mnist 웹캠연동 실습

김 효 관

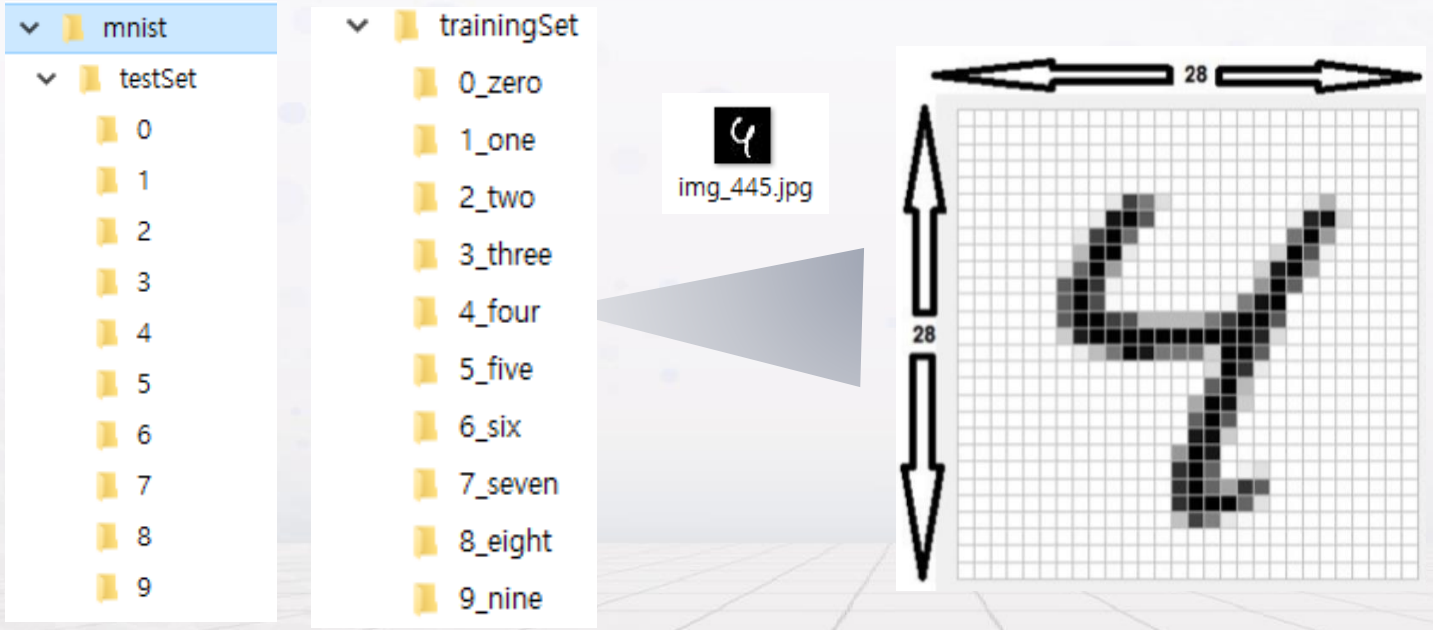
교육목표: 이미지를 학습한 후 실시간 영상에서 학습한 내용을 활용하여 예측

CONTENTS

1. mnist 코어모델 생성
2. 웹캠 연동
3. 핵심정리 및 Q&A

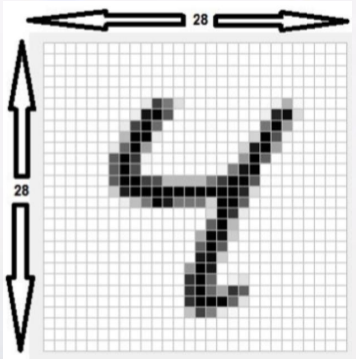
1. mnist 코어모델 생성

데이터 설명



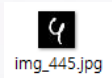
1. mnist 코어모델 생성

Image classification

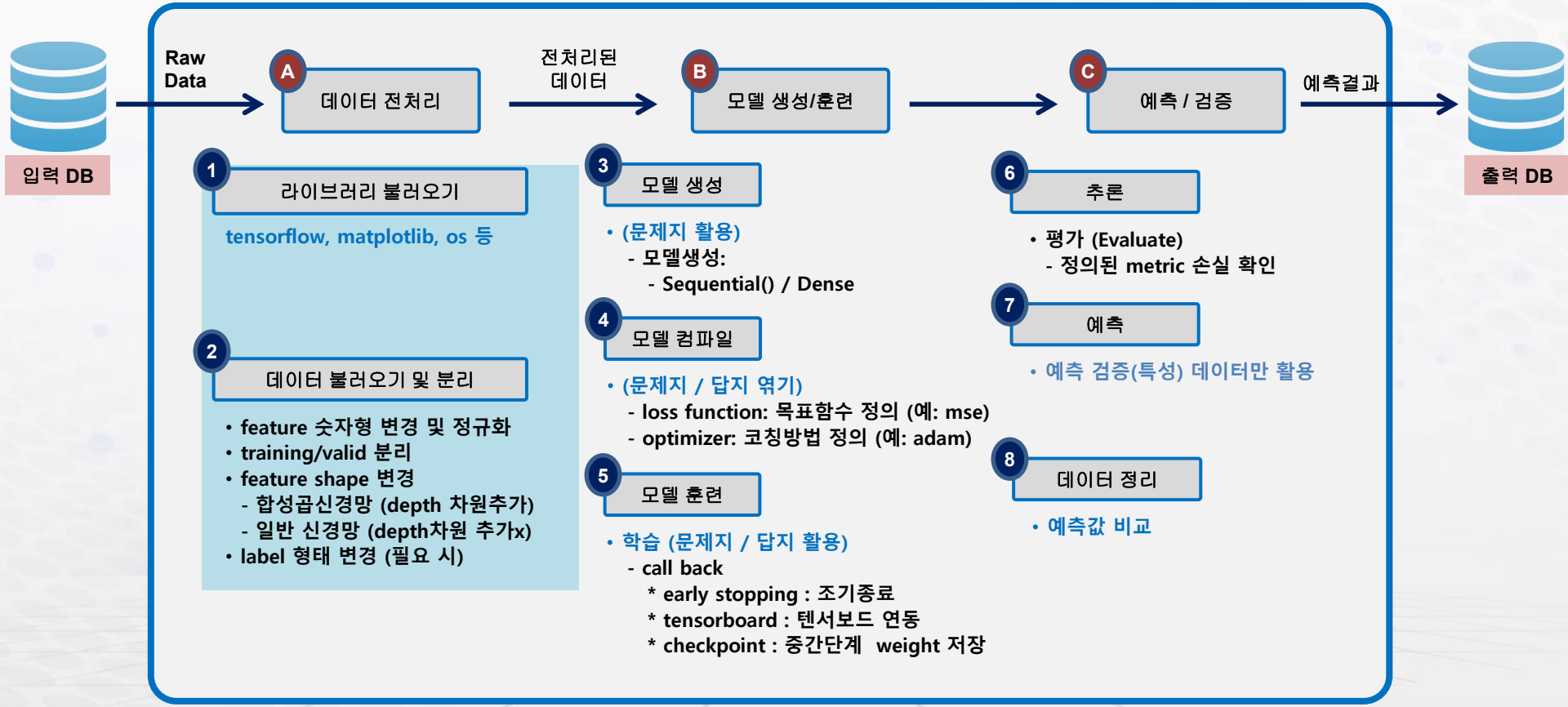


1. mnist 코어모델 생성

Image classification

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

1. mnist 코어모델 생성



1. mnist 코어모델 생성

1. 라이브러리 선언

Intel GPU 활용 방법 (PLAID ML)

```
import numpy as np
import os
import time
import matplotlib.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

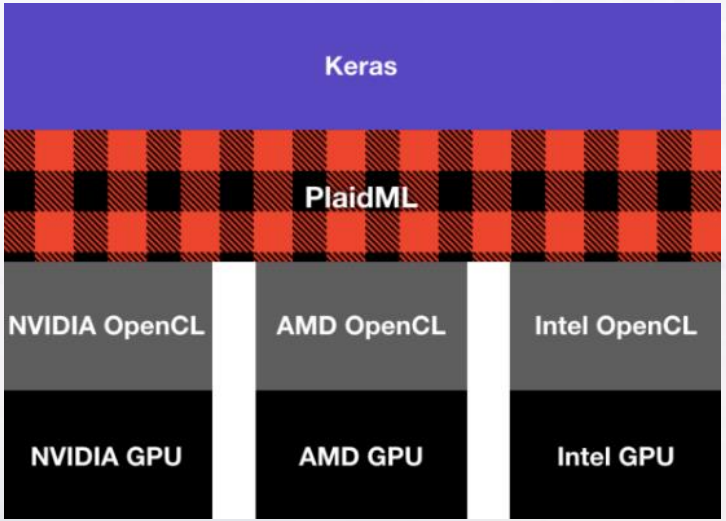
```
import tensorflow as tf
from tensorflow.python.client import device_lib
print(device_lib.list_local_devices())

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

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

1. mnist 코어모델 생성

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. mnist 코어모델 생성

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

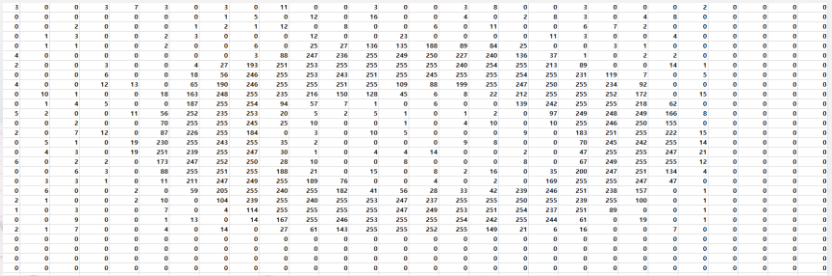
```
1 img
```

```
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. mnist 코어모델 생성

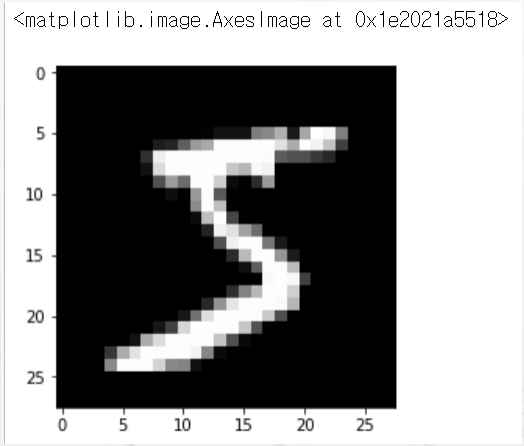
1. 라이브러리 선언

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

1. mnist 코어모델 생성

2. 데이터 불러오기

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

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

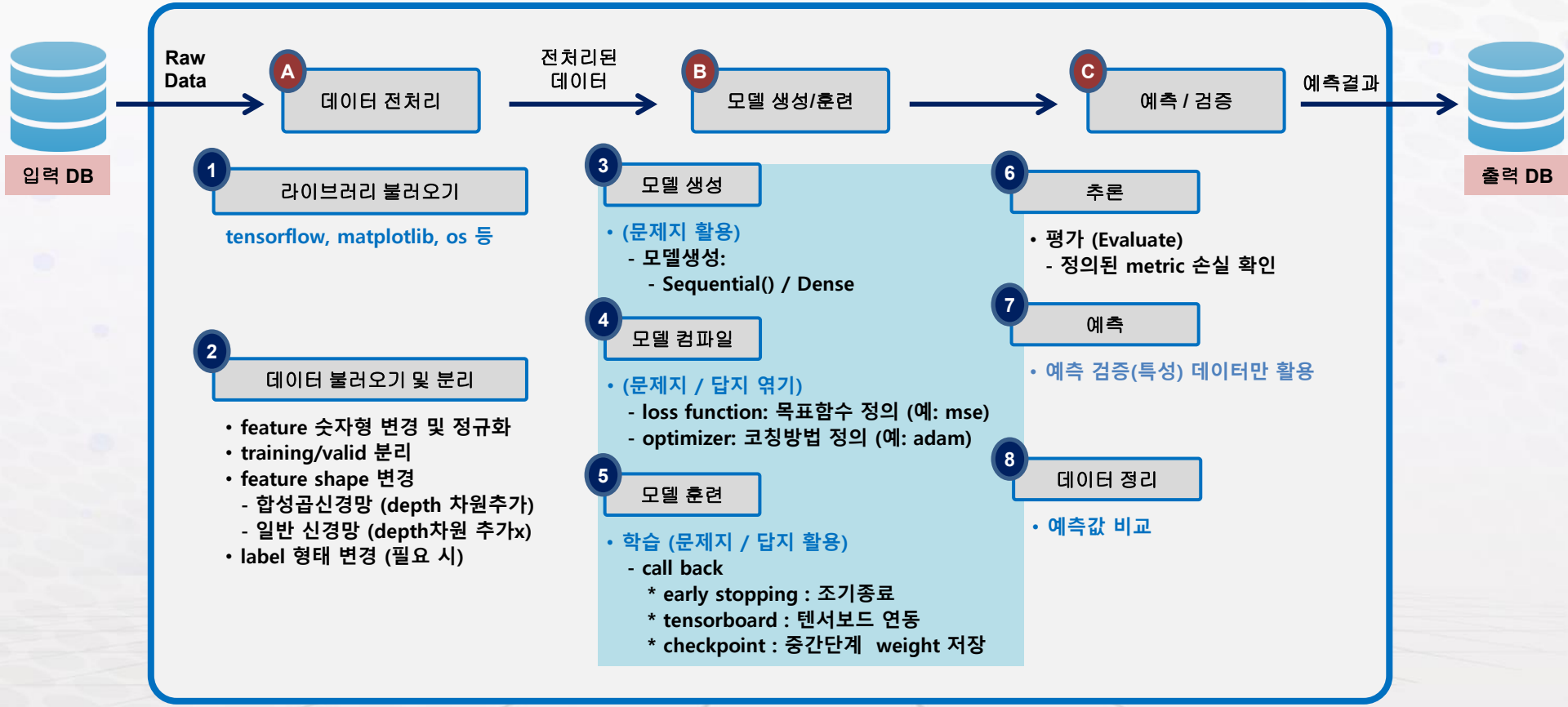


1. mnist 코어모델 생성

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)
```

1. mnist 코어모델 생성



1. mnist 코어모델 생성

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. mnist 코어모델 생성

4. 모델 컴파일

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

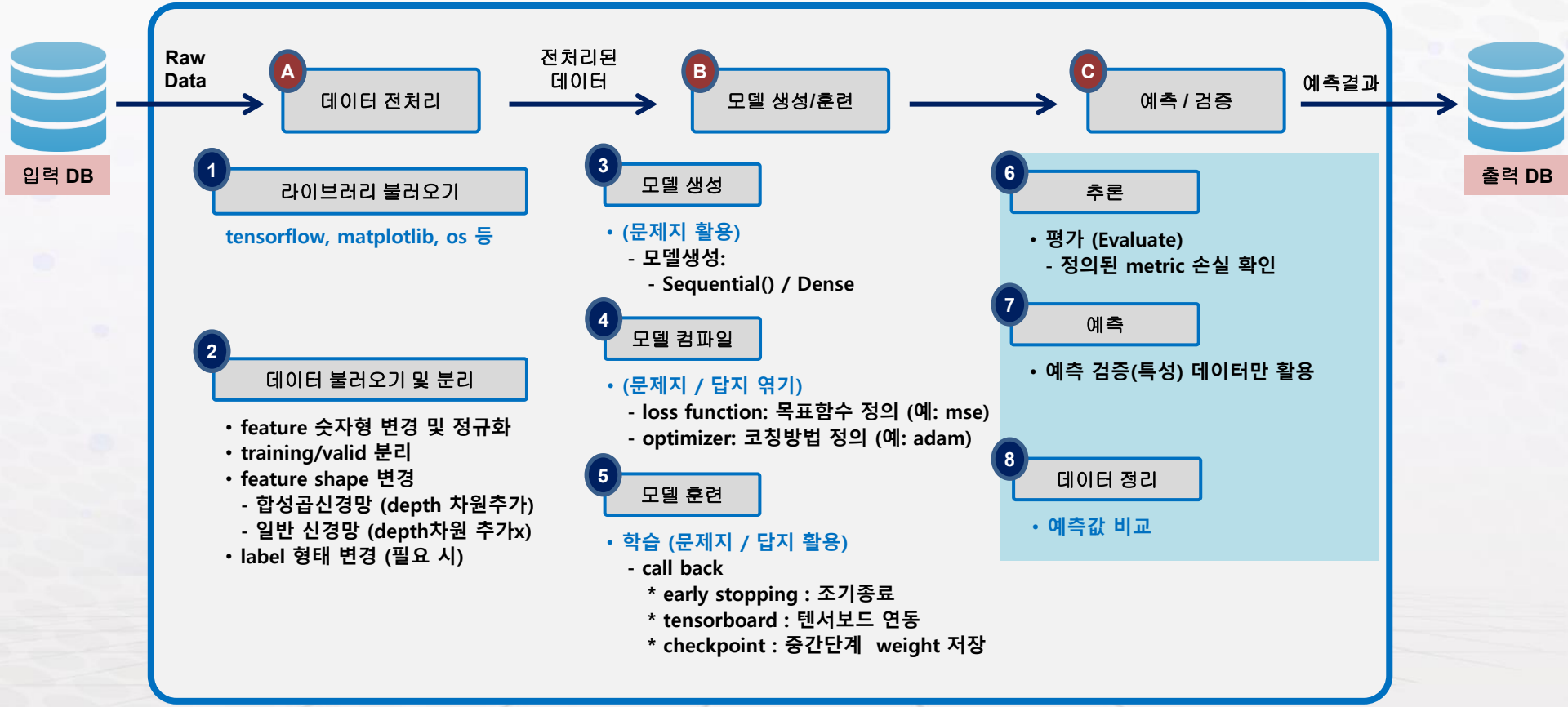
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 28, 28, 32)        160
_________________________________________________________________
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. mnist 코어모델 생성

5. 모델 훈련

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

1. mnist 코어모델 생성



1. mnist 코어모델 생성

6. 모델 추론

model.evaluate(x_test, y_test_one)

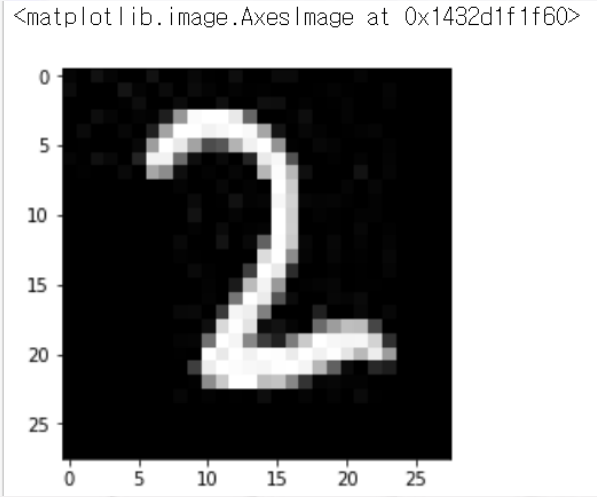
```
1  model.evaluate(x_test, y_test_one)
```

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

1. mnist 코어모델 생성

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))
```



1. mnist 코어모델 생성

모델 저장

모델 선언 구조 저장

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

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

# 모델 가중치 저장
model.save_weights("./model_mnist_weight.h5")
```

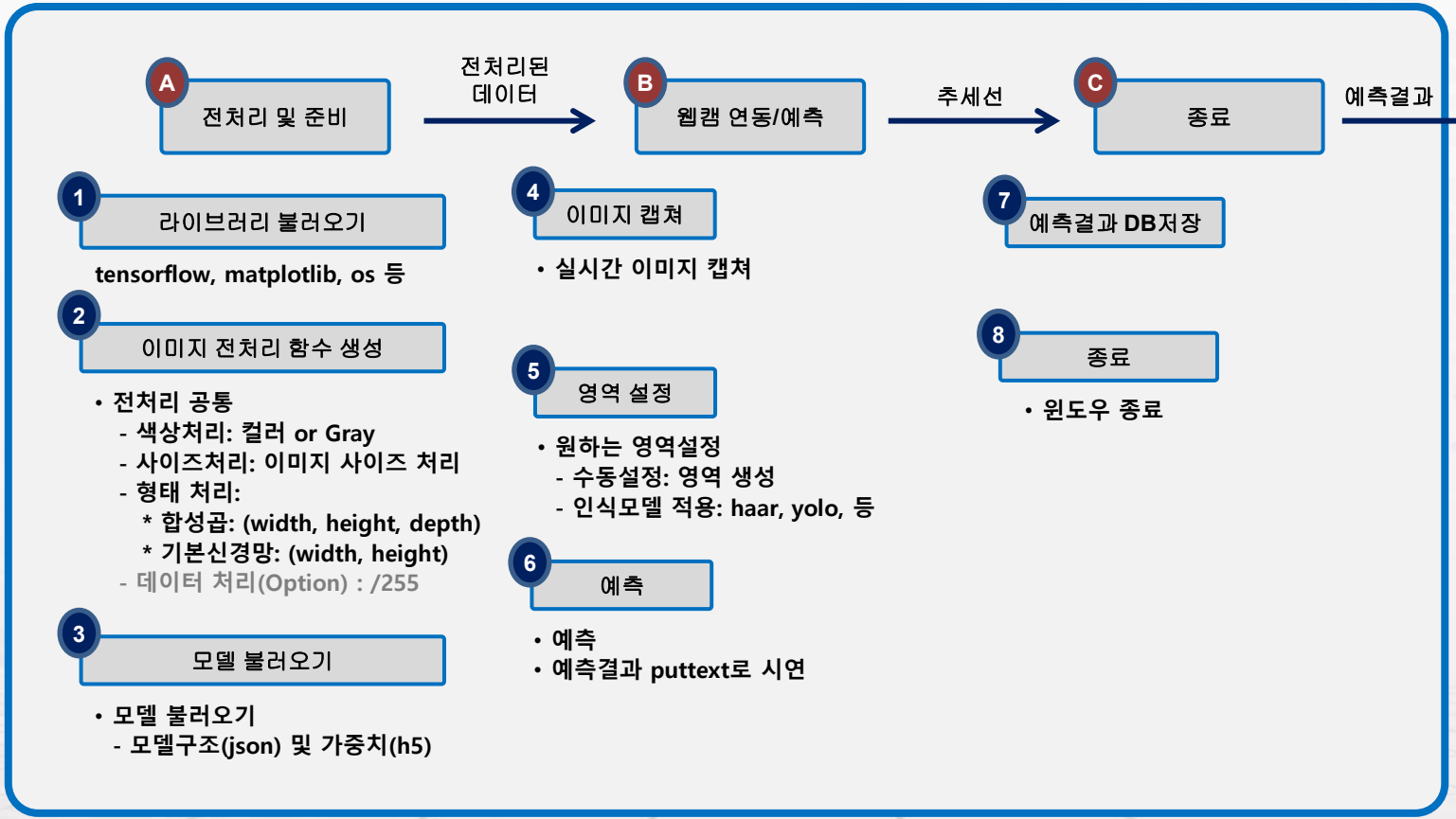
역전파를 통한 업데이트된 가중치 저장

1. mnist 코어모델 생성

GPU 메모리 릴리즈

```
from tensorflow.keras import backend as K

K.clear_session()

from numba import cuda
cuda.select_device(0)
cuda.close()
```

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 (1, 28,28,1 형태로 변경, 합성곱 신경망 활용)
def process(img_input):
    # 이미지 사이즈 변경
    IMG_SIZE = 28
    # 그레이컬러 변환 및 사이즈 조절 (단 이미 gray인 경우 패스)
    if len(img_input.shape) > 2:
        img_input = cv2.cvtColor(img_input, cv2.COLOR_BGR2GRAY)
    img_input = cv2.resize(img_input, (IMG_SIZE, IMG_SIZE))
    # 합성곱 신경망 적용을 위한 설정
    out_img = img_input.reshape(1,IMG_SIZE,IMG_SIZE)
    return out_img
```

2. 웹캠 연동

2. 이미지 전처리 수행

```
# 인풋 이미지를 불러와서 (1, 28,28,1 형태로 변경, 합성곱 신경망 활용)
def process(img_input):
    # 이미지 사이즈 변경
    IMG_SIZE = 28
    # 그레이컬러 변환 및 사이즈 조절 (단 이미 gray인 경우 패스)
    if len(img_input.shape) > 2:
        img_input = cv2.cvtColor(img_input, cv2.COLOR_BGR2GRAY)
    img_input = cv2.resize(img_input, (IMG_SIZE, IMG_SIZE))
    # 합성곱 신경망 적용을 위한 설정
    out_img = img_input.reshape(1,IMG_SIZE,IMG_SIZE)
    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_mnist.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_weight.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

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

2. 웹캠 연동

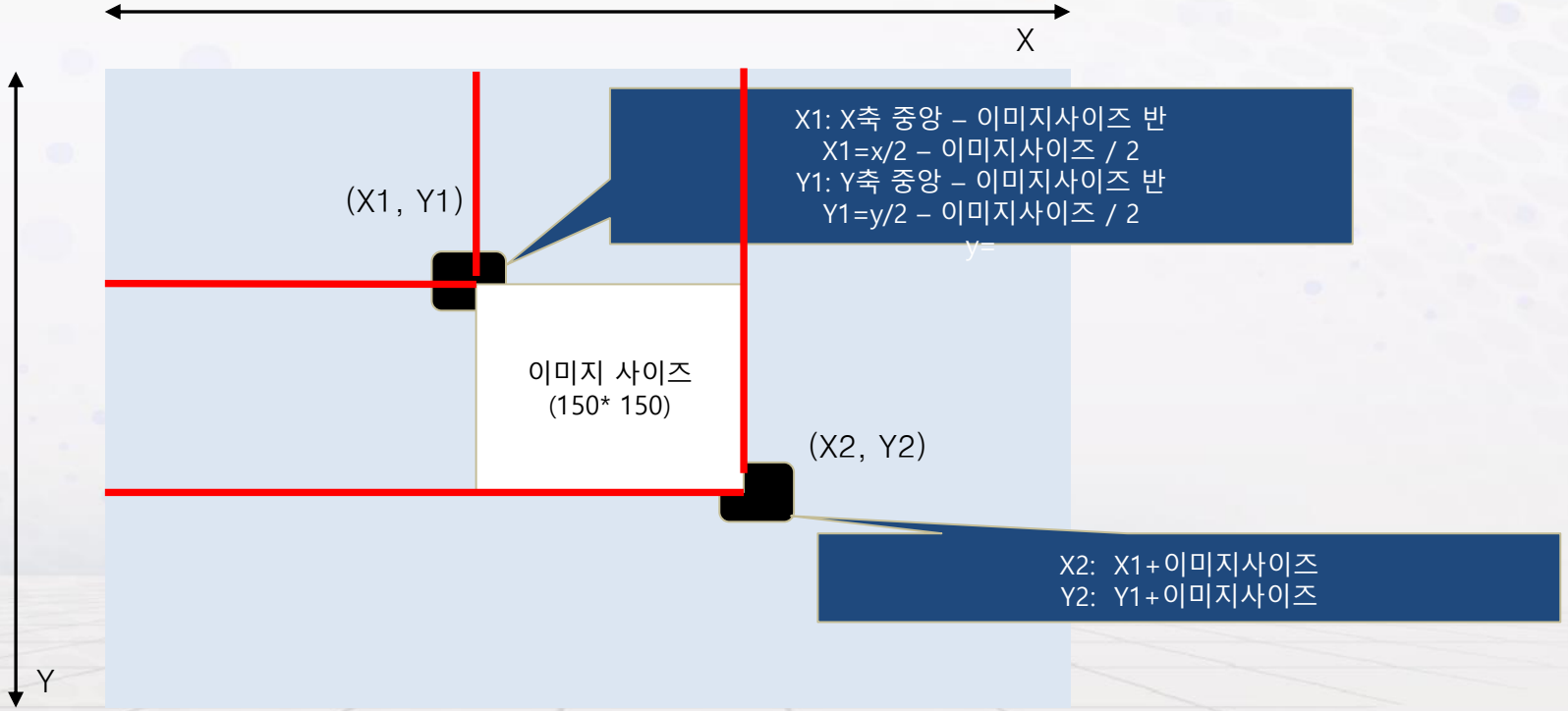
4. 웹캠연동

```
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)
```

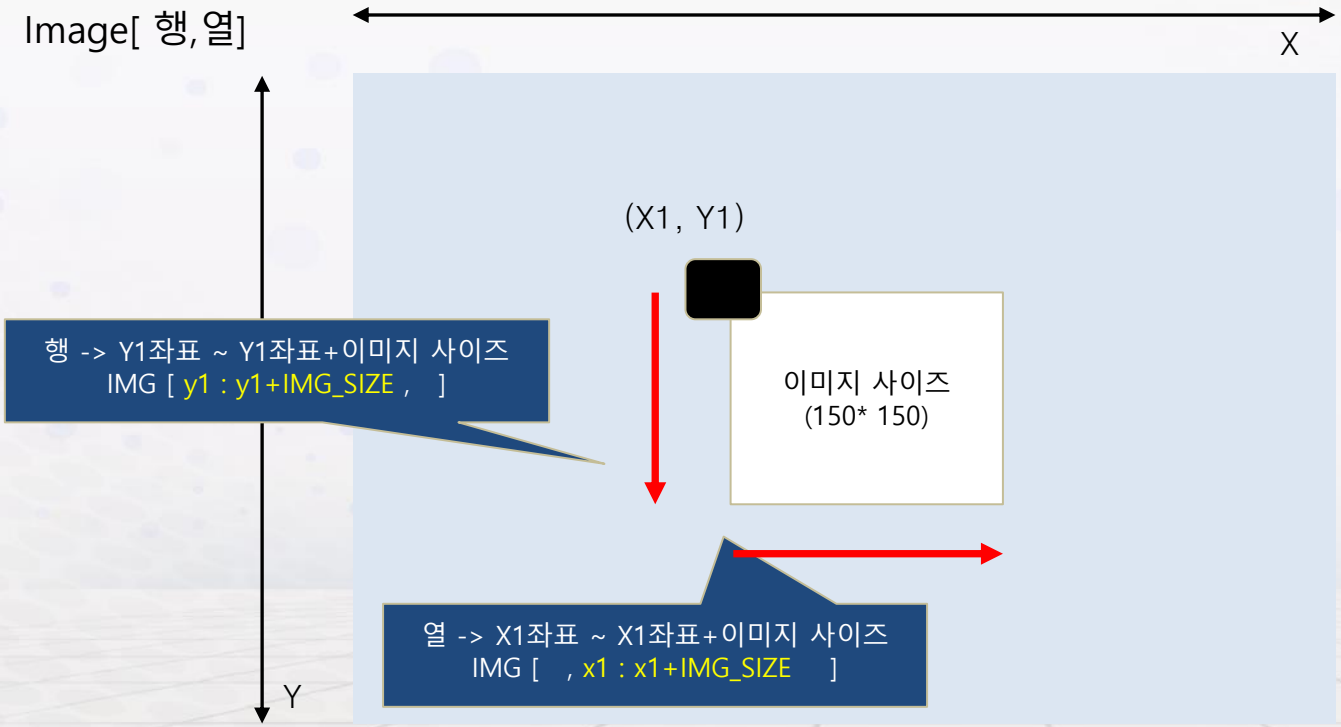
2. 웹캠 연동

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

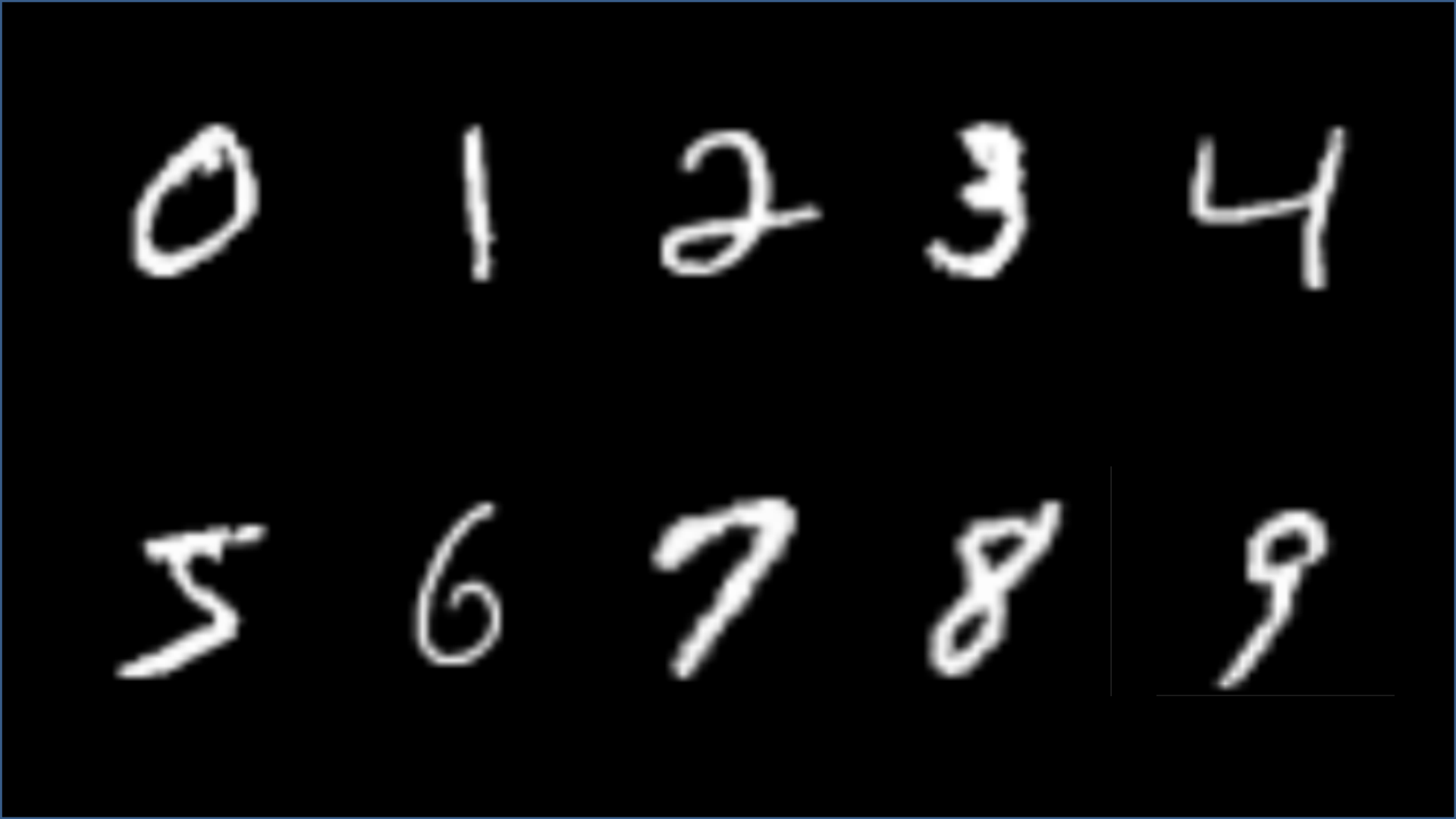
참조. 글자 삽입

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

제공된 mnist 이미지 자료를 활용하여
실습해 봅니다.

이후, emotion_f 폴더 내 자료를 활용하여
(감정인식 모델)
학습한 내용을 응용하여 구현하세요

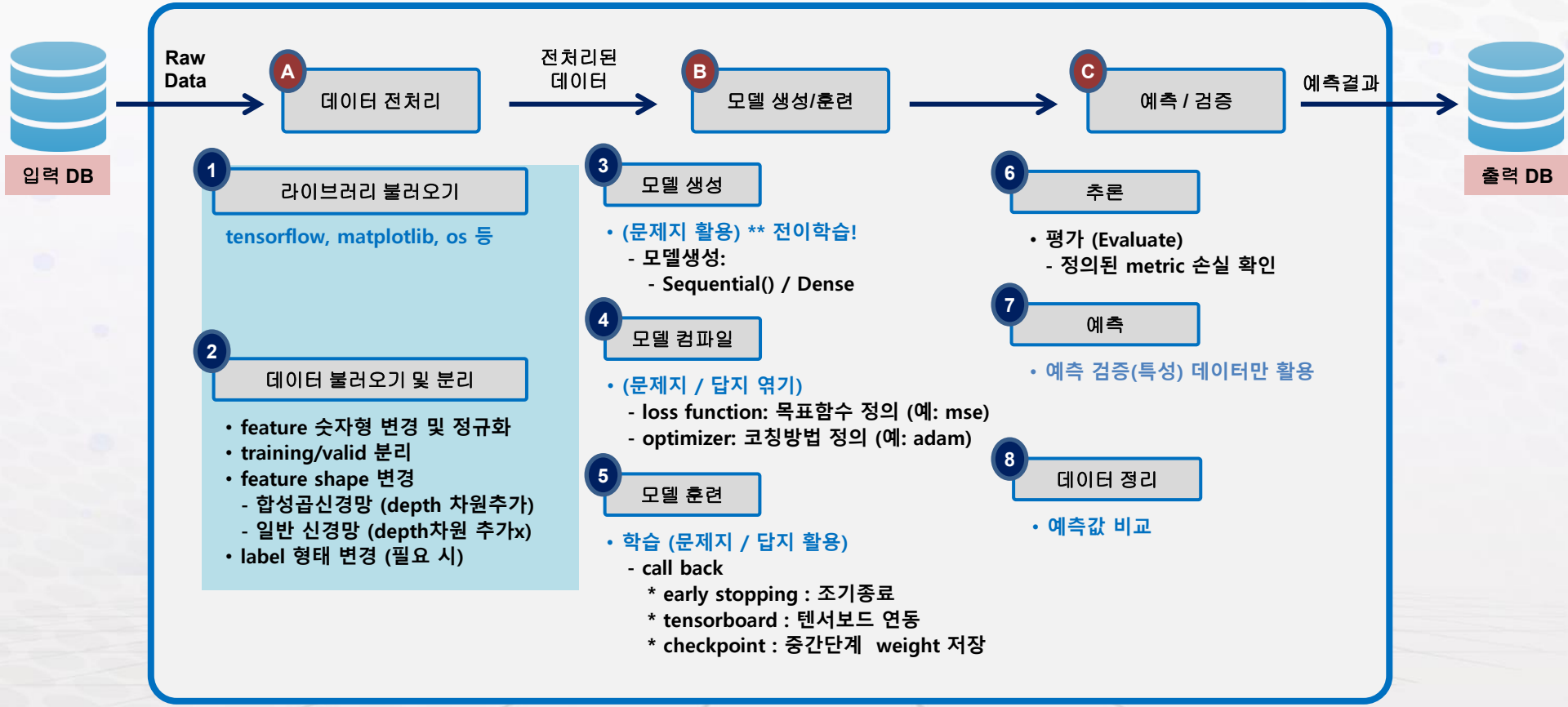
3. 핵심정리 및 Q&A

기억합시다

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

감사합니다.

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) ** 전이학습!
- 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

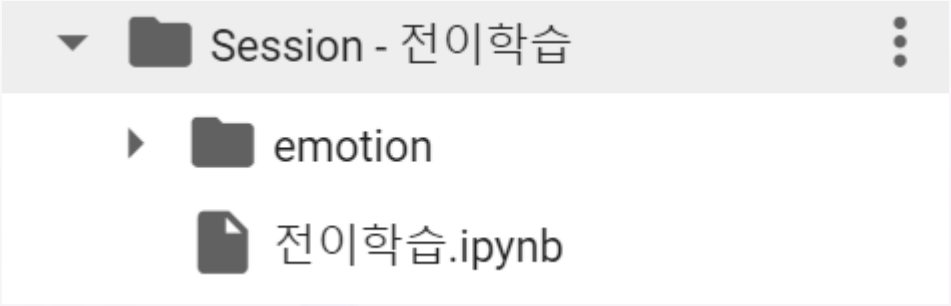
참조. 딥러닝 작동 원리

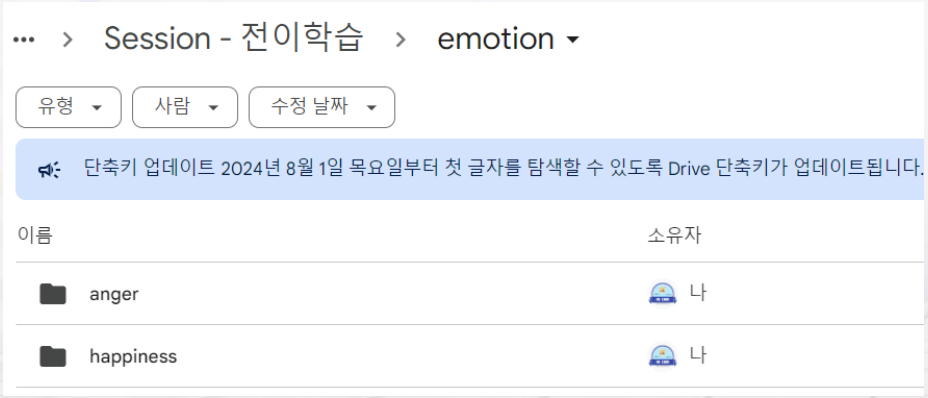
[데이터셋]
https://www.kaggle.com/datasets/sudarshanvaidya/random-images-for-face-emotion-recognition
[코드셋]
https://vo.la/Misvu

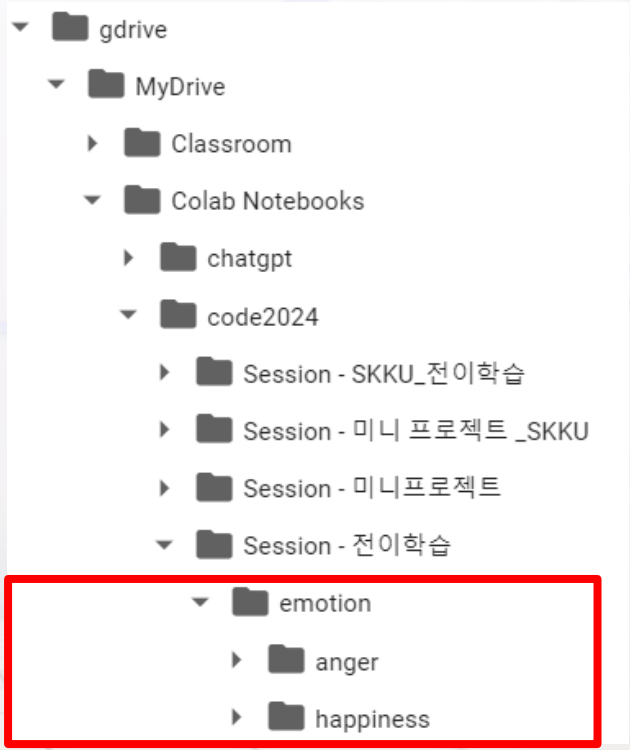
[vgg16 vgg19... 조금튜닝해서~ 쓰겠다~

출력 DB

참조. 딥러닝 작동 원리


Session - 전이학습
emotion
전이학습.ipynb


Session - 전이학습
emotion
유형
사람
수정 날짜
단축키 업데이트 2024년 8월 1일 목요일부터 첫 글자를 탐색할 수 있도록 Drive 단축키가 업데이트됩니다.
이름
소유자
anger
나
happiness
나


gdrive
MyDrive
Classroom
Colab Notebooks
chatgpt
code2024
Session - SKKU_전이학습
Session - 미니 프로젝트 _SKKU
Session - 미니프로젝트
Session - 전이학습
emotion
anger
happiness

참조. 타 모델 활용하기

1. 라이브러리 불러오기

- 기본 라이브러리 및 구글드라이브 연동

```
import os
import pandas as pd
import numpy as np
import cv2

from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.layers import Input, Flatten, Dense
from tensorflow.keras.models import Model
from tensorflow.keras.datasets import mnist
from sklearn.model_selection import train_test_split

from google.colab import drive
drive.mount("/content/gdrive")
```

사이트 들어가서
최소 shape 확인 꼭 하기!

https://keras.io/applications/

참조. 타 모델 활용하기

2. 데이터 불러오기 및 분리

- 데이터 불러오기 함수 활용

```
def collectImg(basePath, inSize=256, splitRatio=0.2):
    # 파라미터 타입변환
    basePath = str(basePath)
    inSize = int(inSize)
    splitRatio = float(splitRatio)

    trainImg = []
    trainLabel = []

    # basePath = "/content/gdrive/MyDrive/Colab Notebooks/downimg"
    totalFolderList = os.listdir( basePath )
    # 한폴더만 지정해서 수집 (반복문으로 변경)
    for folderIndex in range(0, len(totalFolderList)):
        eachFolder = totalFolderList[folderIndex]
        # 한폴더와 basepath 를 조인시켜서 이미지들이 있는 최종 폴더에 접근
        eachFolderFullPath = os.path.join( basePath, eachFolder )
        # print(eachFolderFullPath)
        totalImageList = os.listdir( eachFolderFullPath )
        # 한개의 이미지만 지정해서 저장 (반복문으로 변경)
        for imageIndex in range(0, len(totalImageList)):
            try:
                eachImg = totalImageList[imageIndex]
                eachImgFullPath = os.path.join(eachFolderFullPath, eachImg)
                inImg = cv2.imread( eachImgFullPath, cv2.IMREAD_COLOR )
                resizedImg = cv2.resize( inImg, (inSize, inSize) )
                trainImg.append( resizedImg )
                trainLabel.append( folderIndex )
            except Exception as e:
                print(eachImgFullPath)
                print(e)
                pass
    trainX = np.array( trainImg )
    trainY = np.array( trainLabel )
    trainX, testX, trainY, testY = train_test_split( trainX, trainY, test_size=splitRatio, random_state=11)
    return trainX, testX, trainY, testY
```

colab 코드 활용 설명 예정

폴더에서 이미지 불러오기

참조. 타 모델 활용하기

2. 데이터 불러오기 및 분리

- 데이터 불러오기 함수 활용

```
imgPath = "/content/gdrive/MyDrive/Colab Notebooks/code2024/Session - 전이학습/emotion"
imgLabel = ["anger","happy"]
x_train, x_test, y_train, y_test = collectImg(basePath=imgPath)

x_train[0].shape
IMG_SIZE = x_train[0].shape[0]
from tensorflow.keras.utils import to_categorical
y_train_one = to_categorical(y_train)
y_test_one = to_categorical(y_test)
```

참조. 타 모델 활용하기

Input Layer

- 데이터프레임:
  훈련데이터의 첫번째 행의 shape
- 이미지:
  훈련데이터의 첫번째 이미지의 shap

Hidden Layer
(vgg16)

자유롭게 model

Output Layer

3

참조. 타 모델 활용하기

3. 모델 생성

- vgg16 내 VGG16 application 불러옴 (가중치 제외, top 제외)
  *** 가중치는 다른사람이 열심히 트레인한거 제외, top은 input shape도 mydata의 input shape으로 변경!

```
inputShape = x_train[0].shape
inputShape

outputShape = len(np.unique(y_train))
outputShape

vgg_model = VGG16(weights=None, include_top=False)
vgg_model.summary()
```

참조. 타 모델 활용하기

3. 모델 생성

- 불러온 모델에 input output 레이어만 변경 후 모델 재설계
  *** 다른 사람은 30개 카테고리로 예측한 모델을 나의 모델은 2개 (happy, angry)로만 !!

```
# 커스텀 input_shape 생성
keras_input = Input(shape=inputShape, name = 'custom_input')
keras_input.shape
# 커스텀모델 생성
vgg_model_c = vgg_model(inputs=keras_input)

# Fully Connected Layer 추가 생성
x = Flatten(name='flatten')(vgg_model_c)
x = Dense(64, activation="relu", name='custom_1')(x)
x = Dense(64, activation="relu", name='custom_2')(x)
x = Dense(outputShape, activation='softmax',
name='prediction')(x)

# 커스텀 모델 생성
pretrained_model = Model(inputs=keras_input, outputs=x)
pretrained_model.summary()
```

참조. 타 모델 활용하기

4. 모델 컴파일

- 목표함수 및 코칭방법 설정

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

참조. 타 모델 활용하기

5. 모델 훈련

- 검증데이터 포함하여 모델 훈련 수행

```
epochs = 10
pretrained_model.fit(x=x_train,
                     y=y_train_one,
                     epochs=10,
                     validation_data = (x_test, y_test_one))
```