발표자: 한이루

GIT: HTTPS://GITHUB.COM/HIR260104-HASH/OPEN-CV-DEEP-LEARNING

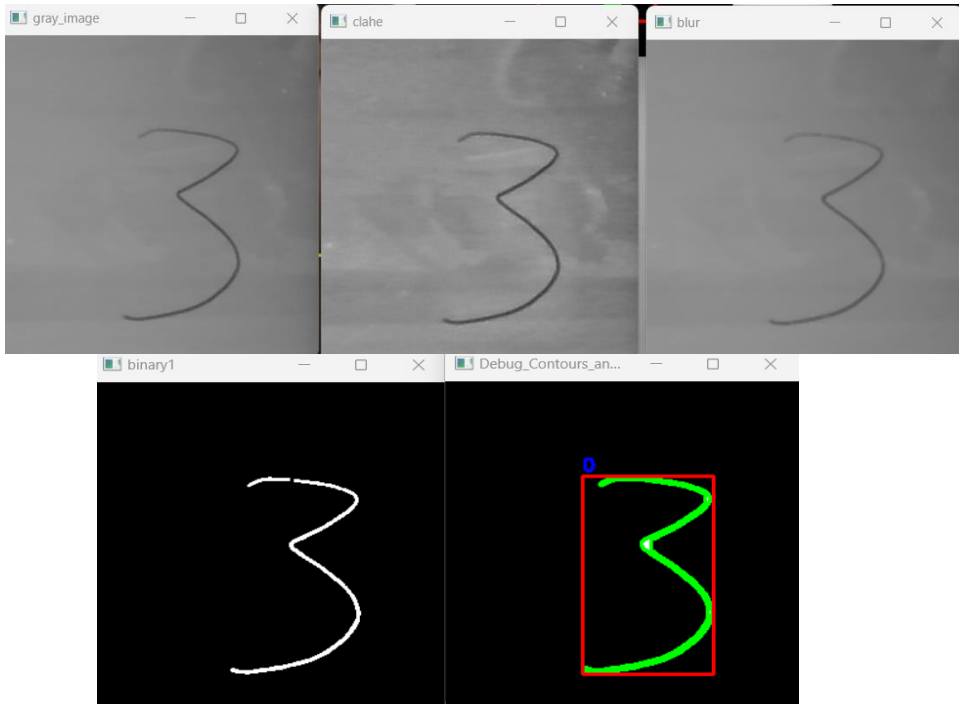
딥러닝 프로젝트

주제

- 카메라 영상 인식을 통한 딥러닝 기반 스마트 비밀번호 인증 시스템
- **목표**: 단순한 이미지 매칭이 아닌, 딥러닝(CNN)을 활용하여 다양한 환경(글씨체, 각도, 조명)에서도 높은 정확도로 숫자를 분류

전처리 단계 (이미지 정제 과정)

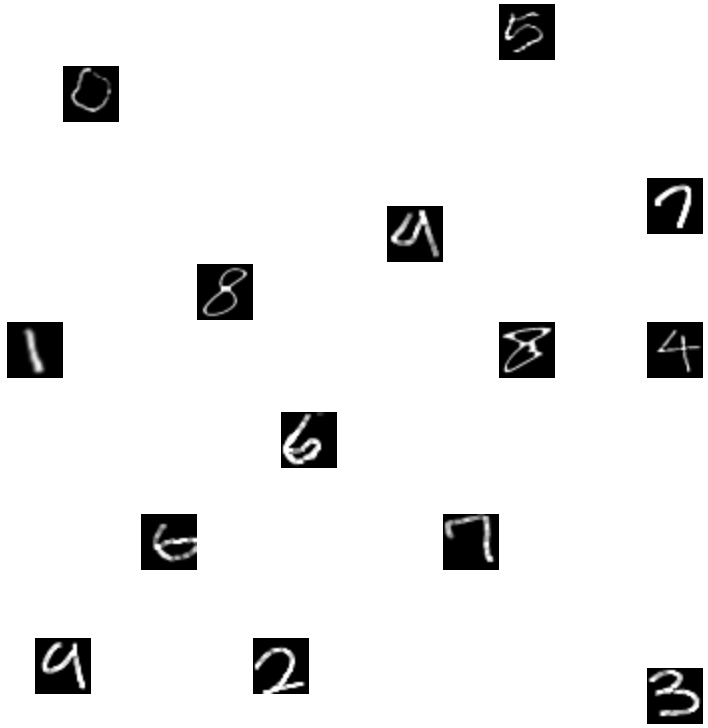
- 경계선 노이즈 제거
  - 테두리의 불필요한 선이 숫자로 인식되는 것 방지
- MNIST 규격 최적화
  - 검은 배경에 흰 글씨인 학습 데이터셋 형식과 일치화
- 지역적 대비 극대화 (CLAHE)
  - 조명 불균형을 해결하여 숫자의 형태를 뚜렷하게 보정
- 조명 변화에 강인한 이진화
  - 위치마다 다른 밝기 값을 계산해 정확한 흑백 분리
- 객체 분할(Contours)
  - 추출된 외곽선을 기준으로 개별 숫자 영역 분리



"딥러닝 모델의 성능을 극대화하기 위해, 영상을 표준화된 학습 데이터 규격으로 정제하는 '데이터 엔지니어링' 단계"

사용자 정의 데이터셋 구축

- 개인별 필체 특성 학습을 위한 실시간 샘플링
- **인터랙티브 레이블링**: ROI 내 입력된 숫자를 키보드 이벤트(0~9)와 매핑하여 자동 데이터 분류 및 저장
- **데이터 증강 적용**: 1회 캡처 시 회전, 이동, 확대/축소 등의 변환을 동시 적용하여 데이터 부족 문제 해결 및 모델의 **일반화 성능** 극대화



학습용 이미지 캡쳐 영상

모델 학습

- **기본 모델 (MNIST)**: 수만 장의 표준 숫자 데이터를 통해 "숫자의 기본 형태"를 학습.
- **튜닝 모델 모델 (사용자 데이터+MNIST)**: 내가 직접 카메라로 찍은 데이터를 추가 학습하여 "내 필체"에 최적화.
- **학습 전략**: 기본 지식 위에 내 데이터를 얹어 정확도를 극대화함.

```
# [1] 모델의 구조 정의 (CNN - 합성곱 신경망)
model = keras.Sequential([
    # 1. 특징 추출 (Conv2D, MaxPooling)
    layers.Conv2D(32, (3,3), activation='relu', input_shape=(28,28,1)),
    layers.MaxPooling2D((2,2)),

    # 2. 데이터 펼치기 (Flatten)
    layers.Flatten(),

    # 3. 최종 판단 (Dense)
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```

"CNN(합성곱 신경망) 구조를 채택하여 이미지의 특징을 스스로 추출하게 하였으며, 사용자 데이터를 결합하여 실전 인식 성공률을 개선하려고 함"

기본 모델

1. 전처리

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

x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0

x_train = x_train.reshape(-1, 28, 28, 1)
x_test = x_test.reshape(-1, 28, 28, 1)
```

2. 모델 설계

```
model = keras.Sequential([
    layers.Input(shape=(28, 28, 1)),
    layers.Conv2D(32, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.2),
    layers.Dense(10, activation='softmax')
])
```

기본 모델

3. 학습 설정

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

4. 학습 시작

```
history = model.fit(
    x_train, y_train,
    epochs=5,
    batch_size=32,
    validation_split=0.2
)
```

튜닝 모델

1. 전처리

```
(x_mnist, y_mnist), (x_test, y_test) = keras.datasets.mnist.load_data()
x_mnist = x_mnist.reshape(-1, 28, 28, 1).astype('float32') / 255.0
x_test = x_test.reshape(-1, 28, 28, 1).astype('float32') / 255.0

user_images, user_labels = [], []
base_path = "dataset"

for digit in range(10):
    dir_path = os.path.join(base_path, str(digit))
    if not os.path.exists(dir_path): continue
    for img_name in os.listdir(dir_path):
        img = cv2.imread(os.path.join(dir_path, img_name), cv2.IMREAD_GRAYSCALE)

        if img is not None:
            img = cv2.resize(img, (28, 28))
            user_images.append(img)
            user_labels.append(digit)

user_x = np.array(user_images).reshape(-1, 28, 28, 1).astype('float32') / 255.0
user_y = np.array(user_labels)

user_x_oversampled = np.repeat(user_x, 20, axis=0)
user_y_oversampled = np.repeat(user_y, 20, axis=0)
```

2. 모델 설계

```
model = keras.Sequential([
    layers.Input(shape=(28, 28, 1)),
    layers.Conv2D(32, (3, 3), activation='relu', padding='same'),
    layers.BatchNormalization(),
    layers.Conv2D(32, (3, 3), activation='relu', padding='same'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu', padding='same'),
    layers.BatchNormalization(),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(256, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(10, activation='softmax')
])
```

튜닝 모델

3. 학습 설정 및 시작1

```
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model.fit(x_mnist, y_mnist, epochs=3, batch_size=128, validation_split=0.1)
```

4. 학습 설정 및 시작2

```
datagen = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.15,
    height_shift_range=0.15,
    zoom_range=0.2,
    shear_range=0.15,
    fill_mode='constant',
    cval=0
)

model.compile(
    optimizer=keras.optimizers.Adam(learning_rate=1e-5),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

user_x_oversampled = np.repeat(user_x, 20, axis=0)
user_y_oversampled = np.repeat(user_y, 20, axis=0)

indices = np.random.choice(len(x_mnist), 39000, replace=False)
mixed_x = np.concatenate((user_x_oversampled, x_mnist[indices]), axis=0)
mixed_y = np.concatenate((user_y_oversampled, y_mnist[indices]), axis=0)
model.fit(datagen.flow(mixed_x, mixed_y, batch_size=64), epochs=5, verbose=1)
```

모델 비교

- 정확도(Accuracy): 전체적으로 얼마나 잘 맞히나?
- 정밀도(Precision): 보안성 (보안 사고 방지)
- 재현율(Recall): 사용자 편의성 (인식 성공률)

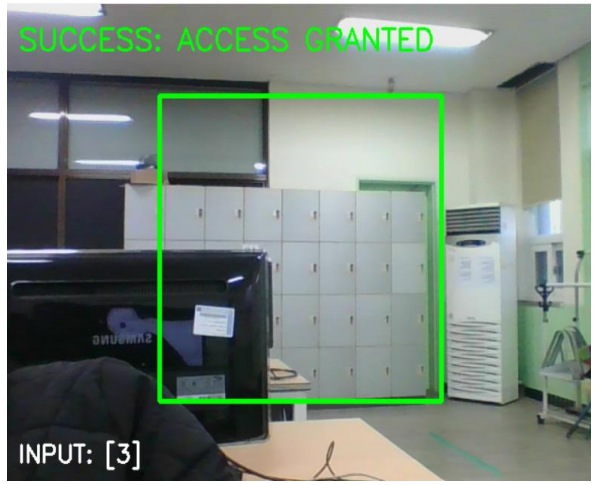
평가지표 (Metric)	기본 모델 (Simple)	맞춤형 모델 (Fine-tuned)	개선 정도
전체 정확도 (Accuracy)	98.52%	98.93%	+0.41%p
정밀도 (Precision)	98.54%	98.92%	+0.38%p
재현율 (Recall)	98.50%	98.92%	+0.42%p

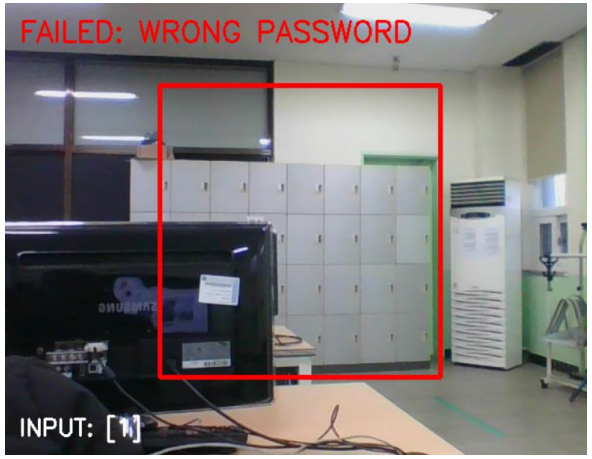
- (의문점) 학습 후 전체적인 성능은 올라갔으나, 숫자 '8'의 경우 재현율이 미세하게 하락함. 하지만 직접 테스트 해봤을 땐 학습 후 '8'이 더 잘 인식되어졌음.

```
--- [숫자 '8'에 대한 집중 분석] ---
Simple Model Recall: 0.9918
Fine-tuned Model Recall: 0.9897
개선도: -0.21% 향상
```

프로그램 흐름

- 설정할 비밀번호 입력
- 숫자 인식
- 입력한 비밀번호와 숫자가 일치한다면 "성공"
  아니면 "실패"





기본 모델 영상

튜닝모델영상

어려웠던 점 및 해결방안

- 맨 처음엔 OpenCV 만으로 해결하려고 했지만, 사진마다 너무 특징이 달라서 어떤 사진에는 특성이 잘 먹고 어떤 사진에는 특성이 잘 안먹기도 했음.
- .손글씨 인식률 문제
  - 해결 방법: 사용자의 필체를 직접 수집하여 미세 조정 학습을 수행.
- 숫자 영역 추출 문제 (객체 분할 문제)
  - 해결 방법: OpenCV의 `findContours` 함수를 사용하여 객체의 외곽선을 탐지.

하드웨어와 연결 방안

- 라즈베리 파이에 카메라 모듈을 달고, 이 파이썬 코드를 그대로 실행.
- 인식된 결과가 설정한 비밀번호와 일치하면, 잠금장치를 구동시켜 문을 열 수 있다.

개선방안

- 나의 손글씨 말고 개개인의 손글씨를 학습시켜 개인화를 시키는 방향도 검토 필요 -> 이렇게 하기 위해서는 캠을 켜서 손글씨 추출하는 방법 외에 파일 업로드 방법이 더 편할 수도 있다고 생각
- 내가 올린 이미지를 검증하는 방향도 고려해야할 것 같음 (실수로 올렸을 때 발생 가능)
- 지금은 분류를 수동으로 해주지만, 자동으로 해주는 것도 고려 필요