Drowsiness Detection for Safe Driving

Madhumita Ghosal
mghosal1@binghamton.edu
Department of Computer Science
Binghamton University

Atsuko Shimizu
ashimiz1@binghamton.edu
Department of Computer Science
Binghamton University

Gregory Wint
gwint1@binghamton.edu
Department of Computer Science
Binghamton University

ABSTRACT

Driving while sleep-deprived is dangerous. Drowsy drivers have significantly lower reaction times and worsened abilities to sustain attention, which increases the chance of getting into vehicle accidents. One of the challenges of preventing drowsy driving is that the drivers themselves may not be aware of the fatigue because the signs of fatigue can be difficult to identify. To solve this problem, we developed an Android application that detects whether a driver is drowsy. Our application uses facial recognition methods to analyze video of the driver at the wheel, and can alert the driver if drowsiness is detected. To encourage habitual safe driving, we incorporated a *streak* feature, which, over frequent uses, keeps track of how long the driver has driven without detected drowsiness. The goal of our project is to remove the burden on the driver of identifying their fatigue, allowing safer driving.

KEYWORDS

facial recognition, driving, drowsiness detection

1 INTRODUCTION

According to the National Security Council, there are approximately 100,000-300,000 accidents per year due to drowsy driving [14]. Fatigued drivers have significantly slower reactions times and are three times more likely to get into a vehicle accident [14]. Not only does this place the driver in a dangerous situation, but drowsy driving can also increase the risk of accidents of other drivers and passengers. Preventing drowsy driving is difficult because the drivers themselves may not know that they are fatigued. Drivers may also be unaware of experiencing *micro-sleep*, which are brief, involuntary periods of inattention. Only 4 to 5 seconds of micro-sleep at highway speeds can lead to detrimental vehicle accidents [14].

To solve this problem, we developed an Android application that can detect whether the driver is drowsy at the wheel. The app continuously captures video of the driver and analyzes the video to detect whether the driver is drowsy. Having an app to detect drowsiness removes the burden on the driver to identify their fatigue themselves, which can be difficult to do while the driver operates a vehicle. In addition to detecting drowsiness, our app also encourages habitual safe driving by keeping track of the driver's *streak* of driving without drowsiness. Our application is a starting point to provide a simple, easy to use, and accessible tool for safer driving.

This project report is organized as follows. In Section 2, we discuss the overall design of our Android application and the details of our implementation. In Section 3, we evaluate the accuracy and performance of our drowsiness detector. Then finally in Section 4, we conclude.

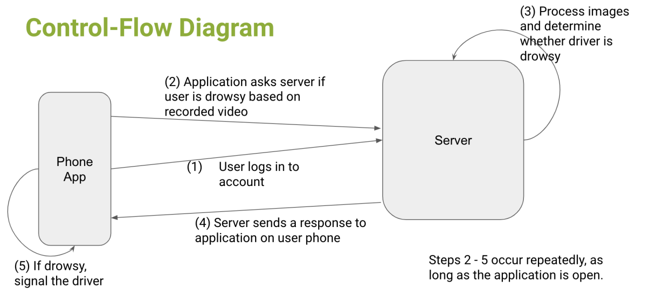

Control-Flow Diagram
(3) Process images and determine whether driver is drowsy
(2) Application asks server if user is drowsy based on recorded video
Phone App
Server
(1) User logs in to account
(4) Server sends a response to application on user phone
(5) If drowsy, signal the driver
Steps 2 - 5 occur repeatedly, as long as the application is open.

Figure 1: Control-flow diagram of our drowsiness detector

2 DESIGN AND IMPLEMENTATION

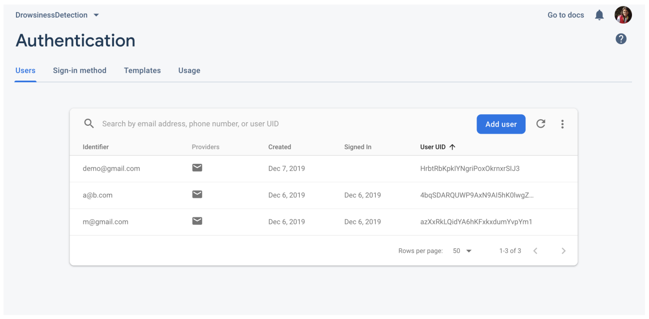
In this section, we first describe the overall design of our drowsiness detection app. Then we describe the tools that we used and the details of our implementation. Our GitHub repository of this project can be found here: https://git.io/JeMKb.

2.1 Control-Flow Diagram

Figure 1 shows the control-flow diagram of our drowsiness detector. First, the user open the drowsiness detection app and log into their account (1). To implement the app, we used Android [10]; we discuss the implementation details of using Android in Section 2.3. To perform account registration and login, we used Firebase [11], which we describe in Section 2.2. The user then starts the camera (2), which starts a background task in the Android app. This background task periodically sends the recorded video to the Amazon Web Services (AWS) [1] server (2). Recorded video is continuously sent as long as the user keeps the camera screen in the Android application open on their phone. AWS runs the detection algorithm (3) on the video that the server receives, and returns the result of the video (whether the driver is drowsy or not) back to the Android app (4). We describe the detection algorithm in Section 2.4. If the driver is drowsy, the app notifies the user (5). The streak value of the user is reset if drowsiness is detected. Else, if the driver drove safely in that session, the streak value is incremented. The streak value is updated when the user logs out and logs back into the application.

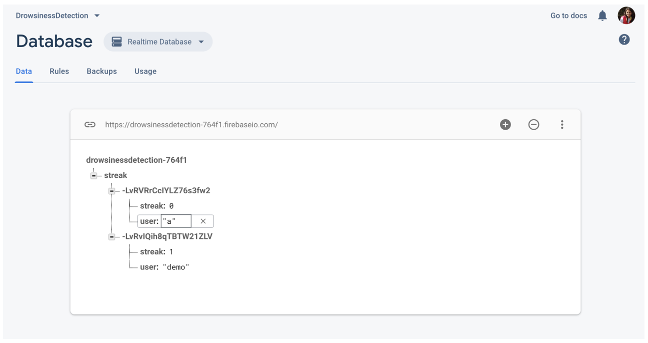
2.2 Firebase Database

To store user credentials and user data, we used Firebase database [11]. The Firebase database is a cloud-hosted NoSQL database that



Identifier	Providers	Created	Signed In	User UID
demo@gmail.com	✉	Dec 7, 2019		HrbtRbKpkIYNgriPoxOkrnxrSIJ3
a@b.com	✉	Dec 6, 2019	Dec 6, 2019	4bqSDARQUWP9AxN9AI5hK0lwgZ...
m@gmail.com	✉	Dec 6, 2019	Dec 6, 2019	azXxRkLQidYA6hKFxkxdumYvpYm1

Figure 2: Authentication tab in Firebase



```
https://drowsinessdetection-764f1.firebaseio.com/

drowsinessdetection-764f1
  streak
    -LvRVRrCclYLZ76s3fw2
      streak: 0
      user: "a"
    -LvRvIQih8qTBTW21ZLV
      streak: 1
      user: "demo"
```

Figure 3: Database tab in Firebase

stores and syncs data between users in real time. The Android application has a login and registration page where any new user would have to register first before using the application by providing a username and password. Once the user clicks the REGISTER button, a record is created in the Firebase database. After the registration, the user would have to login by providing the username and password that was registered.

Another feature that was added to the application was the streak functionality. For each time the driver drove non-drowsy, the streak counter is incremented by 1. If at any point of time the driver drove drowsy, the streak is reset to 0.

2.2.1 Implementation. Once the user provides their email ID and password and clicks on register, we call the `createUserWithEmailAndPassword` method, which creates an entry in the Firebase database under the tab `Authentication`, as shown in Figure 2. When we would have to login, the email id and password gets authenticated in the Firebase database using the `signInWithEmailAndPassword` method, which is defined under the Firebase library. We integrate the Firebase database account to our code using the generated `google-services.json` file that was created while creating the database instance in the Firebase console. We needed to copy the `google-services.json` file in our project under the `app/` directory in our Android code.

In the Firebase database, we created a `Realtime Database` instance called the `drowsinessdetection-764f1` with a document called the `streak`, shown in Figure 3. Each entry created in the document corresponds to each user; username and streak value is stored under every entry. If the driver drives drowsy, the streak is updated in the database to 0. Else, if the driver drives non-drowsy, the streak value is incremented by 1 and updated in the database.

2.3 Android Application

To prototype our application, we used a Motorola moto G6 phone [13] running Android version 8.0 [10]. The moto G6 phone has a 2160x1080 display, with an 8MP front-facing camera. To develop the Android application, we used Android Studio [8].

2.3.1 Activities. The Android app is organized into three Activity classes [2], which are focused tasks that the user can perform on the app. Each Activity typically has its own screen. The `MainActivity` class is the screen that the user sees when the app is first open. `MainActivity` is responsible for interacting with Firebase [11] for user registration and login, which we describe further in Section 2.2. Once the user logs in, `MainActivity` opens `HomeActivity`, which displays the current streak of the user and a button to start the drowsiness detection camera. The login and home screen is shown in Figure 4. To incorporate the camera functionality, we followed the Android Camera API [3] into the Activities. `HomeActivity` is responsible for obtaining the camera permissions from the user before preceding to the detector camera. The `CameraActivity` obtains a `Camera` object [4] and displays the camera preview to the user. When the user taps RECORD, `CameraActivity` creates a background task that records video and periodically sends the video file to the AWS server.

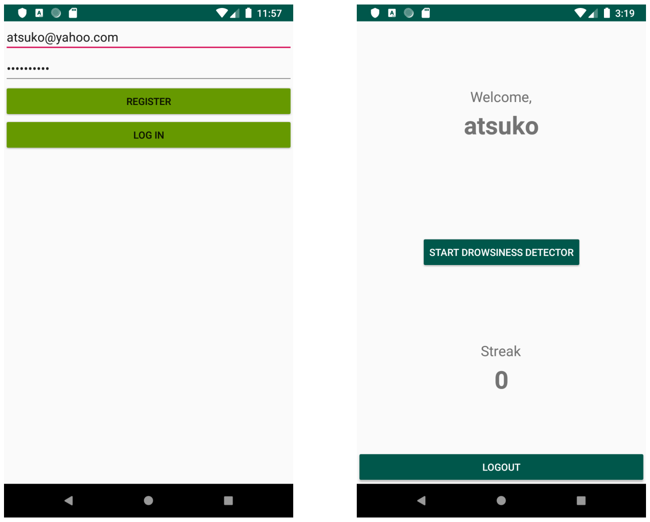


Figure 4: The login and home screen of the detection app

2.3.2 Classes. There are two main Java classes: `CameraPreview` and `VideoSender`. `CameraPreview` allows the user to see what the video is currently recording. There is a `CameraPreview` frame on the `CameraActivity` screen. When the user starts recording, the

`CameraActivity` class creates a `VideoSender`, which is a background task that inherits from `AsyncTask` [5]. `VideoSender` prepares a `MediaRecorder` [6] and an `HttpURLConnection` [7] for every video that it sends. The `MediaRecorder` configures the camera, video format, and output file path (the app uses temporary cache storage, as opposed to permanent storage). The `HttpURLConnection` is used to send the video to the AWS server and retreive the detection algorithm output (drowsy or not drowsy). When drowsiness is detected, the user interface (UI) must be manipulated to alert the user. Since the background task cannot manipulate the UI, the background task must contact the UI thread to create a Toast message and play an alert sound when drowsiness is detected.

2.4 Drowsiness Detection Algorithm

The Android app continuously sends segments of recorded video to the AWS server [1], which performs the following drowsiness detection algorithm. Given a video, a determination is made as to the drowsiness of the driver by first breaking the video into a series of image frames. Since consecutive frames are unlikely to capture any differences in driver position, only every fifth frame is analyzed. The openCV library [9] was used to handle extracting frames from the video provided to the backend. A pre-trained facial feature detector provided by the dlib image processing library [12] was used to identify the eye regions of the face contained within each frame. The *eye aspect ratio*, which measures the ratio of eye height to eye width, was used to determine if the person's eyes are open or closed. Any value below a specified threshold signalled that the person's eyes were closed, while any value above that mark signalled that the person's eyes were open. The number of consecutive frames in which a person's eyes are closed is tracked and used to determine whether or not a person is drowsy. After a specified number of consecutive frames is reached, it is determined that the person depicted in the frames is drowsy. When the person depicted has their eyes open, the counter tracking the number of consecutive frames is reset. The results of the drowsiness detection is sent back to the Android application to notify the user.

3 EVALUATION

A walk through of our app can be seen in our demo video: https://youtu.be/OUERfBvFglA. The detection algorithm determines drowsiness when consecutive frames in the video contain a face with their eyes closed. In our demo video, our application is able to do this, but with a considerable delay. This is because the drowsiness detector needs to analyze the video frames sequentially before processing the next video. We reduce this delay by skipping frames, as described in Section 2.4, however, this is still too slow for instant drowsiness detection. As a future improvement, we can reduce the video processing time by merging frames (by taking the average of consecutive frames), or using a Graphics Processing Unit to process each video frame in parallel.

4 CONCLUSION

In this project, we developed an Android application that detects the drowsiness of a driver while they operate a vehicle. The app continuously records video of the driver and sends the video segments to an AWS server, which performs facial recognition techniques to detect whether the driver is drowsy. When drowsiness is detected, the app alerts the user with a notification message and sound, then suggests the driver to pull over for their safety.

Operating a vehicle while fatigued is dangerous, however signs of sleepiness can be difficult to identify. Our application approaches this challenge by providing a non-intrusive way to detect drowsiness. Although the speed of the video processing is too slow to be used in real-time, our app is a starting point to providing a simple and accessible tool to assist in safer driving.

REFERENCES

[1] Amazon. 2006. Amazon Web Services (AWS) - Cloud Computing Services. https://aws.amazon.com/.

[2] Android. 2007. Activity. https://developer.android.com/reference/android/app/Activity.

[3] Android. 2007. Activity. https://developer.android.com/guide/topics/media/camera.

[4] Android. 2007. Activity. https://developer.android.com/reference/android/hardware/Camera.

[5] Android. 2007. Activity. https://developer.android.com/reference/android/os/AsyncTask.

[6] Android. 2007. Activity. https://developer.android.com/reference/android/media/MediaRecorder.

[7] Android. 2007. Activity. https://developer.android.com/reference/java/net/HttpURLConnection.

[8] Android. 2013. Download Android Studio. https://developer.android.com/studio.

[9] G. Bradski. 2000. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).

[10] Google. 2007. Build Anything with Android. https://developer.android.com/.

[11] Google. 2011. Firebase. https://firebase.google.com/.

[12] Davis E. King. 2009. Dlib-ml: A Machine Learning Toolkit. *Journal of Machine Learning Research* 10 (2009), 1755–1758.

[13] Motorola. 2018. moto g6. https://www.motorola.com/us/products/moto-g-gen-6?ds_rl=1242196&ds_rl=1242193&ds_rl=1260444&ds_rl=1260444&gclid=EAIaIQobChMIydvTrO-P5gIVTV8NCh3JugTiEAAYASAAEgI7dvD_BwE&gclsrc=aw.ds.

[14] National Safety Council. 2019. Drivers are Falling Asleep Behind the Wheel. https://www.nsc.org/road-safety/safety-topics/fatigued-driving.