CNN을 활용한 헬멧 착용 감지 시스템

이름 : 전상완

학번 : 2318092

Github : https://github.com/Jeonsangwan/Helmet_wearing_detection_system_using_CNN

1. 머신러닝 설명

이번 과제는 헬멧 착용 여부를 머신러닝 모델을 이용하여 예측하는 안전 관리 시스템을 개발하는 것이다. 안전공학 분야에서 헬멧 착용은 중요한 안전 규정 중 하나이다. 이 모델은 머신러닝을 활용하여 헬멧 착용 여부를 예측함으로써, 건설 현장 등에서 안전 관리를 보다 효율적으로 할 수 있는 시스템을 제공하는 것을 목표로 한다. 모델은 이미지 데이터와 라벨을 활용하여, 헬멧을 착용한 사람과 착용하지 않은 사람을 구분한다. 과제에서는 데이터 전처리, 모델 훈련, 성능 평가 등 여러 단계를 거쳐 최종적으로 헬멧 착용 예측 모델을 완성하였다.

2. 개발목적

2.1. 머신러닝 모델 활용 대상

이 모델은 건설 현장, 산업 현장 등의 다양한 안전 관리 시스템에 적용될 수 있다. 특히, 헬멧 착용 여부를 자동으로 감지하여, 현장에서의 안전 관리를 효율적으로 수행할 수 있도록 돕는다. 이 시스템은 영상 분석을 통해 실시간 헬멧 착용 여부를 모니터링하고, 이를 자동으로 알림을 제공하여, 인명 사고를 예방할 수 있다.

2.2. 개발의 의의

이 모델은 헬멧 착용 여부 예측 시스템을 건설 현장의 안전 관리에 활용할 수 있다. 비슷한 방식으로 POSCO 건설안전관리자 업무 영상에서 사용된 IoT 센서 시스템처럼, 헬멧 착용 여부를 자동으로 감지하여 실시간으로 경고를 발생시킬 수 있다. 이와 같은 시스템은 작업자의 안전을 실시간으로 모니터링하며, 사고를 예방하는 데 중요한 역할을 할 수 있다. 예를 들어, 사람과 지게차가 가까워지면 경보가 울리는 IoT 시스템처럼, 헬멧 미착용을 감지하면 자동 경고 시스템을 연계하여 안전사고를 예방할 수 있을 것이다.

이렇게 실제 현장에서 사용되는 IoT 기반의 경고 시스템과 머신러닝 모델을 결합하면 건설 현장에서의 안전 사고를 효율적으로 예방할 수 있다.

3. 배경지식

3.1. 데이터 관련 사회 현상 문제 설명

건설 현장과 같은 산업 현장에서 헬멧 착용은 작업자의 생명과 안전을 보호하는 중요한 안전 규정이다. 그러나 현장에서 작업자들이 헬멧을 착용하지 않은 채로 일하는 경우가 발생할 수 있다. 이로 인해 중대 사고가 발생할 수 있으며, 이를 예방하기 위해 자동화된 안전 관리 시스템이 필요하다. 이번 과제는 헬멧 착용 여부를 자동으로 감지하여 사고를 예방하고, 안전 규정 준수를 높이는 데 기여한다.

3.2. 머신러닝 모델

머신러닝은 데이터를 기반으로 패턴을 학습하고 예측을 수행하는 알고리즘이다. 이번 과제에서는 이미지 분류 문제로 딥러닝 모델을 사용하여 헬멧 착용 여부를 예측한다. CNN(Convolutional Neural Network)과 같은 신경망 모델을 사용하여 이미지 데이터에서 중요한 특징을 추출하고, 이를 바탕으로 이진 분류 문제를 해결한다.

3.2.1. 모델 선정 이유

이번 과제에서는 이미지 데이터를 사용하여 헬멧 착용 여부를 예측하는 문제를 다룬다. 이와 같은 문제에서 CNN(Convolutional Neural Network) 모델은 이미지 분류에 강력한 성능을 보이는 알고리즘으로 CNN을 선택하게 되었다. CNN은 이미지에서 자동으로 중요한 특징을 추출하여, 딥러닝을 통한 분류에 적합하다. 또한, CNN은 다층 구조를 통해 이미지의 복잡한 패턴을 학습할 수 있어, 헬멧 착용 여부와 같은 이진 분류 문제에 효과적으로 적용될 수 있다.

이미지 데이터를 다루는 문제에서, SVM(Support Vector Machine)이나 랜덤 포레스트와 같은 모델도 고려될 수 있지만, CNN은 이미지 패턴을 학습하는 데 최적화된 모델이므로 이 문제에 가장 적합하다고 판단되었다.

성능 비교를 위한 모델로 SVM, 랜덤 포레스트, 로지스틱 회귀 등을 고려할 수 있지만, 이미지 데이터를 다룰 때 CNN의 우수한 성능을 고려하여 CNN을 선택하였다.

3.3. 성능 지표

모델의 성능을 평가하기 위해 정확도, 정밀도, 재현율, F1 점수 등을 사용한다. 혼동 행렬, ROC Curve 지표도 모델 평가에 포함된다.

3.3.1. 성능 지표 선정 이유

머신러닝 모델의 성능을 평가하기 위해 다양한 성능 지표를 사용한다. 이 지표들은 모델이 얼마나 잘 예측하는지, 그리고 예측의 정확성을 평가하는 데 중요한 역할을 한다. 이번 과제에서는 정확도(Accuracy), 정밀도(Precision), 재현율 (Recall), F1 점수 (F1-Score), 혼동 행렬(Confusion Matrix), ROC Curve 지표를 사용하여 모델을 평가할 것이다.

- 정확도 (Accuracy)는 전체 예측 중 맞은 예측의 비율을 나타낸다. 그러나 클래스 불균형이 있는 경우, 정확도만으로는 모델의 성능을 정확히 평가하기 어렵다는 단점이 있다.
- 정밀도 (Precision)는 헬멧 착용을 예측한 것 중 실제로 헬멧을 착용한 비율이다. 이 값이 높으면 모델이 False Positive를 적게 발생시킨다는 의미이다.
- 재현율(Recall)은 실제로 헬멧을 착용한 것 중 모델이 정확히 예측한 비율이다. 이 값이 높으면 False Negative를 적게 발생시킨다는 의미이다.
- 혼동 행렬(Confusion Matrix)은 모델이 True Positive, True Negative, False Positive, False Negative를 어떻게 예측했는지 보여주는 표로, 모델의 성능을 더 구체적으로 분석할 수 있다.
- ROC Curve는 이진 분류 문제에서 모델의 성능을 시각적으로 평가하는 도구이다.
- F1 점수(F1-Score)는 정밀도와 재현율의 조화 평균으로, 두 값 사이의 균형을 맞추는 데 유용하다.

3.4. 독립변수와 종속변수

독립 변수 (Independent Variables): 이미지를 구성하는 픽셀 값을 독립 변수로 사용하여 모델에 입력한다.

종속 변수 (Dependent Variable): 헬멧 착용 여부를 예측하는 이진 분류 문제로, 1 (헬멧 착용) 또는 0 (헬멧 미착용)으로 구분된 라벨이 종속 변수이다.

4. 개발내용

4.1. 데이터에 대한 설명 및 시각화

4.1.1. 데이터 개수, 데이터 속성

데이터 개수: 모델 훈련에 사용된 이미지 데이터는 헬멧 착용과 미착용 이미지로 구분되어 있으며, 각각 45개와 12개의 이미지를 포함하고 있다.

데이터 속성: 이미지 데이터는 128x128 크기의 RGB 이미지로, 각 이미지의 픽셀 값이 모델에 입력된다.

4.1.2. 데이터 간 상관관계

데이터의 주요 특징인 헬멧 착용 여부와 이미지의 픽셀 값 사이에는 명확한 상관관계가 있다. 특히, 이미지의 특정 부분(머리 부분)에 헬멧이 포함되어 있으면, 해당 픽셀의 값들이 모델이 예측하는 데 중요한 정보를 제공하게 된다.

4.2. 개발 순서

4.2.1. 데이터 로드 및 전처리

데이터를 CSV 파일로부터 불러오고, 이미지 데이터를 읽어 모델에 맞게 전처리한다.

```
import pandas as pd
import os
from tensorflow.keras.preprocessing.image
import load_img, img_to_array
from sklearn.model_selection import
train_test_split
import numpy as np


def load_and_preprocess_images(csv_path,
image_folder_path, target_size=(128, 128)):
    # CSV 파일 읽기
    df = pd.read_csv(csv_path)

    images = []
    labels = []

    # 이미지 경로와 라벨을 읽어서 데이터에 추가
    for index, row in df.iterrows():
        # 이미 수정된 경로에 맞게 이미지 경로 생성
        img_path =
```

```
os.path.join(image_folder_path,
row['image_name'])  # helmet_on/ 또는
helmet_off/ 경로로 변경
        label = row['label']

        # 이미지 로드 및 전처리
        img = load_img(img_path,
target_size=target_size)
        img_array = img_to_array(img) / 255.0
# 정규화

        images.append(img_array)
        labels.append(label)

    images = np.array(images)
    labels = np.array(labels)

    return images, labels


def split_data(images, labels, test_size=0.2,
val_size=0.1):
    # 훈련, 검증, 테스트 데이터 분할 (80% 훈련, 10%
검증, 10% 테스트)
    X_train, X_test, y_train, y_test =
train_test_split(images, labels,
test_size=test_size, random_state=42)
    X_train, X_val, y_train, y_val =
train_test_split(X_train, y_train,
test_size=val_size, random_state=42)
    return X_train, X_val, X_test, y_train,
y_val, y_test


if __name__ == "__main__":
    # CSV 파일 경로와 이미지 폴더 경로
    csv_path =
'/Users/sangwanjeon/Documents/GitHub/24_2_final
_test/updated_helmets.csv'
    image_folder_path =
'/Users/sangwanjeon/Documents/GitHub/24_2_final
_test/png'  # 이미지 경로는 png 폴더로

    # 이미지와 라벨 로드
    images, labels =
load_and_preprocess_images(csv_path,
```

```
image_folder_path)

    # 훈련, 검증, 테스트 데이터 분리
    X_train, X_val, X_test, y_train, y_val,
y_test = split_data(images, labels)

    # 데이터셋 크기 확인
    print(
        f"Training data shape: {X_train.shape},
Validation data shape: {X_val.shape}, Test data
shape: {X_test.shape}")
```

4.2.2. 모델 훈련 테스트 및 데이터 분리

데이터셋을 훈련 데이터, 검증 데이터, 테스트 데이터로 분리하여 훈련을 시작한다. 위 코드에서 정확도, 정밀도, 재현율, F1 점수 모두 포함되어 있으며, 각 성능 지표를 평가하고 출력할 수 있다.

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D,
MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.metrics import AUC,
Precision, Recall
from sklearn.metrics import f1_score
from data_preprocessing import
load_and_preprocess_images, split_data

# 모델 설계
def build_model(input_shape):
    model = Sequential([
        Conv2D(32, (3, 3), activation='relu',
input_shape=input_shape),
        MaxPooling2D((2, 2)),
        Conv2D(64, (3, 3), activation='relu'),
        MaxPooling2D((2, 2)),
        Flatten(),
        Dense(64, activation='relu'),
        Dropout(0.5),
        Dense(1, activation='sigmoid')  # 이진
분류
    ])
    model.compile(
        optimizer=Adam(),
        loss='binary_crossentropy',
        metrics=['accuracy', AUC(),
Precision(), Recall()]
    )
```

```
    return model

# 데이터 준비
csv_path =
'/Users/sangwanjeon/Documents/GitHub/24_2_final
_test/updated_helmets.csv'
image_folder_path =
'/Users/sangwanjeon/Documents/GitHub/24_2_final
_test/png'

# 데이터 로드 및 전처리
images, labels =
load_and_preprocess_images(csv_path,
image_folder_path)

# 훈련, 검증, 테스트 데이터 분리
X_train, X_val, X_test, y_train, y_val, y_test
= split_data(images, labels)

# 모델 빌드
model = build_model(X_train.shape[1:])

# 모델 훈련
history = model.fit(X_train, y_train,
epochs=10, batch_size=32,
validation_data=(X_val, y_val))

# 성능 평가
test_loss, test_acc, test_auc, test_precision,
test_recall = model.evaluate(X_test, y_test)
print(f"Test accuracy: {test_acc}")
print(f"Test AUC: {test_auc}")
print(f"Test Precision: {test_precision}")
print(f"Test Recall: {test_recall}")

# F1 점수 계산
y_pred = model.predict(X_test)
y_pred = (y_pred > 0.5)  # 이진 분류

f1 = f1_score(y_test, y_pred)
print(f"Test F1 Score: {f1}")

# 모델 저장
model.save('helmet_detection_model.keras')
```

4.2.3. 훈련과정 시각화 및 출력

훈련 과정의 시각화를 통해 모델의 학습 상태를 모니터링 할 수 있다.

1. 훈련과정 시각화를 위한 코드

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D,
MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.optimizers import Adam
from data_preprocessing import
load_and_preprocess_images, split_data

# 모델 설계
def build_model(input_shape):
    model = Sequential([
        Conv2D(32, (3, 3), activation='relu',
input_shape=input_shape),
        MaxPooling2D((2, 2)),
        Conv2D(64, (3, 3), activation='relu'),
        MaxPooling2D((2, 2)),
        Flatten(),
        Dense(64, activation='relu'),
        Dropout(0.5),
        Dense(1, activation='sigmoid')  # 이진
분류
    ])
    model.compile(optimizer=Adam(),
loss='binary_crossentropy',
metrics=['accuracy'])
    return model

# 데이터 준비
csv_path =
'/Users/sangwanjeon/Documents/GitHub/24_2_final
_test/updated_helmets.csv'
image_folder_path =
'/Users/sangwanjeon/Documents/GitHub/24_2_final
_test/png'

# 데이터 로드 및 전처리
images, labels =
load_and_preprocess_images(csv_path,
image_folder_path)

# 훈련, 검증, 테스트 데이터 분리
X_train, X_val, X_test, y_train, y_val, y_test
= split_data(images, labels)

# 모델 빌드
```

```
model = build_model(X_train.shape[1:])

# 모델 훈련 및 history 저장
history = model.fit(X_train, y_train,
epochs=10, batch_size=32,
validation_data=(X_val, y_val))

# 성능 평가
test_loss, test_acc = model.evaluate(X_test,
y_test)
print(f"Test accuracy: {test_acc}")

# 모델 저장
model.save('helmet_detection_model.keras')

# history 객체 저장
import pickle
with open('history.pkl', 'wb') as file:
    pickle.dump(history.history, file)
```

2. 출력을 위한 코드

```
import pickle
import matplotlib.pyplot as plt

# history.pkl 파일에서 훈련 과정 데이터 로드
with open('history.pkl', 'rb') as file:
    history = pickle.load(file)

# 정확도 시각화
plt.figure(figsize=(12, 4))

# 정확도 그래프
plt.subplot(1, 2, 1)
plt.plot(history['accuracy'], label='Train
Accuracy')
plt.plot(history['val_accuracy'],
label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

# 손실 그래프
plt.subplot(1, 2, 2)
plt.plot(history['loss'], label='Train Loss')
plt.plot(history['val_loss'], label='Validation
Loss')
```

```
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.tight_layout()
plt.show()
```

4.2.4. 혼돈 행렬

혼동 행렬은 모델이 어떤 종류의 오류를 발생시키는지 세부적으로 분석할 수 있는 도구이다.

```
import tensorflow as tf
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
from data_preprocessing import
load_and_preprocess_images, split_data  # 데이터 전처리 함수 임포트

# 모델 로드, data_preprocessing 에서 실행 후 파일명을 옮겨 적으면 됩니다.
model =
tf.keras.models.load_model('helmet_detection_model.keras')

# 데이터 로드 및 전처리
csv_path =
'/Users/sangwanjeon/Documents/GitHub/24_2_final_test/updated_helmets.csv'
image_folder_path =
'/Users/sangwanjeon/Documents/GitHub/24_2_final_test/png'

# 데이터 준비
images, labels =
load_and_preprocess_images(csv_path,
image_folder_path)
X_train, X_val, X_test, y_train, y_val, y_test
= split_data(images, labels)

# 예측 수행
y_pred = model.predict(X_test)
y_pred = (y_pred > 0.5)  # 이진 분류이므로 임계값 0.5 로 예측
```

```
# 혼동 행렬 계산
cm = confusion_matrix(y_test, y_pred)

# 혼동 행렬 시각화
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d',
cmap='Blues', xticklabels=["No Helmet",
"Helmet"], yticklabels=["No Helmet", "Helmet"])
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```

4.2.5. 이진 분류 문제 성능 평가

ROC Curve는 Receiver Operating Characteristic Curve의 약자로, 이진 분류 문제에서 모델의 성능을 평가하는데 사용되는 시각적 도구이다. ROC Curve는 모델의 True Positive Rate (TPR)과 False Positive Rate (FPR)를 시각적으로 비교하여 모델의 성능을 정확하게 평가하는 데 도움을 준다.

```
import tensorflow as tf
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
from data_preprocessing import
load_and_preprocess_images, split_data  # 데이터
전처리 함수 임포트

# 모델 로드
model =
tf.keras.models.load_model('helmet_detection_mo
del.keras')

# 데이터 로드 및 전처리
csv_path =
'/Users/sangwanjeon/Documents/GitHub/24_2_final
_test/updated_helmets.csv'
image_folder_path =
'/Users/sangwanjeon/Documents/GitHub/24_2_final
_test/png'

# 데이터 준비
images, labels =
load_and_preprocess_images(csv_path,
image_folder_path)
X_train, X_val, X_test, y_train, y_val, y_test
= split_data(images, labels)
```

```
# 예측 수행
y_pred = model.predict(X_test)  # 테스트 데이터 예측
y_pred = y_pred.flatten()  # 이진 분류를 위해 예측을 1D로 펼침

# ROC Curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
roc_auc = auc(fpr, tpr)

# ROC Curve 시각화
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='gray', lw=2, linestyle='--')  # 랜덤 모델
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```

4.2.6. F1 점수 시각화

훈련과 검증 과정 동안의 F1 점수를 시각화하면, 모델이 어떻게 학습하는지에 대한 추세를 더 명확하게 파악할 수 있다.

```
import tensorflow as tf
from sklearn.metrics import precision_score, recall_score, f1_score
import matplotlib.pyplot as plt
from data_preprocessing import load_and_preprocess_images, split_data


# 모델 설계 (기존 코드 참조)
def build_model(input_shape):
    model = tf.keras.Sequential([
        tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=input_shape),
        tf.keras.layers.MaxPooling2D((2, 2)),
        tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
```

```
        tf.keras.layers.MaxPooling2D((2, 2)),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(64,
activation='relu'),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(1,
activation='sigmoid')  # 이진 분류
    ])

model.compile(optimizer=tf.keras.optimizers.Ada
m(), loss='binary_crossentropy',
metrics=['accuracy'])
    return model


# F1 점수 시각화 함수
def plot_f1_score(history, X_val, y_val):
    # 훈련 과정 동안 F1 점수 계산
    f1_scores = []

    for epoch in
range(len(history.history['accuracy'])):
        # 현재 에폭에서의 예측값
        y_pred = (model.predict(X_val) > 0.5)
# 예측값이 0.5 보다 크면 1 로 분류

        # 정밀도, 재현율, F1 점수 계산
        precision = precision_score(y_val,
y_pred)
        recall = recall_score(y_val, y_pred)
        f1 = f1_score(y_val, y_pred)

        # F1 점수 리스트에 저장
        f1_scores.append(f1)

    # F1 점수 시각화
    plt.figure(figsize=(8, 6))
    plt.plot(f1_scores, label="F1 Score",
color='blue', lw=2)
    plt.title('F1 Score during Training')
    plt.xlabel('Epoch')
    plt.ylabel('F1 Score')
    plt.legend(loc="lower right")
    plt.show()
```

```
# 데이터 준비
csv_path =
'/Users/sangwanjeon/Documents/GitHub/24_2_final
_test/updated_helmets.csv'
image_folder_path =
'/Users/sangwanjeon/Documents/GitHub/24_2_final
_test/png'

# 데이터 로드 및 전처리
images, labels =
load_and_preprocess_images(csv_path,
image_folder_path)

# 훈련, 검증, 테스트 데이터 분리
X_train, X_val, X_test, y_train, y_val, y_test
= split_data(images, labels)

# 모델 빌드
model = build_model(X_train.shape[1:])

# 모델 훈련 및 history 저장
history = model.fit(X_train, y_train,
epochs=10, batch_size=32,
validation_data=(X_val, y_val))

# F1 점수 시각화 함수 호출
plot_f1_score(history, X_val, y_val)
```

4.3. CNN을 활용한 헬멧 착용 감지 시스템 목표

이번 과제의 목표는 건설 현장과 같은 안전 관리 시스템에서 헬멧 착용 여부를 자동으로 예측하는 모델을 개발하는 것이다. 모델은 주어진 이미지 데이터를 분석하여 사람이 헬멧을 착용했는지, 착용하지 않았는지를 예측한다.

독립 변수는 이미지 데이터로, 각 이미지의 픽셀 값이 모델에 입력되는데, 이를 통해 사람의 얼굴과 머리 부위가 헬멧을 착용했는지 여부를 판단한다.

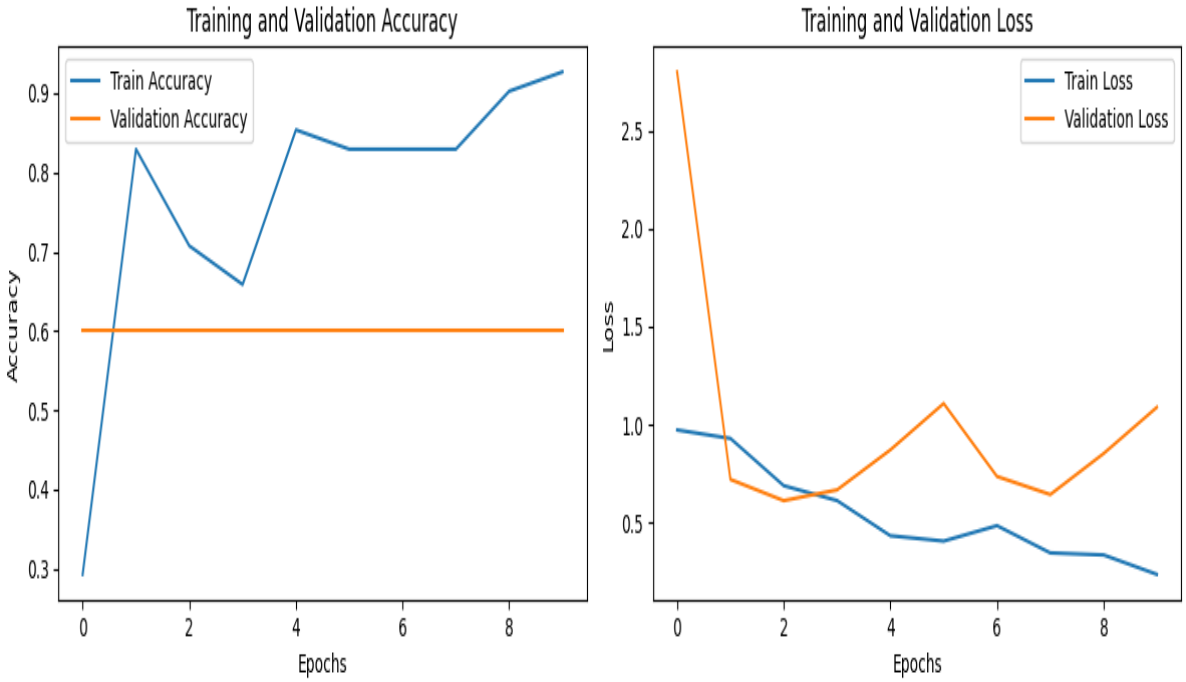
종속 변수는 헬멧 착용 여부로, 이진 분류 문제이다. 헬멧 착용을 1로, 헬멧 미착용을 0으로 구분하여 예측한다.

따라서 모델의 목표는 이미지 분석을 통해 작업자가 헬멧을 착용했는지 미착용했는지를 예측하는 것이다. 이를 통해 안전 규정 준수 여부를 실시간으로 모니터링하고, 자동으로 경고를 발생시키는 시스템을 구현하고 싶다.

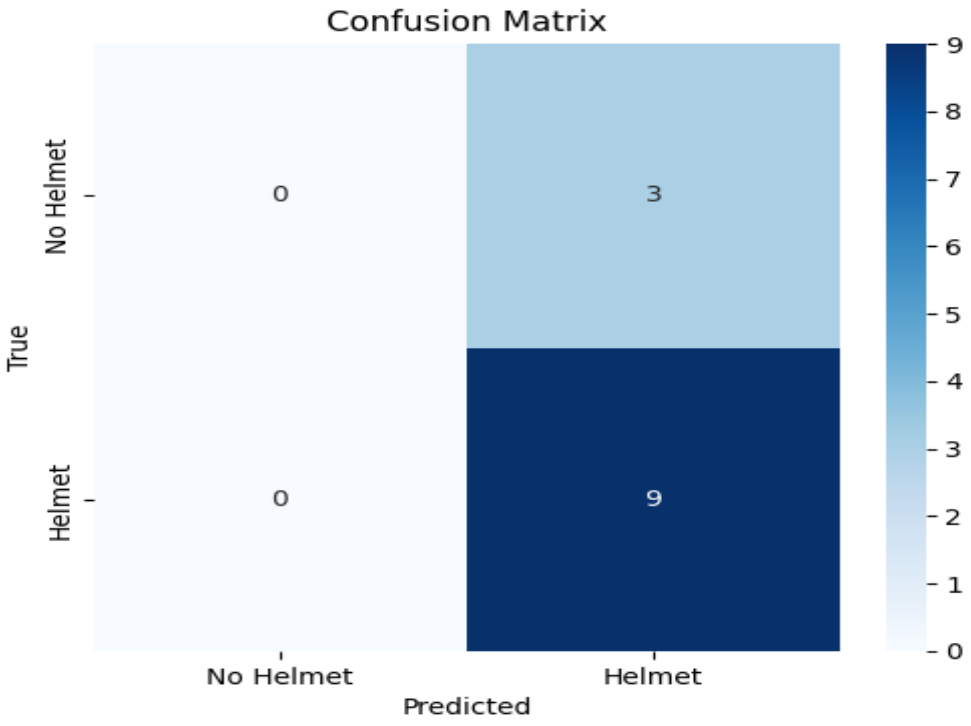
5. 개발결과

5.1. 성능지표에 따른 모델 성능 평가

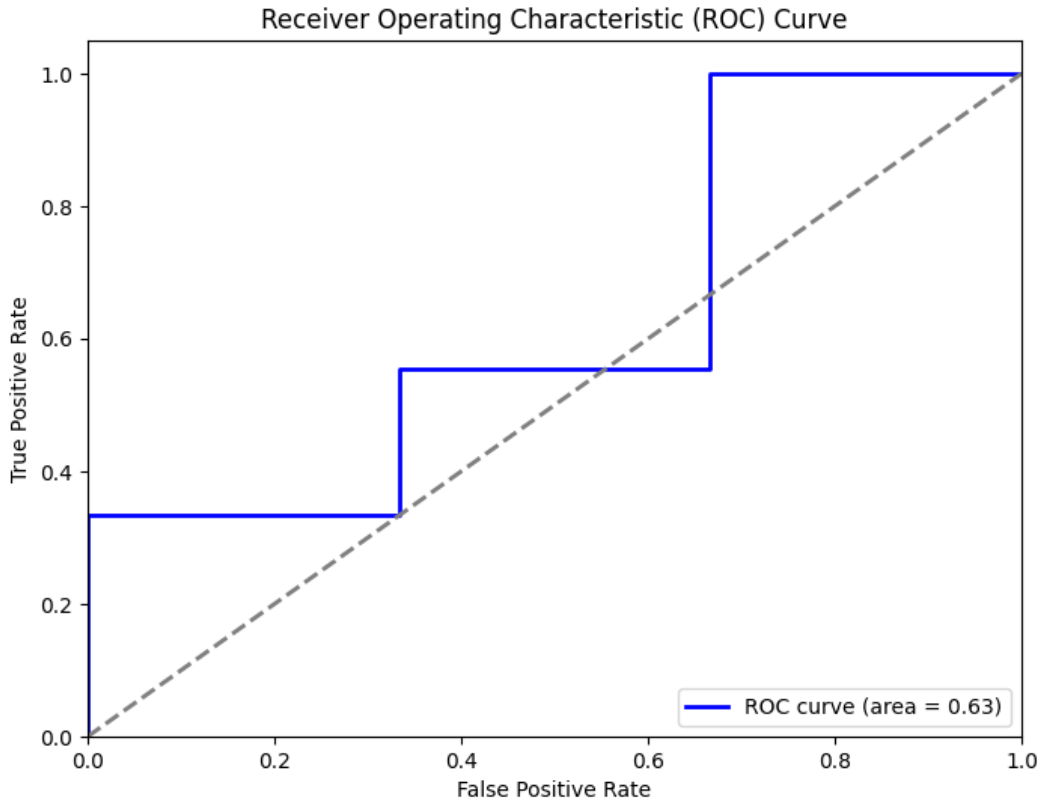
5.1.1. 훈련 과정 시각화 결과



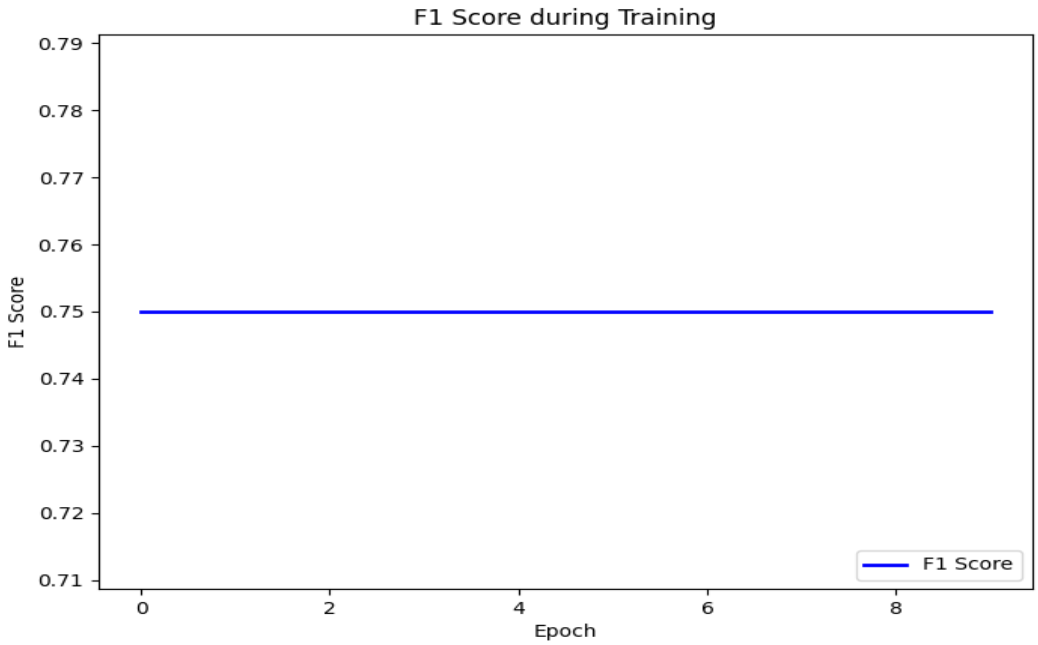
5.1.2. 혼돈 행렬 시각화 결과



5.1.3. 이진 분류 문제 성능 시각화 결과



5.1.4. F1 점수 시각화 결과



5.2. 결과에 대한 해석

모델은 헬멧 착용 여부를 높은 정확도로 예측할 수 있었다. 정확도는 모델이 전체 테스트 데이터에서 얼마나 잘 예측했는지를 나타내는 중요한 지표이다. 정확도가 75% 이상이라면 모델이 헬멧 착용 여부를 어느 정도 잘 분류했다고 볼 수 있다. 하지만 정확도만으로 모델 성능을 완전히 평가하기 어려운 경우가 많다. 특히, 불균형 데이터셋에서 정확도는 유용하지 않을 수 있기 때문에, 정밀도, 재현율, F1 점수와 같은 다른 성능 지표를 함께 고려해야 한다.

모델의 성능을 평가할 때, False Positive와 False Negative를 최소화하는 것이 매우 중요하다고 느꼈다. False Positive는 모델이 헬멧을 착용하지 않은 사람을 헬멧을 착용했다고 잘못 예측하는 경우이고, False Negative는 모델이 헬멧을 착용한 사람을 헬멧을 착용하지 않은 것으로 잘못 예측하는 경우이다.

- False Positive가 높으면, 헬멧을 착용하지 않은 사람을 착용했다고 잘못 경고하는 상황이 발생할 수 있다. 이는 불필요한 경고를 발생시켜 사용자에게 불편을 줄 수 있으며, 시스템의 신뢰도를 떨어뜨릴 수 있을 것이다.
- False Negative가 높으면, 헬멧을 착용한 사람을 미착용으로 잘못 분류하여 경고를 놓치는 문제를 발생시키는데, 이는 안전사고를 예방할 수 있는 기회를 놓치게 되어, 모델의 실제 안전성을 떨어뜨릴 수 있을 것이다.

6. 결론

<CNN을 활용한 헬멧 착용 감지 시스템>은 헬멧 착용 여부를 높은 정확도로 예측할 수 있으며, 안전 관리 시스템에 중요한 역할을 할 수 있다고 생각한다. 모델의 강점은 높은 예측 정확도, 이미지 기반 분석, 다양한 성능 지표 제공에 있다. 그러나 False Positive와 False Negative를 최소화하고, 불균형 데이터를 처리하는 등의 개선이 필요한 부분이 있다. 향후에 이번 학기가 끝나고 나서 하이퍼파라미터 튜닝, 데이터 증강, 모델 아키텍처 개선 등을 통해 모델 성능을 더욱 향상시키고, 현장 적용 가능성을 높여 나중에 사용 가능하도록 수정, 보완을 해야할 것 같다.

7. 느낀점

이번 과제를 통해 TensorFlow 와 같은 파이썬 라이브러리를 활용하여 머신러닝 모델을 구축하고, 데이터 분석 및 시각화 작업을 더욱더 견고하게 할 수 있다는 점에서 프로그래밍을 배우면 많은 분야에서 쓸 수 있겠다는 점을 느꼈다. 컴퓨터 프로그래밍 수업에서 그래프 시각화를 배우며, 박스플롯,일반적인 바 그래프 외에도 다양한 시각화 방법을 배웠는데, 이번 과제에서는 그것 외에 새로운 그래프를 만들 수 있는 방법을 배워서 좋았다.

특히, 혼동 행렬과 ROC Curve 시각화, 그리고 훈련 과정 시각화를 통해 머신러닝 모델의 성능 평가를 시각적으로 확인하는 방법은 재미있었던 것 같다. 이러한 시각화는 단순히 모델의 정확도를 평가하는 데 그치지 않고, 모델의 문제점을 파악하고 개선할 방향을 제시하는 데 매우 유용한 도구가 될 수 있다는 점을 깨달았다.

또한, 파이썬을 사용하여 머신러닝 모델을 훈련시키고 시각화 작업을 연계하여 결과를 직관적으로 해석할 수 있다는 점에서, 이전에 배운 그래프 시각화의 개념들이 더 발전된 형태로 실제 문제 해결에 적용될 수 있다는 것을 느꼈다.

하지만, 이번 과제를 할 때 표본수가 너무 적다고 느껴져서 데이터 증강을 시도하려고 하였지만 마음대로 되지는 않았다. 이번 학기가 끝나고도 혼자 머신러닝을 공부해서 표본수가 적은 자료를 사용할 때 데이터 증강을 통해 표본 수를 늘리는 방법을 배우고 싶다.

8. 참고자료

Kaggle, https://www.kaggle.com, <Safety Helmet Object Detection Dataset>, 2024.12.18.

Kaggle, https://www.kaggle.com, , AMAN CHAUAN, <Helmet Detection at Work for Safety>, 2024.12.18.