

포트폴리오

| | |
|------------|--|
| 프로젝트 명 | 딥러닝 기반의 유방암 분류 신경망 |
| 프로젝트 기간 | 2025.03.25 |
| 내용 요약 | <div>1. 개발 언어: Python, Tensorflow library</div> <div>2. 유방암 데이터셋을 사용하여 이진 분류 문제를 해결하는 신경망 모델을 학습하고 평가하는 프로그램입니다. 데이터를 훈련용 데이터와 테스트용 데이터로 분리하고, 정규화를 통해 모델의 학습을 효율적으로 진행합니다.</div> <div>신경망 모델은 3 개의 Dense 층을 가지고 있으며, 마지막 출력층은 sigmoid 활성화 함수를 사용하여 이진 분류 문제를 해결합니다. 모델을 훈련시킨 후 정확도를 평가하여 화면에 출력합니다.</div> |
| 코드 해설 | <div><pre>from sklearn.datasets import load_breast_cancer from sklearn.model_selection import train_test_split (X, y) = load_breast_cancer(return_X_y=True) print(X[0]) (X_train, X_test, y_train, y_test) = train_test_split(*arrays: X, y, test_size=0.2, stratify=y, random_s</pre></div> <div>유방암 데이터를 불러오고 훈련용 데이터와 테스트용 데이터로 나눕니다.</div> <div><pre>from sklearn.preprocessing import MinMaxScaler scaler = MinMaxScaler() scaler.fit(X_train) scaler.transform(X_train) scaler.fit(X_test) scaler.transform(X_test) print() print(X_train[0]) print(X_train.shape)</pre></div> <div>데이터를 전처리 하는 코드. 전처리는 특정 값들을 1 이나 0 으로 변환하는 과정이고 신경망에서 학습이 더 잘 되도록 합니다.</div> |

```
import keras
import tensorflow as tf
model = keras.Sequential(name='Predict_Cancer')
input_layer = keras.Input(shape=(30,), name="input_layer")
model.add(input_layer)
model.add(keras.layers.Dense(units=64, activation='relu', name="Layer1"))
model.add(keras.layers.Dense(units=64, activation='relu', name="Layer2"))
model.add(keras.layers.Dense(units=32, activation='relu', name="Layer3"))
output_layer = keras.layers.Dense(units=1, activation='sigmoid', name='Output')
model.add(output_layer)
model.summary()
```

신경망 정의

Model: "Predict_Cancer"

| Layer (type) | Output Shape | Param # |
|----------------|--------------|---------|
| Layer1 (Dense) | (None, 64) | 1,984 |
| Layer2 (Dense) | (None, 64) | 4,160 |
| Layer3 (Dense) | (None, 32) | 2,080 |
| Output (Dense) | (None, 1) | 33 |

Total params: 8,257 (32.25 KB)
Trainable params: 8,257 (32.25 KB)
Non-trainable params: 0 (0.00 B)

Summary 함수로 어떤 구조인지 쉽게 파악가능

```
model.compile(optimizer='adam', loss="binary_crossentropy", metrics = ['accuracy'])
model.fit(x=X_train, y=y_train, epochs=1_000, verbose='auto')
evaluation = model.evaluate(x=X_test, y=y_test)
print(f"예측 정확도 : {evaluation}")
```

Adam 최적화 알고리즘을 사용하여 모델을 학습. Epochs 를 1000 으로 설정하여 훈련을 1000 번 반복합니다. 반복이 끝나면 테스트 데이터를 사용해 모델의 성능을 평가합니다

결과

```
15/15 ————— 0s 2ms/step - accuracy: 0.9715 - loss: 0.0619
Epoch 999/1000
15/15 ————— 0s 2ms/step - accuracy: 0.9654 - loss: 0.0749
Epoch 1000/1000
15/15 ————— 0s 2ms/step - accuracy: 0.9769 - loss: 0.0473
4/4 ————— 0s 7ms/step - accuracy: 0.9408 - loss: 0.0940
예측 정확도 : [0.08695278316736221, 0.9561403393745422]
```

손실 값(Loss)과 정확도(Accuracy)가 표기됨

| | |
|-----|---|
| 시각화 | <div data-bbox="311 286 1380 824"> <p>The figure consists of two side-by-side line plots. The left plot, titled 'Model Loss', has 'Epoch' on the x-axis (0 to 1000) and 'Loss' on the y-axis (0.0 to 0.6). It shows a blue line for 'train' loss that drops sharply from 0.6 to near 0.0 by epoch 100, and an orange line for 'val' loss that starts at 0.6, drops to 0.1, spikes to 0.45 at epoch 300, and then fluctuates between 0.1 and 0.2. The right plot, titled 'Model Accuracy', has 'Epoch' on the x-axis (0 to 1000) and 'Accuracy' on the y-axis (0.825 to 1.000). It shows a blue line for 'train' accuracy that quickly reaches 1.0, and an orange line for 'val' accuracy that starts at 0.825, jumps to 0.95, and then fluctuates between 0.925 and 0.975.</p> </div> |
| 소감 | <p>유방암 데이터셋을 사용하여 실제 문제를 해결하는 경험을 통해 데이터 전처리, 모델 학습, 평가 과정을 깊이 이해할 수 있었습니다. 특히 신경망 모델을 설계하고 훈련하는 과정에서 여러 가지 파라미터와 하이퍼파라미터 조정의 중요성을 실감했고, 모델의 성능을 향상시키는 방법을 배웠습니다. 이를 통해 실용적인 머신러닝 모델을 구현하는 능력을 향상시킬 수 있었고, 향후 더 다양한 데이터셋에 대해 적용할 수 있는 자신감을 얻었습니다. 훈련 시 과대적합이 발생하는 것이 보이는데, l1, l2 규제를 통해서 과대적합을 없앨 수 있도록 모델을 다시 설계해보겠습니다.</p> |

포트폴리오

| | |
|------------|--|
| 프로젝트명 | 자료구조를 이용한 도서관리 프로그램 개발 |
| 프로젝트 기간 | 2025.02.26 ~ 2025. 02. 28 |
| 상세내용 | <p>1. C, 자료구조</p> <p>2. 자료구조인 트리를 활용하여 도서 관리 프로그램을 개발합니다. 프로그램을 실행하면 두 가지의 책이 기본값으로 설정되며, 사용자는 메뉴를 번호로 선택하여 작동할 수 있습니다. 메뉴의 기능으로는 새로운 책 추가, 책 검색, 책 삭제, 책 재고 출력 등이 있습니다. 책 추가는 새로운 책을 추가하는 기능이며, 책 검색은 이름을 입력하면 트리 구조를 중위 탐색하여 동일한 책을 찾아냅니다. 책 삭제는 이름을 입력하면 동일한 이름의 책을 찾아서 삭제하는 기능입니다. 재고 출력 메뉴는 현재 보유하고 있는 책들의 정보를 한 번에 출력합니다.</p> |
| Flow chart | <pre> graph TD Start([start]) --> Main{Main} Main -- 1 --> AddBook[Add Book] Main -- 2 --> SearchBook[Search Book] Main -- 3 --> DeleteBook[Delete Book] Main -- 4 --> PrintBook[Print Book] AddBook --> Main SearchBook --> Main DeleteBook --> Main PrintBook --> Main </pre> |
| 실행화면 | <pre> typedef struct Book { char title[100]; // 책 제목 char author[100]; // 저자 char isbn[100]; // ISBN 번호 int year; // 출판 연도 int quantity; // 재고 수량 }Book; </pre> |

```
typedef struct TreeNode
{
    Book book;           // 책 정보 (구조체 포함)
    struct TreeNode *left; // 왼쪽 자식 노드
    struct TreeNode *right; // 오른쪽 자식 노드
}TreeNode;
```

구조체 코드입니다. 책 구조체와 트리 구조체를 구현했습니다.

```
1. 책 추가
2. 책 검색
3. 책 삭제
4. 책 재고 출력
선택 :
```

실행하면 도서관 메뉴가 나옵니다. 번호를 입력하면 각각 메뉴들이 실행됩니다.

```
TreeNode *insertBook(TreeNode *root, Book book)
{
    if(root == NULL)
    {
        TreeNode *newnode = (TreeNode*)malloc(sizeof(TreeNode));
        newnode->book = book;
        newnode->left = NULL;
        newnode->right = NULL;
        return newnode;
    }

    if(strcmp(book.title, root->book.title) < 0)
        root->left = insertBook(root->left, book);
    else if(strcmp(book.title, root->book.title) > 0)
        root->right = insertBook(root->right, book);
    return root;
}
```

책 추가 코드입니다. 책 제목을 기준으로 이진 탐색 트리의 규칙에 따라서 삽입합니다.

```

void deleteBookByTitle(TreeNode **root)
{
    char title[100];
    fputs("삭제할 책 제목: ", stdout);
    fgets(title, sizeof(title), stdin);
    title[strlen(title, "\n")] = 0;

    *root = deleteBook(*root, title);
    fputs("책이 삭제되었습니다.\n", stdout);
}

```

책 삭제 코드입니다. 동일한 책 제목이 있는지 이진 탐색을 하여 찾습니다

```

void printBooks(TreeNode *root)
{
    if(root != NULL)
    {
        printBooks(root->left);
        fputs("-----\n", stdout);
        printf("Title : %s\nAuthor : %s\nISBN : %s\nYear : %d\nQuantity : %d\n",
            root->book.title, root->book.author, root->book.isbn,
            root->book.year, root->book.quantity);
        printBooks(root->right);
    }
}

```

책 목록 출력 코드입니다. 중위순회 방식으로 트리를 탐색하여 책정보를 출력합니다.

```

TreeNode *deleteBook(TreeNode *root, char *title)
{
    if(root == NULL)
        return NULL;

    if(strcmp(title, root->book.title) < 0)
    {
        root->left = deleteBook(root->left, title);
    }
    else if(strcmp(title, root->book.title) > 0)
    {
        root->right = deleteBook(root->right, title);
    }
    else // 현재 노드가 삭제할 노드일 경우

```

| | |
|----|--|
| | <pre>if(root->left == NULL && root->right == NULL) // 자식이 없는 경우 { free(root); return NULL; } else if(root->left == NULL) // 오른쪽 자식만 있는 경우 { TreeNode *temp = root->right; free(root); return temp; } else if(root->right == NULL) // 왼쪽 자식만 있는 경우 { TreeNode *temp = root->left; free(root); return temp; } else // 자식이 두 개 있는 경우 { TreeNode *temp = findminNode(root->right); // 오른쪽 서브트리에서 최소값 찾기 root->book = temp->book; // 찾은 최소값을 현재 노드에 복사 root->right = deleteBook(root->right, temp->book.title); // 중복된 노드 삭제 } } return root;</pre> <p>책 삭제 코드입니다. 자식이 없는 경우 현재 노드를 삭제합니다. 자식이 하나만 있을 경우 자식을 현재 위치로 이동시킵니다. 자식이 둘 다 있을 경우 오른쪽 서브 트리에서 가장 작은 값을 찾아서 현재 노드로 이동시킵니다.</p> |
| 소감 | <p>이 프로젝트를 통해 트리 구조를 이용한 데이터 관리와 검색, 삭제 기능을 구현하면서 자료구조의 중요성을 다시 한 번 느꼈습니다.</p> |

포트폴리오

| | |
|------------|---|
| 프로젝트 명 | 오토인코더를 이용한 딥러닝 기반의 온습도 이상 감지 프로그램 |
| 프로젝트 기간 | 2025.04.03 |
| 상세내용 | <div>1. 개발 언어: Python, C/C++, Tensorflow</div> <div>2. 설명: 이 코드는 온도 및 습도 데이터를 수집하여 오토인코더 기반 이상 탐지 시스템을 구현합니다. 시리얼 포트(COM3)에서 데이터를 읽고, 기존 데이터를 CSV로 저장한 후, 정규분포를 이용해 추가 데이터를 생성합니다. 정규화된 데이터를 학습하여 오토인코더 모델을 훈련하고, 재구성 오차를 기반으로 이상 여부를 감지합니다. 실시간으로 센서 데이터를 분석하여 임계값을 초과하면 부저와 LED가 작동되도록 설계되었습니다.</div> |
| 아두이노 | <div><pre>#include <DHT.h> #define DHTPIN 11 // DHT11 센서 데이터 핀 #define DHTTYPE DHT11 #define BUZZER_PIN 9 // 부저 핀 (수동형이므로 tone() 사용) #define LED_PIN 2 // LED 핀 (2번 포트에 LED 연결) DHT dht(DHTPIN, DHTTYPE); void setup() { Serial.begin(9600); dht.begin(); pinMode(BUZZER_PIN, OUTPUT); pinMode(LED_PIN, OUTPUT); // LED 핀을 출력으로 설정 digitalWrite(BUZZER_PIN, LOW); digitalWrite(LED_PIN, LOW); // LED를 끈 상태로 초기화 // 모니터링 시작 메시지 출력 Serial.println("모니터링 시작..."); }</pre></div> <div>아두이노의 초기 셋업입니다. 연결할 부저와 센서, LED 포트를 설정합니다.</div> |


```

void loop() {
    delay(2000); // 2초마다 측정

    // 센서 값 읽기
    float temperature = dht.readTemperature();
    float humidity = dht.readHumidity();

    // 센서 값을 읽지 못한 경우
    if (isnan(temperature) || isnan(humidity)) {
        Serial.println("ERROR: 센서 값을 읽을 수 없음");
        return;
    }

    // 센서 데이터를 Python 등 외부 프로그램으로 전송 (CSV 데이터 형식)
    Serial.print(temperature);
    Serial.print(",");
    Serial.println(humidity);

    // Python으로부터 명령 수신 및 모니터링용 출력
    if (Serial.available() > 0) {
        String command = Serial.readStringUntil('\n');
        command.trim(); // 개행 문자 제거

        Serial.print("수신된 명령: ");
        Serial.println(command);

        if (command == "BUZZER_ON") {
            tone(BUZZER_PIN, 1000); // 1000Hz 주파수로 부저 작동
            Serial.println("부저 작동 중...");

            // 부저와 함께 LED도 켜기
            digitalWrite(LED_PIN, HIGH); // LED 켜기

            delay(1000); // 1초간 소리와 LED가 켜짐

            noTone(BUZZER_PIN); // 부저 정지
            Serial.println("부저 정지");

            // LED 끄기
            digitalWrite(LED_PIN, LOW); // LED 끄기
        }
    }
}

```

온습도센서 데이터를 파이썬으로 계속 전송합니다. 파이썬에서 이상이 감지되었다는 명령이 수신된다면 부저와 LED가 작동됩니다.

파이썬

```
# 시리얼 포트 설정
ser = serial.Serial( port: 'COM3', baudrate: 9600, timeout=1)
time.sleep(2)

# CSV 파일 준비
filename = "temperature_humidity_data.csv"
if not os.path.exists(filename):
    with open(filename, mode='w', newline='') as file:
        writer = csv.writer(file)
        writer.writerow(["Temperature", "Humidity"])
    print("새로운 데이터 파일 생성 완료!")
else:
    print("데이터 파일이 이미 존재합니다. 기존 데이터를 계속 사용합니다.")

# 데이터 로드
df = pd.read_csv(filename)

# 원본 데이터의 평균과 표준편차 계산
temp_mean, temp_std = df["Temperature"].mean(), df["Temperature"].std()
humi_mean, humi_std = df["Humidity"].mean(), df["Humidity"].std()

# 정규분포를 사용하여 새로운 데이터 2000개 생성
np.random.seed(42)
new_temps = np.random.normal(temp_mean, temp_std, size: 2000)
new_humis = np.random.normal(humi_mean, humi_std, size: 2000)

# 생성된 데이터를 DataFrame으로 변환
new_data = pd.DataFrame({"Temperature": new_temps, "Humidity": new_humis})

# 원본 데이터와 합치기
df_extended = pd.concat( objs: [df, new_data], ignore_index=True)

# 데이터 정규화
scaler = MinMaxScaler()
X = scaler.fit_transform(df_extended[["Temperature", "Humidity"]])
```

원본 데이터의 표준편차를 계산하고 정규분포로 새로운 데이터를 생성하고 합친 뒤 데이터 정규화를 합니다.

```

# 오토인코더 모델 생성
input_dim = X.shape[1]
encoding_dim = 2

input_layer = Input(shape=(input_dim,))
encoded = Dense(units=8, activation='relu')(input_layer)
encoded = Dense(encoding_dim, activation='relu')(encoded)

decoded = Dense(units=8, activation='relu')(encoded)
decoded = Dense(input_dim, activation='sigmoid')(decoded)

autoencoder = Model(*args: input_layer, decoded)
autoencoder.compile(optimizer='adam', loss='mse')

# 모델 학습
print("모델 학습 시작")
autoencoder.fit(X, X, epochs=200, batch_size=4)
autoencoder.save("autoencoder_model.keras")
print("모델 학습 완료!")

# threshold 자동 설정
X_reconstructed = autoencoder.predict(X)
reconstruction_errors = np.mean(np.abs(X - X_reconstructed), axis=1)
threshold = np.mean(reconstruction_errors) + 2 * np.std(reconstruction_errors)
threshold = max(threshold, 0.1) # 최소값을 설정

print(f"🔧 새 임계값: {threshold:.4f}")

```

원본 데이터와 오토인코더가 복원한 데이터 차이를 계산하고, 평균오차 + 2 배 표준편차를 이상탐지 임계값으로 설정합니다.

```

# 실시간 감지 시작
print("실시간 감지 시작")
autoencoder = tf.keras.models.load_model("autoencoder_model.keras", compile=False)

while True:
    try:
        data = ser.readline().decode('utf-8').strip()
        if data and "," in data:
            temp, humi = map(float, data.split(","))
            input_data = pd.DataFrame(data=[temp, humi], columns=["Temperature", "Humidity"])
            input_data_scaled = scaler.transform(input_data)

            # 재구성된 데이터를 원래 스케일로 되돌리기
            reconstructed_scaled = autoencoder.predict(input_data_scaled)
            reconstructed = scaler.inverse_transform(reconstructed_scaled)

            # 원본 데이터와 재구성된 데이터 비교
            reconstruction_error = np.mean(np.abs(input_data.values - reconstructed))

            anomaly_detected = reconstruction_error > threshold

            print(f"🌡 Temp: {temp}, 💧 Humi: {humi}, 🚨 이상 감지: {anomaly_detected}, 🔍 오류율: {reconstruction_error:.4f}")

            if anomaly_detected:
                ser.write(b'BUZZER_ON\n') # 아두이노에 부저 작동 명령 전송

            time.sleep(2)
    except KeyboardInterrupt:
        print("❌ 실시간 감지 중지됨.")
        break

```

프로그램이 실행되면 아두이노의 시리얼 데이터를 읽고 오차를 계산하고 임계값보다 크다면, 아두이노로 신호를 보내 부저와 LED가 작동되도록 합니다.

실행

513/513 ————— 0s 758us/step - loss: 5.8687e-05

Epoch 200/200

513/513 ————— 0s 746us/step - loss: 5.1907e-05

모델 학습 완료!

65/65 ————— 0s 1ms/step

새 임계값: 0.1000

학습결과로 임계값이 결정됩니다.

1/1 ————— 0s 33ms/step

Temp: 28.6, Humi: 38.0, 이상 감지: False, 오류율: 0.0127

1/1 ————— 0s 27ms/step

Temp: 28.6, Humi: 38.0, 이상 감지: False, 오류율: 0.0127

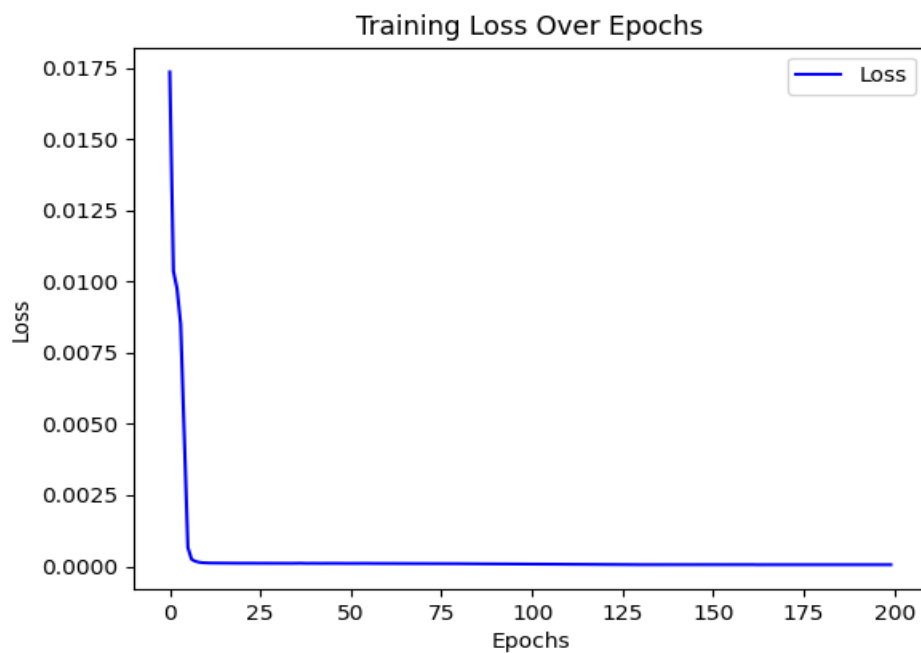
1/1 ————— 0s 29ms/step

Temp: 28.6, Humi: 38.0, 이상 감지: False, 오류율: 0.0127

1/1 ————— 0s 29ms/step

2 초마다 데이터를 읽고 학습된 임계값을 넘어 이상감지가 되면 부저와 LED 가 작동됩니다,

손실
그래프



손실 그래프입니다. 빠르게 0 으로 수렴하여 성공적으로 학습이 된 것을 알 수 있습니다.

| | |
|-----------|---|
| <p>사진</p> | <div data-bbox="309 264 1369 658" data-label="Image"> </div> <p>이상이 감지되면 부저와 LED 가 작동합니다.</p> |
| <p>소감</p> | <p>아두이노를 이용해 실시간 센서 데이터를 수집하고, 이를 파이썬에서 처리하여 오토인코더 모델을 학습시키는 과정에서 데이터 전처리와 모델링 기술에 대한 이해가 깊어졌습니다. 부저와 LED를 이용한 실시간 알림 시스템을 설계함으로써, IoT와 딥러닝을 결합한 실제적인 문제 해결 능력을 키울 수 있었습니다.</p> |

포트폴리오

| | |
|------------|--|
| 프로젝트명 | Pos 프로그램 |
| 프로젝트 기간 | 2025.02.14 ~ 2025.02.21 |
| 상세내용 | <p>1. 개발 언어: C (리눅스)</p> <p>2. 가게에서 돈을 계산하는 POS 기 프로그램입니다. 프로그램 시작 시 로그인 화면이 출력되며 관계자인지 확인하고, 인증에 성공하면 이름과 시작 시간, 가게 잔고가 표시됩니다. 이후 메인 화면으로 이동하며, 메뉴에는 제품 입력, 제품 재고 확인, 제품 입고, 계산, 제품 검색, 근무 종료, 프로그램 종료 기능이 있습니다. 제품 입력은 새로운 제품을 등록하는 기능입니다. 재고 확인은 가게에 있는 물건들의 이름과 개수를 *표로 표시합니다. 제품 입고는 물건 이름을 입력하면 이미 등록된 경우 "이미 존재합니다."라는 메시지가 출력되며, 등록되지 않은 경우 전체 목록에서 선택하여 세부사항을 수정할 수 있습니다. 제품 검색은 원하는 물건을 빠르게 찾을 수 있도록 합니다. 계산 메뉴는 고객이 선택한 물건을 골라서 판매하는 기능으로, 성인 인증이 필요한 경우 관련 메시지가 출력되며, 유통기한이 지난 물건은 경고 메시지가 표시됩니다. 판매된 물건은 재고에서 차감되며, 수익은 가게 잔고에 추가됩니다. 근무 종료 메뉴는 근무 시간을 계산하여 일급을 출력합니다.</p> |
| 플로우 차트 | <pre> graph TD Start([시작]) --> Login[로그인] Login --> MainMenu{메인메뉴} MainMenu --> ProductInput[제품입력] ProductInput --> ProductData[/제품데이터/] ProductData --> MainMenu MainMenu --> ProductInventory[제품재고확인] ProductInventory --> MainMenu MainMenu --> ProductIn[제품입고] ProductIn --> MainMenu MainMenu --> Calculation[계산] Calculation --> MainMenu MainMenu --> ProductSearch[제품검색] ProductSearch --> MainMenu MainMenu --> EndWork[근무종료] EndWork --> MainMenu MainMenu --> End([종료]) End --> End </pre> |

코드

```
struct Product
{
    char name[50]; // 제품명
    char mf[50];   // 제조사
    char exp[50];  // 유통기한
    int price;     // 가격
    int adult;     // 성인여부
    int count;     // 재고
};
```

상품 구조체의 구성 요소입니다.

```
while (1)
{
    int choose = 0;
    printf("\n\n1. 제품 입력 메뉴 \n");
    printf("2. 제품 재고 확인 \n");
    printf("3. 제품 입고 \n");
    printf("4. 계산 메뉴 \n");
    printf("5. 제품 검색 \n");
    printf("6. 근무 종료 \n");
    printf("7. 프로그램 종료 \n");

    // 유효한 숫자 입력이 들어올 때까지 반복
    int result = 0;
    do {
        printf("선택 : ");
        result = scanf("%d", &choose);

        if (result != 1)
        { // 숫자가 아닌 값이 입력된 경우
            printf("잘못된 입력입니다. 숫자를 입력하세요!\n");
        }
        while (getchar() != '\n'); // 입력 버퍼 비우기
    } while (result != 1);
```

반복문 안에 스위치문으로 번호로 메뉴를 선택하면 해당 함수가 실행됩니다.

함수실행이 끝나면 반복문으로 다시 돌아와 메뉴가 다시 출력됩니다

```

void product_input()
{
    fputs("제 품 명 : ", stdout);
    fgetstr(products[product_count].name, sizeof(products[product_count].name));

    fputs("제 조 회 사 : ", stdout);
    fgetstr(products[product_count].mf, sizeof(products[product_count].mf));

    fputs("유통기한 : ", stdout);
    fgetstr(products[product_count].exp, sizeof(products[product_count].exp));

    fputs("성인여부(0: 아니오, 1: 예): ", stdout);
    while (scanf("%d", &products[product_count].adult) != 1) {
        // 입력 오류 처리
        printf("잘못된 입력입니다. 성인여부(0: 아니오, 1: 예): ");
        while(getchar() != '\n'); // 잘못된 입력이 있을 경우 버퍼를 비움
    }
}

```

제품입력 코드입니다. 구조체 내용들을 하나하나 입력할 수 있습니다.

```

    fputs("현금 얼마를 지불하십니까? ", stdout);
    if (scanf("%d", &cash) != 1 || cash < total_price)
    {
        fputs("입력된 현금이 부족하거나 잘못된 금액입니다.\n", stdout);
        return;
    }
    change = cash - total_price;
    fprintf(stdout, "잔돈은 %d원입니다.\n", change);
    balance += total_price; // 잔고에 반영
}
else if (num == 1) // 카드 결제
{
    fputs("카드 결제입니다.\n", stdout);
    balance += total_price;
}

printf("총 결제 금액은 %d원입니다.\n", total_price);
printf("잔고 : %d원 \n", balance);
fputs("구매가 완료되었습니다.\n", stdout);
return;
}

```

계산 함수입니다. 총 결제금액을 계산하고 가게잔고와 잔돈계산, 재고처리를 합니다.

| | |
|------------------------|---|
| <div>프로그램 실행</div> | <div><div><div>tot@iot-VMware-Virtual-Platform:~/Code/c\$./POS2</div><div>ID : asdf</div><div>PASSWORD : 1234</div><div>사원 : KarL</div><div>현재 시간 : 14:30</div><div>잔고 : 1234000원</div></div><div><div><div>1. 제품 입력 메뉴</div><div>2. 제품 재고 확인</div><div>3. 제품 입고</div><div>4. 계산 메뉴</div><div>5. 제품 검색</div><div>6. 근무 종료</div><div>7. 프로그램 종료</div><div>선택 : 1</div></div><div><div>1. 제품 입력 메뉴</div><div>2. 제품 재고 확인</div><div>3. 제품 입고</div><div>4. 계산 메뉴</div><div>5. 제품 검색</div><div>6. 근무 종료</div><div>7. 프로그램 종료</div><div>선택 : 1</div></div><div><div>제품명 : 초콜릿</div><div>제조회사 : tottee</div><div>유통기한 : 2025-05-29</div><div>성인여부(0: 아니오, 1: 예): 0</div><div>가격 : 2500</div><div>갯수 : 10</div></div></div><div><div>상품 목록</div><div>1. 담배 : 7000원 10개</div><div>2. 사탕 : 500원 30개</div><div>3. 포카칩 : 1500원 10개</div><div>4. 콜라 : 1000원 20개</div><div>5. 사만코 : 1000원 10개</div><div>6. 너구리 : 1300원 5개</div><div>7. 빵 : 1400원 8개</div><div>8. 소주 : 2000원 10개</div><div>9. 새콤달콤 : 1000원 10개</div><div>10. 휴지 : 10000원 3개</div><div>구매할 상품 번호를 선택하세요 (0을 입력하면 종료): 1</div><div>이 제품은 성인 인증이 필요합니다. (1: 예, 2: 아니오): 1</div><div>구매할 수량을 입력하세요: 2</div><div>상품을 추가로 구매하시겠습니까? (1: 예, 2: 아니오): 1</div><div>구매할 상품 번호를 선택하세요 (0을 입력하면 종료): 4</div><div>구매할 수량을 입력하세요: 2</div><div>상품을 추가로 구매하시겠습니까? (1: 예, 2: 아니오): 2</div><div>결제 방법을 선택해주세요 (1: 카드, 2: 현금): 2</div><div>현금 얼마를 지불하십니까? 50000</div><div>잔돈은 34000원입니다.</div><div>총 결제 금액은 16000원입니다.</div><div>잔고 : 1250000원</div><div>구매가 완료되었습니다.</div></div><div><div>검색할 제품명 : 콜라</div><div>검색 결과 : 콜라 1000원 18개</div></div></div> |
| <div>소감</div> | <div>C 언어를 활용한 상품 관리 시스템을 구현해 보았습니다. 사용자 입력을 처리하고, 상품의 재고 관리 및 결제 시스템을 구현하면서 데이터 구조와 알고리즘에 대한 이해를 높일 수 있었습니다. 실용적인 프로그램을 만들며 문제 해결 능력이 향상된 기회였습니다.</div> |

포트폴리오

| | |
|------------|--|
| 프로젝트 명 | VGG19 모델을 이용한 딥러닝기반의 이미지예측 프로그램 |
| 프로젝트 기간 | 2025.03.27 |
| 상세내용 | <ol style="list-style-type: none"> 1. 개발 언어: 파이썬, Tensorflow 2. 설명: 이 프로그램은 VGG19 모델을 사용하여 이미지를 분류합니다. 모델을 정의하고, 사전 학습된 가중치를 로드한 후, 주어진 이미지를 전처리하여 모델에 입력합니다. 이미지의 예측 결과를 출력합니다 |
| VGG19 |  <p>VGG -19 Architecture</p> <p>224x224x64, 112x112x128, 56x56x256, 28x28x512, 14x14x512, 7x7x512</p> <p>maxpool, maxpool, maxpool, maxpool</p> <p>depth=64, depth=128, depth=256, depth=512, depth=512</p> <p>3x3 conv, 3x3 conv, 3x3 conv, 3x3 conv, 3x3 conv</p> <p>conv1_1, conv2_1, conv3_1, conv4_1, conv5_1</p> <p>conv1_2, conv2_2, conv3_2, conv4_2, conv5_2</p> <p>conv3_3, conv4_3, conv5_3</p> <p>conv3_4, conv4_4, conv5_4</p> <p>Size=4096, FC1, Size=1000, FC2, Softmax</p> |
| conv | <pre>import keras import tensorflow as tf model = keras.Sequential(name="VGG19_NET") input_layer = keras.Input(shape=(244, 244, 3), name='InputImage')</pre> <p>인풋 레이어 층을 삽입합니다</p> |

```

model.add(keras.layers.Conv2D(filters=64, kernel_size=(3,3), padding='same', activation='relu'))
model.add(keras.layers.Conv2D(filters=64, kernel_size=(3,3), padding='same', activation='relu'))
model.add(keras.layers.MaxPooling2D(pool_size=(2,2), strides=(2,2)))

model.add(keras.layers.Conv2D(filters=128, kernel_size=(3,3), padding='same', activation='relu'))
model.add(keras.layers.Conv2D(filters=128, kernel_size=(3,3), padding='same', activation='relu'))
model.add(keras.layers.MaxPooling2D(pool_size=(2,2), strides=(2,2)))

model.add(keras.layers.Conv2D(filters=256, kernel_size=(3,3), padding='same', activation='relu'))
model.add(keras.layers.Conv2D(filters=256, kernel_size=(3,3), padding='same', activation='relu'))
model.add(keras.layers.Conv2D(filters=256, kernel_size=(3,3), padding='same', activation='relu'))
model.add(keras.layers.Conv2D(filters=256, kernel_size=(3,3), padding='same', activation='relu'))
model.add(keras.layers.MaxPooling2D(pool_size=(2,2), strides=(2,2)))

```

위의 VGG19 모델을 보고 Hidden Layer 를 추가합니다

```

# DNN
model.add(keras.layers.Flatten())
model.add(keras.layers.Dense(units=4096, activation='relu'))
model.add(keras.layers.Dropout(0.5))
model.add(keras.layers.Dense(units=4096, activation='relu'))
model.add(keras.layers.Dropout(0.5))
output_layer = keras.layers.Dense(units=1_000, activation='softmax')
model.add(output_layer)

model.compile(optimizer='Adam', loss='categorical_crossentropy', metrics=['accuracy'])

model.load_weights('vgg19_weights_tf_dim_ordering_tf_kernels.h5')
model.summary()

```

Flatten 을 통해 2D 데이터를 1D 로 변환하고, Dense Layer 를 통해 분류 작업을 수행합니다.

Softmax Activation 을 통해 최종적인 예측 값을 출력합니다.

Model: "VGG19_NET"

| Layer (type) | Output Shape | Param # |
|--------------------------------|-----------------------|---------|
| conv2d (Conv2D) | (None, 244, 244, 64) | 1,792 |
| conv2d_1 (Conv2D) | (None, 244, 244, 64) | 36,928 |
| max_pooling2d (MaxPooling2D) | (None, 122, 122, 64) | 0 |
| conv2d_2 (Conv2D) | (None, 122, 122, 128) | 73,856 |
| conv2d_3 (Conv2D) | (None, 122, 122, 128) | 147,584 |
| max_pooling2d_1 (MaxPooling2D) | (None, 61, 61, 128) | 0 |
| conv2d_4 (Conv2D) | (None, 61, 61, 256) | 295,168 |
| conv2d_5 (Conv2D) | (None, 61, 61, 256) | 590,080 |
| conv2d_6 (Conv2D) | (None, 61, 61, 256) | 590,080 |

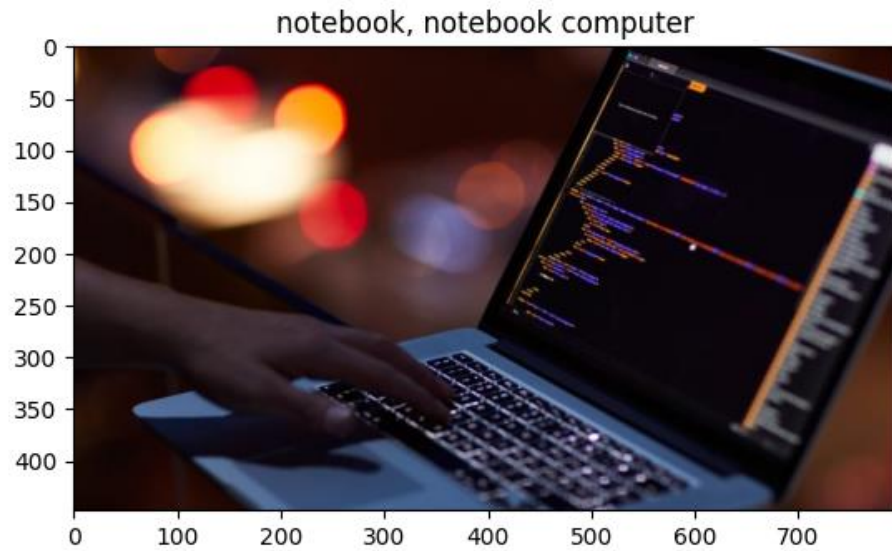
Summary 함수로 Layer 가 잘 형성됐는지 확인할 수 있습니다.

```
y_predict = model.predict(x=image1)
print(y_predict)
finding_key = np.argmax(y_predict)
print(np.argmax(y_predict))
print(f"예상되는 그림은 : {classes[finding_key]}")
```

7.01091701e-07 4.41839447e-05 3.01021188e-05 2.09782303e-06
2.39215751e-06 8.17913860e-06 4.89422348e-07 2.65638464e-05
1.27559099e-06 1.42455076e-06 2.82302972e-06 7.00482303e-07
1.54300460e-06 1.16403908e-06 7.34817093e-07 1.63691539e-06
3.26639156e-06 3.95033453e-07 1.09748307e-05 4.40545846e-05]]
681
예상되는 그림은 : notebook, notebook computer

예상된 이미지를 출력합니다

Figure 1



(x, y) = (215, 44.)
[248, 94, 32]

소감

VGG19 와 같은 강력한 모델을 활용하여 이미지 분류 성능을 극대화하는 방법을 배우며, 이미지 처리 및 모델의 활용 가능성에 대한 이해도가 크게 향상되었습니다. 이 경험을 통해 더 복잡한 컴퓨터 비전 문제를 해결할 수 있는 자신감을 얻었고, 향후 다양한 모델을 적용해 보는 데에 도움이 될 것 같습니다.

포트폴리오

| | |
|------------|--|
| 프로젝트명 | 머신러닝 기반의 Knn 알고리즘을 이용한 강아지 종 판별 프로그램 |
| 프로젝트 기간 | 25.03.25 |
| 상세내용 | <p>사용언어: Python</p> <p>이 코드는 200 개의 랜덤 데이터로 닥스훈트와 사모예드 강아지를 K-최근접 이웃(KNN) 알고리즘을 사용해 분류하는 프로그램입니다. KNN 모델은 각 강아지의 길이와 높이를 바탕으로 강아지의 종을 예측합니다. k=3 는 가장 가까운 3 개 이웃을 참조하여 분류하며, 강아지의 종을 알맞게 출력합니다. 코드에서 생성한 총 400 개의 데이터는 정규 분포(Normal distribution)를 사용하여 생성되었습니다. 즉, 무작위 데이터가 생성되긴 하지만, 닥스훈트와 사모예드의 길이와 높이에 대해 평균과 표준편차를 바탕으로 데이터가 생성됩니다.</p> |
| 초기값 설정 | <pre># 닥스훈트의 길이와 높이 특성 데이터 dachshund_length = [77, 78, 85, 83, 73, 77, 73, 80] dachshund_height = [25, 28, 29, 30, 21, 22, 17, 35] # 사모예드의 길이와 높이 특성 데이터 samoyed_length = [75, 77, 86, 86, 79, 83, 83, 88] samoyed_height = [56, 57, 50, 53, 60, 53, 49, 61]</pre> |

Knn
알고리즘

```
# knn 알고리즘
dachshund_length_mean = np.mean(dachshund_length)
dachshund_height_mean = np.mean(dachshund_height)

samoyed_length_mean = np.mean(dachshund_length)
samoyed_height_mean = np.mean(dachshund_height)

new_dachshund_length_data = np.random.normal(dachshund_length_mean, scale: 10.0, size: 200)
new_dachshund_height_data = np.random.normal(dachshund_height_mean, scale: 10.0, size: 200)

new_samoyed_length_data = np.random.normal(samoyed_length_mean, scale: 10.0, size: 200)
new_samoyed_height_data = np.random.normal(samoyed_height_mean, scale: 10.0, size: 200)

dachshund_data = np.column_stack((new_dachshund_length_data, new_dachshund_height_data))
print(dachshund_data)
print(dachshund_data.shape) # 텐서 보기
dachshund_label = np.zeros(len(dachshund_data)) # 0번으로 설정
print(dachshund_label)

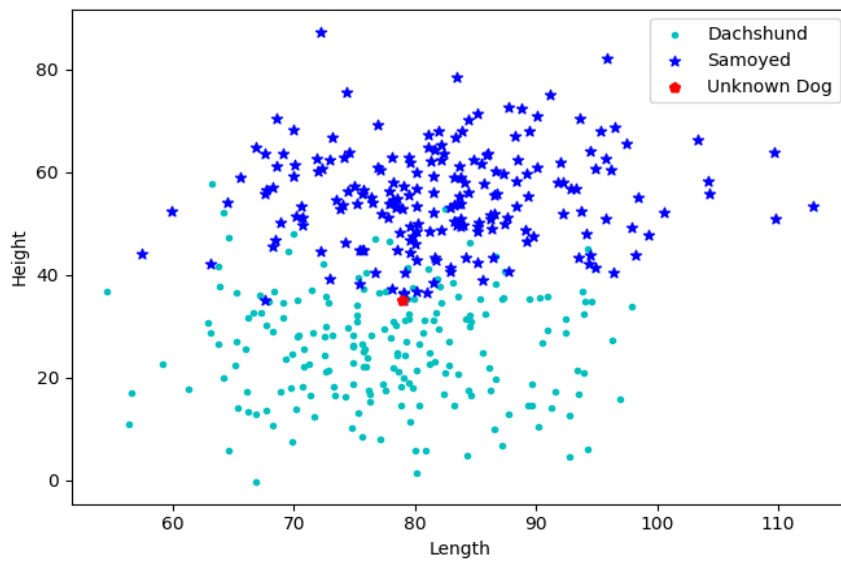
samoyed_data = np.column_stack((new_samoyed_length_data, new_samoyed_height_data))
print(samoyed_data)
print(samoyed_data.shape)
samoyed_label = np.ones(len(samoyed_data)) # 1번으로 설정
print(samoyed_label)

unknown_dog = [[75, 35]]

dogs = np.concat( arrays: (dachshund_data, samoyed_data), axis=0)
labels = np.concat( arrays: (dachshund_label, samoyed_label), axis=0)
print(dogs)
print(dogs.shape) # 행렬 타입
print(labels)
print(labels.shape) # 벡터 타입
dog_classes = {0: "닥스훈트", 1: "사모예드"}
```

사모예드와 닥스훈트 레이블 추가하는 코드입니다

분포도



8 개는 너무 적은 샘플이므로 정규 분포로 데이터를 각각 200 개로 정규 분포를 이용해 만들었습니다

결과

```
from sklearn.neighbors import kneighbors_graph, KNeighborsClassifier
k = 3
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(X=dogs, y=labels)
print(knn.classes_) # 클래스 레이블 표시
y_predict = knn.predict(unknown_dog)
print(y_predict) # 넘파이형태
print(f"이 강아지는 : {dog_classes[y_predict[0]]}") # 넘파이에서 값만 꺼내기
```

(400,)

[0.]

이 강아지는 : 닥스훈트

입력 된 데이터를 바탕으로 닥스훈트라고 판단함.

포트폴리오

| | |
|------------|---|
| 프로젝트명 | CNN 을 이용한 딥러닝기반의 손글씨예측 프로그램 |
| 프로젝트 기간 | 2025.03.28 |
| 상세내용 | <ol style="list-style-type: none"> 1. 개발 언어: python, tensorflow 2. 설명: MNIST 데이터셋을 불러와 전처리한 후, CNN 모델을 학습하거나 저장된 모델을 불러옵니다. 손글씨 숫자 이미지를 28x28 크기로 조정하고, 색상을 반전한 후 정규화하여 모델에 입력합니다. 모델이 숫자를 예측하면 확률 값을 출력하고, 가장 높은 확률을 가진 숫자를 최종적으로 추론합니다 |
| 알고리즘 | <pre> # MNIST 데이터셋 로드 (X_train, y_train), (X_test, y_test) = load_data() # print(X_train.shape) # print(X_test.shape) # 데이터 모양 변경 X_train = X_train.reshape((-1, 28, 28, 1)) X_test = X_test.reshape((-1, 28, 28, 1)) # print(X_train.shape) # print(X_test.shape) # 데이터 정규화 X_train = X_train / 255.0 X_test = X_test / 255.0 print(X_test[0]) </pre> <p>손글씨를 인식하기 위해 MNIST 데이터를 불러오고 정규화합니다.</p> <pre> # 학습 model_path = "2025_03_27_CNN.keras" if os.path.exists(model_path): print("저장된 모델을 불러옵니다...") model = keras.models.load_model(model_path) print(f"예측 정확도 : {model.evaluate(x=X_test, y=y_test)}") else: model = keras.Sequential(name="CNN") input_layer = keras.Input(shape=(28,28,1), name="InputLayer") model.add(input_layer) model.add(keras.layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu', name="conv2D_1")) model.add(keras.layers.MaxPooling2D(pool_size=(2,2), name="MaxPool2D_1")) model.add(keras.layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu', name="conv2D_2")) model.add(keras.layers.MaxPooling2D(pool_size=(2,2), name="MaxPool2D_2")) model.add(keras.layers.Conv2D(filters=32, kernel_size=(3, 3), activation='relu', name="conv2D_3")) model.add(keras.layers.MaxPooling2D(pool_size=(2,2), name="MaxPool2D_3")) model.add(keras.layers.Flatten()) model.add(keras.layers.Dense(units=64, activation='relu', name='HiddenLayer1')) model.add(keras.layers.Dense(units=64, activation='relu', name='HiddenLayer2')) model.add(keras.layers.Dense(units=32, activation='relu', name='HiddenLayer3')) model.add(keras.layers.Dense(units=10, activation='softmax', name='OutputLayer')) model.summary() model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy']) model.fit(x=X_train, y=y_train, epochs=10, verbose='auto') print(f"예측 정확도 : {model.evaluate(x=X_test, y=y_test)}") </pre> <p>CNN 신경망을 이용해 이미지의 특징을 추출합니다. 그 후에 Dense layer 에서 숫자를</p> |

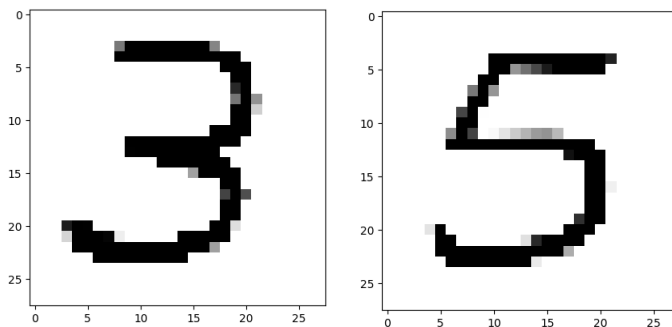
분류해서 Softmax 활성화 함수를 이용하여 가장 높은 확률의 숫자를 출력합니다.

```
import cv2 as cv
# 새로운 이미지 예측
original = cv.imread('test8.png', cv.IMREAD_GRAYSCALE)
image = cv.resize(original, dsize=(28, 28))
plt.imshow(image, cmap="gray")
plt.show()
# 정규화
image = cv.bitwise_not(image) #반전 추가
image = image.astype('float32') / 255.0
image = image.reshape(1, 28, 28, 1)
#숫자 예측
predict_image = model.predict(image, batch_size=1)
print(f'그림 이미지 값은 : {predict_image}')
print(f'추정된 숫자는 : {predict_image.argmax()}')
```

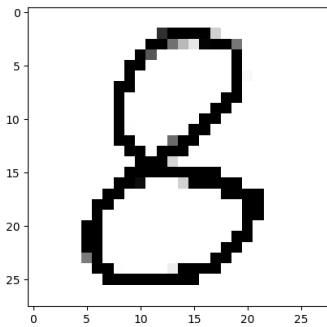
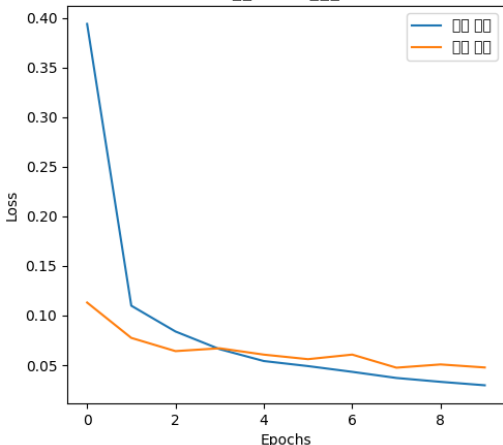
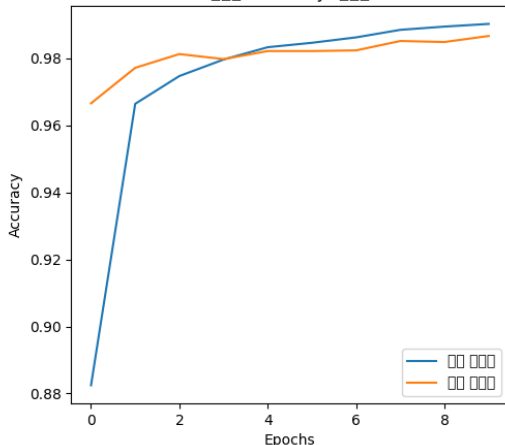
새로운 이미지를 불러와서 정규화를 거친 뒤, 숫자를 예측합니다.

결과

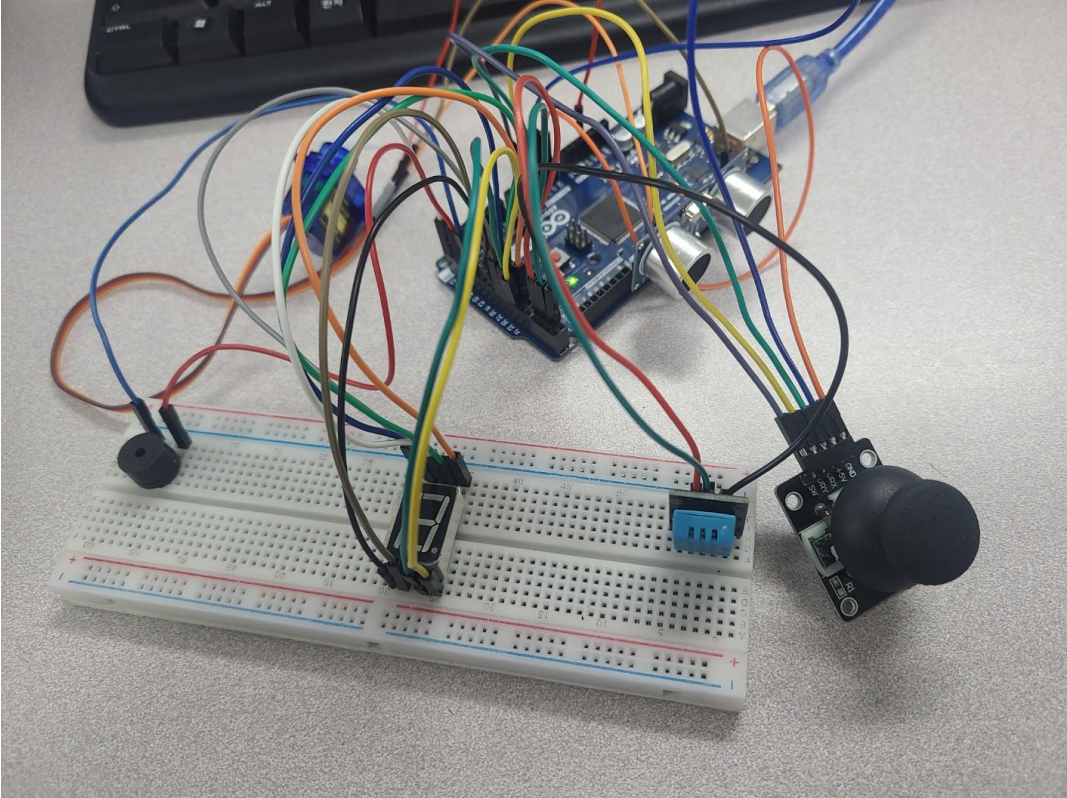
```
313/313 ————— 1s 2ms/step - accuracy: 0.9816 - loss: 0.0704
예측 정확도 : [0.057233329862356186, 0.9843000173568726]
1/1 ————— 0s 88ms/step
그림 이미지 값은 : [[4.8789103e-16 3.2060335e-08 2.9970698e-07 9.9999952e-01 1.4027450e-13
 2.7950831e-08 1.3130926e-13 2.1457423e-09 1.6607355e-08 4.3712984e-09]]
추정된 숫자는 : 3
```



```
313/313 ————— 1s 2ms/step - accuracy: 0.9816 - loss: 0.0704
예측 정확도 : [0.057233329862356186, 0.9843000173568726]
1/1 ————— 0s 86ms/step
그림 이미지 값은 : [[1.1917678e-11 4.3466047e-10 7.1932251e-09 1.1695238e-06 1.8747001e-10
 9.9999774e-01 5.3832270e-08 8.0054907e-11 6.3374046e-08 8.9767263e-07]]
추정된 숫자는 : 5
```

| | <div>313/313 ————— 1s 2ms/step - accuracy: 0.9816 - loss: 0.0704</div> <div>예측 정확도 : [0.057233329862356186, 0.9843000173568726]</div> <div>1/1 ————— 0s 114ms/step</div> <div>그림 이미지 값은 : [[1.6336171e-09 2.0099047e-09 1.0274935e-05 2.4741780e-07 5.5903382e-10 4.9506809e-07 5.6955177e-08 1.8425243e-10 9.9998891e-01 1.5419950e-08]]</div> <div>추정된 숫자는 : 8</div> <div></div> <div>프로그램을 실행하면 그림과 같이 불러온 손글씨의 이미지와 예측 정확도, 추정 한 숫자가 출력됩니다.</div> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--------------------|--|---------------------|------------|-----------------|---|------|------|---|------|------|---|------|------|---|------|------|---|------|------|---|------|------|---|------|------|---|------|------|---|------|------|---|------|------|--------|----------------|---------------------|---|------|------|---|------|------|---|------|------|---|------|------|---|------|------|---|------|------|---|------|------|---|------|------|---|------|------|---|------|------|
| Loss Acc 그래프 | <div><div>손글씨(Loss) 손글씨</div><table><tr><th>Epochs</th><th>Train Loss</th><th>Validation Loss</th></tr><tr><td>0</td><td>0.40</td><td>0.12</td></tr><tr><td>1</td><td>0.11</td><td>0.08</td></tr><tr><td>2</td><td>0.09</td><td>0.07</td></tr><tr><td>3</td><td>0.07</td><td>0.07</td></tr><tr><td>4</td><td>0.06</td><td>0.07</td></tr><tr><td>5</td><td>0.05</td><td>0.06</td></tr><tr><td>6</td><td>0.05</td><td>0.06</td></tr><tr><td>7</td><td>0.04</td><td>0.05</td></tr><tr><td>8</td><td>0.04</td><td>0.05</td></tr><tr><td>9</td><td>0.04</td><td>0.05</td></tr></table></div> <div><div>손글씨(Accuracy) 손글씨</div><table><tr><th>Epochs</th><th>Train Accuracy</th><th>Validation Accuracy</th></tr><tr><td>0</td><td>0.88</td><td>0.97</td></tr><tr><td>1</td><td>0.97</td><td>0.98</td></tr><tr><td>2</td><td>0.98</td><td>0.98</td></tr><tr><td>3</td><td>0.98</td><td>0.98</td></tr><tr><td>4</td><td>0.99</td><td>0.98</td></tr><tr><td>5</td><td>0.99</td><td>0.98</td></tr><tr><td>6</td><td>0.99</td><td>0.98</td></tr><tr><td>7</td><td>0.99</td><td>0.99</td></tr><tr><td>8</td><td>0.99</td><td>0.99</td></tr><tr><td>9</td><td>0.99</td><td>0.99</td></tr></table></div> | Epochs | Train Loss | Validation Loss | 0 | 0.40 | 0.12 | 1 | 0.11 | 0.08 | 2 | 0.09 | 0.07 | 3 | 0.07 | 0.07 | 4 | 0.06 | 0.07 | 5 | 0.05 | 0.06 | 6 | 0.05 | 0.06 | 7 | 0.04 | 0.05 | 8 | 0.04 | 0.05 | 9 | 0.04 | 0.05 | Epochs | Train Accuracy | Validation Accuracy | 0 | 0.88 | 0.97 | 1 | 0.97 | 0.98 | 2 | 0.98 | 0.98 | 3 | 0.98 | 0.98 | 4 | 0.99 | 0.98 | 5 | 0.99 | 0.98 | 6 | 0.99 | 0.98 | 7 | 0.99 | 0.99 | 8 | 0.99 | 0.99 | 9 | 0.99 | 0.99 |
| Epochs | Train Loss | Validation Loss | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | 0.40 | 0.12 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 0.11 | 0.08 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | 0.09 | 0.07 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | 0.07 | 0.07 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 4 | 0.06 | 0.07 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 5 | 0.05 | 0.06 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 6 | 0.05 | 0.06 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 7 | 0.04 | 0.05 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 8 | 0.04 | 0.05 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 9 | 0.04 | 0.05 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Epochs | Train Accuracy | Validation Accuracy | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | 0.88 | 0.97 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 0.97 | 0.98 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | 0.98 | 0.98 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | 0.98 | 0.98 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 4 | 0.99 | 0.98 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 5 | 0.99 | 0.98 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 6 | 0.99 | 0.98 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 7 | 0.99 | 0.99 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 8 | 0.99 | 0.99 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 9 | 0.99 | 0.99 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 소감 | <p>이 프로젝트에서는 CNN 모델을 사용하여 MNIST 데이터셋에서 손글씨 숫자를 분류하는 작업을 했습니다. 데이터를 전처리하고, 모델을 학습시키고, 기존 모델을 로드하여 예측하는 과정을 통해, 딥러닝 모델이 이미지 데이터를 어떻게 처리하고 학습하는지에 대한 깊은 이해를 얻을 수 있었습니다. 특히, 데이터 전처리 과정에서 이미지 크기 조정, 색상 반전, 정규화 등의 기술이 모델 성능에 중요한 영향을 미친다는 점을 알게 되었습니다.</p> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

포트폴리오

| | |
|----------------|---|
| 프로젝트명 | Joystick Control Hub |
| 프로젝트 기간 | 2025.03.05 ~ 2025.03.07 |
| 상세내용 | <p>1. 개발 언어: C++</p> <p>2. 설명: 아두이노에 장착된 조이스틱을 조작하여 여러가지 장치들을 동작 시킬 수 있습니다. 조이스틱은 상, 하, 좌, 우, 스위치버튼 5 가지 동작 방법이 있습니다.</p> <p>오른쪽으로 조작하면 초음파 센서가 동작하며 센서 앞의 장애물과 센서에서의 거리를 계산하여 모니터에 표기합니다.</p> <p>왼쪽으로 조작하면 부저가 동작하며 경고음이 울립니다.</p> <p>위로 조작하면 온습도 센서가 작동하며 모니터에 현재 온도와 습도가 출력됩니다.</p> <p>아래로 조작하면 서브모터가 작동하며 작성된 코드에 따라 회전하게 됩니다.</p> <p>조이스틱을 누르면 스위치가 작동하며 작성된 코드에 따라 세그먼트에 숫자가 표기됩니다.</p> |
| 실제 구현 사진 |  |

| | |
|-----------|---|
| 플로우 차트 | |
| 결과 | <div>Temperature : 30.20[c] Humidyty : 59.00[%] X: 4 Y: -5 OFF</div> <div>X: 4 Y: 0 OFF 36.21cm</div> <div>X: 4 Y: 0 OFF 7.58cm</div> |
| 소감 | 이 프로젝트는 다양한 센서와 장치를 조이스틱을 통해 제어하는 재미있는 경험이었습니다. 각 기능을 구현하며 아두이노와 센서의 작동 원리를 깊이 이해할 수 있었습니다 |