**Senior Design Final Report**

Smart Eye Drops

Version 1.0 – 7 April 2020

Team Members:

Marco Aguilar

Aaron Alvarez

Ana Cortes

Jonathan Nunez

Feng You

Ivan Yu

Faculty Advisor:

Dr. Navid Amini

Kevin Delao

Liaisons:

Haley Kirk

Table of Contents

[1. Introduction 3](#_Toc39847672)

[1.1. Background 3](#_Toc39847673)

[1.2. Importance 3](#_Toc39847674)

[1.3. Design Principles 4](#_Toc39847675)

[1.4. Design Benefits 5](#_Toc39847676)

[1.5. Achievements 5](#_Toc39847677)

[2. Hardware and Technologies Used 6](#_Toc39847678)

[2.1. MBientLab Sensors 6](#_Toc39847679)

[2.2. Force Sensitive Resistor 6](#_Toc39847680)

[2.3. Amazon Web Services 6](#_Toc39847681)

[3. System architecture 7](#_Toc39847682)

[3.1. Overview 7](#_Toc39847683)

[3.2. Data Flow 8](#_Toc39847684)

[3.3. Implementation 9](#_Toc39847685)

[3.3.1. Data Transmission 9](#_Toc39847686)

[3.3.2. Mobile Application 9](#_Toc39847687)

[3.3.3. Server 9](#_Toc39847688)

[3.3.4. Machine Learning 9](#_Toc39847689)

[3.3.5. Database 9](#_Toc39847690)

[3.3.5. Visualization 10](#_Toc39847691)

[3.3.6. Adherence Graph 10](#_Toc39847692)

[4. Algorithms 11](#_Toc39847693)

[4.1. Thresholding 11](#_Toc39847694)

[4.2. Long Short-Term Memory 12](#_Toc39847695)

[5. Conclusions 13](#_Toc39847696)

[5.1. Results 13](#_Toc39847697)

[5.2. Future 14](#_Toc39847698)

[6. References 16](#_Toc39847699)

# 1. Introduction

## 1.1. Background

Adherence to eye drop medication has been a significantly challenging factor in treating eye diseases such as glaucoma. Despite there being numerous methods of identifying nonadherence (i.e. self-reporting, physician report), there is still no quantitative model for measuring adherence to eye drop medication. In addition, studies have shown that physicians have difficulty spotting patients who do not adhere to their prescribed medication regimens and often inflate their reported adherence rates [1].

Because glaucoma is the second leading cause of irreversible blindness worldwide and is one of the prime examples of nonadherence in eye medication, we built our project, Smart Eye Drops, with glaucoma treatment at the core of what we set out to do. Hence, the requirements we identified, and our design inspiration came largely in part with increasing glaucoma eye drop medication in mind. As there are other diseases that can be controlled or treated using eye drops, such as corneal disease and conjunctivitis (pink eye), we believe Smart Eye Drops can also be utilized to increase medication adherence for such diseases as well.

Previous studies on glaucoma medication adherence have shown us issues we, as software engineers, have the ability to address. From the identified barriers to adherence, 24 – 47% of patients had difficulties with their medication schedule, 23 -59% of patients stated that they had poor self-efficacy, and 23 – 62% of patients stated that they were forgetful [2]. A possible solution that we have seen other teams try to implement involved a sensor unit attached to an eye drop bottle that only recorded when, and how long the bottle was tipped upside down by using a simple thresholding algorithm on data provided by their accelerometer. Our solution to these concerns was more accurate by also taking account the squeezing of the bottle, when and how long the bottle was tipped upside down, and incorporated machine learning to classify when an eye drop was administered. We built a sensor system that could be fitted to any cylindrical shaped eye drop bottle and a supporting Android application to facilitate data transmission from our sensors to our machine learning algorithms on Amazon Web Services.

The objective of this project was to develop a completely new eye drop adherence monitoring system and that goal was realized under the guidance of Dr Navid Amini, and sponsorship of Vodafone Group Plc.

## 1.2. Importance

Since we designed Smart Eye Drops with increasing patient adherence to their eye drop medication, an obvious benefit would be the improvement of their ocular health. Preventing the progression of vision loss, disability and blindness is a global burden and disproportionally affects the world. Currently in the US, there are 3 million Americans living with glaucoma and that number is projected to reach more than 7 million people in 2050, growing at a rate of 28% per decade [3]. Glaucoma costs the US health care system an estimated $2.5 billion annually: $1.9 billion in direct costs and $0.6 billion in indirect costs [4]. Considering that demographic, it is imperative that effective treatment is provided to patients and they adhere to their prescription regiments. Smart Eye Drops is designed to collect eye drop usage data directly from patients whenever they use their medication, process that information on our cloud, and provide physicians with meaningful information to help then make better decisions on treating their patients. By increasing adherence, we aim to reduce the direct and indirect health care costs in the US.

## 1.3. Design Principles

On the hardware side, the Smart Eye Drops sensor system consists of a force sensitive resistor attached to a main sensor module with Bluetooth capabilities. This allows us to collect and transmit data to our Android application in real time. The sensor system was also required to fit most cylindrical shaped eye drop bottles, small enough to carry around, and resistant to accidental drops.

On the software side, our Android application facilitates the transmission of data from our sensor system, to our cloud, then retrieves processed data from our machine learning module to display for both physician and user to view. The data is visualized using a scatter chart that displays eye drop appliances for a month of a year. This gives the user a wider view of how often their patient is applying their eye drops and if they are applying them at the correct time.

Data collection is collected using various methods to create a single stream of packaged data that allows the cloud to process raw data given from the sensors. As the application is opened, it creates a secure connection between the sensors and the mobile application. A timer is then placed to allow the mobile application to continuously connect until its allowed to collect data from the sensors. Once the application has connected with all the needed sensors it then proceeds the following steps to successfully package raw data to send to the cloud.

Firstly, the application creates different threads to continuously read data from all required sensors such as the accelerometer sensor and force sensor. Threads allows the application to continuously read raw data from all sensors at the same time. This allows us to gather data from both the accelerometer and force sensor which are needed to make predictions in the cloud.

Secondly, a timer is trigged which allows the application to only gather raw data from the sensors. A timer allows the sensors to only send data to application when they are triggered. All sensors stop sending data to the application and collection of raw data is halted as the timer expires. A timer allows all sensors to only send raw data to the application when is needed. This allows the sensors to only work when the sensors are triggered which allows the smart eye drop bottle to extend its battery life. Gather data only when is in use.

Thirdly, all raw data is then packaged and sent to the cloud for future processing. Packaged data is processed by generating a format streamed of data containing the time, accelerometer and force sensor data which is later sent to the cloud for classification using our machine learning algorithms.

All steps are then repeated as the consumer applies the next eye drop.

## 1.4. Design Benefits

The benefits of creating a sensor system that has an external force sensor is that data collection only occurs when force and accelerometer readings exceed a certain threshold simultaneously. This increases the accuracy of our data and allows us to extend the battery life of our hardware and makes it easier to use for patients. Smart Eye Drops’ small form factor makes it easy to carry around, further increasing its ease of use for patients who may be required to use their medication during the day and increase self-efficacy.

The decision to display the data on a scatter chart in our Android application is beneficial because it gives an overall look on how the patient is adhering to the eye drop regimen in relation to the previous days of the month. If a certain treatment plan is unsuitable for a patient, the physician can make personalized schedules for each patient based on the data we collected and reduce the barriers patience face regarding their medication schedule. Our team’s goal was to design a friendly user interface that would allow both physicians and patients to easily navigate through the app to view the adherence analytics.

Another design benefit is the utilization of reusable services provided by AWS. These services are decoupled and can be improved or replaced without interfering with other modules of the Smart Eye Drops system. By having the heavy computational tasks handled on the cloud, we reduce the processing load on the users’ smartphones.

## 1.5. Achievements

Our team built an end-to-end system that allows patients to apply an eye drop and have it displayed in the Smart Eye Drops mobile application in 5 seconds. We were able to build a reusable and cohesive backend using the AWS cloud services. The cloud services used are modularized so that they can be used or replaced with other components. The Smart Eye Drops backend is loosely coupled with the user interface, which allows future developers to update or completely change the look and feel of the application.

Our team was able to build an algorithm that uses thresholding and machine learning to predict when an eye drop was being applied. The accuracy of our classification model is at 96.97%

# 2. Hardware and Technologies Used

## 2.1. MBientLab Sensors

There were two MBientLab Sensors used for this project, MetaMotionC (MMC) and MetaMotionR (MMR). Both sensors have many features, offering real-time information of motion and environmental sensor data. Our team focused on retrieving data from the accelerometer. Both sensors transfer data wirelessly via Bluetooth and weigh approximately 0.2oz. The MMR is rectangular while the MMC is circular. Both sensors came enclosed with a case. With their cases the MMC is 25mm x 4mm and the MMR is 27mm x 27mm x 4mm. The MMR has a 100mAH micro-USB rechargeable battery. The MMC has a 200mAH coin-cell battery. Both sensors have GPIO pins allowing the connection of external sensors. We were able to connect an external Force Sensitive Resistor to these sensors through the GPIO pins. Our prototype right now has the MMR because of the rechargeable battery. Realistically the MMC is a better fit because of its circular design, but since a patient would have to continuously change the battery rather than just recharge it, we used the MMR instead [5].

## 2.2. Force Sensitive Resistor

Our team used Interlink Electronics FSR 406. This Force Sensitive Resistor (FSR) was used to calculate the squeezing motion of the eye bottle. This FSR was chosen because it fit the best with the MBientLab Sensors. This FSR has a 39.6mm x 39.6mm active area. It has solder tabs in case we want to solder it onto the MBientLab Sensors, for now the FSR is glued on. Its force accuracy ranges from 5% to 24% and has a sensing range from 0.2N to 20N. Considering only small amounts of force is required when squeezing an eye drop bottle during its use, this force sensor was the best choice. The FSR wraps around the eye drop bottle [6].

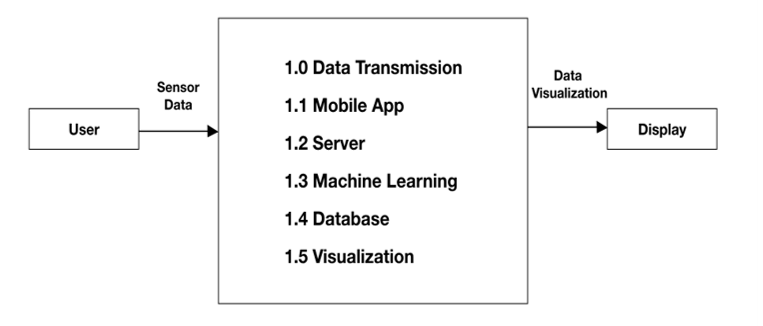
## 2.3. Amazon Web Services

Amazon Web Services (AWS) was our cloud platform of choice. AWS’ suite of cloud services allowed for endless potential in terms of development and scaling. In total, our project used 5 AWS services. We started off with API Gateway, which allowed us to create and manage our own REST API. Our REST API would allow us to make GET, POST, PUT, and DELETE requests to the rest of our application. Next, we used Lambda to make serverless applications to write, read and delete from our database. Additionally, we used AWS EC2, a web service that provides secure, and resizable compute capacity to process all our raw data and invoking our machine learning endpoints. Next, we used DynamoDB. DynamoDB is a popular noSQL database service. We are using it because it supports key-value data structures. We are using it to host the physician and patient tables. Finally, we used SageMaker to host all our machine learning instances, models, and endpoints.

# 3. System architecture

## 3.1. Overview

The system architecture for Smart Eye Drops is shown in Figure 1 below.

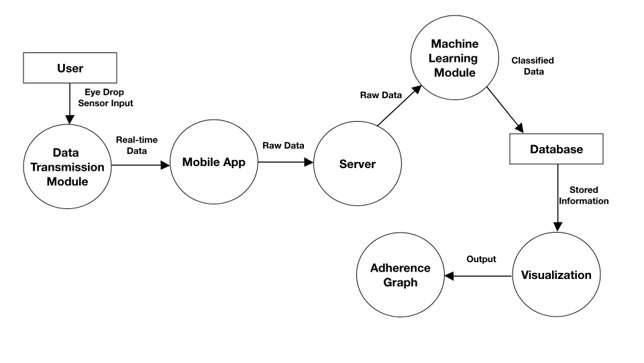


*Figure 1: Smart Eye Drops DFD-0*

To give a brief overview of how Smart Eye Drops works; from left going right, the user is the beginning of our data flow. When the user applies their medication, raw data is collected and passed our main processing application, comprised of six smaller modules, and stored in the database module until the processed information is retrieved to be displayed for the patient or the physician to view.

## 3.2. Data Flow

Figure 2 shows a more detailed outline of Smart Eye Drops and its various components.



*Figure 2: Smart Eye Drops DFD-1*

Description of each component:

* User: represents the patient using Smart Eye Drops. Raw data produced by the motion and pressure they apply to the sensor system will be tracked, recorded, and transmitted through the Data Transmission Module to the Mobile App (methods explained in more detail in section 4.1.).
* Data Transmission Module: used to send raw data collected by the sensor module to the Mobile App in real time.
* Mobile App: collects real time data from the Data Transmission Module and forwards that data to our Server. Is also used to view a patient’s adherence analytics.
* Server: relays Raw Data from Mobile App to our Machine Learning Module.
* Machine Learning Module: processes Raw Data using an artificial recurrent neural network architecture to classify the data into different labels: e.g. “No Motion”, “One Drop”, and “Walking” (methods explained in more detail in section 4.2.). That information is then stored in our database.
* Database: stores labeled data processed by our Machine Learning Module and sends that information to our Visualization module when requested.
* Visualization: takes information from our Database and forwards the information to be displayed on a patient’s Adherence Graph.
* Adherence Graph: takes information from the Visualization Module and visualizes that information on a scatter chart to both patient and physician to view.

## 3.3. Implementation

Smart Eye Drops was split into the following six sections to allow for more efficient and modularized development: Data Transmission, Mobile Application, Server, Machine Learning, Database, Visualization, and Adherence Graph. Each component plays a key role in presenting the progression of the project and can be altered and swapped out in future development without affecting other existing components.

### 3.3.1. Data Transmission

Raw data is collected through the sensors and packaged within the mobile application. External sensors are plugged in onto the a MetaWear Board, MMC/MMR. The application gathers multiple raw data points from the Smart Eye Drops bottle allowing cloud processing to make accurate predictions. Data is collected from the sensors through the MetaWear application programming interface (API). This allows us to connect the sensors and the application safely, allowing continuous data collection (live data). All raw data is then processed, packaged, and sent to the cloud for additional processing.

### 3.3.2. Mobile Application

All raw data is stored onto a single stream data and sent to the cloud. Once the cloud makes its predictions with the raw data, we use those results and display it using a graphical user interface on the mobile application. Within the user interface, the user sees a list of patients in which the he/she can then select from. When a user selects a patient, the app presented a history of that patient and their eye drop adherence index. All results are presented in a friendly graphical user interface through the mobile application.

### 3.3.3. Server

The server is created by running a Java application on EC2 where it will be waiting for a socket connection through the EC2’s IP. Once a connection has been established the Java application saves all the raw data that it receives into a csv file. Then it will run a python script that will read the csv file that was created and create all the features needed for our machine learning algorithm and save it into a new csv file. Then these features are sent over to our machine learning module for classification.

### 3.3.4. Machine Learning

The machine learning module can be created on either another an EC2 server or SageMaker. Our algorithm uses Python with the Keras and TensorFlow libraries. Once our module has been trained it will be waiting for data from the server. The module will then classify our data to see if an eye drop bottle was used or not. If it was not used then then the data will be discarded, else this module will send an update to the Database.

### 3.3.5. Database

The database is created using DynamoDB. It holds two tables, one for patients, and another for physicians. The physician table contains their ID, first name, last name, list of patients, and their image. The patients table contains their ID, first name, last name, email, their physician's ID, their image, and a list of records for all the dates and times they used the eye drop. Currently, only the patient’s table can be updated through the machine learning module. The update affects the patient’s records by adding the date and time of when the eye drop bottle was applied.

### 3.3.5. Visualization

The data inside the database can be visualized using the Visualization module. This module and provides a pleasant interface for the user so that he or she can view the adherence analytics. Visualization of the data is achieved via the Smart Eye Drops mobile application, which uses open source libraries to create the user-interface. In particular, the mobile application uses the MPAndroidChart to display the adherence data points on a scatter chart.

### 3.3.6. Adherence Graph

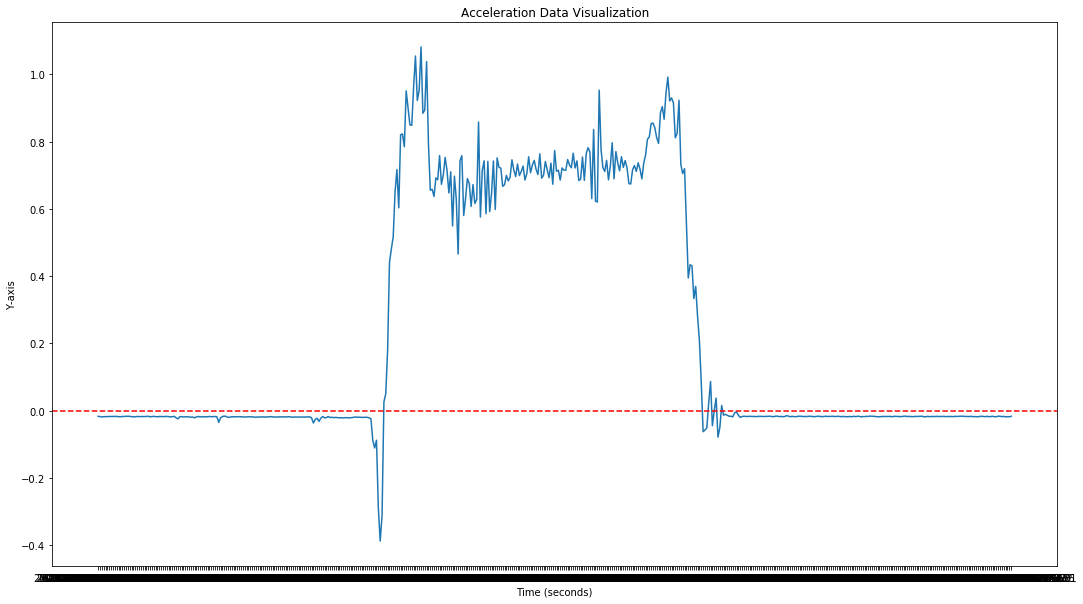
The adherence graph is created using the MPAndroidChart library to display the eye drop adherence data on a scatter chart. On the graph there are two types of data points. The first type of data point is the scheduled eye drop use, which is displayed using grey data points on the graph. The second type of data point is when the patient applied an eye drop, which is highlighted using a blue data point on the graph.

# 4. Algorithms

## 4.1. Thresholding

The MBientLab sensor module we used, has a built in XYZ 3-axis acceleration sensor (accelerometer) that detects gravitational acceleration values at 33 Hz. The force sensitive resistor we attached to the MBientLab sensor via its GPIO pins measures pressure applied to the bottle at the same frequency.

We defined the Y-axis of the accelerometer to be the vertical axis of the eye drop bottle, so that when the bottle is tilted, there would be a sharp change in values detected by the accelerometer on that axis (Figure 3). Our Smart Eye Drops sensor system constantly runs our thresholding algorithm to detect whether the bottle is lifted, tilted upside down, and is squeezed to apply an eye drop. To elaborate further, the sensor system is constantly running to see if the Y-axis values and the pressure values simultaneously exceed a set of predetermined values, indicated by the red horizonal dotted line in Figure 3.



*Figure 3: Acceleration values recorded by our accelerometer on the Y-axis*

This method, in theory, can be sufficient to a certain degree to label the set of motions detected by our sensor system but it is not accurate enough for our requirements. Thresholding simply does not have the capabilities to filter out false data: movements that are similar to the motions of applying an eye drop e.g. carrying the eye drops in a bag, accidental drops. However, we implemented this method to trigger a data logger to collect 20 seconds of acceleration and force data and send it to our machine learning algorithm for more accurate classifications.

## 4.2. Long Short-Term Memory

We used long sort-term memory (LSTM) as our machine learning algorithm. Because LSTM is an artificial recurrent neural network architecture [7], we were able to acheive a better accuracy when compared to some machine learning algorithms. With the 20 seconds, we got 491 data sets. Each set contained the time of application, acceleration on the X-axis, Y-axis, and Z-Axis, and force applied. Since the time of application is not important to determine if the eye drop bottle was used, it gets put on the side. This leaves us with four data points for each set. We used the data sets to create 36 features that would be useful in training our machine learning model. For each of the four data points, across the entire data set, we got the kurtosis, absolute kurtosis, min, absolute min, max, absolute max, mean, absolute mean, and median.

Once we obtained a set of features, the set was labeled according to the type of application. We had six categories; no movement, applied to one eye, applied to both eyes, lifting, dropping, and walking. For the base of our model, we drew inspiration from Dharmitha Ajerla’s model [8] to start up our model. It was a good starting point since the type of activity we both were trying to detect happened within a short period of time. The only type of activity we focused on was applying to one eye or both. This was useful in determining if the eye drop bottle was used or not. Since most of the time the eye drop bottle will not be moving. We tested the data we gathered, applying one or two drops, with the machine learning model. By using an average for the type of application we were able to determine how close the type of application resembled one of our six categories (Table 1).

*Table 1: Data comparison for one or two drops*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Category | No Motion | One drop | Two drops | No drops | Dropped | Walking |
| Two drops | 0.88% | 1.21% | 88.57% | 4.07% | 1.02% | 4.21% |
| One drop | 0.49% | 96.97% | 0.7% | 1.01% | 0.43% | 0.38% |

# 5. Conclusions

## 5.1. Results

Even though the dataset that we used was slightly artificial, we were still able to achieve an accuracy high enough to determine if an eye drop was applied or not. This was the most crucial data that we needed, because determining if the eye drop bottle was used or not would determine if DynamoDB would be updated. One of the possible errors that we found can occur when applying a single eye drop is that it may be categorized as two drops instead. We believe that this is due the way that the eye drop was applied, where the user will squeeze the eye drop once, but the eye drop does not fall, so they release their squeeze and try again.

When testing data such as no motion, and zero drops we get similar results (Table 2). Because the eye drop bottle will be in no motion for most of the time. The purpose of the no motion category was to clear out false signals where the eye drop bottle would be moved slightly and trigger sending data. The same can be said for zero drops where the motion of applying the eye drop is the same, but no force was used to squeeze the bottle.

*Table 2: Data comparison for no motion or zero drops*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Category | No Motion | One drop | Two drops | No drops | Dropped | Walking |
| No motion | 94.43% | 1.74% | 0.86% | 0.73% | 0.93% | 1.29% |
| Zero drops | 0.5% | 0.83% | 3.17% | 88.94% | 1.873% | 4.71% |

The categories dropped and walking are considered rare activates that happen while holding the eye drop bottle. They are less frequent than applying an eye drop, however if someone is moving while holding the eye drop bottle it can be assumed that they will still be moving for either a shot or long time. Therefore, the data gathered for walking was moving is a straight line or walking throughout a house. Due to the unique way these activities are stored it is easy to distinguish them from the other categories (Table 3).

*Table 3: Data comparison for dropping or walking*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Category | No Motion | One drop | Two drops | No drops | Dropped | Walking |
| dropping | 0.79% | 1.09% | 0.74% | 2.7% | 92.6% | 2.07% |
| walking | 0.24% | 0.97% | 0.7% | 2.42% | 0.35% | 95.32% |

When training and testing the data on itself the accuracy was 100%. Since most of the data were copies with slight variations of real data gathered this was not a surprise. The ratio of real data to copies was 1:80, but as we started to decrease the amount of copies, we started to get more realistic results (Table 4). However as less data is trained in LSTM it becomes less accurate.

*Table 4: ratio comparison*

|  |  |
| --- | --- |
| Ratio | Accuracy |
| 1:80 | 100% |
| 1:40 | 100% |
| 1:20 | 94.5% |
| 1:3 | 75.83% |

Our eye drop adherence monitor would help physicians learn about their patient’s compliance in using an eye drop bottle. Our application detects if a patient used the eye drop bottle in real-time and will update the patient’s record. For the data we would use, we decided to use the ratio of 1:20 for a more realistic approach to the data we gathered. Gathering data in a controlled environment was not as ideal as we though in this situation. Mainly, because there was a lot of rigid movement when applying the eye drop and it was not very natural, especially since all the test subjects have never used an eye drop bottle before. Our LSTM was able to detect the type of activity with a 97% accuracy, where a ~2% inaccuracy can be contributed from to the inability to distinguish a false two drop as a one drop as mentioned before. There are two ways we could deploy our LSTM model, either on SageMaker or another EC2 server. We were able to accurately gather the data from both the accelerometer and force sensor using our Java application on Android. This raw data is then pass over to our server on EC2 using a socket in Java where the data is then converted into features using Python. The features are then passed over to either another EC2 or SageMaker depending on how the LSTM model was implemented. After classification, DynamoDB will be updated, where our Android application will be able to gather all the data from DynamoDB using API gateway and create a chart with all the dates and times when the eye drop bottle was used.

## 5.2. Future

Further development may see an improved machine learning module that is capable of detecting who is using Smart Eye Drops by considering movement of the eye drop bottle on all three axes mentioned in section 4.1. One suggestion would be to use Gated Recurrent Units (GRU), a newer generation of Recurrent Neural networks [9], instead of LSTM. This would need the user of Smart Eye Drops to calibrate the machine learning model to recognize their unique movements when applying eye drops and even the time taken for the action to complete. There would also need adjustments to be made to tweak the sensors and dampen the readings from accelerometer and pressure values for more precise data. By learning who is using the eye drops, we can further increase the accuracy of our classification algorithm. Having a more realistic approach on gathering data rather than a controlled environment will better help reduce certain inaccuracies.

The next iteration of Smart Eye Drops software could head in the direction of relocating the analytics and visualizations of patient adherence to a web-based platform. This would greatly benefit medical professionals as they will be able to view their patients’ information across many platforms. Implementing this would be fairly straightforward as the cloud services we used are modularized and can be replaced with other components, as discussed in section 1.5. It is also worthwhile to mention that the sensor system can also be programmed to communicate with the MBientLab MetaHub to transmit data to our cloud services, thus eliminating the need for a smartphone.

# 6. References

1. Robin, A. L., & Muir, K. W. (2019). Medication adherence in patients with ocular hypertension or glaucoma. *Expert Review of Ophthalmology*, *14*(4–5), 199–210. https://doi.org/10.1080/17469899.2019.1635456
2. Tsai, J. C., McClure, C. A., Ramos, S. E., Schlundt, D. G., & Pichert, J. W. (2003). Compliance barriers in glaucoma: A systematic classification. *Journal of Glaucoma*, *12*(5), 393–398. https://doi.org/10.1097/00061198-200310000-00001
3. Feldman, R. M., Cioffi, G. A., Liebmann, J. M., & Weinreb, R. N. (2020). Current knowledge and attitudes concerning cost-effectiveness in glaucoma pharmacotherapy: A glaucoma specialists focus group study. *Clinical Ophthalmology*, *14*, 729–739. <https://doi.org/10.2147/OPTH.S236030>
4. Lee, P. P., Walt, J. G., Doyle, J. J., Kotak, S. V., Evans, S. J., Budenz, D. L., Chen, P. P., Coleman, A. L., Feldman, R. M., Jampel, H. D., Katz, L. J., Mills, R. P., Myers, J. S., Noecker, R. J., Piltz-Seymour, J. R., Ritch, R. R., Schacknow, P. N., Serle, J. B., & Trick, G. L. (2006). A multicenter, retrospective pilot study of resource use and costs associated with severity of disease in glaucoma. In *Archives of Ophthalmology* (Vol. 124, Issue 1, pp. 12–19). American Medical Association. <https://doi.org/10.1001/archopht.124.1.12>
5. FSR 406. (n.d.). Retrieved May 9, 2020, from https://www.interlinkelectronics.com/fsr-406
6. MetaMotionC – MbientLab. (n.d.). Retrieved May 9, 2020, from https://mbientlab.com/metamotionc/
7. Illustrated Guide to LSTM’s and GRU’s: A step by step explanation. (n.d.). Retrieved May 9, 2020, from https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21
8. Ajerla, D., Mahfuz, S., & Zulkernine, F. (2019). A Real-Time Patient Monitoring Framework for Fall Detection. https://doi.org/10.1155/2019/9507938
9. Hochreiter, S., & Urgen Schmidhuber, J. J. (1997). (No Title). In MEMORY Neural Computation (Vol. 9, Issue 8). http://www7.informatik.tu-muenchen.de/~hochreithttp://www.idsia.ch/~juergen