Project 1

camera를 활용한 선,원,엣지 검출

이미지 처리 프로세스

1. 카메라 입력 영상 실행.
2. 입력 받은 영상을 Circle Detection 활용한 이미지 처리 진행
3. Line Detection을 이용한 이미지 처리 진행
4. Canny Edge Detection을 이용한 이미지 처리 진행.

이미지 처리

```python
import cv2
import numpy as np

# 웹캠을 엽니다.
cap = cv2.VideoCapture(0)

# 웹캠이 정상적으로 열렸는지 확인합니다.
if not cap.isOpened():
    print("웹캠을 열 수 없습니다.")
    exit()

# 웹캠 영상을 계속해서 표시합니다.
while True:
    ret, frame = cap.read()

    # 프레임을 제대로 읽었는지 확인합니다.
    if not ret:
        print("프레임을 읽을 수 없습니다.")
        break

    # 원 검출을 위해 이미지를 그레이스케일로 변환합니다.
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # 가우시안 블러를 적용하여 노이즈를 제거합니다.
    gray_blurred = cv2.GaussianBlur(gray, (5, 5), 0)

    # Hough Circle Transform을 적용하여 원을 검출합니다.
    circles = cv2.HoughCircles(gray_blurred, cv2.HOUGH_GRADIENT, dp=1, minDist=20, param1=150, param2=30, minRadius=5, maxRadius=

    # 선 검출을 위한 캐니 엣지 검출기 적용
    edges = cv2.Canny(gray_blurred, 50, 150)

    # 검출된 선을 그릴 빈 화면 생성
    line_image = np.zeros_like(frame)

    # 허프 선 변환을 사용하여 선을 검출합니다.
    lines = cv2.HoughLinesP(edges, 1, np.pi/180, threshold=100, minLineLength=50, maxLineGap=10)

    # 검출된 선을 원본 이미지에 그립니다.
    if lines is not None:
        for line in lines:
            x1, y1, x2, y2 = line[0]
            cv2.line(line_image, (x1, y1), (x2, y2), (255, 0, 0), 5)

```

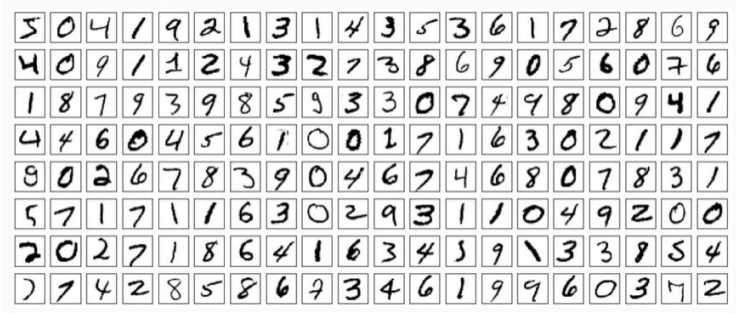
Project 2

Deep Learning Workflow

Deep Learning Workflow 5단계

1. 데이터 수집 및 전처리:
    a. 데이터를 수집하고 필요한 형식으로 변환합니다.
    b. 데이터를 정규화하고 전처리하여 모델 학습에 적합한 형태로 준비합니다.
2. 모델 설계:
    a. 딥러닝 모델의 구조를 설계합니다. 이는 층(layer) 및 연결 방법을 정의하는 것을 포함합니다.
    b. 필요에 따라 사전 학습된(pre-trained) 모델을 사용하거나 새로운 모델을 만들 수 있습니다.
3. 모델 학습:
    a. 학습 데이터를 사용하여 모델을 학습시킵니다.
    b. 손실 함수를 최소화하기 위해 가중치를 조정하고, 역전파(backpropagation) 등의 알고리즘을 사용하여 모델을 최적화합니다.
4. 모델 평가:
    a. 학습된 모델을 테스트 데이터셋에 적용하여 성능을 평가합니다.
    b. 평가 지표(metrics)를 사용하여 모델의 정확도, 정밀도, 재현율 등을 평가합니다.
5. 모델 배포 및 유지 보수:
    a. 학습된 모델을 실제 환경에 배포하고 사용 가능하게 만듭니다.
    b. 모델이 실제 환경에서 잘 작동하도록 모니터링하고 필요할 때 업데이트하거나 유지 보수합니다

Mnist dataset을 이용한 손글씨 인식



```python
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np

mnist = tf.keras.datasets.mnist

(image_train, label_train), (image_test, label_test) = mnist.load_data()

print("Train Image shape :",image_train.shape)
print("Train Labe : ", label_train,"\n")
print(image_train[0])

num = 20
plt.figure(figsize=(15,15))
for idx in range (num):
    sp = plt.subplot(5,5,idx+1)
    plt.imshow(image_train[idx])
    plt.title(f'Label: {label_train[idx]}')
plt.show()

model = tf.keras.Sequential()
model.add(tf.keras.Input(shape=(28, 28))) #인풋 사이즈
model.add(tf.keras.layers.Flatten()) # 플렛 평평히 펴주기 플렛트을 할 경우 원본 형상이 날라갈 수 있음 (굴곡 등등)
model.add(tf.keras.layers.Dense(64, activation='sigmoid')) # 줄이기
model.add(tf.keras.layers.Dense(10, activation='softmax')) # 줄이기

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

model.fit(image_train, label_train, epochs = 10, batch_size=10) #트레이닝

num = 3
predic = model.predict(image_test[0:num])
print(predic)

print( "* prediction, ", np.argmax(predic, axis = 1))
plt.figure(figsize = (15, 15))
for idx in range(num):
    sp = plt.subplot(1, 3, idx+1)
    plt.imshow(image_test[idx])

plt.show()
```