**Candidate Name:** Kanimozhi M

**Project: Smart Helmet Detection System with Automatic Fine Notification**

**Code:**

import os

import numpy as np

import cv2

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.utils import to\_categorical

with\_helmet\_images = ["//content//with1.jpeg", "//content//with2.jpeg", "//content//with3.jpeg", "//content//with4.jpeg", "//content//with5.jpeg"]

without\_helmet\_images = ["//content//without1.jpeg", "//content//without2.jpeg", "//content//without 3.jpeg", "//content//without4.jpeg", "//content//without5.jpeg"]

def load\_and\_preprocess\_image(image\_path, resize\_shape=(64, 64)):

    img = cv2.imread(image\_path)  # Read the image

    img = cv2.resize(img, resize\_shape)  # Resize to the desired shape for the model

    img = img.astype('float32') / 255.0  # Normalize the image

    return img

images = []

labels = []

for image\_path in with\_helmet\_images:

    img = load\_and\_preprocess\_image(image\_path)

    images.append(img)

    labels.append(1)

for image\_path in without\_helmet\_images:

    img = load\_and\_preprocess\_image(image\_path)

    images.append(img)

    labels.append(0)

X = np.array(images)

y = np.array(labels)

y = to\_categorical(y, num\_classes=2)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.8, random\_state=42)

model = Sequential([

    Conv2D(32, (3, 3), activation='relu', input\_shape=(64, 64, 3)),

    MaxPooling2D(pool\_size=(2, 2)),

    Conv2D(64, (3, 3), activation='relu'),

    MaxPooling2D(pool\_size=(2, 2)),

    Conv2D(128, (3, 3), activation='relu'),

    MaxPooling2D(pool\_size=(2, 2)),

    Flatten(),

    Dense(128, activation='relu'),

    Dense(2, activation='softmax')

])

model.compile(optimizer=Adam(), loss='categorical\_crossentropy', metrics=['accuracy'])

history = model.fit(X\_train, y\_train, epochs=10, batch\_size=2, validation\_data=(X\_test, y\_test), verbose=0)

def predict\_image(image\_path):

    img = load\_and\_preprocess\_image(image\_path)

    img = np.expand\_dims(img, axis=0)

    prediction = model.predict(img)

    label = np.argmax(prediction)

    labels = ["Without Helmet", "With Helmet"]

    predicted\_label = labels[label]

    print(f"Prediction: {predicted\_label}")

    if predicted\_label == "Without Helmet":

        vehicle\_number = input("Enter the vehicle number: ")

        fine\_amount = 100

        print(f"Sending message to the responsible person: 'Fine applied for vehicle {vehicle\_number}. Fine amount: rs{fine\_amount}'")

image\_path = input("Enter the image path to predict: ")

predict\_image(image\_path)

**Dataset:**

<https://drive.google.com/drive/folders/179r3tPqgW7rYxp-PaH1d6rMEmIPbCYEI?usp=drive_link>