DriveWise

Authors: Sirisha Brahmandam, Yasser Corzo, and Elinora Stoney

Department of Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—DriveWise is a system that assists drivers with becoming safer on the road. It works by keeping track of a user's eye movements, mouth movements, and overall face position to check for signs of inattention and sleepiness. It will then provide real-time audio feedback to drivers accordingly as well as keep logs of their driving history. We used cameras with computer vision to track the driver's face as well as an accelerometer to determine when the car is in motion and meets conditions that don't require feedback. This device aims to provide more driver-focused feedback as many modern safety features in cars don't currently do that. It will also be able to be easily integrated into most cars so anyone can use the device. The domains in the project will be software, hardware, and signals and systems.**

***Index Terms*—computer vision, driving assistant, embedded system, eye tracking, facial detection, machine learning, web application**

I. Introduction

Close to a million car accidents occur annually due to distracted driving in the United States alone, with young and inexperienced drivers particularly at risk. Most cars on the road today don't give real-time feedback about a driver's attention levels, making it difficult to identify and adjust dangerous behaviors. Even vehicles with preventative features are not particularly driver-focused, reporting on symptoms (ex. irregular car movement) rather than the cause of distracted driving. DriveWise is an automated driver-feedback system that can be integrated easily into any semi-modern car. The device uses facial detection to identify when a driver is inattentive to the road or showing signs of drowsiness. Real-time audio feedback is provided to the driver alerting the driver of their distractedness. The device also works in the cases of low-light conditions and an obstruction to the eyes (hat or sunglasses) with a slightly lower accuracy, while still meeting user requirements. Similar facial detection devices for drivers exist (Cipia, Smart Eye, Eyeware Tech, and Seeing Machines), but they are marketed towards car manufacturers rather than individuals. DriveWise is able to easily integrate into any car that a user currently owns.

II. Use-Case Requirements

To ensure that DriveWise adequately addresses our user's needs, we consider the following use-case requirements.

Clear and concise feedback must be provided when the driver reaches two seconds of inattention. We define inattention as either looking away from the road or exhibiting signs of drowsiness. More specifically, (1) looking anywhere other than the windshield or rear/side view mirrors and (2) yawning and/or having closed eyes.. We chose two seconds as our time limit because the US National Highway Traffic Safety Administration (NHTSA) [4] has determined it is unsafe to look away from the road for greater than two seconds. Clear feedback in these cases will provide the user with an opportunity to notice and correct their behavior.

Inattention detection accuracy should be at least 90% in ideal conditions and 85% in non-ideal conditions. We define ideal conditions as high back lighting and no obstruction of the driver's eyes. In contrast, non-ideal conditions include instances of low lighting and when the driver is wearing sunglasses or a hat. We are considering these non-ideal conditions to more accurately reflect the driving experience. We chose these percentages because similar research papers on facial detection for inattention using convolutional neural networks (CNNs) have reported approximately 95% accuracy for our defined ideal conditions and 90% accuracy for non-ideal conditions [6].

The computation time for detecting inattention should be less than 1 second. Because the NHTSA determines 2 seconds [4] as the dangerous limit for inattention, we want our computation time to be less than one second to ensure there is enough remaining time to deliver feedback before the two second threshold is reached.

The frame rate for the video camera should be at least 5 frames per second (fps). We are using CNNs to classify whether the video stream of the driver's face is showing signs of inattention. An fps of five will allow enough time to use CNNs to give a more accurate classification.

Users should be able to view all of their past data. This is motivated by the fact that users want to observe and monitor their progress to motivate further safe driving practices. We will accomplish this with a web application that the user can log into to view all of their past data.

User data should be private and not shared with other users or insurance companies. In this way we can protect user privacy concerns and prevent negative insurance implications, such as higher rates due to higher inattention levels while driving.

The device should be able to plug into a power source in the user's car. Direct powering from the car avoids the burden of charging in advance or changing batteries and also allows the device to be easily integrated into any semi-modern car. We define any semi-modern car as one with a 12V auxiliary power outlet or any kind of USB power port.

Feedback should not be given while the car is stationary, turning, or reversing. The user does not need to pay as much attention to the road while the car is parked, reversing out of a parking space, at a red light, or looking right/left to make a turn and any audio feedback given during this time would be annoying to the user.

Lastly, **the device should not obstruct the user's view of the road**, as this would jeopardize driver safety.

III. Architecture and/or Principle of Operation

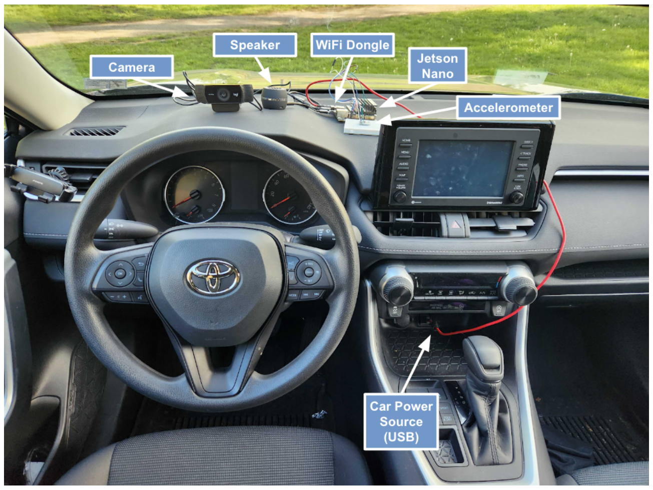


Fig. 1. This is the setup of the DriveWise device in an actual car.

The device is powered by the 12V USB-A port in the user's car, connected to the Jetson via a USB-A to USB-C adaptor. Note that a different adaptor could be used to allow the device to be powered by an auxiliary power outlet instead, but the cars that we are testing the device in have a USB-A port. The Jetson houses all of the machine learning and computer vision capabilities, as well as the classification and feedback logic.

When the car is powered on, the device will immediately turn on and the camera will begin recording a video of the driver's face. At this point, the calibration sequence will be initiated and the user will be instructed (via audio) to look at the center of the camera with their mouth closed for five seconds. After calibration, the device will be enabled to provide feedback once the accelerometer (connected to the Jetson via I2C) detects that the car is moving. All of the computer vision and machine learning algorithms will be stored in the Jetson. Connected to a live camera feed, frames will be sent (once the car is in motion) to the OpenCV DNN module and its artificial neural network where, along with the landmark detector, will help classify whether a driver is distracted or sleepy through eye and mouth detection. As a result, if a driver is determined to be distracted or sleepy, audio feedback will be relayed to the driver through the speaker connected to the Jetson.

The Jetson will send classification data in real-time (over the WiFi dongle connected to the user's phone hotspot) to the Firebase Firestore, which we are using for storage. This Firestore is connected to our web application and users are able to view their own feedback logs once they log in. The web application itself is also hosted on Firebase, with the frontend programmed using React. The overall system is shown in figures 1 and 2, with figure 1 displaying the setup of the device while plugged into a car. See Figure 3 for the more detailed block diagram of the system.

The main changes to our system since the design report were the switch from using the Jetson Xavier AGX to the Jetson Nano, using DLib instead of cnn-facial-landmarks, using a wifi dongle instead of a cellular dongle, getting new adaptors for the different ports on the Jetson Nano, and using Firebase for the backend and hosting instead of AWS.

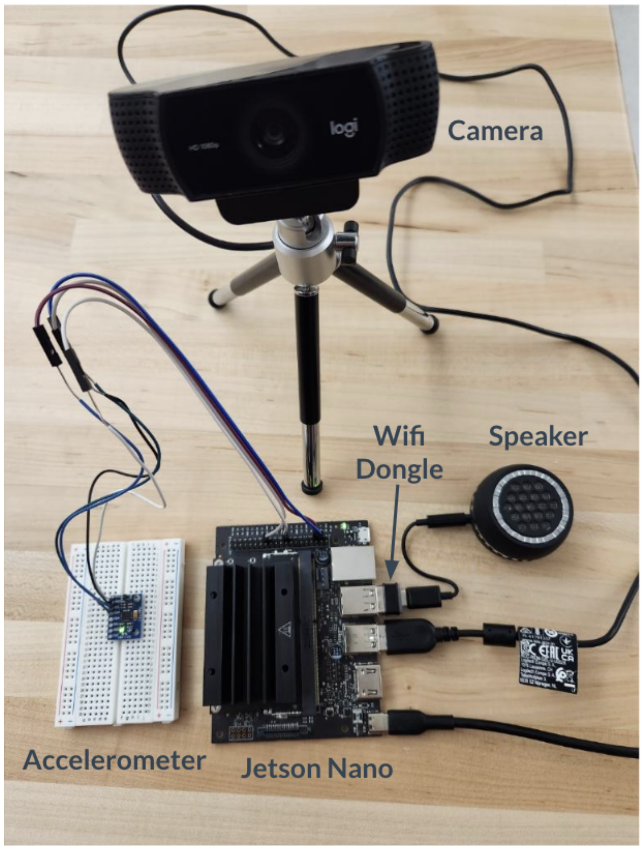


Fig. 2. This is a closer-up labeled diagram of the completed device.

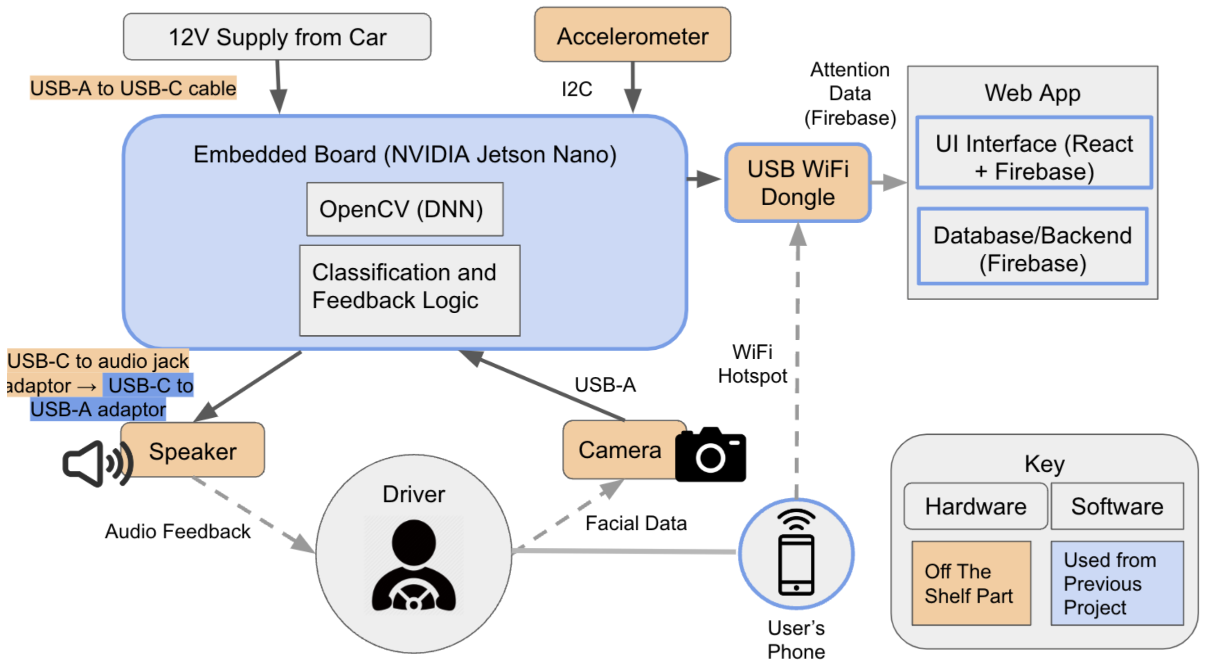

12V Supply from Car
Accelerometer
USB-A to USB-C cable
I2C
Attention Data (Firebase)
Web App
UI Interface (React + Firebase)
Database/Backend (Firebase)
Embedded Board (NVIDIA Jetson Nano)
OpenCV (DNN)
Classification and Feedback Logic
USB WiFi Dongle
USB-C to audio jack adaptor → USB-C to USB-A adaptor
USB-A
WiFi Hotspot
Speaker
Camera
Driver
Audio Feedback
Facial Data
User's Phone
Key
Hardware
Software
Off The Shelf Part
Used from Previous Project

Fig. 3. This is the functional block diagram for the device

IV. Design Requirements

We require the audio feedback to be less than one second long as, according to the National Highway Traffic Safety Administration (NHTSA) [4], looking away from the road for more than two seconds is considered unsafe. With the feedback being given immediately in under one second, this will give the driver enough time to hopefully adjust their behaviors accordingly. As a result, we want to make sure the feedback is clear and concise to make it clear to the driver that changes to their driving behavior are needed.

We require our facial detection algorithm to be unfazed by large head movement. This is necessary for our use case requirement of having the inattention detection accuracy be at least 90% in ideal conditions and at least 85% in non-ideal conditions as having our facial detection algorithm fazed by sudden head movements will affect the overall accuracy of our detection model. This is especially important since similar research studies using CNNs produced 95% accuracy in ideal conditions, ensuring our product does actually promote safe driving practices in an accurate manner.

We require our facial & landmark detection algorithms to complete computation in less than one second. This design requirement targets the use case requirement of having the computation time for detection attention to the road and drowsiness be less than one second. Our justification for this requirement is the NHTSA's research concluding that inattention of the road for more than two second is unsafe [5], therefore having our computation time be less than one second gives enough time for the system to give feedback in under two seconds, ensuring the driver immediately changes behaviors as soon as the two second limit is reached.

We require our camera capture video at a frame rate of at least five fps. This ensures we reach the use case requirement of having a frame rate of five fps because we want to use CNNs as they provide a more accurate facial detection when the user's faces are at different angles. The fps of the camera can then be lowered to around five fps in order to reach our computation time.

For the web app we want to host it and use the storage feature on Firebase. This is required to give the ability to users to view all their past data in a web app to motivate further safe driving practices.

Web app accounts should have a login and be linked to a specific DriveWise device. This is required to address the use case requirement of having user data being private (i.e., not viewable by anyone other than the user). This is needed to address user privacy concerns as not keeping this data private may have negative insurance implications. We are only considering the case that there is one driver per DriveWise device.

The Jetson Nano must be powered by a 12V auxiliary port or USB-A or -C port in the vehicle. This addresses the use case requirement of having the device being able to be plugged into the power source of a user's vehicle. This also eases the user's experience with DriveWise by helping the user avoid the time and burden of recharging and replacing batteries as well as making the device compatible with any semi-modern car.

The device should be under 100x100x150mm as this will not significantly obstruct the driver's view through the windshield in order not to jeopardize the driver's safety.

V. Design Trade Studies

A number of design trade-offs have been made thus far regarding the design process of our system. Specifically, they have been made for the hardware component selection, the computer vision algorithms, the scope of the project, calibration of the device, and the integration of a web application with our physical device and how communication between the two will occur.

A. Hardware Component Selection

Embedded Board: For our embedded board, we ended up choosing the NVIDIA Jetson Nano. For machine learning and computer vision applications, NVIDIA Jetsons are often used due to their relatively compact size and high computation power. Since purchasing our own Jetson for the project would take up a significant portion of our budget, we decided to borrow one from the 18-500 class part inventory. We had the choice between a variety of Jetsons (Xavier NX, AGX Xavier, Nano, and TX2) and since price was no longer an issue, we originally wanted to use the model with the highest computation power (most CPU cores and GPU frequency) to ensure that we are able to meet our design requirement of completing computation/classification in under 1 second. The Jetson AGX Xavier has 8 CPU cores and a 512-core GPU, compared to the 6 CPU cores and 384-core GPU of the Jetson NX Xavier, which is the next closest in computation power. However, we knew that there was a chance of the Jetson AGX Xavier not being able to work with the car, and its setup process was much more complicated than we had envisioned. Because of this, we decided to switch to the Jetson Nano, which would have less computation power (at 4 CPU cores and a 128-core GPU) but require a lower wattage to run and it was much more straightforward to set up.

Camera: Three viable options at our disposal were the C920 Logitech Webcam, ELP 5 USB camera, and the C922 Logitech Webcam, as these were the three cameras mostly used in CV projects online. After further research we decided on the C922 Logitech Webcam because it had the best characteristics for a real-time computer vision camera: low latency, adequate low-light, and reasonable resolution. The ELP camera has a frame rate of 30 fps and latency of around 105 ms while the C920 camera has a frame rate of 30 fps at 720p and a latency of around 250 to 300 ms and the C922 camera has a frame rate of 60 fps at 720p with latency of around 200 to 250 ms and has adequate backlight compensation, working better than the other two in low light conditions. In addition, the C922 comes with a tripod which was needed for the camera to be placed on a car's dashboard. In order to meet our use case requirements of having our audio feedback given in less than 1 second, a low latency is required from our camera. Therefore the C922 camera is a better option.

B. Facial Detection Models

Facial detection is a crucial aspect of this project as it allows us to detect a driver's face, crucial in identifying facial features to determine whether a driver is drowsy or distracted. Different face detection models in Python exist with strengths and weaknesses unique to each, making it critical to choose the best overall model for this project. After researching, we were left with two models, OpenCV's DNN and Dlib frontal face detector, as the most viable options available to us.

Dlib is a C++ toolkit binded to run in python containing machine learning algorithms used to detect faces. The frontal face detector works using features extracted from Histogram of Oriented Gradients (HOG) which are then passed into an SVM [1].

OpenCV's deep neural network is a Caffe model based on the Single Shot-Multibox Detector (SSD) and uses ResNet-10 architecture as its backbone [3].

From our preliminary testing we found that Dlib's frontal face detector was inaccurate when it came to large angled head movements, specifically when it comes to side faces. OpenCV's DNN module was much more accurate. In addition, the frame rate with Dlib was much lower at 5.41 fps while OpenCV's DNN was much higher at 12.95 fps [1]. All in all OpenCV's DNN performed much better than Dlib which is why we have originally decided to use DNN over Dlib. As a result of this decision, we initially used CNN-facial-landmarks [2] as a landmark detector that estimates the location of 68 points that map to facial structures on a user's face instead of the keypoint facial landmark detector from Dlib.

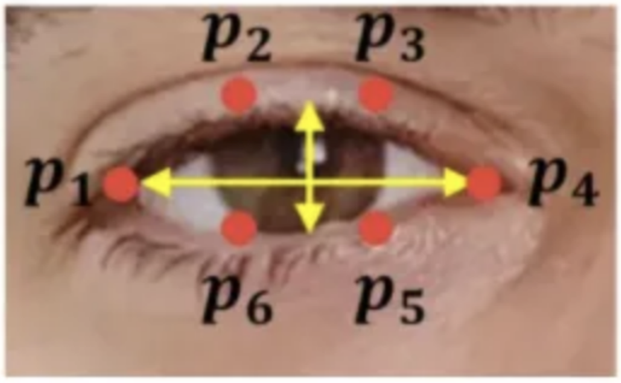


Fig. 4. Eye detection points shown on image.

However, CNN-facial-landmarks was not accurate with tracking eye features and in turn, was not accurate with calculating the eye aspect ratio (EAR) needed for determining whether a driver had their eyes closed (Fig 4). As a result, we first tried using both CNN-facial-landmarks and Dlib, with CNN-facial-landmarks being used for eye tracking, mouth detection, and head pose estimation and Dlib being used in calculating EAR. However, after integrating the features together, fps suffered significantly (reaching approximately 3fps) due to using both models at the same time. With a significant decrease in fps, head pose was not accurately tracked in cases of sudden head movements, causing a significant amount of false negatives. After testing head pose with Dlib, because even though CNN-facial-landmarks had higher accuracy, Dlib was still able to meet the accuracies defined in our use case requirements. After confirming that head pose was still accurate enough using Dlib, we switched the model for eye tracking and mouth detection to Dlib. This change led to a significant increase in fps (from 3fps to 7fps) which in turn increased our overall accuracies.

C. Calibration Sequence

The calibration sequence is important in ensuring that the models we use for detecting signs of distraction and drowsiness are effective for each user, accounting for their facial dimensions and seat position/distance from the camera. Initially, we had planned on including a calibration sequence instructing the user to look at the four corners of their windshield and the two sideview mirrors to determine the thresholds for “safe” viewing. This choice was made because, at that point, we had determined that viewing anywhere within the windshield or either side view mirrors was considered safe.

While looking at the side view mirrors or the edges of the windshield do technically constitute attentiveness of the driver, we have decided that doing so for greater than two seconds (at speeds above the 5mph threshold at which feedback is enabled) would be considered unsafe. In these cases, the driver is looking too far away from the section of the road directly in front of them and exposing themselves to risks of distraction. Due to this new definition of where is safe to look, we altered our calibration process to be based only on the position of the user relative to the camera and a predefined range of what constitutes as looking towards the road.

Our chosen calibration sequence begins when the user starts the car and the user will be instructed (via audio) to look at the center of the camera with their mouth closed for five seconds, after which the device has gained all information needed to customize the facial detection algorithms to the user. By requiring less calibration points, we also improved the ease-of-use of the device and decreased the time needed for calibration at the start of each drive.

D. Web Application

We wanted a way to be able to display information about a user’s driving attention on either a phone or web application. While a phone app may be more convenient for the user, we settled on a web app because our team has more experience with programming web apps using React during prior internships and classes at CMU.

For hosting and storage on the web app, we had originally considered using AWS (EC2 instance and S3 storage bucket) because of our team members’ experience using AWS in past internships. From the start, we had planned on using authentication through Firebase, and after more research learned that Firebase has the capabilities of hosting and cloud storage as well. We ultimately chose to use Firebase for hosting and cloud storage instead of AWS because it was very straightforward to incorporate into our web application and we were already using it for Google OAuth authentication.

Another design decision we had to make regarding the web application was how we would send driver attention data to it from the user’s DriveWise device. Because the device must operate in a moving vehicle that does not have its own WiFi network, we considered two options for sending the data: connecting the Jetson over WiFi to a hotspot on the user’s phone and connecting to the web app or connecting directly to the web app using cellular data. Each of these would require the addition of a card or dongle for WiFi or cellular data. While using cellular data may require less effort on the user’s part, we ultimately settled on the WiFi dongle option because it didn't require the purchase of a cellular data plan.

E. Scope

In regard to the scope of the project, there have been several changes along the way. The most notable one has been how the device would operate under ideal and non-ideal conditions.

We had initially hoped to have the device work to a near perfect accuracy under ideal conditions, which we classified as with good lighting (not extremely bright or direct sunlight and not during the night) and with no obstructions that would drastically alter the computer vision algorithms (items like sunglasses and hats which could prevent the camera from picking up on a driver’s eyes), and a little less than perfect accuracy under nonideal conditions, which means poorer lighting conditions or certain types of obstructions to the face. However, when experimenting with the computer vision algorithms, the difficulty of tracking a user’s eye movements under nonideal conditions became very evident and we opted to go about those situations through detecting a person’s head position instead. This new method required more time and deliberation since it would’ve required potentially using different models which can be an issue when integrating the code, so it became a bit of a stretch goal to include the nonideal conditions. However, we were able to successfully incorporate this feature, and it did require us to make a design trade-off very late into the project and change the models of the other features as already discussed.

VI. System Implementation

A. Calibration

The software components of this project consists of two cycles. When a driver first uses this device, they will be required to complete a calibration process, which they are guided through with audio instructions played over the device's speaker. Afterwards, the driver will be classified as being drowsy or distracted using computer vision and machine learning algorithms. This process is summarized in Figure 4.

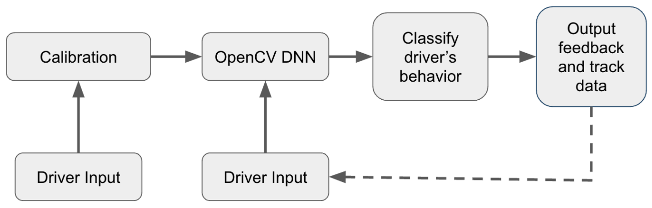


Fig. 5. This is the software specification diagram

The calibration process entails the device reading a driver's facial dimensions to help better determine whether a driver is distracted or sleepy. In particular, we are measuring the distance between the lips when the mouth is closed as this will serve as a point of reference for a non-distracted/non-drowsy driver. The calibration process will also help to ensure that adequate background light is available as this will enable our computer vision and machine learning algorithms to correctly determine whether a driver is distracted or sleepy.

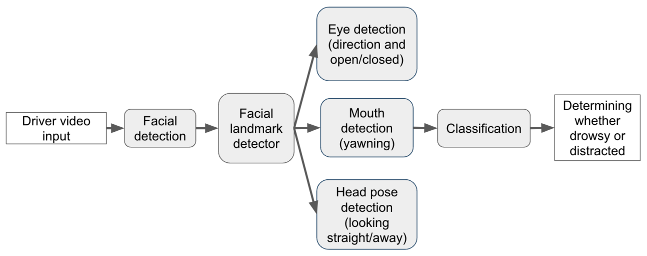

Eye detection (direction and open/closed)
Driver video input
Facial detection
Facial landmark detector
Mouth detection (yawning)
Classification
Determining whether drowsy or distracted
Head pose detection (looking straight/away)

Fig. 6. Block Diagram for Computer Vision Algorithm

B. *Computer Vision*

The computer vision aspect of this project will allow us to classify whether a driver is distracted or sleepy. A vital aspect of this approach is detecting the mouth and eyes of a driver as these facial structures will be used to determine driver awareness. This is a two step press: localizing the face in the image and detecting key facial structures. We accomplish facial localization through face detection and detecting key facial structures through the use of facial landmarks.

In order to obtain facial detection, we use OpenCV and deep learning [4]. In particular, we are using OpenCV's "deep neural networks" (DNN) module which supports a number of deep learning frameworks such as Caffe, TensorFlow, and PyTorch. OpenCV's deep learning face detector is based on the Single Shot Detector (SSD) framework with a ResNet base network.

When using OpenCV's DNN module, a model is required to be imported from various frameworks compatible with DNN. The framework we will use in this project is the Caffe framework. When using OpenCV's DNN module with Caffe models, two sets of files are required: .prototxt and .caffemodel. Prototxt files define the model architecture (i.e. the layers themselves in the neural network) and .caffemodel files contain the weights for the actual layers. Both of these files allow us to read and load a network model from disk by utilizing cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"]) which explains why both files are required when using models trained using Caffe for deep learning. This returns an artificial neural network where we can pass our images as passed as a blob image (through cv2.dnn.blobFromImages['path to image'] where mean subtraction and scaling are used to preprocess these images and prepare them for classification) to obtain detections and predictions of a driver's face. It is important to note that we aren't training a neural network-rather, we are making use of a pre-trained network. Therefore we are just passing the image through the network (i.e. forward propagation) to obtain the result (no back-propagation). Our predictions are represented by localized faces though drawn bounded boxes around a user's face.

Facial landmarking is used to localize and label facial regions such as the eyes and mouth. Therefore, we use this method for eye and mouth detection. In particular, we are using Dlib's facial landmark detection. It is trained on the iBUG 300-W dataset [6]. This model gives 68 landmarks that map to 68 (x, y) coordinates of important facial structures on faces. By passing the Dlib landmark model along with the predicted face detection rectangles to a keypoint detector, we can detect our desired key facial structures in an image.

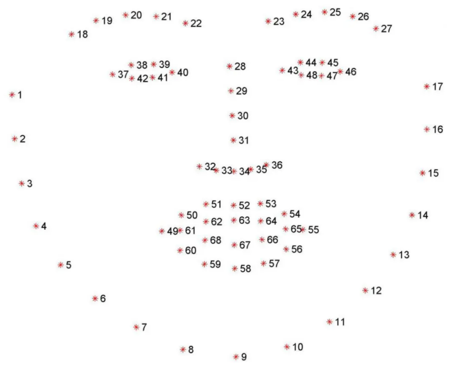


Fig. 7. Visualizing the 68 facial landmark coordinates from the iBUG 300-W dataset

In order to detect the eyes and mouth we must get the (x, y) coordinates from the facial landmark model that corresponds to these facial structures. In addition, we create a black mask using NumPy of the same dimensions as our webcam frame. Using the stored (x, y) coordinates of the eyes and mouth we draw these points on the mask using the OpenCV method .fillConvexPoly. Doing this returns a black mask where the eye and mouth areas are drawn in white. Using bitwise operations on images (provided by methods from OpenCV), we can apply the mask on the image to segment out the eyes and mouth. In addition, thresholding is used to create a binary mask. To obtain a more precise mouth and eye detection, thresholding processing steps, namely erosion, dilation, and median blur will be used. With respect to the eye, the horizontal distance of the eye is measured by the endpoints of the eye (keypoints 37 and 40 in Fig. 8) as well as the vertical distance (yellow dotted line in Fig. 8). From these distances, we can calculate the horizontal and vertical threshold boundaries. We classify whether a driver is looking away from the road when the pupil crosses the threshold boundaries (noted as the purple and blue dotted lines in Fig 8). Note that this eye tracking method continues for each frame in the video. With respect to detecting whether a driver's eyes are closed, we check whether the eye aspect ratio falls below 0.20. Note that this method continues for each frame in the video. With respect to the mouth, this method requires us to calculate the distance of the lips and compare this distance with the closed mouth distance recorded in the calibration step. When a driver yawns, their mouth will be open and will have a lip distance greater than that of a closed mouth, indicating the driver is sleepy.

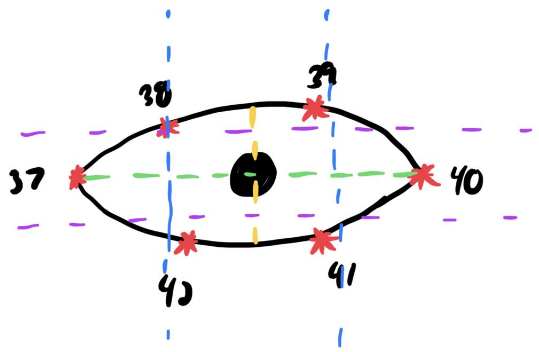


Fig. 8. Diagram of eye-detection points and boundaries

In order to detect the position of the driver's head, six points of the face are needed i.e nose tip, chin, extreme left and right points of lips, and the left corner of the left eye and right corner of the right (Fig 9). The 3D coordinates of these features are taken and using this, the rotational and translational vectors at the nose tip are measured. Using these vectors we can estimate the position of the driver's head by calculating the angle between the tip of the nose and the x and y axis. For classification, there is a preset angle threshold for the vertical and horizontal axis which determines whether a driver is looking up, down, left, or right. This head pose method continues for every frame of the video as well.

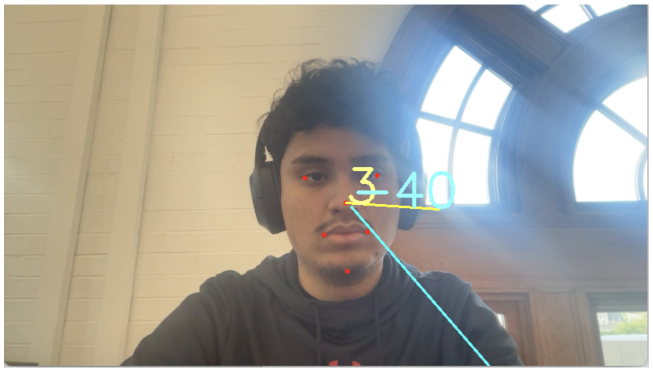


Fig. 9. Head pose estimation visual feedback during testing

C. *Sensor Input and Feedback*

The device will take and process input from two different sensors: an accelerometer and a camera. The accelerometer will report data on the speed of the car and will be essentially used as an on/off switch for the auditory feedback component of the device. Once the accelerometer has detected that the car is moving forward at speeds greater than 5mph, the feedback logic in the Jetson will be able to be triggered when the user has been classified to show signs of inattention. The accelerometer is used in this way to account for situations in which the driver may be safely looking away from the road while parked, stopped at a red light, or reversing (in which case the user could be looking backwards over their shoulder). The camera acts as another sensor, providing images at a rate of 7 fps to be processed as described earlier in the Computer Vision section.

Audio feedback is provided through the speaker plugged into the Jetson. For each of the feedback cases (looking away from the road for two seconds and exhibiting signs of drowsiness) we have pre-made sound files that the Jetson will trigger to play through the speaker when the classification logic determines that the corresponding conditions have been met. Specifically, when distractedness is detected, the device plays "look at the road" and "drowsiness detected" is played in the case of drowsiness.

D. *Web Application*

We are using React.js to program the frontend of the web app. Firebase is used for hosting, cloud storage (Firestore), and authentication for the web app.

When visiting the DriveWise web application, users will first see the login page shown in Fig. 10. Authentication is implemented using Google OAuth through Firebase. The user is prompted to log in directly with a signed-in Google account (via the "Sign in with Google" button) or to manually enter an email and password. Choosing to sign in with a Google account redirects the user to the page shown in Fig. 11. If the user does not already have an account, they will be prompted to add a device ID from their DriveWise device (assuming each device will have a unique ID printed visibly on it) upon logging in for the first time. This device ID will be used to filter Firestore data so the user is only able to view data from their device.

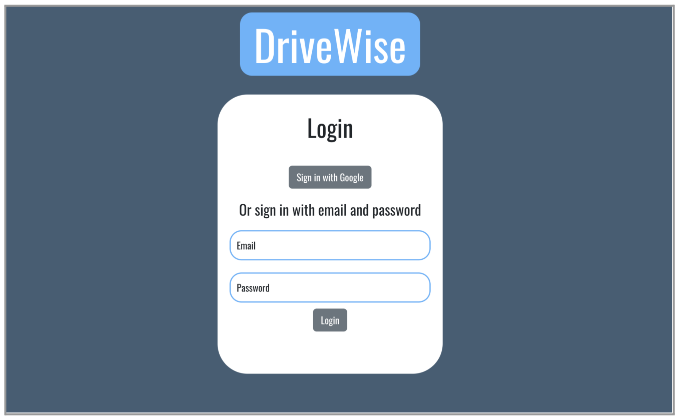
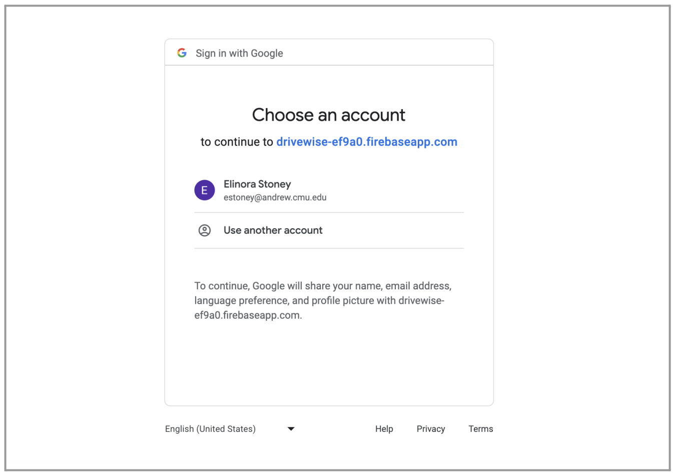


Fig. 10. This is the completed web application login page



Sign in with Google

Choose an account

to continue to drivewise-ef9a0.firebaseapp.com

Elinora Stoney
estoney@andrew.cmu.edu

Use another account

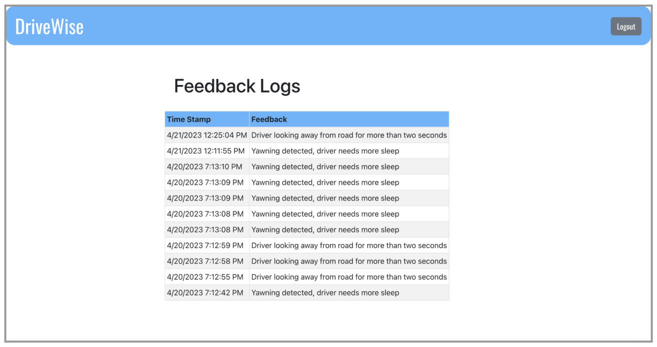
To continue, Google will share your name, email address, language preference, and profile picture with drivewise-ef9a0.firebaseapp.com.

English (United States) Help Privacy Terms

Fig. 11. The Google OAuth page that the user is redirected to by "Sign in with Google" button

After logging in or creating an account, the user will be redirected to the Logs Page (i.e. home page) seen in Fig. 12. This page displays each instance that the DriveWise device has flagged as unsafe driving with a timestamp and the corresponding feedback. These logs will be updated real-time as the device provides feedback in the car.

We had originally planned on including a Metrics Page with statistics related to the percentage of driving time that the user had spent distracted over the past month and over all time. Unfortunately due to integration taking far longer than planned for, we were unable to implement the Metrics page.



Feedback Logs

Time Stamp	Feedback
4/21/2023 12:25:04 PM	Driver looking away from road for more than two seconds
4/21/2023 12:11:55 PM	Yawning detected, driver needs more sleep
4/20/2023 7:13:10 PM	Yawning detected, driver needs more sleep
4/20/2023 7:13:09 PM	Yawning detected, driver needs more sleep
4/20/2023 7:13:09 PM	Yawning detected, driver needs more sleep
4/20/2023 7:13:08 PM	Yawning detected, driver needs more sleep
4/20/2023 7:13:08 PM	Yawning detected, driver needs more sleep
4/20/2023 7:12:59 PM	Driver looking away from road for more than two seconds
4/20/2023 7:12:58 PM	Driver looking away from road for more than two seconds
4/20/2023 7:12:55 PM	Driver looking away from road for more than two seconds
4/20/2023 7:12:42 PM	Yawning detected, driver needs more sleep

Fig. 12. Logs page

VII. Test, Verification and Validation

Table I. Overall Test Results

Requirement	Metric	Results
Driver shouldn't take eyes off of the road for >2 seconds	Eyes looking away for >2 seconds using frontal view	95% in ideal conditions 80% in non-ideal conditions
Driver shouldn't fall asleep at the wheel	Changes in yawning and eyes closed	~100% in ideal conditions 95% in non-ideal conditions
Device accuracy in ideal and non-ideal conditions	Aiming for 90% in ideal conditions and 85% in non-ideal conditions	94% in ideal conditions 86% in non-ideal conditions
Driver is classified and feedback is given in <1 second	Feedback is given in <1 second so user can react in <2 seconds	~0 second latency

A. *Driver Shouldn't Take Their Eyes Off of the Road for More Than Two Seconds*

In order to meet the safety guidelines, we require that the driver cannot take their eyes off of the road for greater than two seconds at a time. The metric we used to check for this was timing how long the user would look away from a front view and checking that auditory feedback was provided if that length of time was greater than 2 seconds. We did several trials of having the user look away twenty times and recorded the number of correct classifications.

Our results showed that our device has a 95% accuracy in ideal conditions (adequate lighting and no obstructions to the face) and an 80% accuracy in non-ideal conditions.

B. *Driver Shouldn't Fall Asleep at the Wheel*

Another one of our requirements was that the driver cannot fall asleep at the wheel. The metric we used to check for this was seeing if a user closed their eyes for more than two seconds or kept their mouth open for greater than

5 seconds, to indicate a yawn. Again, we ran several trials of the user closing their eyes and simulating a yawn in a random order twenty times and recorded the number of correct classifications.

Our results showed that the device has around a 100% accuracy in ideal conditions and a 95% accuracy in non-ideal conditions. We recognize that the number of tests we conducted is not enough to declare a 100% accuracy for our device on this metric, but all tests that we conducted for detecting drowsiness in ideal conditions did pass.

C. Device Accuracy in Ideal and Non-Ideal Conditions

We didn't want the facial detection algorithm to be fazed by natural head movement that would be considered safe when driving, so we had a use case requirement of an accuracy of at least 90% in ideal conditions and at least 85% in non-ideal conditions. We tested for this by having a user simulate a series of all the gestures as if they were driving and recorded the number of correct classifications.

Our results showed that the device has around a 94% accuracy in ideal conditions and around an 86% accuracy in non-ideal conditions.

D. Driver is Classified and Feedback is Given in Under One Second

Our last requirement was that the driver would be classified and feedback would be given in under one second. We want the audio feedback to happen within one second so that the total time taken for computation, classification, and feedback is under two seconds which allows the driver to have time to correct their behavior. We tested this by timing how long it took between a classification and the end of the feedback being given. This process was repeated five times.

The results showed that there was virtually no latency between these two steps.

E. Jetson Subsystem Test

We wanted to perform tests on the Jetson itself before we tested the entire system as a whole. For these tests, we specifically wanted to see if the Jetson was able to be powered by the car, successfully connect to the user's personal hotspot and access the computer vision code, and if it met the product specifications we had. To test this, we plugged the Jetson into the car's USB-A port and checked if it would be powered and stayed on for an extended period of time, and we tested this several times at different times during the day. We also tried to connect to the personal hotspot and ssh into a laptop to access the code, and we also did this several times during the day. To see if the product met our specifications, we measured the size of it to make sure it wouldn't be a distraction to the user, which we determined would be a size around or less than that of a smartphone. All of these tests passed successfully.

F. Accelerometer Subsystem Test

To test the accelerometer, we wanted to see if it was able to detect changes in speed and provide feedback only when it needed to, and this meant that feedback would be turned off if the user is coming to a stop or reversing since these actions warrant a driver to look away from the road. We tested this with a rolling cart in the classroom, but we still have yet to test it in a car. It mostly works, with the velocity calculated from the acceleration working roughly half the time since the sensor we have isn't the best quality.

G. Web Application Subsystem Test

To test the web application, we wanted to see if the data for different users would be displayed appropriately. We tested this by making different user accounts and testing our device with each user to see if their respective data would be sent to their accounts and displayed on the metrics and logs page. This test was passed successfully.

VIII. Project Management

A. Schedule

The updated Gantt chart of the schedule is located in the appendix. It contains each part of the project and how the tasks are broken down. We divided it by each member of the team. The red boxes represent all of us, the purple is Elinora, the blue is Sirisha, and the green is Yasser.

B. Team Member Responsibilities

In terms of division of labor, we aimed to give each person two areas to focus on, one being a primary and the other being a secondary. This division was based on each person's strengths, so there will also be a good amount of overlap. The plan was for Yasser to focus on software in regards to machine learning and computer vision. Elinora was planning to focus on hardware and the user interface part of the software. Sirisha was planning to work on the user interface part of the software as well as the machine learning part of the software, and she was also going to provide assistance with hardware as needed. The actual breakdown of the work was that Yasser focused on the majority of the computer vision and some of the machine learning, Elinora focused on the hardware setup and user interface part of the software, and Sirisha focused on all three parts somewhat equally.

For the presentations and reports, we all tried to contribute as equally as possible. Yasser contributed marginally less but he spent that time working very diligently on the computer vision as that part was taking the longest to get working, so it was a decision we needed to make. We also all worked equally in terms of integrating the entire system.

C. *Bill of Materials and Budget*

TABLE II. BILL OF MATERIALS

Component	Cost	Status
NVIDIA Jetson Xavier AGX	$1,200-$2,000 x1	DID NOT USE
NVIDIA Jetson Nano	$100-$150 x1	Borrowed from department for free
K-Tech Mini Portable Speaker	$14.99 x1	Ordered from Amazon, has been delivered
ZENVAN USB-C to Headphone Jack	$7.98 x1	Ordered from Amazon, has been delivered
ELP 2 megapixel Hd Free Driver USB Camera	$40.99 x1	DID NOT USE
C922 Logitech Webcam	$66.28 x1	Ordered on Amazon
USB-A to USB-C Charging Cables	x1	Already had this part
HiLetGo MPU-6050 MPU6050 6-axis Accelerometer Gyroscope Sensor	$9.99 x1 (pack comes with 3)	Ordered on Amazon
ZTE MF833V 4G LTE USB Modem Dongle	$54.99 x1	DID NOT ORDER
EDiMAX EW-7611ULB USB WiFi Dongle	$12.84 x1	Order on Amazon
SanDisk 128GB microSD card	x1	Borrowed from friend
USB-C female to USB-A male adaptor	x1	Borrowed from friend
Total Budget	$112.08 used	$487.92 left over

D. *Risk Management*

Our first potential risk was having malfunctioning hardware components. We planned to mitigate this by testing our parts very early on so we could repurchase anything if necessary since after buying everything, we still had a very large part of our budget. Luckily, none of our parts malfunctioned during the process of working on this project, but we did need to buy parts throughout the semester, much later than we had planned. This was a result of needing to either change some hardware or realizing that we needed extra components, such as dongles, adaptors, and more storage.

Another risk we knew we could run into was that we might need to switch between the computer vision models since certain features work better with certain models, and we didn't know how the integration of two different models would work. To mitigate this, we spent a lot of time testing each individual feature before and after integration, but due to unexpected errors and the scope of our project changing, we did need to switch the models. Luckily, we had planned for this possibility early enough that the change didn't take too much time.

Our last major risk that we considered was being able to test safely and effectively in a car and make sure we had enough time. Ideally, we wanted to start testing in the car much earlier, but integration ended up taking much longer than we had anticipated. We started by testing in a classroom setting because we could make sure that the code would properly run on the Jetson and it was an easy way to see the outputs being displayed on the monitor. However, each individual step we needed to take for integration caused many issues, such as realizing the Jetson didn't have enough storage for all of the libraries we needed to download and we had to buy a microSD card, or dealing with just how sensitive the code is to lighting conditions and needing to spend much more time with tuning and calibration. Installing the needed computer vision libraries on the Jetson surprisingly took the longest – days more than we anticipated. We also had a few issues when it came to setting up the device to work in a car, such as getting the personal hotspot to work. This lengthy process also ate into some of the time we had allocated for tuning the accelerometer and working on our metrics page. Despite all of these unexpected issues that came up, we were still able to get our highest priority tasks working in time, and we were lucky that we were able to start integration as early as we did.

IX. Ethical Issues

The most relevant ethical issue with our product is that because it doesn't have a 100% accuracy, we had to figure out the best balance of false negative and false positives so we could minimize the risks to safety since our product deals with a car. Too many false negatives and a user won't get notified of a sign of inattention or drowsiness, and it may lead to an accident. Too many false positives, and a user may get distracted or startled by the feedback going off, and it could also lead to an accident. The best way to mitigate this issue at the moment would be to choose false positives over false negatives since it is better to be safer and as long as the audio feedback is presented in a way that wouldn't startle the driver, it shouldn't be as much of a distraction since most people are used to listening to sounds, such as music, other passengers, and talking on the phone, in their cars. We also made sure to use a neutral-sounding AI voice to generate the audio feedback samples and tested that the volume was not loud enough to be especially startling.

The fact that this device is powered by the car could also be considered an ethical issue because it may drain the car of its battery much faster than a user expects, leading to a car that can't start unexpectedly. Additionally, one of our peers brought up that, for some cars, the device may continue to drain the car's battery even after the engine is shut off. To mitigate these concerns, we can instruct the driver to ensure that the device isn't plugged in when the car is off, and since we are using the Jetson Nano over a more computational device, it shouldn't use much more power than, say, charging a device in the car.

Since our device aims at assisting a driver when they are exhibiting signs of inattention and drowsiness, the driver may become too reliant on the system and fail to self-monitor their driving habits. We hope to mitigate this through the web application so the user can track their progress and try to make changes to their habits.

Our device uses an auditory feedback system, so it relies on the fact that the user must be able to hear relatively well. This means that it doesn't cater to the deaf or hard of hearing community which is an issue since those people may also want to benefit from the use of a device that can assist them with driving. This could be mitigated by changing the method of feedback to something that is visual, like flashing lights, or something that produces a haptic feedback that they can feel when driving, similar to how some cars have a feature in their steering wheel that makes it vibrate when the car is veering too much in either direction.

X. Related Work

There have been a few products created on the market similar to what we are proposing to implement. Companies like Cipia, Smart Eye, Eyeware Tech, and Seeing Machines have created quite comprehensive products that track many aspects of a driver's behaviors, and some even go as far as to track how other people in the car are behaving. From research on these companies and their products, they all seem more focused on commercial applications rather than the average daily driver, which is what DriveWise aims to do.

We have also come across research in this field that has aimed to do similar things as us, particularly in the area of tracking a driver's eye in ideal and nonideal conditions.

We have some ideas for future work on DriveWise if time permits or if we have the chance to work on it even past this course. These ideas include adding detection of hands on a steering wheel and some way to remind driver's to check their blind spots when changing lanes.

XI. Summary

Our system was able to meet mostly all of our design specifications and requirements. The biggest issue that we have yet to tackle is with the accuracy of the accelerometer because there hasn't been much we were able to do from our end since the quality of the sensor itself needs to be better, and spending more of our time on that feature was not a priority as we cared more about getting the computer vision code to work at a high enough accuracy and properly on the Jetson. Aside from that, everything else works as we had intended. We were able to achieve around a 94% accuracy overall in ideal conditions and around an 86% accuracy overall in non-ideal conditions. If we had more time, we would love to get a better accelerometer and fully incorporate that feature as we had planned. We would also like to work on tuning the computer vision code and maybe even get a better quality webcam to hopefully increase the accuracies. We also did not get to add all of the features to the web application that we had hoped, and with more time we would add a metrics page with a time-series plot showing the percentage of driving time that a driver spends distracted to motivate future improvement of driving practices.

For future semesters, we want to share some of the lessons we have learned. Primarily, computer vision is challenging. It takes much longer than one might think due to how finicky it can be in different lighting conditions. It takes a lot of trial and error to find the best thresholds for certain conditions, and even then, they will always be changing simply because of all other variables that are out of control. We also learned how long integration actually takes. It may seem like it wouldn't require that much time, but issues crop up at every step of the way, and they are usually issues that no one expects. Nonetheless, we hope to encourage work in this field and with safe driving practices through the use of our system.

Glossary of Acronyms

CV	Computer Vision
NHTSA	National Highway Traffic Safety Administration
DNN	Deep Neural Networks
CNN	Convolutional Neural Networks
I2C	Inter-Integrated Circuit
SVM	Support Vector Machine

References

[1] Vardan Agarwal, *Face Detection Models: Which to Use and Why?*, July 2, 2020, [Online]. Available: https://towardsdatascience.com/face-detection-models-which-to-use-and-why-d263e82c302c

[2] Yin Guobing, *cnn-facial-landmark*, April 22, 2019. Available: https://github.com/yinguobing/cnn-facial-landmark

[3] Adrian Rosebrock, *Deep Learning with OpenCV*, August 21, 2017. Available: https://pyimagesearch.com/2017/08/21/deep-learning-with-opencv/

[4] Eric A. Taub, *2-Second Rule for Distracted Driving Can Mean Life or Death*, September 27, 2018. Available: https://www.nytimes.com/2018/09/27/business/distracted-driving-auto-industry.html#:~:text=The%20National%20Highway%20Traffic%20and,a%20time%2C%20the%20agency%20says.

[5] Rateb Jabbar, Mohammed Shinoy, Mohamed Kharbeche, Khalifa Al-Khalifa, Moez Krichen, et al.. "Driver Drowsiness Detection Model Using Convolutional Neural Networks Techniques for Android Application." ICIoT 2020, 2020, Doha, Qatar. pp.237-242, ff10.1109/ICIoT48696.2020.9089484ff. ffhal02479367f

[6] Adrian Rosebrock, *Facial landmarks with dlib, OpenCV, and Python,* April 3, 2017, Available: https://pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/

APPENDIX

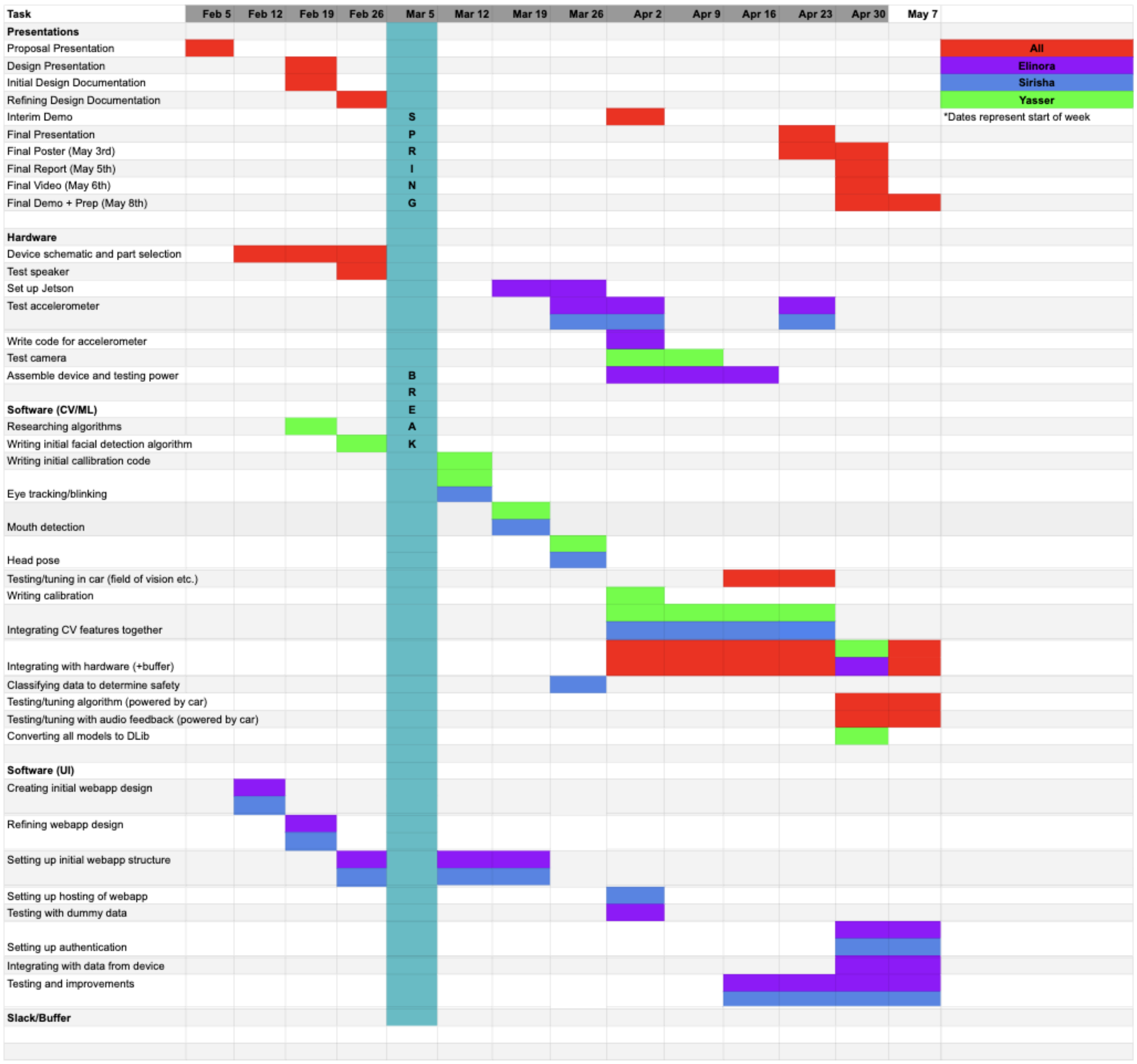


Fig. 1. Gantt chart with team member responsibilities