손글씨 인식하기

welcome to 4조

곽동혁,김현준,이민

목차 Index

프로젝트 설명

인식 과정

VSCODE

시연 영상

프로젝트 설명 Project

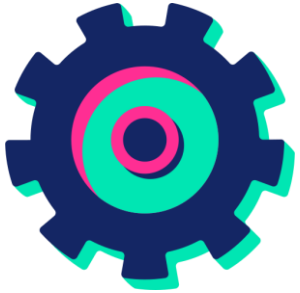
저희의 프로젝트는 **손글씨 인식**하기 입니다.

Open cv와 **Tensorflow**를 사용하여 **손글씨 인식**하기에 몰두 하였습니다.

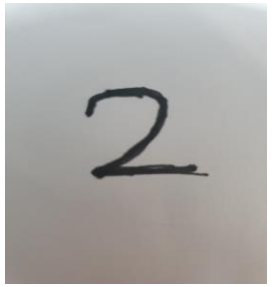
Tensorflow를 사용하여 손으로 쓴 글자들을 **AI**에 학습시켜 인식이 원활 하도록 만들었습니다.

그 후 카메라가 인식을 하게되면 **Open cv**로 이미지를 따 그 글자가 무엇 인지 표시하는 코드를 짜게 되었습니다.

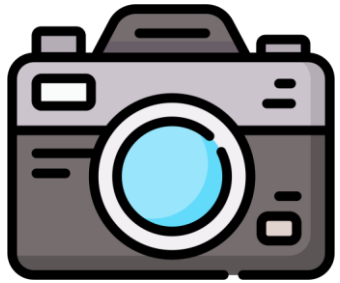
인식 과정



시스템 구동



손글씨 비추기



카메라 작동

OpenCV

Open cv로 이미지 따기

Vs code의 터미널에 작성한 숫자 표시

학습 VSCODE (PY1)

```
import tensorflow as tf


mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(512, activation=tf.nn.relu),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])

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

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)

model.save_weights('mnist_checkpoint')
```

손글씨숫자를 인식하기위해 뉴럴 네트워크를 학습시키는 코드입니다.

뉴럴 네트워크 학습이란
최대한 데이터 셋에 가까운 출력을
내게끔 선의 값을 계속 수정하면서
완성시키는 것.

구동 VSCODE (PY2)

```
import tensorflow as tf
import cv2
import numpy as np
import math


def process(img_input):

    gray = cv2.cvtColor(img_input, cv2.COLOR_BGR2GRAY)

    gray = cv2.resize(gray, (28, 28), interpolation=cv2.INTER_AREA)


    (thresh, img_binary) = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)


    h,w = img_binary.shape


    ratio = 100/h
    new_h = 100
    new_w = w * ratio

    img_empty = np.zeros((110,110), dtype=img_binary.dtype)
    img_binary = cv2.resize(img_binary, (int(new_w), int(new_h)), interpolation=cv2.INTER_AREA)
    img_empty[:img_binary.shape[0], :img_binary.shape[1]] = img_binary

    img_binary = img_empty


    cnts = cv2.findContours(img_binary.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

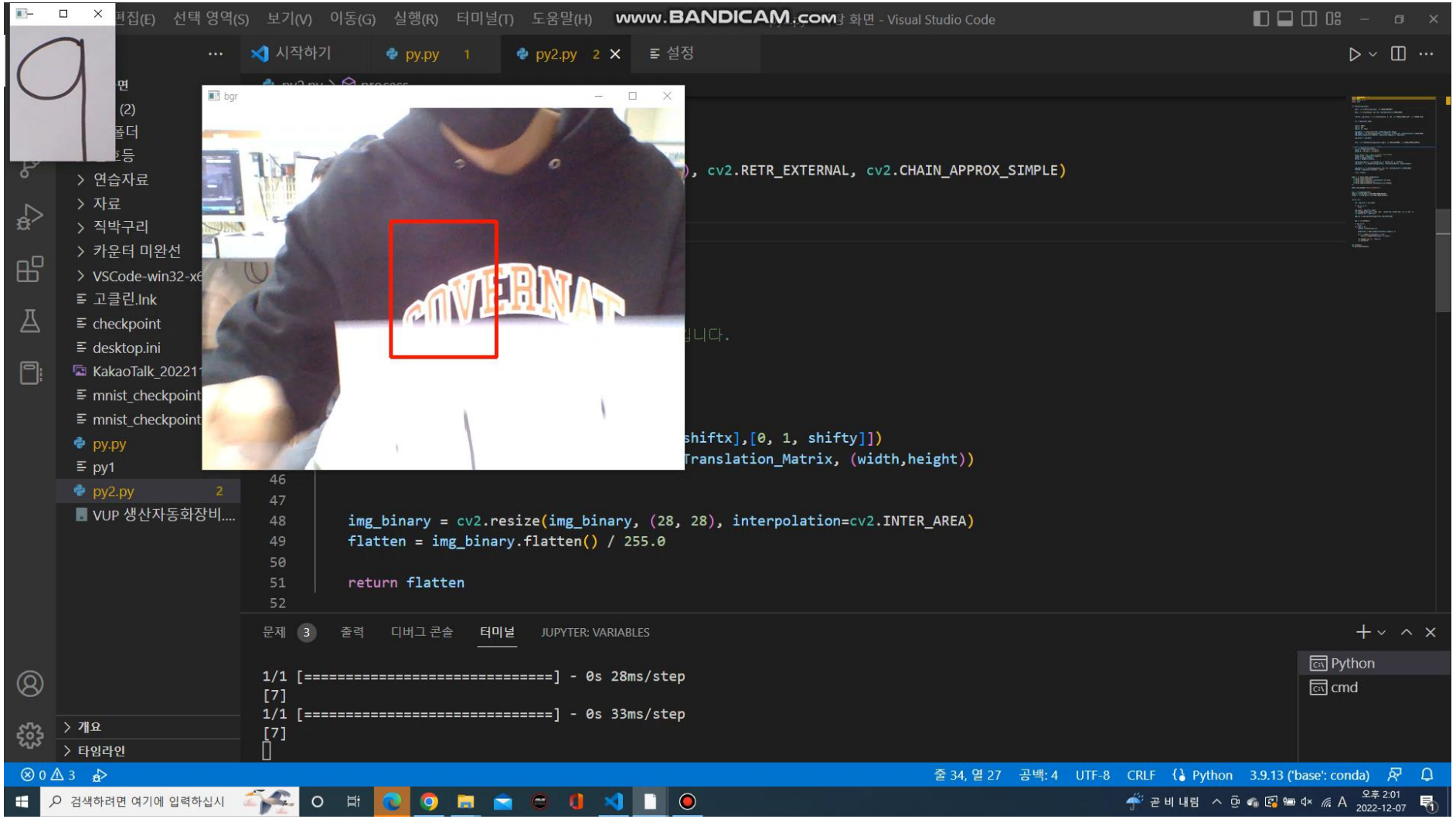
```
    # 컨투어의 무게중심 좌표를 구합니다.
    M = cv2.moments(cnts[0][0])
    center_x = (M["m10"] / M["m00"])
    center_y = (M["m01"] / M["m00"])

    # 무게 중심이 이미지 중심으로 오도록 이동시킵니다.
    height,width = img_binary.shape[:2]
    shiftx = width/2-center_x
    shifty = height/2-center_y

    Translation_Matrix = np.float32([[1, 0, shiftx],[0, 1, shifty]])
    img_binary = cv2.warpAffine(img_binary, Translation_Matrix, (width,height))


    img_binary = cv2.resize(img_binary, (28, 28), interpolation=cv2.INTER_AREA)
    flatten = img_binary.flatten() / 255.0

    return flatten



model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(512, activation=tf.nn.relu),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])

model.load_weights('mnist_checkpoint')


cap = cv2.VideoCapture(0)
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
```

```
cap = cv2.VideoCapture(0)
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))


while(True):

    ret, img_color = cap.read()

    if ret == False:
        break;

    img_input = img_color.copy()
    cv2.rectangle(img_color, (250, 150),  (width-250, height-150), (0, 0, 255), 3)
    cv2.imshow('bgr', img_color)

    img_roi = img_input[150:height-150, 250:width-250]


    key = cv2.waitKey(1)

    if key == 27:
        break
    elif key == 32:
        flatten = process(img_roi)

        predictions = model.predict(flatten[np.newaxis,:])

        with tf.compat.v1.Session() as sess:
            print(tf.argmax(predictions, 1).eval())

        cv2.imshow('img_roi', img_roi)
```

```
cap.release()
cv2.destroyAllWindows()
```

PY.1에서 학습된 뉴럴 네트워크를 사용하여 웹캠에 이미지를 Opencv 로 출력해서 손 글씨를 인식하는 코드입니다.

시연 영상


www.BANDICAM.com
bgr
py.py
py2.py
설정
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
shiftx],[0, 1, shifty]])
Translation_Matrix, (width,height))
img_binary = cv2.resize(img_binary, (28, 28), interpolation=cv2.INTER_AREA)
flatten = img_binary.flatten() / 255.0
return flatten
1/1 [==============================] - 0s 28ms/step
[7]
1/1 [==============================] - 0s 33ms/step
[7]

Thank you for watching

4조