TFlite vs. MediaPipe: A Comparative Study

IFS DriverAlert

Date: Jan. 03, 2025

Objective

The purpose of this document is to compare the performance of the two methods to determine which one is best to be used.

Methods

We developed two different methods to achieve the same goals:

- **Method 1:** Uses both TFllite and MediaPipe. TFlite is used for detecting the closed eyes where the MediaPipe is to detect the yawn.
- **Method 2:** Uses Mediapipe for both detecting the closed eyes and detecting the yawn.

Testing Procedure

- Both methods were tested individually on a Raspberry Pi (RPi).
- For each test case, we conducted 10 trials to obtain consistent results.
- Yawn detection results are not included in the table below, as it uses MediaPipe in both methods and provides high and identical results.

Test Case	TFlite + MediaPipe	MediaPipe Only
Eye detection at close distance (less than ½ meter)	10/10	10/10
Eye detection at average distance (1 meter)	9/10	10/10
Eye detection at a far distance ([1.3, 2] meters)	3/10	10/10
Eye detection with glasses	No difference between with/without glasses	No difference between with/without glasses
Eye and yawn detected at same time	Alerts are triggered 3 times	Displayed both alerts correctly
Eye detection in daylight.	10/10	10/10
Eye detection at night.	8/10	10/10
Resource Utilization		
CPU Usage	51.7%	46.2%
Memory Usage	7.2%	7.1%

We can see that the detection of the eye is almost the same for both methods, except for the far distance where the “TFlite+MediaPipe” method struggles to detect the closed eyes. We can also notice that if the eye and the yawn were detected at the same time, the “TFlite+MediaPipe” method is incorrectly triggering the alerts three times, even though the implementation for handling that case is done in the same way in both methods. The issue of the eye and the yawn getting detected at the same time in the first method is because the TFlite model processes frames slower than MediaPipe, causing a delay in synchronizing the detection events. Thus, overlapping detections are misinterpreted, leading to multiple redundant alerts. These results make the MediaPipe method more ideal for our system.

In addition, we compared Resource Utilization to evaluate the efficiency of each method on the Raspberry Pi, as it impacts the system’s performance and ability to run smoothly in a resource constrained environment. From the results, we can see that the MediaPipe method uses slightly less CPU (46.2% vs. 51.7%) and memory (7.1% vs. 7.2%) compared to the “TFlite+MediaPipe” method. While the difference is not significant, the reduced resource usage in the MediaPipe method makes it a more efficient choice.