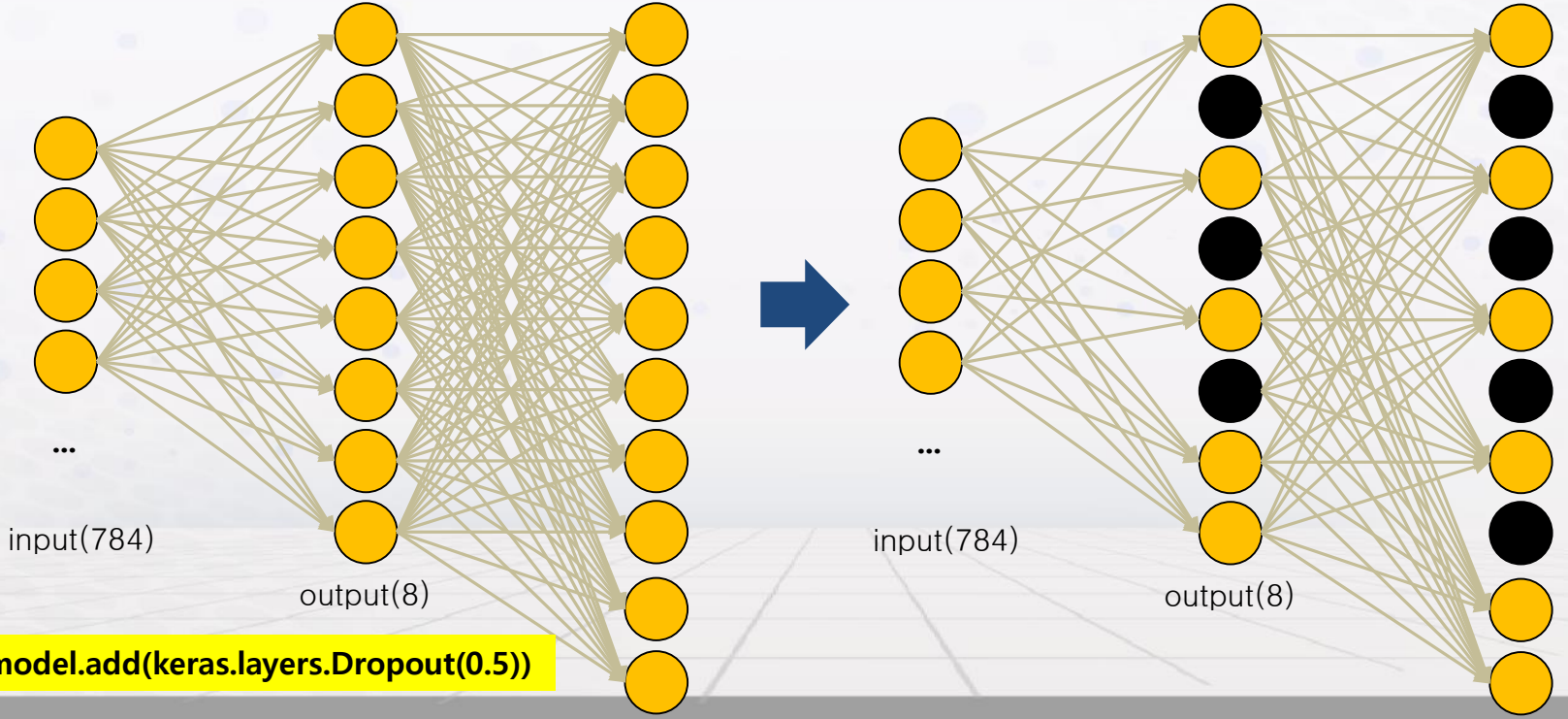
필수는 아님 (선택적 사용)

참조. 합성곱 신경망 원리



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))
```

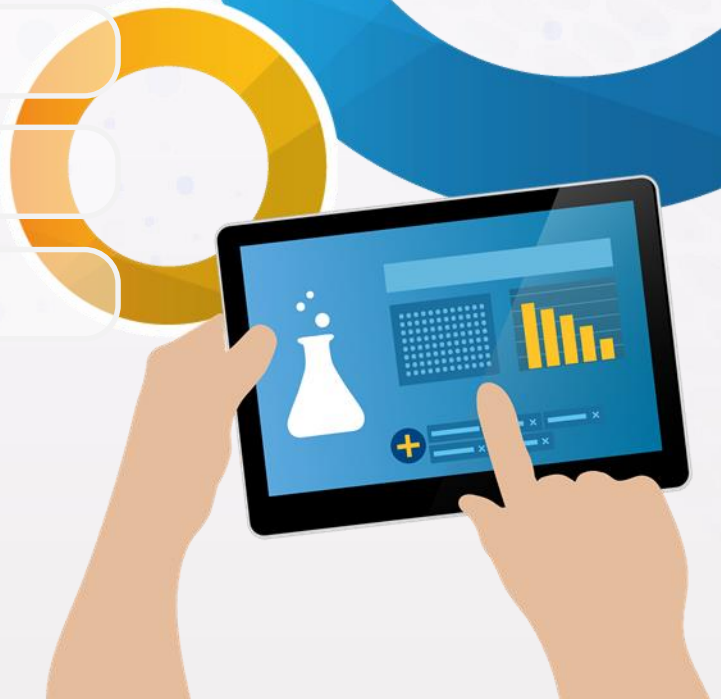
mnist 웹캠연동 실습

김 효 관

교육목표: 이미지를 학습한 후 실시간 영상에서 학습한 내용을 활용하여 예측

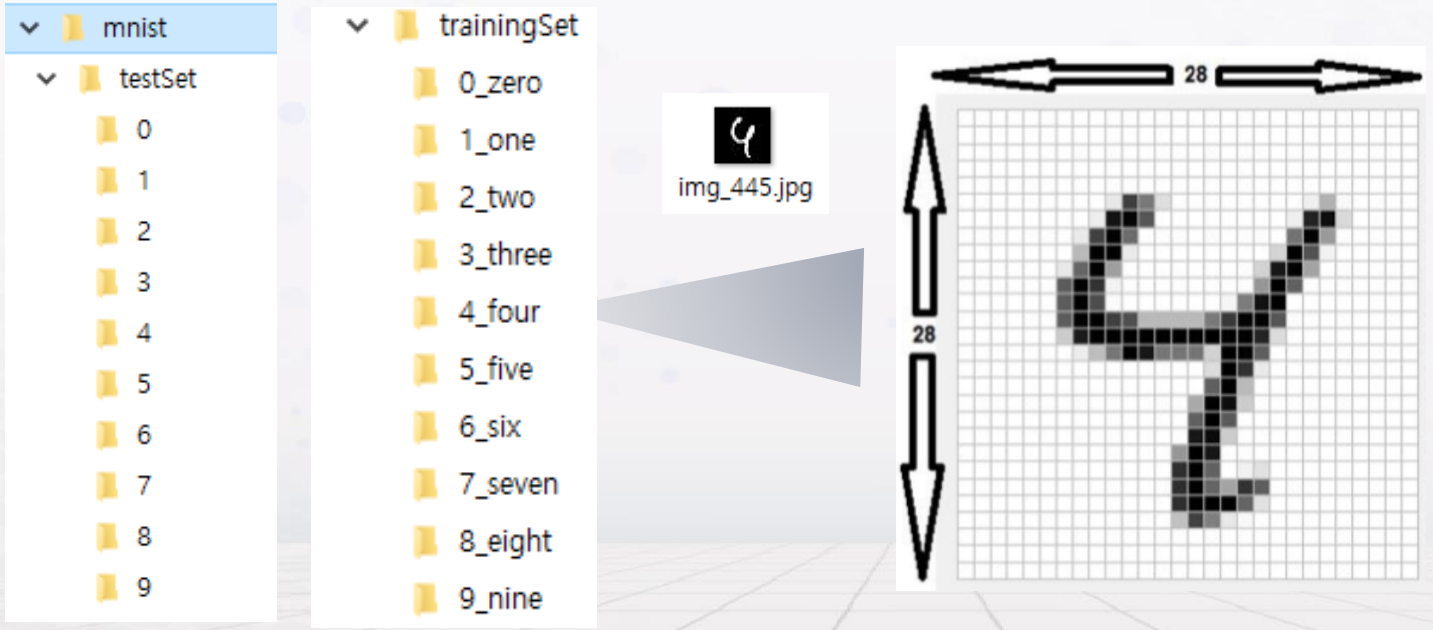
CONTENTS

1. Image Classification 문제
2. 웹캠 연동
3. 핵심정리 및 Q&A



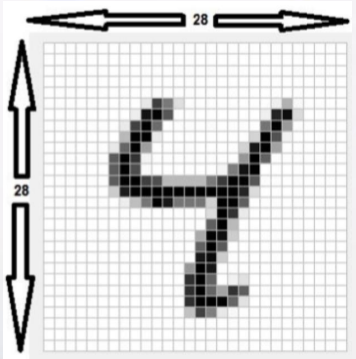
1. Image Classificatino 문제

데이터 설명



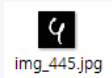
1. Image Classificatino 문제

Image classification

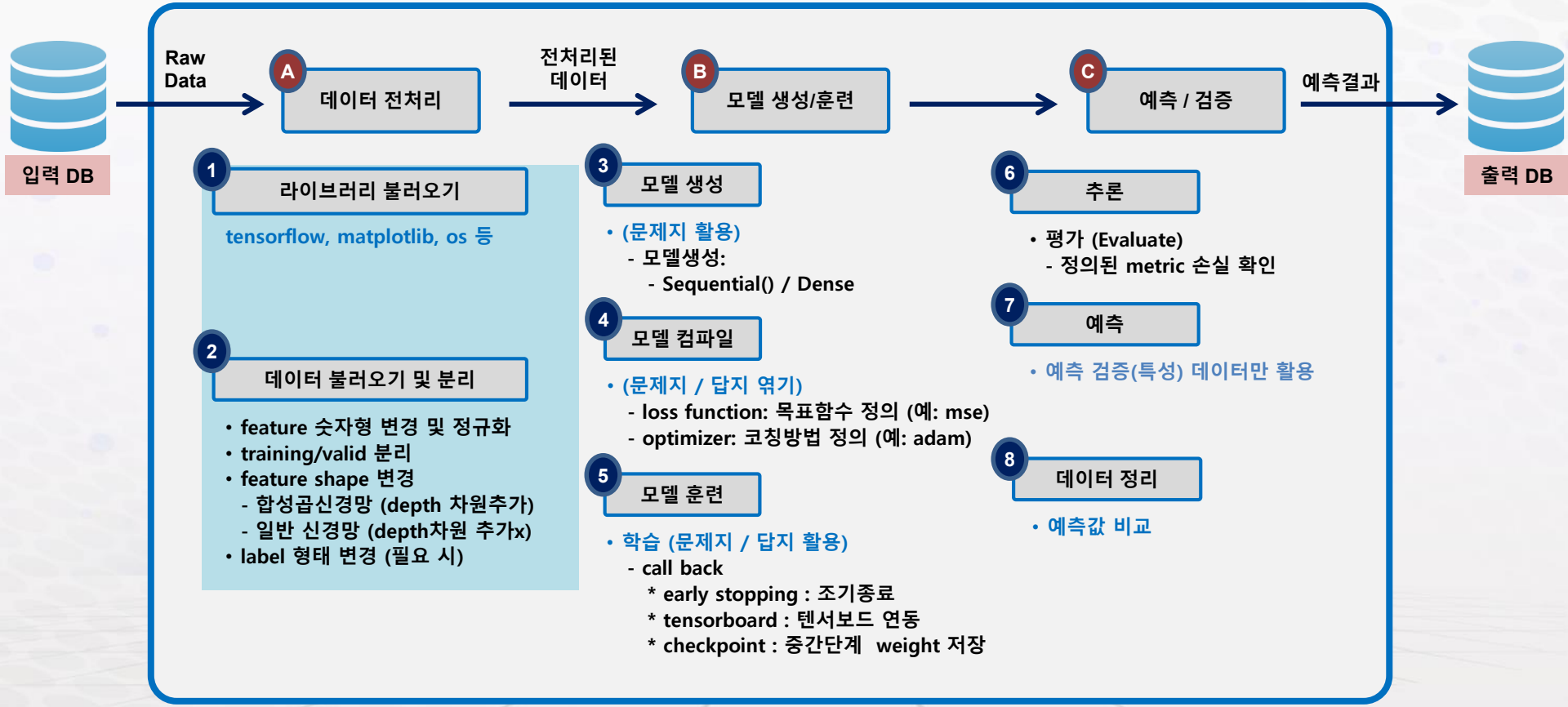


1. Image Classificatino 문제

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	

참조. 딥러닝 작동 원리


입력 DB
Raw Data
A
데이터 전처리
전처리된 데이터
B
모델 생성/훈련
C
예측 / 검증
예측결과
출력 DB
1
라이브러리 불러오기
tensorflow, matplotlib, os 등
2
데이터 불러오기 및 분리
• feature 숫자형 변경 및 정규화
• training/valid 분리
• feature shape 변경
- 합성곱신경망 (depth 차원추가)
- 일반 신경망 (depth차원 추가x)
• label 형태 변경 (필요 시)
3
모델 생성
• (문제지 활용)
- 모델생성:
- Sequential() / Dense
4
모델 컴파일
• (문제지 / 답지 엮기)
- loss function: 목표함수 정의 (예: mse)
- optimizer: 코칭방법 정의 (예: adam)
5
모델 훈련
• 학습 (문제지 / 답지 활용)
- call back
* early stopping : 조기종료
* tensorboard : 텐서보드 연동
* checkpoint : 중간단계 weight 저장
6
추론
• 평가 (Evaluate)
- 정의된 metric 손실 확인
7
예측
• 예측 검증(특성) 데이터만 활용
8
데이터 정리
• 예측값 비교

1. Image Classification 문제

1. 라이브러리 선언

Intel GPU 활용 방법 (PLAID ML)

```
import numpy as np
import os
import time
import mtplotlib.pyplot as plt
%matplotlib inline

# intel gpu 적용
os.environ["KERAS_BACKEND"] = "plaidml.keras.backend"
import keras
# mnist 데이터셋
from keras.datasets import mnist
```

Cuda GPU 활용 방법

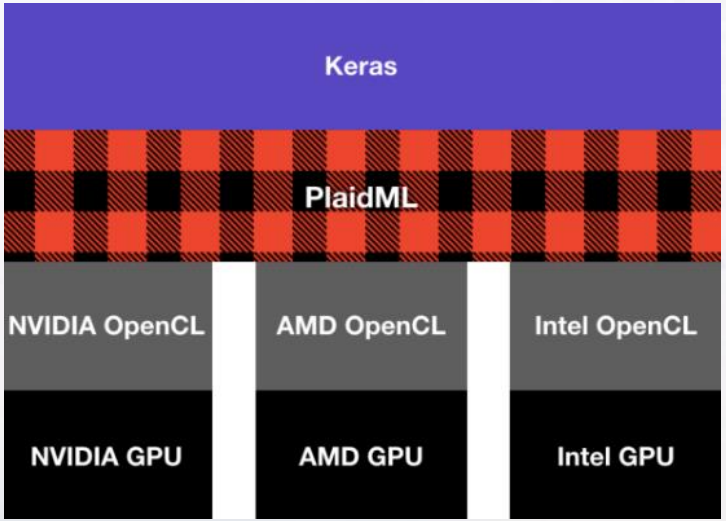
```
import tensorflow as tf
from tensorflow.python.client import device_lib
print(device_lib.list_local_devices())

import os
os.environ["CUDA_VISIBLE_DEVICES"]='1'

# cuda gpu 적용
from tensorflow import keras
# mnist 데이터셋
from tensorflow.keras.datasets import mnist
```

1. Image Classification 문제

1. 라이브러리 선언

일반 CPU 활용

```
import numpy as np
import os
import time
import mtplotlib.pyplot as plt
%matplotlib inline

from tensorflow import keras
```



mnist 압축파일을 해제 후
x_train (이미지), y_train (정답) 변수에 저장하세요.

- 이미지 컬러: GRAYSCALE
- 이미지 사이즈: 28*28
- 정답: 0,1,2,~

1. Image Classification 문제

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

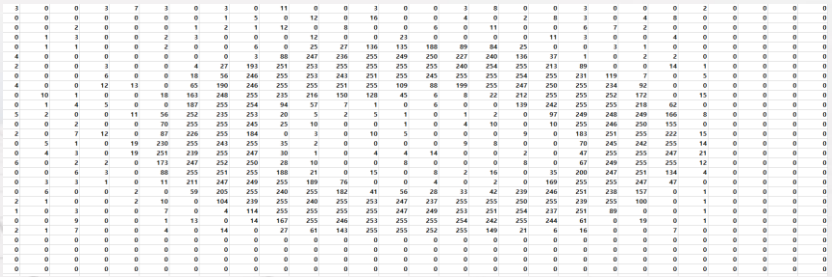
```
1 img
```

```
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

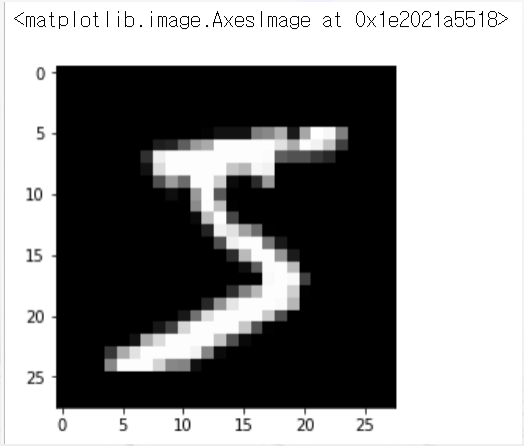
```
img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
import pandas as pd

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



1. Image Classification 문제

2. 데이터 불러오기

```
(x_train, y_train), (x_test, y_test) = mnist.load_data()

plt.imshow(x_train[0], cmap="gray")
```

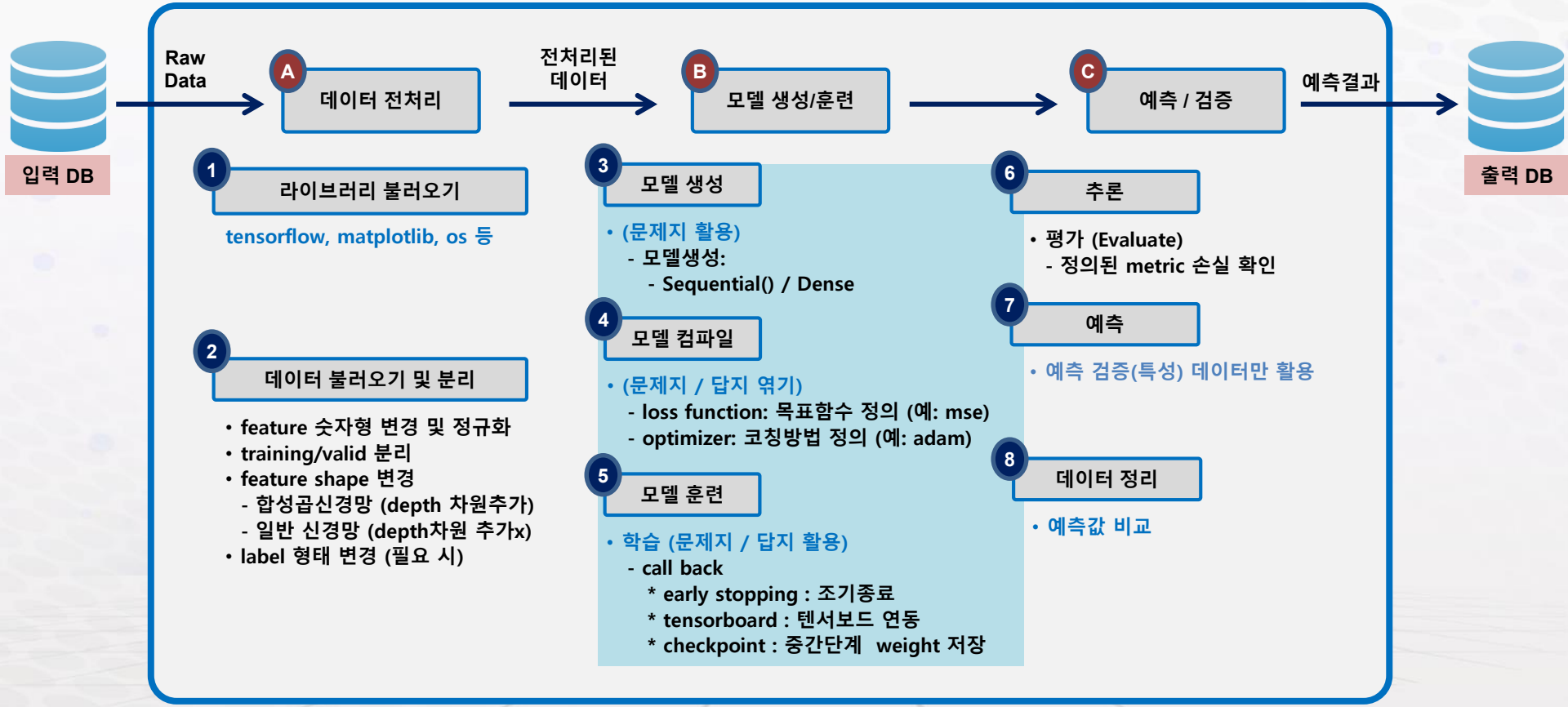


1. Image Classification 문제

2. 데이터 불러오기

```
from tensorflow.keras.utils import to_categorical

IMG_SIZE = 28

# 합성곱 신경망 depth 추가
x_train = x_train.reshape(len(x_train),IMG_SIZE,IMG_SIZE,1)
x_test = x_test.reshape(len(x_test),IMG_SIZE,IMG_SIZE,1)

# 원핫인코딩 적용
y_train_one = to_categorical(y_train)
y_test_one = to_categorical(y_test)
```

참조. 딥러닝 작동 원리


입력 DB
Raw Data
A
데이터 전처리
전처리된 데이터
B
모델 생성/훈련
C
예측 / 검증
예측결과
출력 DB
1
라이브러리 불러오기
tensorflow, matplotlib, os 등
2
데이터 불러오기 및 분리
• feature 숫자형 변경 및 정규화
• training/valid 분리
• feature shape 변경
- 합성곱신경망 (depth 차원추가)
- 일반 신경망 (depth차원 추가x)
• label 형태 변경 (필요 시)
3
모델 생성
• (문제지 활용)
- 모델생성:
- Sequential() / Dense
4
모델 컴파일
• (문제지 / 답지 엮기)
- loss function: 목표함수 정의 (예: mse)
- optimizer: 코칭방법 정의 (예: adam)
5
모델 훈련
• 학습 (문제지 / 답지 활용)
- call back
* early stopping : 조기종료
* tensorboard : 텐서보드 연동
* checkpoint : 중간단계 weight 저장
6
추론
• 평가 (Evaluate)
- 정의된 metric 손실 확인
7
예측
• 예측 검증(특성) 데이터만 활용
8
데이터 정리
• 예측값 비교

1. Image Classification 문제

3. 모델 생성

```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Flatten, Dropout, MaxPool2D
import numpy as np

modelDim = x_train[0].shape
nclass = x_train
nclasses = len(np.unique(y_train))

model = keras.Sequential()
model.add(Conv2D(filters=32, kernel_size=2, padding='same', activation="relu",
input_shape=modelDim))
model.add(MaxPool2D(pool_size=2))
model.add(Dropout(0.3))
model.add(Conv2D(filters=32, kernel_size=2, padding='same', activation="relu"))
model.add(MaxPool2D(pool_size=2))
model.add(Dropout(0.3))
model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(units=nclasses, activation="softmax"))
model.summary()
```

Layer (type)	Output Shape	Param #
conv2d_5 (Conv2D)	(None, 28, 28, 32)	160
max_pooling2d_5 (MaxPooling2	(None, 14, 14, 32)	0
dropout_7 (Dropout)	(None, 14, 14, 32)	0
conv2d_6 (Conv2D)	(None, 14, 14, 32)	4128
max_pooling2d_6 (MaxPooling2	(None, 7, 7, 32)	0
dropout_8 (Dropout)	(None, 7, 7, 32)	0
flatten_3 (Flatten)	(None, 1568)	0
dropout_9 (Dropout)	(None, 1568)	0
dense_3 (Dense)	(None, 10)	15690

Total params: 19,978
Trainable params: 19,978
Non-trainable params: 0

1. Image Classification 문제

4. 모델 컴파일

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

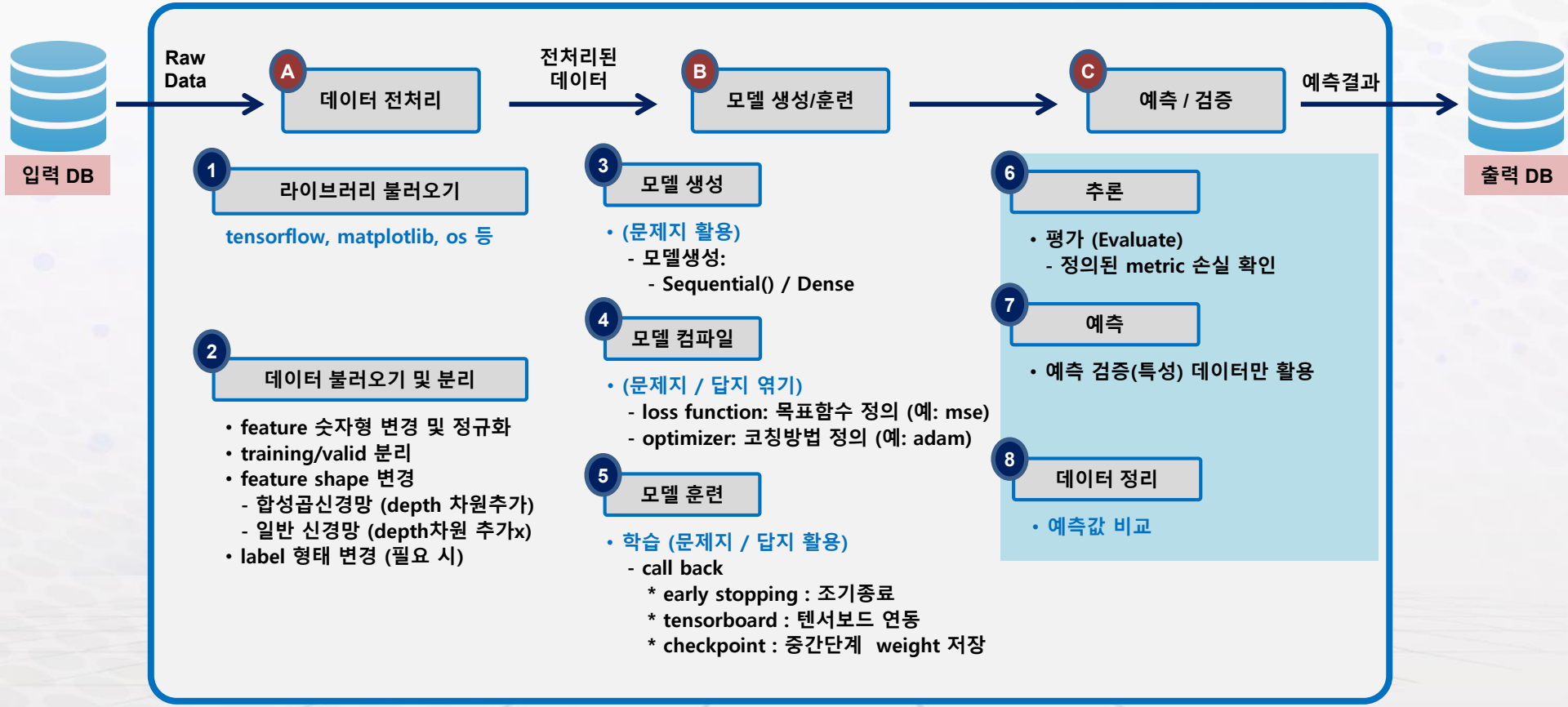
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 28, 28, 32)        160
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
dropout_7 (Dropout)          (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 14, 14, 32)        4128
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 7, 7, 32)          0
_________________________________________________________________
dropout_8 (Dropout)          (None, 7, 7, 32)          0
_________________________________________________________________
flatten_3 (Flatten)          (None, 1568)              0
_________________________________________________________________
dropout_9 (Dropout)          (None, 1568)              0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                15690
=================================================================
Total params: 19,978
Trainable params: 19,978
Non-trainable params: 0
_________________________________________________________________
```

1. Image Classification 문제

5. 모델 훈련

```
model.fit(x_train, y_train_one, epochs=10,
validation_split=0.2)
```

참조. 딥러닝 작동 원리


입력 DB
Raw Data
A
데이터 전처리
전처리된 데이터
B
모델 생성/훈련
C
예측 / 검증
예측결과
출력 DB
1
라이브러리 불러오기
tensorflow, matplotlib, os 등
2
데이터 불러오기 및 분리
• feature 숫자형 변경 및 정규화
• training/valid 분리
• feature shape 변경
- 합성곱신경망 (depth 차원추가)
- 일반 신경망 (depth차원 추가x)
• label 형태 변경 (필요 시)
3
모델 생성
• (문제지 활용)
- 모델생성:
- Sequential() / Dense
4
모델 컴파일
• (문제지 / 답지 엮기)
- loss function: 목표함수 정의 (예: mse)
- optimizer: 코칭방법 정의 (예: adam)
5
모델 훈련
• 학습 (문제지 / 답지 활용)
- call back
* early stopping : 조기종료
* tensorboard : 텐서보드 연동
* checkpoint : 중간단계 weight 저장
6
추론
• 평가 (Evaluate)
- 정의된 metric 손실 확인
7
예측
• 예측 검증(특성) 데이터만 활용
8
데이터 정리
• 예측값 비교

1. Image Classification 문제

6. 모델 추론

model.evaluate(x_test, y_test_one)

```
1  model.evaluate(x_test, y_test_one)
```

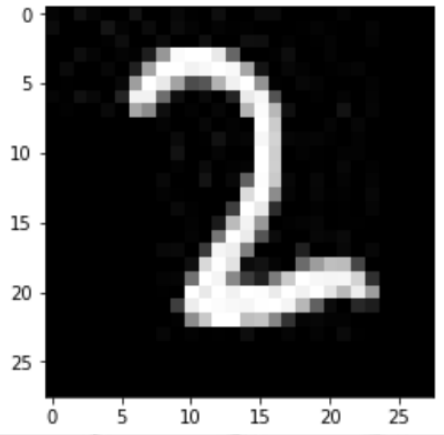
```
10000/10000 [==============================] - 7s 690us/step
[0.06331238393485546, 0.9813]
```

1. Image Classification 문제

7. 모델 예측

```
import cv2

testimg = cv2.imread("../images/mnist/trainingSet/2_two/img_10247.jpg",
cv2.IMREAD_GRAYSCALE)

import matplotlib.pyplot as plt
%matplotlib inline

plt.imshow(testimg, cmap="gray")

refinedImg  = testimg.reshape(1,28,28,1)

answer = np.argmax(model.predict(refinedImg))
```

```
<matplotlib.image.AxesImage at 0x1432d1f1f60>
```



1. Image Classification 문제

모델 저장

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")
```

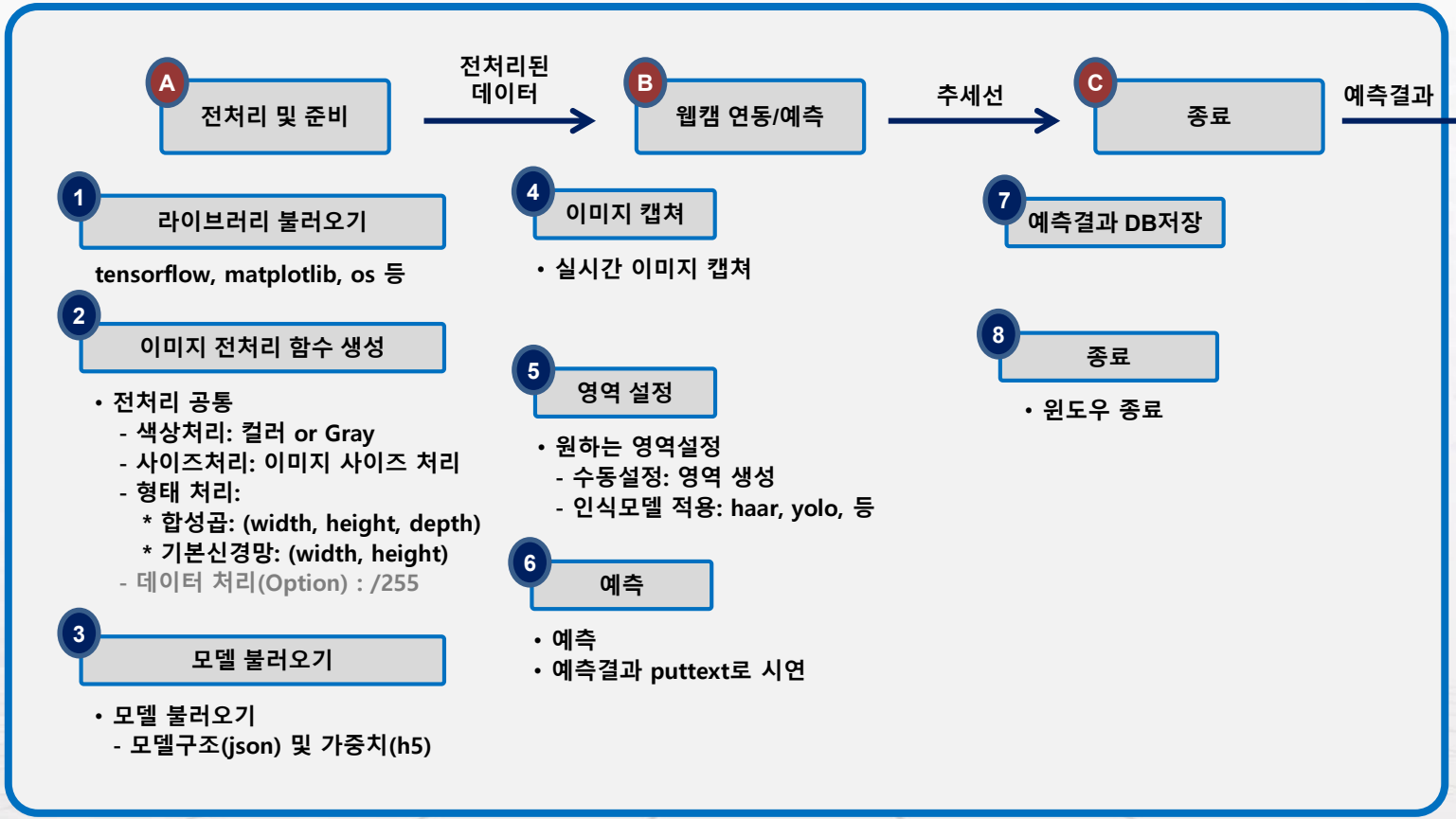
역전파를 통한 업데이트된 가중치 저장

```
# 모델 불러오기
from tensorflow.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 tensorflow.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])
```

참조. 웹캠연동 프로세스


A
전처리 및 준비
전처리된 데이터
B
웹캠 연동/예측
추세선
C
종료
예측결과
출력 DB
1
라이브러리 불러오기
tensorflow, matplotlib, os 등
2
이미지 전처리 함수 생성
• 전처리 공통
- 색상처리: 컬러 or Gray
- 사이즈처리: 이미지 사이즈 처리
- 형태 처리:
* 합성곱: (width, height, depth)
* 기본신경망: (width, height)
- 데이터 처리(Option) : /255
3
모델 불러오기
• 모델 불러오기
- 모델구조(json) 및 가중치(h5)
4
이미지 캡쳐
• 실시간 이미지 캡쳐
5
영역 설정
• 원하는 영역설정
- 수동설정: 영역 생성
- 인식모델 적용: haar, yolo, 등
6
예측
• 예측
• 예측결과 puttext로 시연
7
예측결과 DB저장
8
종료
• 윈도우 종료

2. 웹캠 연동

1. 라이브러리 선언

```
import tensorflow as tf
import cv2
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

2. 웹캠 연동

2. 이미지 전처리 수행

```
# 인풋 이미지를 불러와서 (28,28,1 형태로 변경, 합성곱 신경망 활용)
def process(img_input):
    # 이미지 사이즈 변경
    IMG_SIZE = 28
    # 그레이컬러 변환 및 사이즈 조절
    gray = cv2.cvtColor(img_input, cv2.COLOR_BGR2GRAY)
    gray = cv2.resize(gray, (IMG_SIZE, IMG_SIZE))
    # 합성곱 신경망 적용을 위한 설정
    out_img = gray.reshape(IMG_SIZE,IMG_SIZE,1)
    return out_img
```

2. 웹캠 연동

3. 모델 불러오기

```
import json
from tensorflow.keras.models import model_from_json

# 모델 구조 불러오기
with open( ' model_mnist2.json ' ,  ' r ' ) as json_file:
    loaded_model = model_from_json(json_file.read())
loaded_model.summary()

# 모델 가중치 불러오기
loaded_model.load_weights("./model_mnist.h5")
```

2. 웹캠 연동

4. 웹캠연동

```
cap = cv2.VideoCapture(0)
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
CAP_SIZE = 150
while(True):
    ret, img_color = cap.read()
    if ret == False:
        break;

    cv2.imshow('bgr', img_color)
    key = cv2.waitKey(33)

    if key==27: # esc key
        cap.release()
        cv2.destroyAllWindows()
    elif key==32: # space bar key
        cv2.imwrite("d:/test_capture.jpg", img_color )
cap.release()
cv2.destroyAllWindows()
```

30 fps 를 적용 시 (초당 30프레임 업데이트 시)
33 설정하면 됨.
* 0은 무한 대기

30fps -> 30 frame : 1000ms -> 1frame : 33ms
이미지 하나를 33ms 마다 업데이트 해야함

*** NTSC**는 미국, 캐나다, 대한민국 등에서 널리 사용하는 아날로그 텔레비전 방식이다.

2. 웹캠 연동

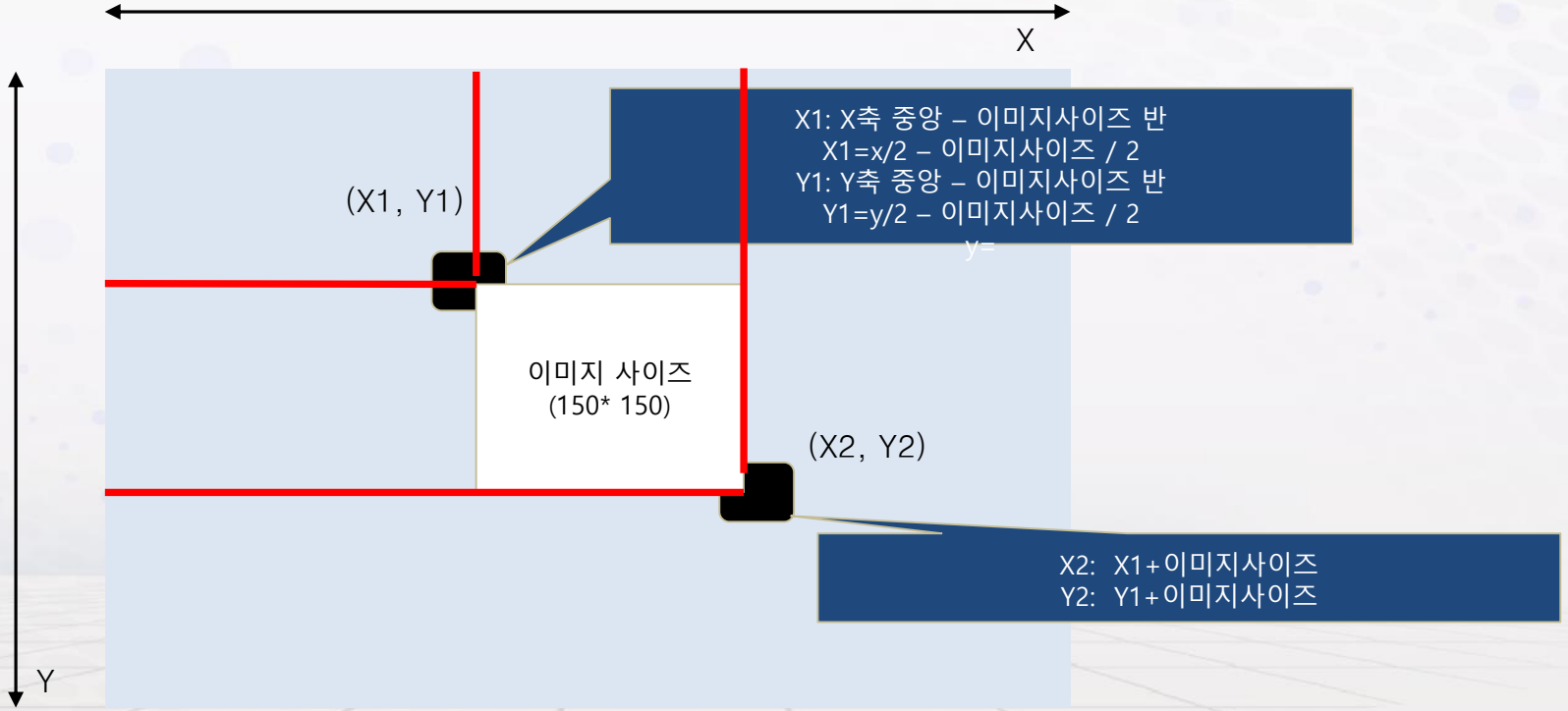
4. 웹캠연동

```
cap = cv2.VideoCapture(0)
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
CAP_SIZE = 150
while(True):
    ret, img_color = cap.read()
    if ret == False:
        break;
    x1= int(width/2-CAP_SIZE/2)
    y1=int(height/2-CAP_SIZE/2)
    x2= x1+CAP_SIZE
    y2= y1+CAP_SIZE
    cv2.rectangle(img_color, ( x1,y1 ), ( x2,y2 ),  (0, 0, 255), 3)
    cv2.imshow('bgr', img_color)

    img_roi = img_color[y1:y1+CAP_SIZE, x1:x1+CAP_SIZE]
    key = cv2.waitKey(33)
    # 코드 Here
cap.release()
cv2.destroyAllWindows()
```

```
if key==27: # esc key
    cap.release()
    cv2.destroyAllWindows()
elif key==32: # space bar key
    try:
        target_img = process(img_roi)
        p_value = loaded_model.predict(target_img)
        p_value2 = np.argmax(p_value)
        print(p_value2)
        cv2.imwrite("d:/test_capture.jpg", img_roi)
    except Exception as e:
        print(e)
```

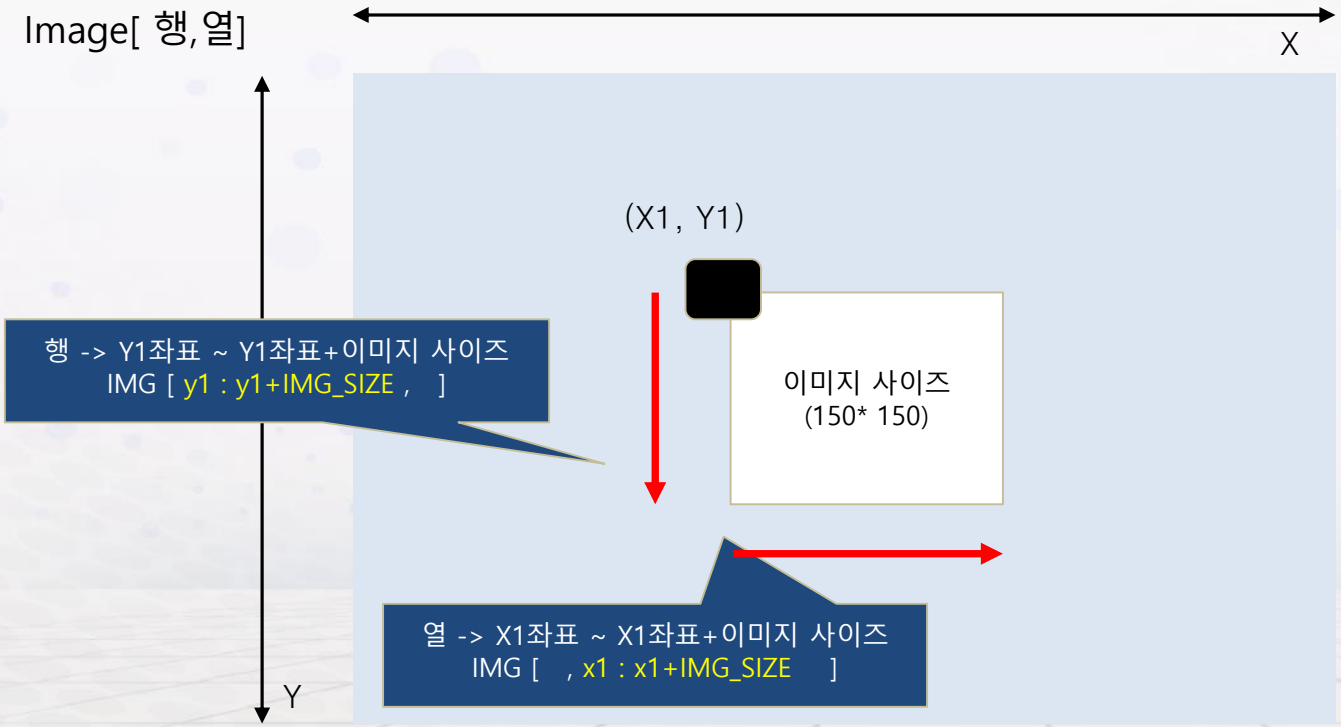
2. 웹캠 연동

4. 웹캠연동 (사각형 그리기 Tip)



2. 웹캠 연동

4. 웹캠연동 (이미지 영역)



참조. 파이썬 주요 키 입력

파이썬 주요 키 입력 (event key code)

순번	숫자	키입력
1	**27**	ESC 키
2	**26**	Ctrl + Z
3	**24**	Ctrl + X
4	**3**	Ctrl + C
5	**0**	아무키
6	**32**	스페이스바
7	**q**	q

https://keycode.info/

참조. 도형 삽입

참조 (이미지 위에 사각형 삽입)

- 사각형 그리기: cv2.rectangle(“이미지변수명”, (시작x축, 시작y축), 색상, 색상채널, 굵기)

```
import cv2

# 이미지 불러오기 및 크기 조절
sampleImg = cv2.imread("../images/it_show.jpg")
sampleImg = cv2.resize(src=sampleImg, dsize=(256,256))

# 불러온 이미지 내 사각형 그리기
rectImg = cv2.rectangle(sampleImg, (10,10), (200,200), (255,0,0))

cv2.imshow("img_show", rectImg)

k = cv2.waitKey(0)

if k==27:
    cv2.destroyAllWindows()
else:
    cv2,destroyAllWindows()
```

(10,10) 좌측상단 기준 10, 10 에서 시작
(200,200) x,y 축 길이
(255,0,0) :BGR 컬러코드

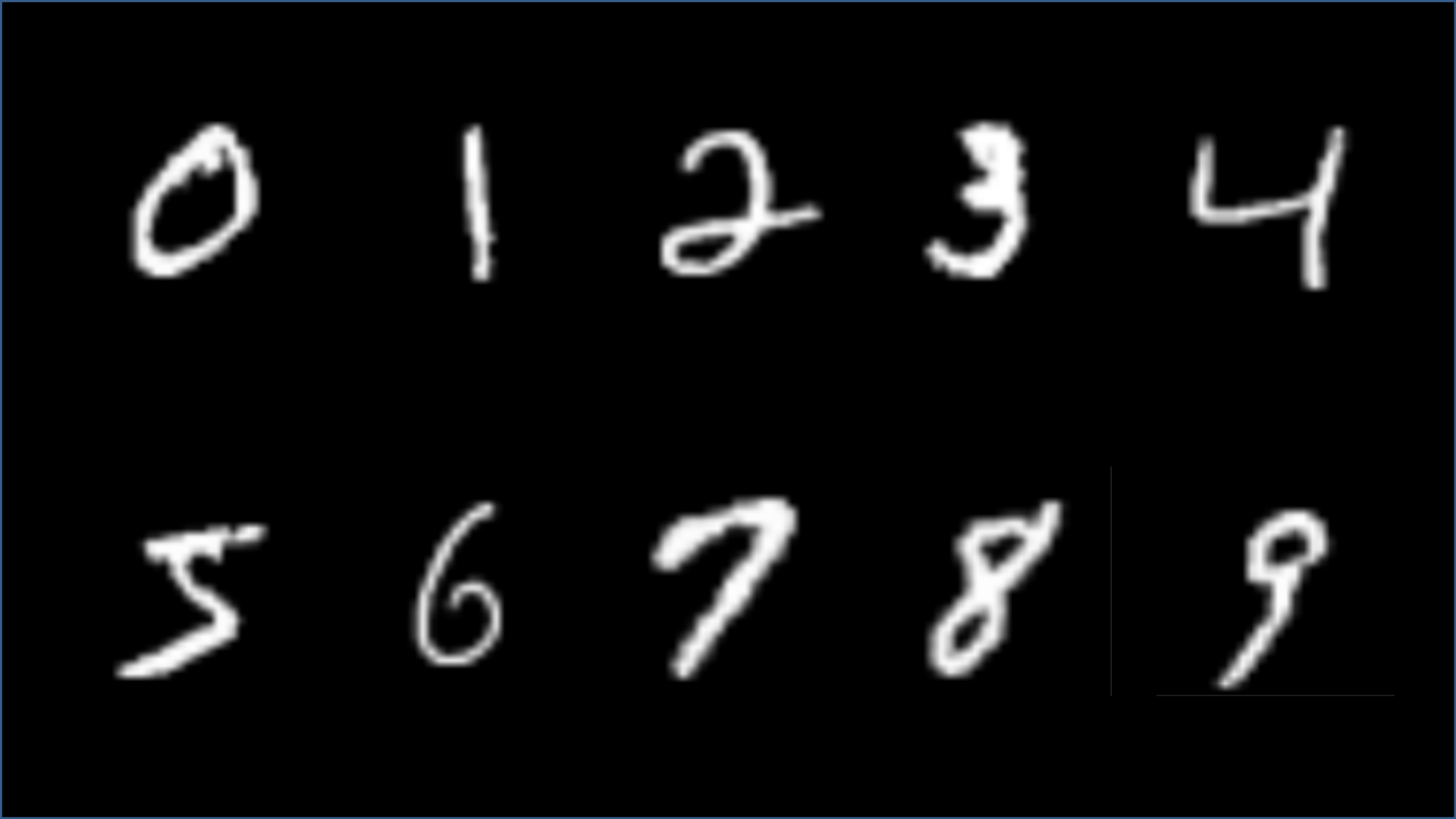
참조. 글자 삽입

참조 (이미지 위에 글자 삽입)

- 글자 삽입: cv2.putText(“이미지변수명”, (시작x축, 시작y축), 폰트, 크기, 색상채널, 굵기)

```
import cv2

# 이미지 불러오기 및 크기 조절
sampleImg = cv2.imread("../images/it_show.jpg")
sampleImg = cv2.resize(src=sampleImg, dsize=(256,256))

# 불러온 이미지 내 사각형 그리기
rectImg = cv2.rectangle(sampleImg, (10,10), (200,200), (255,0,0))
cv2.putText(rectImg,"haiteam",(0,100), cv2.FONT_HERSHEY_SIMPLEX,1,(0,255,0))

cv2.imshow("img_show", rectImg)

k = cv2.waitKey(0)

if k==27:
    cv2.destroyAllWindows()
else:
    cv2,destroyAllWindows()
```

(0,100) 좌측상단 기준 10, 10 에서 시작
cv2.FONT_HER... : 폰트속성
(0,255,0) :BGR 컬러코드


0 1 2 3 4
5 6 7 8 9

제공된 mnist 이미지 자료를 활용하여
실습해 봅니다.

이후, mnist fashion 자료를 활용하여
학습한 내용을 응용하여 구현하세요

3. 핵심정리 및 Q&A

기억합시다

1. 합성곱 신경망 사용법을 재학습 합니다.
2. 웹캠 연동 시 예측모델과 연동 포인트를 정확히 이해합니다.

감사합니다.