DON BOSCO INSTITUTE OF TECHNOLOGY

Kumbalagodu, Mysore Road, Bengaluru – 560074

www.dbit.co.in ph: +91-80-28437028/29/30 Fax: +91-80-28437031

Together for Tomorrow! Enabling People

Education for Future Generations

Department of Computer Science & Engineering

Under DBIT-SIC

Title: "Distracted Driver Detection"

Team Members:

1) Shashank Gowda R [1DB20CS098]

2) Usha N [1DB20CS121]

Abstract:

This research leverages computer vision and deep learning techniques to achieve real-time identification of distractions and tiredness in the context of driving safety improvement. A multifaceted system has been developed, encompassing drowsiness detection, hands-on-the-wheel detection, and drive distraction detection. Each of these elements plays a pivotal role in sustaining the driver's focus and attentiveness throughout the journey.

A deep learning model, created with TensorFlow and Keras, is employed to detect distractions in real time by analysing visual data from the driver and the vehicle's surroundings. It identifies behaviors like phone usage and inattention, while hand detection on the steering wheel, implemented using OpenCV and 'cvzone,' monitors hand positions for issuing alerts and guidance. Additionally, the system tracks drowsiness by analyzing blink patterns and facial landmarks with Dlib, issuing real-time warnings to maintain driver alertness and road safety.

Problem Statement

Distracted driving and driver fatigue are significant threats to road safety, with distractions like phone use and inattentiveness and fatigue during long or late-night trips causing accidents. Current solutions often rely on subjective judgment and unreliable technology.

Objectives:

1. **Distraction Detection:** Employ deep learning to identify and categorize distractions in real-time video data, issuing immediate visual or auditory alerts to the driver when distractions like phone use, eating, or inattention are detected.

2. **Drowsiness Detection:** Implement an algorithm that continuously tracks the driver's eye blink patterns to detect drowsiness, providing timely warnings to keep the driver awake and attentive.

3. **Hands-on-the-Wheel Detection**: Utilize computer vision to monitor the driver's hand position on the wheel, issuing real-time guidance and warnings to ensure safe vehicle control and operational instructions.

4. **Real-Time Feedback**: Integrate distraction, drowsiness, and hand detection technologies to provide real-time feedback to the driver, ensuring it is user-friendly and minimally distracting.

5. **Safety Enhancement**: Promote safe hand positioning techniques, particularly for novice drivers, to reduce accidents and injuries caused by distractions and fatigue.

6. **Scalability and Integrity**: Design a versatile system that can be integrated into various vehicles, making it suitable for both consumer and commercial cars, thereby enhancing road safety.

7. **Testing and Validation**: Rigorously test and validate the distraction, drowsiness, and hand detection modules under diverse driving conditions to ensure accuracy and reliability.

8. **Future Planning**: Consider potential enhancements, including integration with advanced driver-assistance systems (ADAS), and explore data analytics and reporting options for further improving driving safety.

Libraries Used

NumPy

- NumPy is a fundamental library for numerical and array operations in Python. It provides support for multi-dimensional arrays and various mathematical functions.
- NumPy is used for the efficient handling and processing of numerical data and arrays in various aspects of your project.

Pandas

- Pandas is a Python library for data manipulation and analysis. It offers data structures like DataFrames and Series, making it easier to work with structured data.
- Pandas are used for managing and processing data, especially for data preprocessing and analytics.

os

- The 'os' module is part of Python's standard library and provides functions to interact with the operating system. It allows us to perform various file and directory operations.
- 'os' is used for tasks such as file manipulation, directory traversal, and handling file paths in our project

cv2 (OpenCV)

- OpenCV (Open Source Computer Vision Library) is a comprehensive library for computer vision tasks, including image and video processing, object detection, and feature extraction.
- OpenCV is essential for face and eye detection, hand tracking, and image preprocessing.

TensorFlow

- TensorFlow is a popular deep-learning framework developed by Google. It is widely used for creating and training machine learning and deep learning models.
- TensorFlow is used to build and train the distraction detection model in your project.

Adam

- Adam is an optimization algorithm used in training deep learning models. It is an extension of stochastic gradient descent (SGD) and is effective for minimizing loss functions during model training.
- Adam is utilized as the optimization algorithm when compiling the deep learning model in your project.

image_dataset_from_directory

- 'image_dataset_from_directory' is a function provided by TensorFlow that simplifies the process of creating image datasets from directories. It's used for data preparation in deep learning tasks.
- This function is used to load and preprocess image data for training and validation.

Conv2D, MaxPooling2D, Dense, Flatten, Rescaling, Dropout

- These are layers and operations commonly used in deep neural networks. 'Conv2D' and 'MaxPooling2D' are for convolution and pooling, 'Dense' is for fully connected layers, 'Flatten' reshapes data, 'Rescaling' normalizes pixel values, and 'Dropout' prevents overfitting.
- These layers and operations are integral to the design and architecture of your deep learning model.

EarlyStopping

- 'EarlyStopping' is a callback in Keras (a part of TensorFlow) that allows early stopping of model training when a monitored metric stops improving, preventing overfitting.
- It is employed to halt training when validation accuracy stops improving in your project.

joblib

- Joblib is a library for lightweight pipelining in Python. It's used for saving and loading machine learning models, especially for non-Python data types.
- 'joblib' may be used to save and load trained models or data preprocessing pipelines.

dlib

- Dlib is a C++ library for various computer vision tasks, including facial landmark detection, image processing, and machine learning.
- Dlib is critical for detecting facial landmarks and monitoring eye blink patterns for drowsiness detection.

face_utils

- 'imutils' is a library for simplifying computer vision and image processing tasks in OpenCV. 'face_utils' includes utility functions for working with facial features.
- Usage in Project: 'face_utils' is used to facilitate face landmark detection and analysis.

GlobalAveragePooling2D, ImageDataGenerator, MobileNet

- These are specific components and pre-trained models available in Keras/TensorFlow. 'GlobalAveragePooling2D' computes the average output, 'ImageDataGenerator' augments image data, and 'MobileNet' is a pre-trained deep learning model.
- They can be part of the deep learning model architecture, enhancing its capabilities.

cvzone

- 'cvzone' is a Python library designed to simplify computer vision tasks, making hand tracking and other computer vision tasks more accessible and user-friendly.
- 'cvzone' is used for hands detection on the steering wheel, streamlining the process of hand tracking and position monitoring.

HandTrackingModule

- 'HandTrackingModule' is a module from the 'cvzone' library that specializes in hand tracking and detecting hand gestures.
- This module plays a central role in monitoring and guiding the driver's hand positioning on the steering wheel.

Description of Coding

Model building: The constructed deep learning model follows a sequential architecture, serving as an image classifier for identifying distracted driving behaviors. It commences with data preprocessing, specifically rescaling the input images to normalize pixel values within the [0, 1] range. The subsequent layers encompass Conv2D for feature extraction, MaxPooling2D for spatial dimension reduction, and Dropout layers to curb overfitting. The Flatten layer reshapes the data for fully connected layers, which include units of 1024, 512, and 256, each employing ReLU activation. The conclusive layer, activated by softmax, delivers the final classification, with 10 neurons representing diverse distracted driving behaviors. This model efficiently extracts image features and accurately categorizes these behaviors, significantly contributing to the enhancement of road safety.

Drowsiness Detection

Detecting drowsiness and driver distraction through facial landmarks. It uses OpenCV, dlib, and imutils libraries to process a video stream. The program identifies faces in the video, tracks facial landmarks and measures eye blink ratios. Based on the blink ratio, it categorizes the driver's state into three categories: awake, drowsy, or asleep. It then displays this status on the video feed and visually marks facial landmarks. The code effectively tracks and monitors the driver's condition, making it a valuable component in a driver monitoring system for improving road safety and preventing accidents.

Hands Detection

Hands Detection module demonstrates a driver assistance system utilizing hand tracking and computer vision. It captures video from a camera feed and uses the "cvzone" library for hand tracking, counting the number of hands detected. Initially, if two hands are detected, it prompts the message "Engine is started, you can drive. Happy Journey." If no hands are present, it

displays a message advising the driver to place their hands on the steering wheel. Based on the count of detected hands, it dynamically updates the message, indicating whether the driver can accelerate at the maximum speed, drive at a reduced speed of 25 km/h with one hand, or if the brakes are being applied with no hands detected. This system assists drivers in adhering to safe driving practices by providing real-time feedback through the camera feed.

Code

Distracted Driver Detection Model building code:

```
import numpy as np
import pandas as pd
import os
import cv2 as cv
from tqdm.notebook import tqdm
import tensorflow as tf
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import image_dataset_from_directory
from tensorflow.keras.layers import
Conv2D,MaxPooling2D,Dense,Flatten,Rescaling,Dropout
from tensorflow.keras.callbacks import EarlyStopping
train_data = tf.keras.utils.image_dataset_from_directory(
  '/content/drive/MyDrive/Project/Dataset/Train',
  validation_split=0.2,
  subset="training",
  seed=123,
  image_size=(100, 100),
  batch_size=128,label_mode='categorical',)
val_data = tf.keras.utils.image_dataset_from_directory(
  '/content/drive/MyDrive/Project/Dataset/Train',
  validation_split=0.2,
  subset="validation",
  seed=123,
  image_size=(100, 100),
```

```
  batch_size=128,label_mode='categorical',)
classes = train_data.class_names
import matplotlib.pyplot as plt
plt.figure(figsize=(10,10))
for images,labels in train_data.take(1):
    labels = labels.numpy()
    for i in range(25):
        ax = plt.subplot(5, 5, i + 1)
        plt.imshow(images[i].numpy().astype("uint8"))
        plt.title(classes[labels[i].argmax()])
        plt.axis("off")
# creating our model
model = tf.keras.models.Sequential([
    Rescaling(scale = 1/255,input_shape=(100,100,3)),
    Conv2D(32,(3,3),activation='relu'),
    MaxPooling2D((2,2)),
    Dropout(0.1),
    Conv2D(64,(3,3),activation='relu'),
    MaxPooling2D((2,2)),
    Conv2D(32,(3,3),activation='relu'),
    MaxPooling2D((2,2)),
    Dropout(0.1),
    Flatten(),
    Dense(1024,activation='relu'),
    Dropout(0.1),
    Dense(512,activation='relu'),

    Dense(256,activation='relu'),
    Dropout(0.1),
    Dense(10,activation='softmax'),
])
# compiling our model
```

```
model.compile(optimizer = Adam(lr=0.01),loss = 'categorical_crossentropy',metrics=['acc'])
model.summary()
es = EarlyStopping(monitor='val_acc',min_delta=0.01,patience=2)
history = model.fit(train_data,epochs=10,validation_data=val_data)
model.save('/content/drive/MyDrive/Project/Models/model1.h5')
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs_range = history.epoch
plt.figure(figsize=(10,10))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')
plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
```

Hands Detection on a steering wheel

```
import cv2 as cv
from cvzone import HandTrackingModule
video = cv.VideoCapture(0)
detector = HandTrackingModule.HandDetector()
prev_num_hands = -1
count = 0
while True:
    isTrue, frame = video.read()
    hands, img = detector.findHands(frame)
```

```
    num_hands = len(hands)
    if count == 0 and num_hands == 2:
        cv.putText(frame, "Engine is started you can drive. Happy Journey", (50, 50),
cv.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2, cv.LINE_AA)
        count += 1
    if count == 0:
        cv.putText(frame, "Place your hands on the steering to start the engine", (50, 50),
cv.FONT_HERSHEY_SIMPLEX, 0.75, (0, 255, 0), 2, cv.LINE_AA)
    if count > 0:
            if num_hands == 2:
                cv.putText(frame, "You can drive at the max speed", (50, 50),
cv.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2, cv.LINE_AA)
            elif num_hands == 1:
                cv.putText(frame, "You can drive at max speed of 25Km/h", (50, 50),
cv.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2, cv.LINE_AA)
            elif num_hands == 0:
                cv.putText(frame, "Brakes are being applied", (50, 50),
cv.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2, cv.LINE_AA)
    cv.imshow("Hands Detected", frame)
    key = cv.waitKey(10)
    if key & 0xFF == ord("d"):
        break
video.release()
cv.destroyAllWindows()
```

Driver Drowsiness Detection

```
import cv2
import numpy as np
import dlib
from imutils import face_utils
cap = cv2.VideoCapture(0)
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")
sleep = 0
```

```
drowsy = 0
active = 0
status = ""
color = (0, 0, 0)
def compute(ptA, ptB):
    dist = np.linalg.norm(ptA - ptB)
    return dist
def blinked(a, b, c, d, e, f):
    up = compute(b, d) + compute(c, e)
    down = compute(a, f)
    ratio = up / (2.0 * down)
    if ratio > 0.25:
        return 2
    elif 0.21 <= ratio <= 0.25:
        return 1
    else:
        return 0
face_frame = None
while True:
    _, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = detector(gray)
    for face in faces:
        x1 = face.left()
        y1 = face.top()
        x2 = face.right()
        y2 = face.bottom()
        face_frame = frame.copy()
        cv2.rectangle(face_frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
        landmarks = predictor(gray, face)
        landmarks = face_utils.shape_to_np(landmarks)
```

```
        left_blink = blinked(landmarks[36], landmarks[37], landmarks[38], landmarks[41], landmarks[40], landmarks[39])
        right_blink = blinked(landmarks[42], landmarks[43], landmarks[44], landmarks[47], landmarks[46], landmarks[45])
        if left_blink == 0 or right_blink == 0:
            sleep += 1
            drowsy = 0
            active = 0
            if sleep > 6:
                status = "SLEEPING !!!"
                color = (255, 0, 0)
        elif left_blink == 1 or right_blink == 1:
            sleep = 0
            active = 0
            drowsy += 1
            if drowsy > 6:
                status = "Drowsy !"
                color = (0, 0, 255)
        else:
            drowsy = 0
            sleep = 0
            active += 1
            if active > 6:
                status = "Active :)"
                color = (0, 255, 0)
        cv2.putText(frame, status, (100, 100), cv2.FONT_HERSHEY_SIMPLEX, 1.2, color, 3)
        for n in range(0, 68):
            (x, y) = landmarks[n]
            cv2.circle(face_frame, (x, y), 1, (255, 255, 255), -1)
    cv2.imshow("Frame", frame)
    if face_frame is not None:  # Check if 'face_frame' is defined
        cv2.imshow("Result of detector", face_frame)
    key = cv2.waitKey(1)
```

```
if key == 27:
    break
```