Smart Eye Drops

Marco Aguilar, Aaron Alvarez, Ana Cortes, Jonathan Nunez, Feng You, Ivan Yu

CSULA College of Engineering, Computer Science, and Technology

California State University, Los Angeles, CA 90032

{aalva190, acortes7, fyou, iyu, jnunez94, maguil53}@calstatela.edu

***Abstract—* Patient adherence to eye drop medication has been a constant challenge to treatment plans for ocular diseases such as glaucoma. Failure to take medications as prescribed can significantly reduce treatment effectiveness and ultimately results in irreversible blindness. This research efforts aim to deliver a sensor system that can help physicians effectively monitor their patients’ adherence and accurately classify movements of an eye drop bottle using machine learning using data we collect. Our strategy makes use of sensors made by MBientLab and Interlink Electronics, an Android mobile application we developed, and a suite of Amazon Web Services cloud service such as API Gateway, EC2, DynamoDB, and SageMaker. Our completed system has a 5 second latency from data collection to visualization on an adherence chart, and our classification algorithm has an accuracy of 97%.**

***Keywords—* Adherence; Amazon Web Services; cloud; glaucoma; machine learning**; **Smart Eye Drops; thresholding.**

I. Introduction

Adherence to eye drop medication has been a significantly challenging factor in treating eye diseases such as glaucoma. Nonadherence poses a considerable burden on a country’s health system and remains significant global health concern [1]. 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 [2].

Because glaucoma is the second leading cause of irreversible blindness worldwide, accounting for slightly more than twelve percent of all global blindness [3], and is one of the prime examples of nonadherence in eye medication, we built Smart Eye Drops is built 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 adherence 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 indicated to us issues we can 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 [4]. Often, patients who no longer feel symptoms no longer feel the need to comply with their adherence schedule [5]. 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.

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. Typically, visual acuity tests are given to individuals to assess how well one is able to distinguish the shapes and details of objects. Research has shown that decline in visual acuity is often associated with substantial decline and deterioration in health-related quality of life domains such as driving, dependency, role limitations, and mental health [[6]](https://www.sciencedirect.com/science/article/pii/S0039625719302541). Overall, there has been an increase in the number of smart phone applications that aim to assist physicians to oversee ophthalmic conditions. Since poor vision is often correlated with a lower quality of life in a person, especially with the elderly, it makes sense that there are more apps out in the market to help mend this issue.

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 [7]. 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 [8]. 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.

II. Related Work

An application was designed by a team of students at the University of California, San Diego, used a standard eye drop container with a few sensors attached to it. Their team made use of the Google cloud service such as Google Firebase Realtime Database to store their data and configuration settings on which enable care providers or family members to adjust reminders based on the patient’s adherence progress [9].

A solution proposed by Nishimura et al. [10] used a 3-axis acceleration sensor that collected data to be processed by their Convolutional Neural Network. Their method of recording whether the eye drops were applied or not was based on when, and how long the bottle was tipped upside down by using a simple thresholding algorithm on data provided by their accelerometer. That data is stored on a SD card and processed at the end of their research.

Because of the uniqueness of Smart Eye Drops, there are scarcely any research efforts focused on detecting movement of an eye drop bottle. However, classifying the movement of an eye drop bottle is the same as classifying any other type of activity, we examined other types of activity recognition the implemented machine learning. Luna-Perejon et al. [11] proposed a system using 16 Recurrent Neural Networks (RNN) fall detection based on accelerometer data. Welhenge et al. [12] proposed using a Long Short-Term Memory (LSTM) network, an extension of RNN, to identify activities of daily living, which had a final accuracy of 90%. These studies indicated to us the requirements needed to detect the application of an eye drop using machine learning algorithms based on classifications of rare movement time series data.

III. Design Principles

Data collection from a cylindrical shaped eye drops bottle enabled us to decide on minimal sensor that allows us to collect data from a user as they apply a dose. Data collection from is a key essential part of this research as we need lots of data to analyze to make the right predictions as whether a user has applied a medical dose. The same cylindrical shaped eye drops bottle needed to be small enough to carry around, and resistant to accidental drops. In addition, eye drops bottle needed to track user usage of its medical dose efficiently and accurately all by retaining long life battery life.

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.

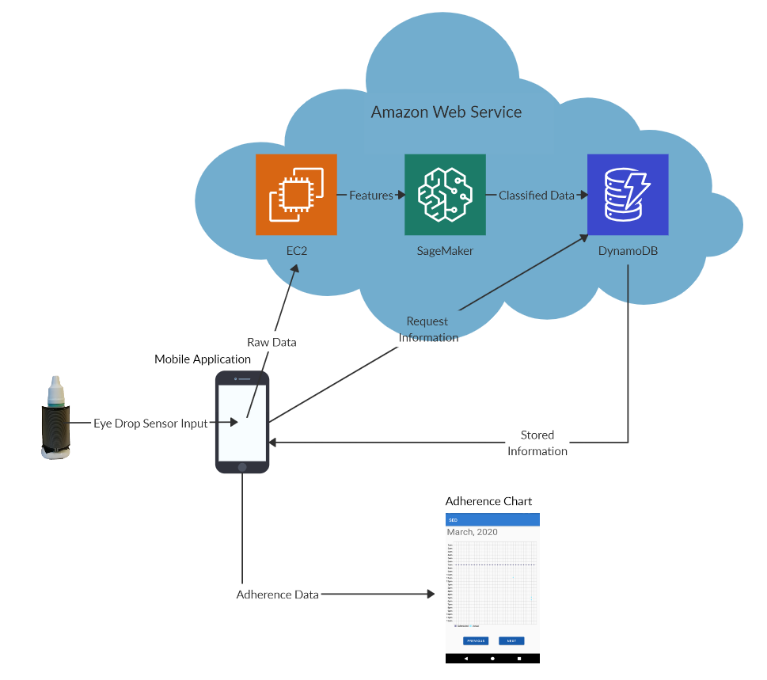
Secondly, a timer is trigged which allows the application to only gather raw data from the sensors. A timer allows the sensors to only send data to application when they are triggered. All sensors stop sending data to the application and collection of raw data is halted as the timer expires. A timer 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 medical dose.

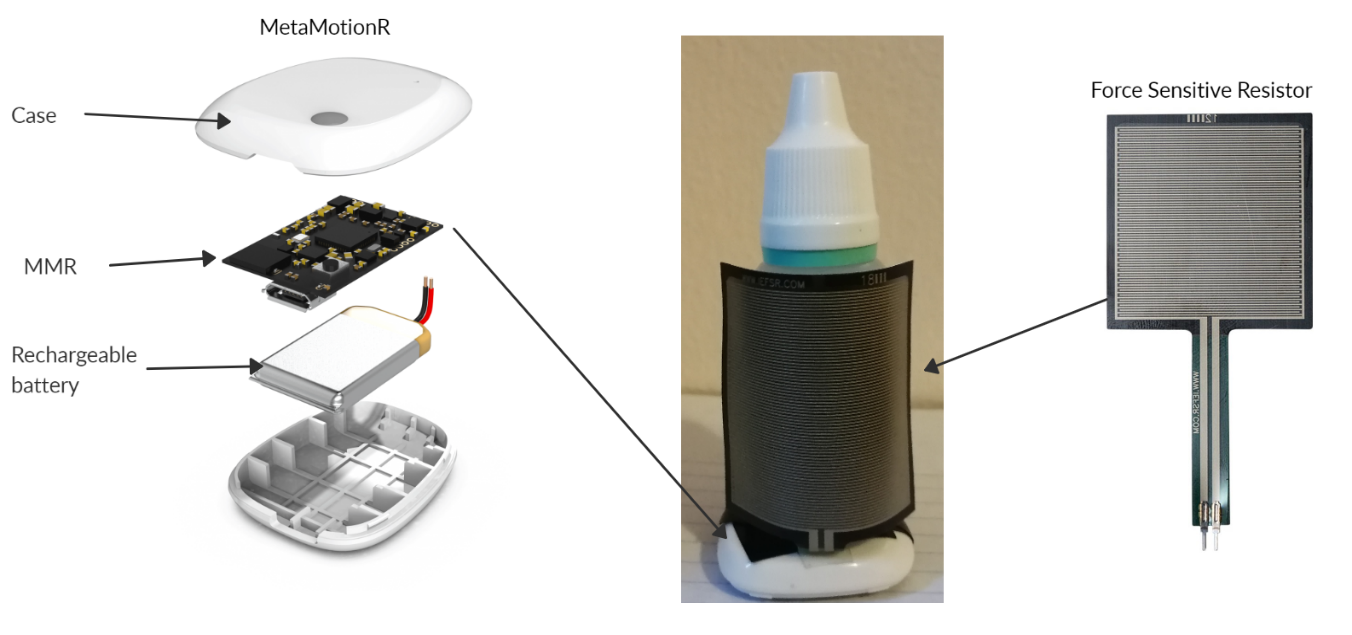
The benefits of creating a sensor system that has an external force sensor is that data collection only occurs when force and accelerometer readings exceed a certain threshold simultaneously. This increases the accuracy of our data and allows us to extend the battery life of our hardware and makes it 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.



*Figure 1: Flow Diagram*



*Figure 2: Smart Eye Drops Prototype*

IV. Hardware and Technologies Used

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. We 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 [13,14].

Our team used Interlink Electronics FSR 406 [15]. This Force Sensitive Resistor (FSR) was used to calculate the squeezing motion of the eye drop bottle. This FSR was chosen because it fit the best with the MBientLab Sensors. This FSR has a 39.6mm x 39.6mm pressure sensitive area. 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 as seen in Figure 2.

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 [16]. Next, we used Lambda to make serverless applications to write, read and delete from our database [17]. 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 [18]. 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 [19].

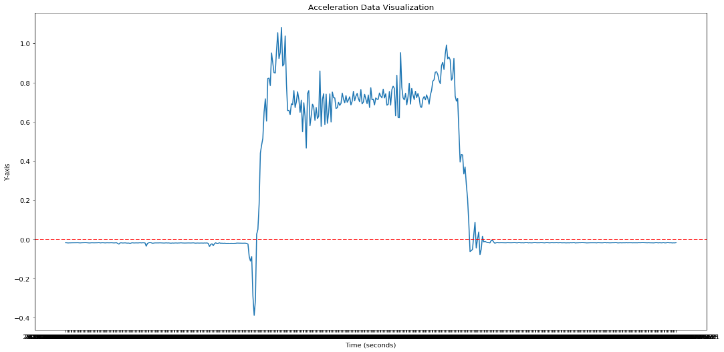
V. Methods

*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.

We determined the accelerometer data threshold as the acceleration value on the Y-axis when the eye drop bottle is laying on its side since this is the first orientation at which eye drops can be instilled. The pressure data threshold was experimentally determined by placing the eye drop bottle on its side and squeezing the bottle until a drop is displaced and pressure is immediately removed. The force data is collected and the point just before pressure is removed is the point at which an eye drop is ejected, and we used that point as the threshold value for force data.



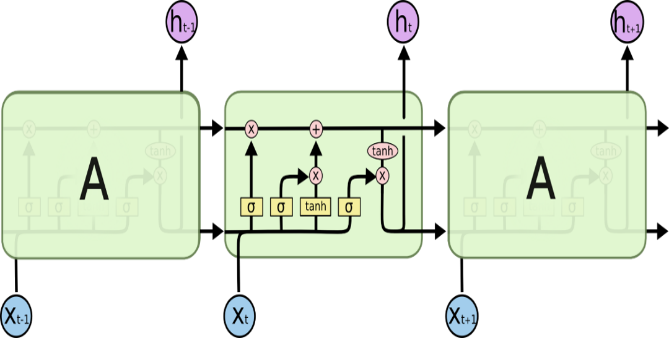
*Figure 3: Acceleration values recorded by our accelerometer on the Y-axis*

This method, in theory, can be sufficient to a certain degree to label the set of motions detected by our sensor system but its accuracy of 40% is inadequate 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.

*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 (RNN) [20], we were able to achieve 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.

When looking at the network for LSTM (Figure 4) [21], xt is the input, A is the network, h is the output hidden layer, and C is the cell state where all the information is stored. All 36 features are our input and they will go into the neural network. Because LSTM is a type of RNN, the output for each A would also be the input for the next A. The input is passed into the layers of the neural network, such as the forget gate, input gate, and tanh. The forget gate looks at the previous output and current input and determines if the current input is forgotten or kept. The input gate looks at the previous output and current input and determines which values to update. The tanh gets creates a vector for the new values to be saved into the cell state. Depending on the number of hidden layers the cell state will be trained and when testing, any new input will not affect the cell state. We used 30 LSTM units for our hidden layer and a dropout of 0.25, where if the forget gate produced a number less than 0.25 then that input would be forgotten.



*Figure 4: Long Sort-Term Memory Network*

For the base of our model, we drew inspiration from Dharmitha Ajerla’s model [22] 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% |

VI. 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 synthetic with slight variations of real data gathered this was not a surprise especially since synthetic data works well across a wide variety of tasks and domains [23]. The ratio of real data to synthetic was 1:80, but as we started to decrease the amount of synthetic data, we started to get less accurate results (Table 4). As less data is trained in our LSTM model 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, we used the highest accuracy rating we got when testing the data multiple times. There were a few inaccuracies such as 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 of when the eye drop bottle was used.

VII. Conclusions

In this paper, we analyzed a system for detecting the adherence rate of a patient using an eye drop bottle with the following sensors, accelerometer, and force sensitive resistor. We used LSTM and thresholding to determine if the eye drop bottle was used, with a 97% accuracy. We analyzed how having synthetic data mixed in with read data would affect the results. Having an eye drop adherence monitor will help physicians to accurately monitor their patients and modify their prescriptions based on the patient’s need.

References

1. *WHO | Priority eye diseases*. (2018). WHO; World Health Organization.
2. 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
3. Baum, A., Revenson, T. A., & Singer, J. E. (2001). *Hangbook of health psychology*. Lawrence Erlbaum Associates Publishers.
4. 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
5. Schwartz, G. F. (2005). Compliance and persistency in glaucoma follow-up treatment. *Current Opinion in Ophthalmology*, *16*(2), 114–121. https://doi.org/10.1097/01.icu.0000156139.05323.26
6. Hogarty, D. T., Hogarty, J. P., & Hewitt, A. W. (2020). Smartphone use in ophthalmology: What is their place in clinical practice? *Survey of Ophthalmology*, *65*(2), 250–262. https://doi.org/10.1016/j.survophthal.2019.09.001
7. 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*, *Volume 14*, 729–739. https://doi.org/10.2147/OPTH.S236030
8. 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. *Archives of Ophthalmology*, *124*(1), 12–19. https://doi.org/10.1001/archopht.124.1.12
9. Aguilar-Rivera, M., Erudaitius, D. T., Wu, V. M., Tantiongloc, J. C., Kang, D. Y., Coleman, T. P., Baxter, S. L., & Weinreb, R. N. (2020). Smart Electronic Eyedrop Bottle for Unobtrusive Monitoring of Glaucoma Medication Adherence. *Sensors*, *20*(9), 2570. https://doi.org/10.3390/s20092570
10. Nishimura, K., Tabuchi, H., Nakakura, S., Nakatani, Y., Yorihiro, A., Hasegawa, S., Tanabe, H., Noguchi, A., Aoki, R., & Kiuchi, Y. (2019). Evaluation of automatic monitoring of instillation adherence using eye dropper bottle sensor and deep learning in patients with glaucoma. *Translational Vision Science and Technology*, *8*(3). https://doi.org/10.1167/tvst.8.3.55
11. Luna-Perejon, F., Civit-Masot, J., Amaya-Rodriguez, I., Duran-Lopez, L., Dominguez-Morales, J. P., Civit-Balcells, A., & Linares-Barranco, A. (2019). An automated fall detection system using recurrent neural networks. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11526 LNAI*, 36–41. https://doi.org/10.1007/978-3-030-21642-9\_6
12. Malshika Welhenge, A., & Taparugssanagorn, A. (2019). Human activity classification using long short-term memory network. *Signal, Image and Video Processing*, *13*(4), 651–656. https://doi.org/10.1007/s11760-018-1393-7
13. *MetaMotionC – MbientLab*. (n.d.). Retrieved May 17, 2020, from https://mbientlab.com/metamotionc/
14. *MetaMotionR – MbientLab*. (n.d.). Retrieved May 17, 2020, from https://mbientlab.com/metamotionr/
15. *FSR 406*. (n.d.). Retrieved May 17, 2020, from https://www.interlinkelectronics.com/fsr-406
16. *Amazon API Gateway tutorials - Amazon API Gateway*. (n.d.). Retrieved May 17, 2020, from https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-tutorials.html
17. *Reading and Writing A Single Item in DynamoDB - AWS SDK for JavaScript*. (n.d.). Retrieved May 17, 2020, from https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/dynamodb-example-table-read-write.html
18. *What Is a Key-Value Database?* (n.d.). Retrieved May 17, 2020, from https://aws.amazon.com/nosql/key-value/
19. *Deploy trained Keras or TensorFlow models using Amazon SageMaker | AWS Machine Learning Blog*. (n.d.). Retrieved May 17, 2020, from https://aws.amazon.com/blogs/machine-learning/deploy-trained-keras-or-tensorflow-models-using-amazon-sagemaker/
20. Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735
21. *Understanding LSTM Networks -- colah’s blog*. (n.d.). Retrieved May 17, 2020, from https://colah.github.io/posts/2015-08-Understanding-LSTMs/
22. Ajerla, D., Mahfuz, S., & Zulkernine, F. (2019). *A Real-Time Patient Monitoring Framework for Fall Detection*. https://doi.org/10.1155/2019/9507938
23. Nikolenko, S. I. (2019). *Synthetic Data for Deep Learning*.