IntoxiGait Deep Learning

Investigating deep learning to predict intoxication levels using smartphones and smartwatches

A **Major Qualifying Project**
Submitted to faculty of
Worcester Polytechnic Institute
In partial fulfillment of the requirements for the
Degree of Bachelor of Science in **Computer Science**

Submitted By:

Joseph Bremner
Nicholas Cheung
Quoc Ho Lam
Sam Huang

Submitted To:

_____

Emmanuel Agu, *advisor*

Date: March 2, 2018

alcodeep@wpi.edu

Abstract

Alcohol abuse has been a pervasive problem worldwide, causing 88,000 annual deaths. In recent years, several projects have attempted to estimate a user's level of intoxication using gait and other mobile sensors. The goal of this project was to compare deep learning to existing machine learning methods in its ability to predict the blood alcohol concentration of a user by training a convolutional neural network and creating a mobile app which could accurately determine intoxication level. We gathered data from 38 participants over the course of 12 weeks, collecting accelerometer and gyroscope data simultaneously from both a smartwatch and smartphone. After performing our data preprocessing, the data is fed into our neural network. Our neural network's accuracy is roughly 64% on the test set and 69% on the training set into 5 BAC ranges[0.00, 0.00-0.08, 0.08-0.15, 0.15-0.25, 0.25-0.35] for a segment containing two seconds of data. Using only data from the phone's sensors, the classifier gives an accuracy of 71% on the training set and 62% on the test set into five bins. With only data from the watch, the accuracy is 34% for training and 28% for testing. When classifying into two bins, the test accuracy is 85% for both devices, 80% for the phone, and 57% for the watch. We employed a hyperparameter search to tune the architecture of our model, which features dual inputs into individual convolutional layers which act as feature extractors for a multilayer perceptron classifier. We also investigated data augmentations as methods to simulate a larger dataset, which we determined resulted in too much overfitting to be useful in practice. Lastly, we built and created a prototype app for testing purposes. The results of our evaluation show that deep learning is a viable method for processing smartphone and smartwatch accelerometer and gyroscope signals to detect intoxication.

Table of Contents

1.0 Introduction

This Major Qualifying Project (MQP) is a continuation of two previous studies: *Normalization and Personalization in Detecting Blood Alcohol Levels of Smartphone Users* (Aiello, 2015) and *AlcoWear: Detecting Blood Alcohol Levels from Wearables* (Bianchi, McAfee, & Watson, 2017). Both of these past studies investigated the use of machine learning to estimate Blood Alcohol Content (BAC) by classifying accelerometer and gyroscope sensor data gathered from their smartphones. In addition to gathering data off smartphone sensors, the second study included data gathered from smartwatches as well. We expand on these studies in Chapter 2.

Given recent advancements in the field of deep learning (University of Wisconsin), we investigate the use of deep convolutional neural networks rather than classic machine learning methods for our study. Since neural networks generally require more data, we attempted to gather data from a much larger sample size. Throughout our study, we attempt to determine if deep learning can be used to produce a more accurate BAC estimate of users.

1.1 Alcohol and Blood Alcohol Content

Alcohol is a common recreational substance that causes euphoria along with a decrease in anxiety and fine motor control in small doses, and stupor, loss of consciousness, and anterograde amnesia in large doses (Sacks, Gonzales, Bouchery, Tomedi, & Brewer, 2015). Long term misuse of alcohol is associated with many psychiatric disorders and can result in dependency, liver disease, cancer, and chronic pancreatitis.

Alcohol intoxication can be measured using the Blood Alcohol Content (BAC) metric, which is defined in North America as the mass in grams of alcohol (ethanol) in blood for every deciliter of blood. BAC can be estimated using breathalyzers, which measure the alcohol content of one's breath, field sobriety tests, which measure physical impairment due to alcohol, and urine tests, which measure the concentration of alcohol byproducts in urine. A direct measurement of BAC can be achieved through a blood test and has an accepted range of accuracy of 10% (Karch, MD, FFFLM, Steven B, 2016). In the United States, a properly administered breath or blood test is admissible. Although the exact laws vary by state, the portable breathalyzer preliminary test is often not considered accurate enough to be used as evidence of driving under the influence.

1.2 Alcohol Abuse

Excessive alcohol use accounts for 88,000 deaths in the United States each year. Excessive drinking cost the United States $249 billion in 2010 (Mokdad, Marks, Stroup, & Gerberding, 2004). This cost was a result of losses in workplace productivity, health care expenditures, and criminal justice costs, among other expenses. The most common and deadly pattern of excessive alcohol use is binge drinking. Binge drinking accounted for about 70% of the aforementioned cost (Sacks et al., 2015). The National Institute on Alcohol Abuse and Alcoholism defines binge drinking as any form of drinking that brings a person's blood alcohol content (BAC) to 0.08 or higher. About one in six adults in the US binge drink around four times a month (Kanny, Liu, Brewer, & Lu, 2013). Excessive alcohol use carries serious risks such as unintentional injuries, violence, fetal alcohol spectrum disorders, sudden infant death syndrome, chronic diseases, various cancers, and alcohol dependence.

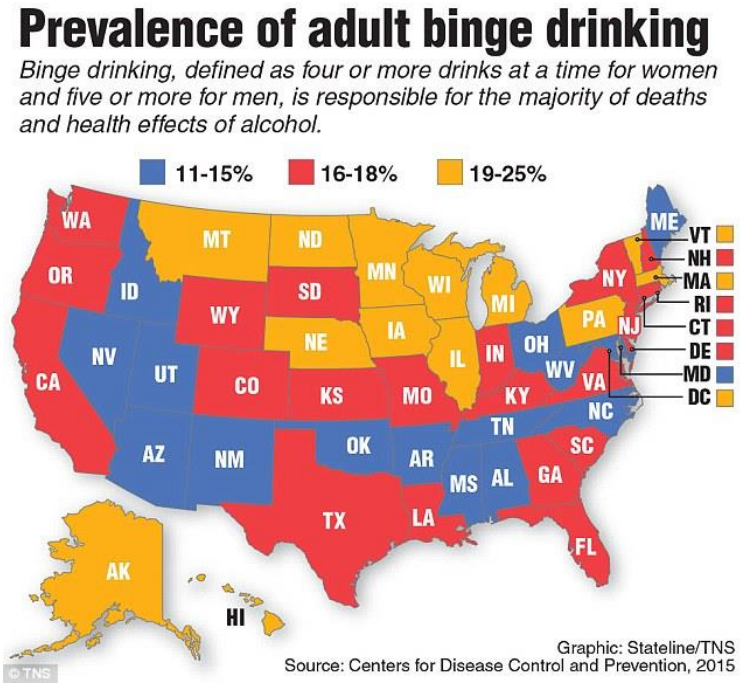

Figure 1: Prevalence of binge drinking throughout U.S. states according to the CDC. Percentages were calculated using data expressing the average number of drinks people claimed to drink per session (Smith, 2016).

Those who suffer from alcohol dependence often desire to live a healthier lifestyle and many organizations have been established to encourage their sobriety. Notably, there are two million members of the mutual aid fellowship Alcoholics Anonymous worldwide and 14,500 specialized drug addiction treatment facilities in the United States (NIDA, 2012). However, in 2012, the Substance Abuse and Mental Health Services Administration reported that less than

11% of the 23 million estimated drug addicts received treatment at a facility (SAMHSA, 2013). Even so, the addiction rehabilitation industry grossed $35 billion in 2014 (Munro, 2015).

1.3 Driving Under the Influence

Alcohol consumption slows a person's reaction time and impairs their judgement and coordination. A study on perceived dangers while intoxicated found that intoxicated individuals perceive drunk driving to be less dangerous than when they are sober. The study found that perceived danger when estimated BAC level decreases was strongly associated with willingness to drive (Morris, Treloar, Niculete, & McCarthy, 2013). Another study in 2016 published in the open access journal *BMC Public Health* found that while people are intoxicated and in a drinking environment, they based their own drunkenness relative to others around them. People are more likely to underestimate their own level of intoxication when surrounded by others who are also intoxicated (S. Moore et al., 2016). Both of these studies raise the issue that people may drink and drive due to poor assessment of their condition.

In the recent decades, drunk driving has been a major cause of traffic-related accidents. According to the National Highway Traffic Safety Administration (NHTSA) 2015 Traffic Safety Facts report, there were 10,265 fatalities in motor vehicle traffic crashes involving drivers with BACs of 0.08 g/dL or higher; this number accounts for 29% of all traffic-related deaths (National Highway Traffic Safety Administration, 2016). When compared to other age groups, the 21-24 year old age group had the highest percentage (28%) of drunk drivers involved in fatal crashes in 2015.

Table 4
Total and Alcohol-Impaired Driving Fatalities,* 2014 And 2015

	2014	2015	Change	% Change
Total Fatalities	32,744	35,092	+2,348	+7.2%
Al-Driving Fatalities	9,943	10,265	+322	+3.2%
Alcohol-Impaired Drivers in Fatal Crashes by Vehicle Type				
Passenger Cars	3,892	4,085	+193	+5.0%
Light Truck - Vans	246	214	-32	-13.0%
Light Truck - Utility	1,494	1,529	+35	+2.3%
Light Truck - Pickups	1,936	1,900	-36	-1.9%
Motorcycles	1,370	1,365	-5	-0.4%
Large Trucks	68	60	-8	-11.8%

Source: FARS 2014 Final File, 2015 ARF
*See definition in text.

Figure 2: Data collected by the NHTSA showing the increases in alcohol-related car crashes from 2014 to 2015 (National Highway Traffic Safety Administration, 2016).

Traffic crashes due to drunk driving also impose an enormous financial burden on the United States. A study by the Pacific Institute for Research and Evaluation (PIRE) found that drunk driving costs the United States more than $132 billion in 2009 alone. This costs includes $61 billion in monetary costs and $71 billion in quality-of-life losses. Aside from the physical consequences, victims of these crashes suffer great costs from insurance payments, deductibles, uncovered costs, and uninsured expenses (National Highway Traffic Safety Administration, 2015). In the event that the victims were involved in a fatal crash, their dependents suffer immediate emotional and economic hardship (National Highway Traffic Safety Administration, 2015). Society as a whole is also affected by these crashes. The cost of medical care due to these crashes "is borne by society through higher insurance premiums and through the diversion of medical resources away from other medical needs" (National Highway Traffic Safety Administration, 2015).

1.4 Wearables

Wearable technology consists of smart electronic devices that are worn on the body as accessories. Examples of wearable technology are smart watches, sport watches, and fitness trackers. Most recent wearable devices have built-in sensors that can collect and report information about their users and surroundings. In this sense, owners of smart devices might find use in applications that can constantly run in the background, monitoring and providing feedback on different aspects of their lifestyles that can be measured with sensors. In this report, the sensors of interest are gyroscopes and accelerometers. Using deep learning, we seek to classify gait data gathered from the gyroscope and accelerometer built into smartwatches and smartphones.

1.5 Using Wearable Technology to Measure Gait

Gait is a person's manner of walking. As it relates to this project, one common side effect of alcohol intoxication is an altered gait (Nieschalk et al., 1999). As such, we will build upon the successes of past groups by using gait as our indicator of intoxication. While gait analysis has been conducted since the late 19th century, the introduction of wearable sensors in recent decades has opened new avenues for the collection of gait-related data. Today, nearly all smartphones and smartwatches are natively equipped with both accelerometers, which measure acceleration, and gyroscopes, which determine rotational speed.

Figure 3: Smartwatches (left) and smartphones (right) used in our study

Due to the nature of smartphones and smartwatches as wearable motion sensors, they are naturally an excellent source of big data. In gait analysis, singular, atomic data points are often measured in terms of bipedal gait cycles, which consists of the time period during human locomotion between one foot contacting the ground and the same foot contacting the ground again. In other words, a data point corresponds to two separate steps. In Chapter 2, we will provide more information about gait analysis.

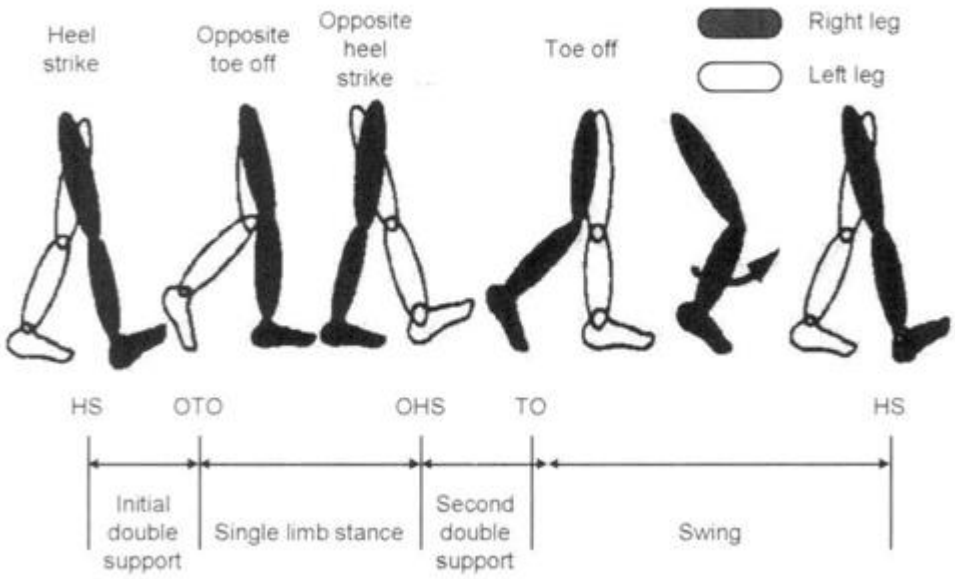

Figure 4: A visual depiction of one gait cycle (Choi, Chou, Su, & Lin, 2003).

1.6 Machine Learning

Machine learning is an application of artificial intelligence that allows computers to train a model that learns from a set of features extracted from data and then generalize predictions on new data. While there are many different methods and algorithms used in machine learning, this report will focus on the theory and application of artificial neural networks and deep learning.

Artificial neural networks (sometimes abbreviated ANNs) are loosely modeled after the biological neural networks found in brains, featuring a directed system of interconnected nodes or neurons that each process inputs and produce outputs. Each connection between nodes is associated with a weight and activation function, forming a directed, weighted graph. When data is processed through this network, the weights are adjusted such that they model a mathematical function which allows a given input to produce a predicted output. Once the network has been trained on a large quantity of data, it can generalize new data and yield similar favored results (Goodfellow, Bengio, & Courville, 2016).

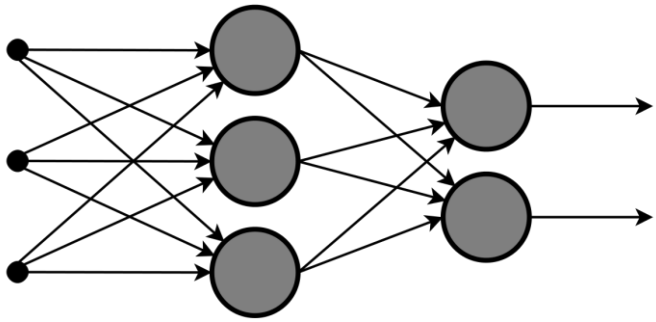

Figure 5: Depiction of nodes with dense connections in an ANN.

Nodes in an artificial neural network are organized into layers, the first of which is the input layer, and the last being the output layer. In between, information is passed only to other neurons and is thus not visible. These layers are called hidden layers. In a traditional neural network, sometimes called a multilayer perceptron, one hidden layer is used to process and represent the nonlinearity of data. Recently, researchers have been experimenting with multiple hidden layers to model more complex data, giving rise to deep learning (University of Wisconsin).

1.7 Deep Learning

Deep learning offers an enhanced ability to learn based on data while having the drawback of requiring more data for training. Recently, it has been sensationally successful in

fields such as computer vision, speech recognition, natural language processing, and machine translation, where breakthroughs in deep neural network architecture has allowed computers to excel in complex learning tasks for the first time (Goodfellow, Bengio, & Courville, 2016).

While all deep neural networks contain multiple hidden layers, some have specialized structure and content that allow for specialized use. This study involves a convolutional neural network or CNN, which contains one or more sparsely connected "convolutional" layers along with the standard fully connected layer, and is commonly used for image processing. By having locally shared edge weights, CNNs can preserve the spatial and temporal relationships in data, which is naturally useful in computer vision, speech recognition, and other applications.

Deep learning is most often applied when there is significant complexity of data. The ability to transform data in hidden layers into higher level features or representations removes the need for feature engineering, which was a major task in previous similar studies (see section 2.2 AlcoWatch). We hypothesized that the accelerometer data collected from a smartwatch while walking may have complexities beyond that which we could currently project and we hypothesized that deep learning would help learn these patterns in the data. Comparatively, data gathered from the cell phone is more straightforward in that it can be used to estimate the amount of swaying about one's trunk, a known estimator of intoxication level (Arnold & LaRose, 2015). To expand upon existing research, deep learning was the method chosen to best model gait data gathered from smartwatches and smartphones with respect to inebriation.

1.8 The Goal of this MQP

Through this project, we investigated the effectiveness of deep learning models to detect intoxication using data from accelerometer and gyroscope sensors. In order to accomplish this, we expanded on the work previously done by the AlcoGait (Arnold & LaRose, 2015) and AlcoWatch (Bianchi et al., 2017) teams through the following objectives:

1. Recruit 50 subjects: As deep learning is most effective with a large amount of data, we aimed to expand on the number of test subjects gathered by previous studies. Specifically, we wanted to gather roughly 100 test subjects through various means.

2. Use sensors on a smartwatch and smartphone to gather data: With the data collecting applications inherited from the prior teams, we conducted test studies with each participant in order to gather as much data as possible.

3. Train a model using deep learning: In order to improve upon the previous team's results, we planned on classifying our data using deep learning techniques. Our goal is to create a

model more accurate than the previous MQP project, AlcoGait (Aiello, 2015). Christina Aiello was able to achieve 69% accuracy splitting with a 66% - 33% percentage split on the training set.

4. Create and test an app to measure intoxication levels: We built a new application, AlcoDeep BAC Estimator, allowing users to use our work to estimate intoxication levels.

5. Compare to previous results: We compared the result of our model trained with deep learning to the previous studies to draw conclusions on the effectiveness of DCNNs for gait analysis using accelerometer and gyroscope data.

1.9 Significance of Study

When over the alcohol driving limit, many people are not able to determine their exact level of intoxication. (S. C. Moore et al., 2016) By investigating the effectiveness of deep learning, we hope that further studies are able to detect drunk drivers just in time and reduce the number of accidents relating to drivers who were unaware of their intoxicated state (Nieschalk et al., 1999).

Future research and development in this area, such as analysis of the correlation between arm swing changes and intoxication levels, could result in an alcohol tracking app that will alert those who are past a certain intoxication level of their current physical state. While existing technology such as breathalyzers have a similar usage, they are much more expensive, less convenient and require purchase beforehand and diligent use every time the user drinks. It's possible that further exploration into this area could expand the application of studies like this to give a general intoxication level measurement during activities beyond walking.

2. Related Work

2.1 AlcoGait

The Smartphone Gait Inference Android application, also known as AlcoGait, was originally developed by Worcester Polytechnic Institute (WPI) students as a Major Qualifying Project (MQP) (Arnold & LaRose, 2015). The objective of Zachary Arnold and Danielle LaRose's study was to test various machine learning methods in order to use a smartphone's accelerometer to infer how many drinks a user has consumed. Over a period of two weeks, they used a mobile application to measure the gaits of seven users. After extracting features such as cadence from the accelerometer readings, the team applied several machine learning classifiers to roughly determine the number of drinks consumed. Using the random forest classifier, they were able to successfully infer how many drinks the user had consumed 56% of the time on the training set and 70% of the time on the validation set, using three "bins" (sober, tipsy, drunk) for classification. The classifier was then used in a mobile application that could measure intoxication. The authors of AlcoGait noted that it was strange to have a data set with a greater accuracy in the validation set than the training set.

After the first app was distributed and the report was finalized, the project was expanded upon by Christina Aiello, a graduate student at WPI (Aiello, 2015). While the previous team only used accelerometer data, Aiello incorporated gyroscope data into her project to generate more features. Additionally, the initial application was built upon to record additional information about the users, such as height and gender. Aiello found more success after having incorporated the gyroscope, obtaining 89.45% accuracy using the J48 classifier and labeling the data into five bins. While Aiello found that adding user personalization, or creating a model solely based on the data of a single individual, rendered inconclusive effects, it was determined that adding personal physical characteristics to the classification process improved accuracy overall.

2.2 AlcoWatch

Following Aiello's master thesis, another WPI MQP team consisting of Benjamin Bianchi, Andrew McAfee, and Jacob Watson expanded AlcoGait by adding smartwatches to the smartphones and testing similar functionality in a project titled AlcoWatch (Bianchi, McAfee, & Watson, 2017). In addition to building their application to work on a smartwatch, the team also conducted a detailed study into the benefits and downfalls of using smartphones versus using smartwatches. In their research, the team found that smartwatches provided the added benefit

of being continuously tethered to the user throughout the day. According to a study by A.K. Dey et al., smartphones are only within arm's reach of the user on average around 54% of the phone's powered on time (Dey et al., 2011). The team found success using smartwatches by performing classification using the Random Forest method on extracted arm swing features, which achieved an accuracy of 79.68%, labeling the data into one of two bins (above and below the legal limit, 0.08 BAC). However, they also found that adjusting the original AlcoGait smartphone app to label the data into two bins instead of five resulted in a 100% success rate using the same technique, with only the phone.

While it was evident to the team that the smartphone was still yielding better results than the smartwatch using machine learning approaches, the fact that the success of their classification was possible only by classifying intoxication levels into two bins is absolutely crucial for further studies such as ours, as it paved the way for more complex methods of classification to be performed on data received from smartwatches. We hope to expand this to allow the watch and phone combination to excel in the five-bin classification category for a more accurate and usable estimation of BAC.

2.3 Mobile Phone-Based Drunk Driving Detection

Dai et al. conducted a study with the goal of creating an Android mobile system that aimed to detect any dangerous vehicle maneuvers that would indicate drunk driving. The system they proposed would rely on accelerometer and gyroscope sensors. Their detection algorithm design was as follows: (1) process and graph sensor data (2) compare the resulting graphs with typical drunk driving patterns. (3) if any evidence of drunk driving is present, the mobile phone will alert the driver or call the police.

Dia et al. measured the performance of their system in terms of false negative (FN), when drunk driving related behaviors show up but the system misses them, and false positive (FP), when the system reports drunk driving related behaviors when the vehicle is actually under regular driving conditions. In their experiments, they tested the detection performance in abnormal curvilinear movement and problem in maintaining speed, both of which are two main categories of drunk driving related behaviors. In detection of abnormal curvilinear movement, the FN rate was 0% and the FP rate was 0.49%. In detection of speed control problem, the FN rate was also 0% and the FP rate was 2.39%. The energy efficiency of the detection system was also evaluated using an Android G1 phone. The G1 phone was fully charged and the power state was monitored continuously for 7 hours in two different scenarios: (1) the G1 phone runs

without the drunk driving detection system and (2) the system is running on the G1 phone. Figure 6 shows a graph of the power consumption curve in both of these scenarios.

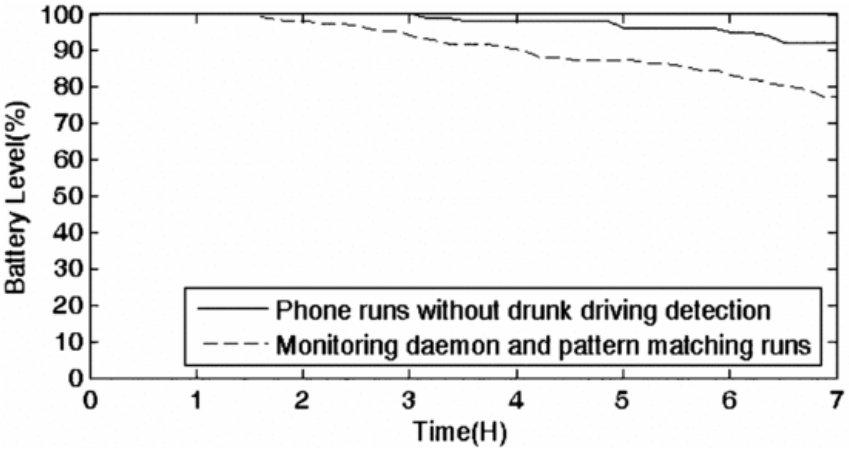

Figure 6: A graph showing the battery level over time of both when the drunk driving detection app was being run and when it was not (Dai et al.).

2.4 Human Activity Recognition using Wearable Sensors by Deep CNNs

Wenchao Jiang and Zhaozheng Yin of the Missouri University of Science and Technology researched activity recognition using ubiquitous wearable devices, such as smartphones and smartwatches (Jiang & Yin, 2015). The activities in their study were walking, standing, and lying down. Unlike previous studies which extracted features independently from multiple time series sensors, Jiang and Yin proposed transferring all the signals from the accelerometer and gyroscope into an activity image. This activity image was then fed to a deep convolutional neural network (DCNN), which outputted a probability distribution over activity categories. This proposed approach to activity recognition was evaluated on three public datasets.

The architecture of Jiang and Yin's proposed DCNN is summarized in figure 7 below. In short, Jiang and Yin's activity recognition procedure consisted of three steps: "(1) generating an activity image from raw signals; (2) apply DCNN to the activity image and produce the probability distribution over all activities; and (3) if there is a sharp peak in the distribution, the activity categorization is confidently determined. Otherwise, bi-class SVM classifier will be applied to classify uncertain classes" (Jiang & Yin, 2015).

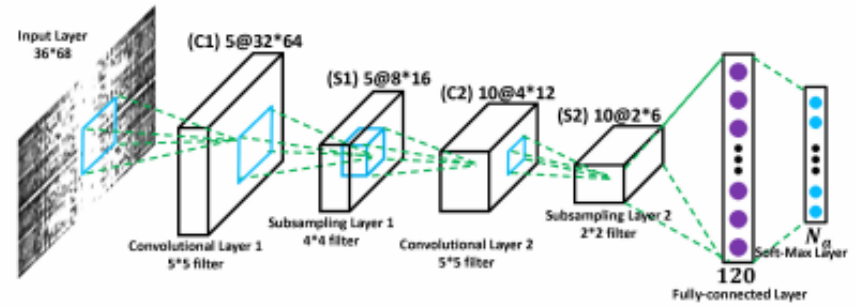

Figure 7: Architecture of Jiang's and Yin's deep convolutional neural network (Jiang & Yin, 2015).

Jiang and Yin compared their proposed methods, DCNN and DCNN+ (the DCNN with the aid of SVM for uncertain cases), with two other state-of-the-art classification methods: Support Vector Machine (SVM) and Random Forest with feature selection. In their comparison, they found that the DCNN and DCNN+ achieved superior performances than the other two methods in terms of recognition accuracy and computational cost. They performed a quantitative comparison in terms of recognition accuracy and computational costs using three public datasets. Figure 8 shows a table of the performance comparison of all the methods. The DCNN+ and DCNN methods achieved the best performance in accuracy and computational cost, respectively. Although the SVM method had the second best accuracy, it had a computational cost that was significantly higher than the other three methods. In contrast, the Random Forest feature selection method had the second lowest computational cost but it also had lowest accuracy rate among the four methods.

This study, like our study was based on Accelerometer and Gyroscope data and used a CNN. As such, it was one of the first studies we looked at to see if similar studies had been conducted in the past. Likewise, we used this study as a way to see how feasible it would be to use MATLAB as a deep learning framework, a route which we eventually decided against.

Dataset	Accuracy (%)				Computational Cost (ms)			
	DCNN+	DCNN	SVM[1]	Feature Selection[4]	DCNN+	DCNN	SVM[1]	Feature Selection[4]
UCI[1]	**97.59**	95.18	96.40	91.31	3.85	**1.56**	10.06	1.81
USC[2]	**97.83**	97.01	97.28	96.77	2.66	**1.54**	11.78	1.86
SHO[3]	**99.93**	99.93	99.93	99.58	1.56	**1.53**	9.87	1.56

Figure 8: A table showing a comparison in performance of Jiang and Yin's proposed DCNN and DCNN+ models to Support Vector Machine (SVM) and Feature Selection models. (Jiang and Yin)

2.5 Improvements from Machine Learning to Deep Learning in Visual Recognition

Francois Fleuret of the Idiap Research Institute used a boosted machine learning algorithm to categorize different images with Binary Classification. The study is titled *Comparing machines and humans on a visual categorization test* (François Fleuret et al., 2011). Despite the simplicity of the machine learning tasks, the performance of the machine learning method was still far below that of human performance. The study concluded that using their methods, it would be unlikely that the machine learning algorithm used would be able to surpass human performance without a larger data set.

Samuel Dodge and Lina Karam of Arizona State University created a study titled, *A Study and Comparison of Human and Deep Learning Recognition Performance Under Visual Distortions (Dodge & Karam, 2017)*. The goal of the study was to train a deep neural network for the purpose of image recognition and to understand the effects of distorted images.

This study by Dodge and Karam referenced the study by Fleuret and reiterated that the tests were not able to create a machine learning algorithm that could approach human-level accuracy in image recognition. However, they proposed that their use of deep learning would yield more success than the previous studies. By giving the network a larger set of images including a set of distorted images, they hoped to minimize the difference between human performance and machine performance. The team believed that the deep neural network's proficiency in identifying features without being specified would be an improvement on the manually extracted features used in the past machine learning algorithms. They found that their classifier performed roughly as well as humans on clean images, but still could not reach human levels of accuracy on distorted images. While this is an improvement over machine learning, it leaves room for further research to be done.

Fluret's experiment used machine learning while Dodge and Karam used deep learning with a CNN, and the result was a better accuracy using machine learning. The improved accuracy of deep learning over machine learning is promising when comparing our project with the AlcoGait and AlcoWatch projects as we also implemented a CNN, even if our project does not feature image recognition.

2.6 Sleep Quality Prediction from Wearable Data Using Deep Learning

Sathyanarayana et al. conducted a study that investigated the feasibility of using deep learning to predict sleep quality given data from wearables (Sathyanarayana et al., 2016). Their

experiment included 92 adolescents and was conducted over a full week. They collected data from their participants by having each of them wear actigraphy sensors or wearable medical devices used to study sleep and physical activity patterns, around their wrists. Participants were told to not remove the sensors at any time during the week this experiment was conducted. They were also given a diary to report times during which they were asleep.

The deep learning models compared in this study were deep multi-layer perceptrons (MLPs), convolutional neural networks (CNNs), recurrent neural networks (RNNs), long short-term memory (LSTM) RNNs, and a time-batched long short-term memory (TB-LSTM) RNNs. Sathyanarayana et al. compared the performance of these models to logistic regression (LR), a standard machine learning model, and found that deep learning models were able to more accurately predict the quality of sleep based on data from awake periods, times in which the participants were not sleeping, as shown in Figure 9. Sensitivity or recall refers to the percentage of the correctly predicted "good-quality sleep" to the total number of "good-quality sleep" instances in the dataset. Specificity refers to the percentage of the correctly predicted "poor-quality sleep" to the total number of "poor-quality sleep" instances.

Data models	Sensitivity or recall	Specificity
LR[a]	0.9714	0.333
MLP[b]	0.8857	0.9048
CNN[c]	0.9714	0.8571
RNN[d]	0.9143	0.2381
LSTM[e]	0.9714	0.4762
TB-LSTM[f]	1.000	0.7143

[a]LR: logistic regression.

[b]MLP: multilayer perceptrons.

[c]CNN: convolutional neural networks.

[d]RNN: recurrent neural networks.

[e]LSTM: long short-term memory.

[f]TB-LSTM: time-batched LSTM.

Figure 9: A table showing the results of Sathyanarayana et al.'s study on the performance of different learning models in predicting sleep quality. (Sathyanarayana et al.)

2.7 Deep Learning for Driving Detection on Mobile Phones

Ramaiah et al. researched the effectiveness of using CNNs combined with accelerometer and gyroscope data for driving detection (Ramaiah, Tran, Cox, & Mohler, 2016). In their paper, the group notes that two big advantages to CNNs are (1) avoiding the time-intensive process of

designing features and (2) the higher accuracy rate compared to the common approach of inputting manually engineered features into a discriminative classifier. Their experiment includes data collected from 2,133 drivers, using Android smartphones. Two minute samples from each device were recorded at random points in time. For each sampling, there was a platform specific activity labeling that went along with the accelerometer and gyroscope measurements. The group removed samples when the activity labels and sensor measurements suggested that the phone was inactive. In total, there was about 5,400 hours of data, excluding samples where the phone was inactive. Each of the remaining samples were split into 30 seconds windows with 50 percent overlap. Each window was then labeled as walking, stationary, or driving for the purposes of supervised learning. To avoid the risk that driving could occur in other vehicles, the group selected drivers that only had a single vehicle listed on their associated insurance policy.

The group experimented with 3 different CNN architectures: (1) a spectrogram CNN, (2) a time series CNN, and (3) a multi-stream spectrogram CNN. The architecture of their CNNs is shown in Figure 10. All of their convolution and dense layers are activated with ReLu and dropout was used for regularization, which is common for convolutional neural networks.

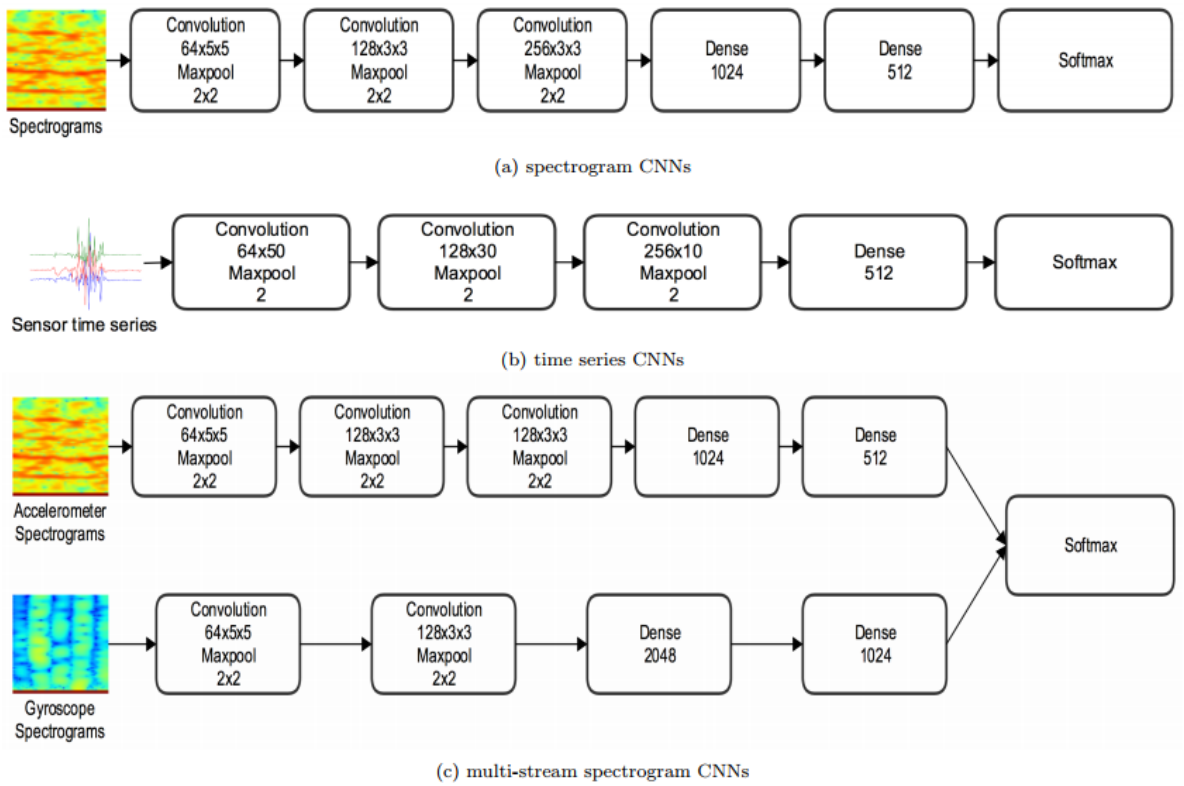

Figure 10: CNN architecture used by Ramaiah et al. to detect walking, driving, or standing still. The numbers indicate filter/kernel sizes. The approaches shown are (a) spectrogram CNN, (b) 1D convolutions over raw sensor data, and (c) multi-stream CNN with accelerometer and gyroscope spectrograms

The group evaluated the efficiency of their approach when used for transportation mode detection and driving detection. Transportation mode detection is a classification problem amongst three classes in their dataset (1) Walking, (2) Automotive, and (3) Stationary. Driving detection is simply classifying among two classes, either driving or not driving. In both classification problems, the group compared their results to random forest classifier, a discriminative classifier. As shown in Figure 11, the group found that classification accuracy of deep learning techniques improve significantly with increasing dataset size.

Dataset Size	Baseline	Accelerometer CNN
50 hours	**0.82**	0.78
150 hours	**0.82**	0.82
500 hours	0.84	**0.86**
1000 hours	0.84	**0.88**
1300 hours	0.83	**0.89**

Figure 11: This shows the AUC scores for the transportation mode detection problem with varying dataset sizes. The baseline in this figure is the random forest classifier.

This study shares our study's involvement with convolutional neural networks operating on accelerometer and gyroscope data for classification purposes. Their architecture design featuring the feature extraction convolutional layers followed by the dense and softmax classification layers is similar to ours, though our neural network has an additional input for our watch data. We also evaluate the performance of our neural network to the result of a random forest classifier.

2.8 Investigating Neural Network Architectures in Related Studies

Title	Authors	Architecture
A Deep Learning Approach to on-Node Sensor Data Analytics for Mobile or Wearable Devices (Ravi, Wong, Lo, & Yang, 2017)	Daniele Ravi, Charence Wong, Benny Lo, Guang-Zhong Yang	CNN
Human Activity Recognition using Wearable Sensors by Deep Convolutional Neural Networks (Jiang & Yin, 2015)	Wenchao Jiang, Zhaozheng Yin	CNN
A Deep Learning Approach to Human Activity Recognition Based on Single Accelerometer (Chen & Xue, 2015)	Yuqing Chen, Yang Xue	CNN
Deep Activity Recognition Models with Triaxial Accelerometers (Alsheikh et al., 2015)	Mohammad Abu Alsheikh, Ahmed Selim, Dusit Niyato, Linda Doyle, Shaowei Lin, Hwee-Pink Tan	DBN
Human Activity Recognition From Accelerometer Data Using Convolutional Neural Network (Lee S., Yoon S., & Cho H., 2017)	Song-Mi Lee, Sang Min Yoon, Heeryon Cho	CNN
DeepSense: a Unified Deep Learning Framework for Time-Series Mobile Sensing Data Processing (Yao, Hu, Zhao, Zhang, & Abdelzaher, 2016)	Shuochao Yao, Shaohan Hu, Yiran Zhao	CNN, RNN
Deep Residual Bidir-LSTM for Human Activity	Yu Zhao, Rennong Yang,	LSTM RNN

Recognition Using Wearable Sensors (Zhao, Yang, Chevalier, & Gong, 2017)	Guillaume Chevalier, Maoguo Gong	
Human activity recognition with smartphone sensors using deep learning neural networks (Ronao & Cho, 2016)	Charissa Ann Ronao, Sung-Bae Cho	CNN
Robust and accurate feature selection for humanoid push recovery and classification: deep learning approach (Semwal, Mondal, & Nandi, 2017)	Vijay Bhaskar Semwal, Kaushik Mondal, G. C. Nandi	MFNN
Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition (Ordóñez & Roggen, 2016)	Francisco Javier Ordóñez, Daniel Roggen	CNN, LSTM RNN
An Early Resource Characterization of Deep Learning on Wearables, Smartphones and Internet-of-Things Devices (Lane, Bhattacharya, Georgiev, Forlivesi, & Kawsar, 2015)	Nicholas D. Lane, Sourav Bhattacharya, Petko Georgiev, Claudio Forlivesi, Fahim Kawsar	CNN
Deep, Convolutional, and Recurrent Models for Human Activity Recognition using Wearables (Hammerla, Halloran, & Ploetz, 2016)	Nils Y. Hammerla, Shane Halloran, Thomas Ploetz	MFNN, CNN, LSTM RNN
Time Series Classification Using Multi-Channels Deep Convolutional Neural Networks (Zheng, Y., Liu, Q., Chen, E., Ge Y., Zhao, J.L., 2016)	Yi Zheng, Qi Liu, Enhong Chen, Yong Ge, and J. Leon Zhao	CNN
Human activity recognition with inertial sensors using a deep learning approach (Tahmina Zebin, Patricia J Scully, & Krikor B. Ozanyan, 2016)	Tahmina Zebin, Patricia J Scully, Krikor B. Ozanyan	CNN

Table 1: Studies on related data, showing chosen neural network architectures.

2.8.1 Selection of CNNs as Type of Neural Network

Table 1 highlights studies that work with data similar to ours, namely accelerometer and/or gyroscope time series data on smartphones and/or wearables. Of the fourteen studies listed, only three did not use convolutional neural networks. During our preliminary research, we found their justifications as well as the general theory behind CNNs to be aligned with our goals for the project. Specifically, the freedom from the need for manual feature engineering and the ability to preserve the temporal locality in the signal data were the most desirable facets of CNNs with respect to studies involving wearable sensors. All of the studies found great accuracy in their classifications using CNNs and in Hammerla et al.'s comparison of Multi-layer Feedforward Neural Networks, Long-Short Term Memory Recurrent Neural Networks, and Convolutional Neural Networks, the CNN produced the most success. This was common across studies with similar nature to ours, and for all of these reasons, we decided to implement a CNN for this project.

3. Methodology

In building upon the previous work conducted by the AlcoGait and AlcoWatch teams, we recruited 100 participants from whom we collected raw sensor data. Our main focus for the project was to perform deep learning tasks on the data we collected. After designing our convolutional neural network architecture, we analyzed and optimized it to best model the data. Then, after implementing our server and client app, we were able to test the effectiveness of the model in a practical environment. Finally, we evaluated the success of our results, particularly in comparison to the results obtained by past project groups. Our methodology is illustrated at a high level in Figure 12.

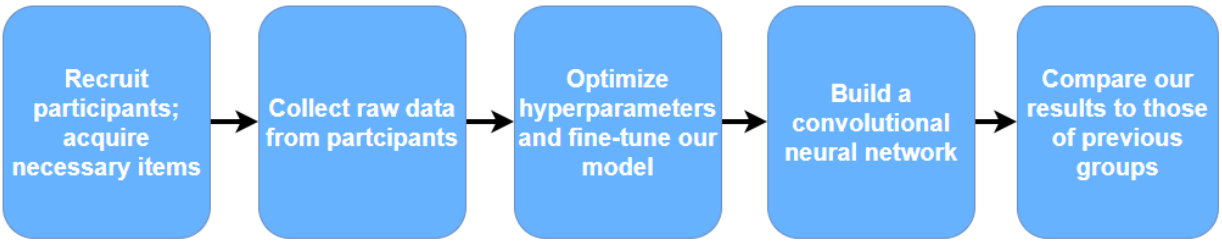

Figure 12: A flow diagram illustrating our methodology.

3.1 Study Logistics and Procedure

As deep learning requires large amounts of data, we aimed to recruit 100 participants in order to improve our application's classification accuracy. We gathered participants through WPI's Social Science Research Participation System (SONA), which consists of a pool of students studying psychology who receive credit for participating in various user studies. Additionally, we recruited other subjects via verbal and email solicitations.

3.1.1 Watch Selection

Our study required equipping subjects with a smartwatch that gathered data while they walked. When we were searching for smartwatches to use for this study, we had two considerations. Firstly, our study required a watch that has accelerometer and gyroscope sensors. Secondly, the watch needs to be able to run Android Wear, which enabled the smartphone's sensors to be programmed. We also considered price as well as how easily and quickly they could be obtained.

There were a surprisingly large number of candidates that appeared to fit these criteria. During research however, it was difficult to tell the specific functions of each watch outside of

comments written on e-commerce websites such as Amazon.com, by customers that had recently bought them. The model we chose is called the Asus ZenWatch. We chose this watch not only because it met all our specified requirements above, but also because we knew of a prior team that had successfully used this watch's hardware.

3.1.2 Sensor-Impairment Goggles

Administering alcohol to subjects or following them to drinking locations involved liability risks, which WPI's IRB board was not comfortable with. In order to gather data corresponding to different BACs, our team provided test subjects with "Drunk Busters" sensor-impairment goggles as an alternative to providing them with alcohol, due to liability concerns. These goggles use vision distortion to "simulate the effects of alcohol consumption on the body" and are shown in Figure 13. Each pair of goggles is intended to simulate visual effects generally associated with a particular BAC range. For our purposes, we used four different pairs of goggles. The pairs of goggles corresponded with the ranges [0.00-0.08), [0.08-0.15), [0.15-0.25), and [0.25-.35). According to the goggle manufacturer, wearers of the goggles are supposed to "experience the effects of reduced alertness, delayed reaction time, confusion, visual distortion, alteration of depth and distance perception, reduced peripheral vision, double vision, and lack of muscle coordination" (Drunk Busters).

While Drunk Busters goggles are commonly used by various groups, such as colleges, to educate individuals on the potential effects alcohol can have on their motor skills, they have not been used in any published research studies. Concerning the scientific accuracy of the goggles, Drunk Busters has stated that their testing "has been done in house, with nothing available to release to the public. It is instructive to note that each alcohol goggle only simulates an approximate BAC range and is not exact. Essentially, 20 people at the same BAC level would all behave a bit differently with varying test results of their physical behavior and such."

Figure 13: A set of Drunk Busters goggles.

3.1.3 Study Procedure

In order to perform a controlled and comprehensive study, we performed the exact same procedure on each participant that was involved, as shown in Figure 14. In order to control the experiment and ensure the safety of participants, we ensured that the test study always took place on a flat surface and in a wide open space so that the participants had plenty of space to maneuver through. Our procedure was as follows: Firstly, we ensured that the participants had read through and signed the letter of consent as required by the IRB. The participants were then handed a smartwatch to put on, as well as a smartphone to put in their pocket. Next, the participants were handed a random pair of goggles to put on. We gave participants the goggles in a random order as opposed to a fixed order so that the data would be unaffected by their perceptions of how impaired they were supposed to feel and their gradual acclimation to wearing vision distorting goggles. We then instructed the participants to initiate the recording on the smartphone, and to walk for one minute as the application collected accelerometer and gyroscope data. The participants were intentionally not instructed to walk at a particular speed or in any specific patterns, with the intent to simulate as much of a real-world scenario as possible and to maintain the complexity of data necessary to justify deep learning. However, they were told to walk normally and to try their best to avoid behaviors that would introduce noise to our data, such as holding onto walls or putting their hands in their pockets.

After participants completed the trial once with a random pair of goggles, they were given another random pair of goggles and instructed to do the trial again, a process which was repeated for each set of goggles. Participants were instructed to perform the trial without wearing goggles as well. After the participant had completed five total trials (four with goggles, one without) the test study was concluded.

Figure 14: A participant walking while wearing the drunk goggles.

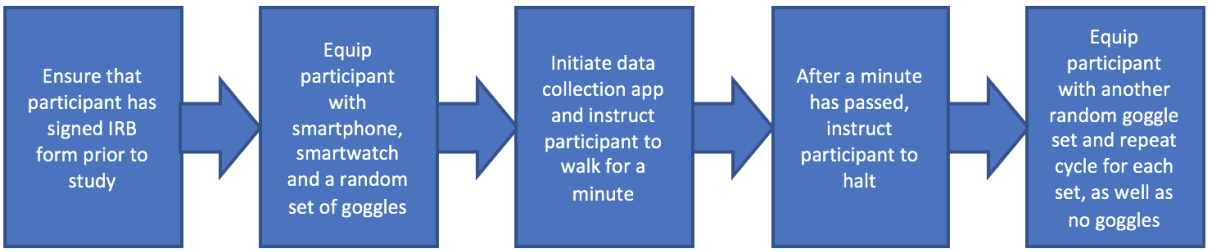

Figure 15: A flow diagram illustrating our test study procedure.

3.2 Deep Learning

3.2.1 Framework

The most important criteria for selecting a deep learning framework were 1) ability to process time-series data 2) ability to efficiently deploy the classifier to mobile devices 3) high level of abstraction, and 4) updated and thorough documentation and support.

Framework	Time-Series	Mobile	High Level	Support
MATLAB	No	No	Yes	No
Torch	Yes	Yes	Yes	No
TensorFlow	Yes	Yes	No	Yes
Caffe	No	Yes	Yes	No
Theano	Yes	Yes	No	Yes
PyTorch	Yes	No	Yes	Yes
Keras	Yes	Yes	Yes	Yes

Table 2: A comparison of deep learning frameworks.

(Keras, 2018; LISA Lab, 2017; MathWorks, 2018; PyTorch, 2017; Ronan, Clement, Koray, & Soumith, 2018; TensorFlow, 2018; University of California, 2018)

Table B shows a comparison of deep learning frameworks according to our criteria. After evaluating the common options, we determined that the Keras library, which is written in Python, was best suited for our purpose. One beneficial feature of Keras is that it is designed to be an interface, rather than a full deep learning framework, and thus rests on top of multiple

deep learning frameworks. Given the prevalence of TensorFlow in research and industry, we elected to use it as the backend for Keras.

3.2.2 Implementation Overview

Our implementation involved significant data preprocessing techniques, such as trimming, augmentation, and segmenting, which are detailed in Section 4.2. With this processed input, our deep neural network featured a series of convolutional and max pooling layers before a fully connected layer and a softmax layer. We expanded upon this architecture in Section 4.3 Neural Network Architecture. To achieve maximum accuracy without overfitting, we then used a form of Bayesian optimization called Tree-structured Parzen estimators to tune our hyperparameters. This process is further detailed in section 4.4 Tuning and Hyperparameter Optimization.

3.3 Experimental Hypotheses

As previously mentioned, many gait-related studies have utilized deep learning models and have found that these models perform better than regular machine learning models. Additionally, we suspected that data from the accelerometer sensor in a smartwatch may contain complexities beyond the scope of standard machine learning models. Thus, we hypothesized that the classification accuracy of our deep learning method combined with the introduction of the smartwatch sensors would surpass the AlcoGait study's accuracy of 89.45% in classification into five bins with a 99:1 data split.

4. Implementation

4.1 Hardware

The smartphones that we used to collect data were a Samsung Galaxy S6 and a Motorola Moto G4. They both ran Android 7.0 Nougat, which allowed us to use functions in the app up to API level 24. We determined empirically that their accelerometer and gyroscopes are comparable in quality, both able to sample data at above 100Hz.

The machine that we used to train our neural network is a compute node on the WPI ARC research cluster. It features 20 CPUs (Intel Xeon E5-2620 v4 @ 2.10GHz), two GPUs (NVIDIA Tesla K20m), and 128 GB of RAM. We specifically requested its compatibility with NVIDIA CUDA and CuDNN, which Tensorflow leverages to perform computations using GPU acceleration. Even with this capability, it took roughly four hours to train the model enough times to tune hyperparameters with a limit of 100 evaluations.

4.2 Data Manipulation

Before designing and training our neural net, we needed to consider ways to manipulate our raw data to ensure that we would be training the neural net with optimally formatted data. As such, we opted to trim and preprocess all of our raw data. Additionally, we researched data augmentation techniques to simulate a greater volume of our data without compromising its integrity. In the following sections, we will go into greater detail about the different methods we used to manipulate our data.

4.2.1 Trimming

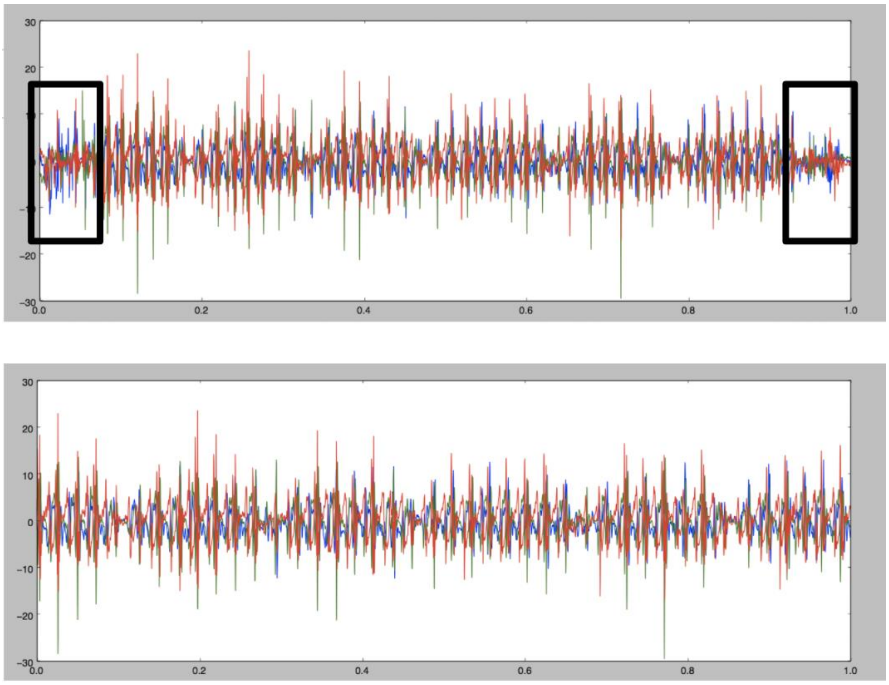

Figure 16: A sample graph of accelerometer data, showing the trimming of the noisy data contained in the two black boxes in the first image

Due to the nature of our test study, in which we generally instructed the app to start collecting data a few seconds before the participants began walking and did not terminate it until a few seconds after they were done, most of our data files had noise typically at the beginning and the end, as shown in Figure 16 above. Since we aimed only to use data collected from people while walking and not while they were manipulating the phone, we decided to trim these noisy sections from our data. We ultimately decided to perform this trimming manually, so as to ensure that as much valid data was extracted as possible to be fed to the neural net. To aid us in performing this task, we plotted the signals as shown and normalized the timestamps on the data (as the app outputs nanoseconds for phone data and milliseconds for watch data) to ensure that we would be able to capture and extract the precise corresponding data samples from the phone and the watch. Visually observing plots of all of our data additionally helped us get a better idea of trends in the data, and allowed us to spot flawed data samples that would have reduced our classification accuracy.

4.2.2 Data Augmentation

As shown in our related work section, in recent years convolutional neural networks have improved greatly and performed well on classification problems with large-scale datasets.

However, creating or obtaining such datasets in preferred quantities is not always feasible and without such significantly large datasets, it can be difficult to learn and produce reasonable results. Data augmentation is one technique used to help alleviate this issue by expanding the dataset artificially using transformations. In short, data augmentation is the process of using already collected data in order to simulate more data.

In our preliminary assessment, we evaluated three data augmentation methods: inverting, rotating by 90 degrees, and scaling the signal's amplitude by 5%. Other augmentations such as permutation or cropping were not used because that watch and phone had different sampling rates, making it difficult to preserve the validity of the data. The augmentations are performed on a random sample of the data and used to train the model. The model is then evaluated using test and validation sets of data previously unseen to the model, which contains only actual data. We experimented with different proportions of augmented and actual data as well as permutations of the different methods assessing their effects on our classifier.

As our dataset consisted of only 1,882 clean samples, we evaluated data augmentation as a way to mitigate our lack of data. We noted changes in our model's loss with respect to the amount of augmented data introduced to determine its effectiveness as well as the optimal ratio of augmented data and actual data.

Figure 17 shows examples of each augmentation. The normal signal is on the top left, flip is shown on the top right, scale appears on the bottom left, and rotate on the bottom right.

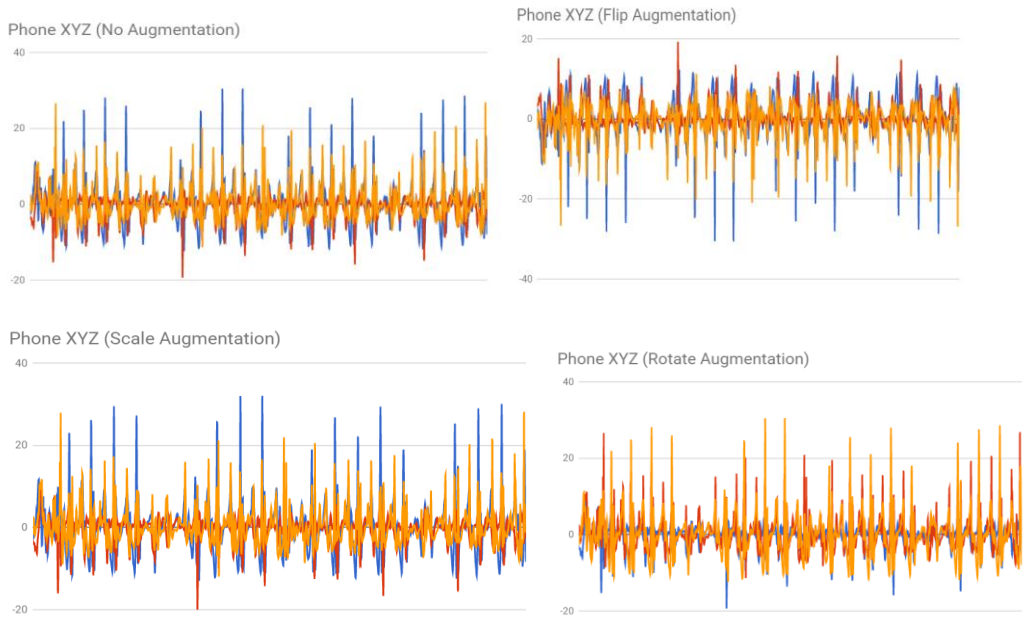

Figure 17: Data Augmentation – No Augmentation, Flip Augmentation, Scale Augmentation and Rotate Augmentation (From Top to Bottom, Left to Right)

We evaluated our data augmentations by repeatedly training our model with different amounts of augmentations introduced, plotting its loss function compared to each augmentation. These results can be found in Section 5.3 Data Augmentation Results.

4.2.3 Simulating a Fixed Sensor Sampling Rate

We also decided to write a script that would modify the data to simulate a perfectly fixed sampling rate. We decided to investigate this because while the data collection app aims to collect data at a rate of about 100 Hz for the phone and about 20 Hz for the watch, these rates are not perfectly achieved in practice. Notably, the phone data only rarely deviates from this rate, however, the watch data's poll rate is inconsistent enough to have an empirically higher standard deviation than its mean, a drawback that we will expand upon in Section 6. Apart from the increase in accuracy from having consistently collected data points, the input layer of the neural network also absolutely requires fixed dimensions for the data.

We elected to resolve the sampling rate so that the data points were equally spaced in order to avoid padding the input. For segments where the phone did not record enough data, instead of adding placeholder values that may reduce accuracy, we used the timestamps to fill in more accurate values. The Android sensor documentation states that sensors trigger interrupt routines whenever they detect a notable change. If no change is detected, then the sensor does not output new data. Though our hardware is very sensitive and rarely does not detect a change,

we still account for situations in which the sensor did not note a change. For situations where the phone sampled faster than usual, we took the average of the readings that occurred within the same 10 milliseconds Ultimately, we were able to write a script that took in phone data as input, systematically mapped the values over to a new dataframe with fixed time interval, and outputted a new CSV file containing the new dataframe.

Unfortunately, due to the incoherence and inconsistency of the watch data, we were not able to execute this process on the watch data. To fix its sampling rate, we simply used padding and truncation.

4.2.4 Preprocessing

To preprocess data, we first segmented the signals (for all three axes and both sensors) into windows of two seconds. Two seconds is about the time of a single walk cycle, though with variance among subjects, we felt that three seconds would encompass the step time of the vast majority of people. However, possibly due to the amount of data available, segmenting them further resulted in empirically better accuracy. To segment the data, we iterated through the time steps of one device's data and counted the time elapsed, filling a window with values until two billion nanoseconds on the phone, or two thousand milliseconds on the watch, had elapsed. For the phone, this was roughly 200 readings and for the watch, it was roughly 50. We then matched that window of elapsed time with that of the other device, and if both devices were sampling with no connection issues detected within that window, then both segments were appended to the dataset.

For various reasons such as devices freezing, losing connection, or missing entries, it was important for us to filter out segments with missing data points in this way, such that the relationship between the watch and phone signals are preserved.

4.3 Our Neural Network Architecture

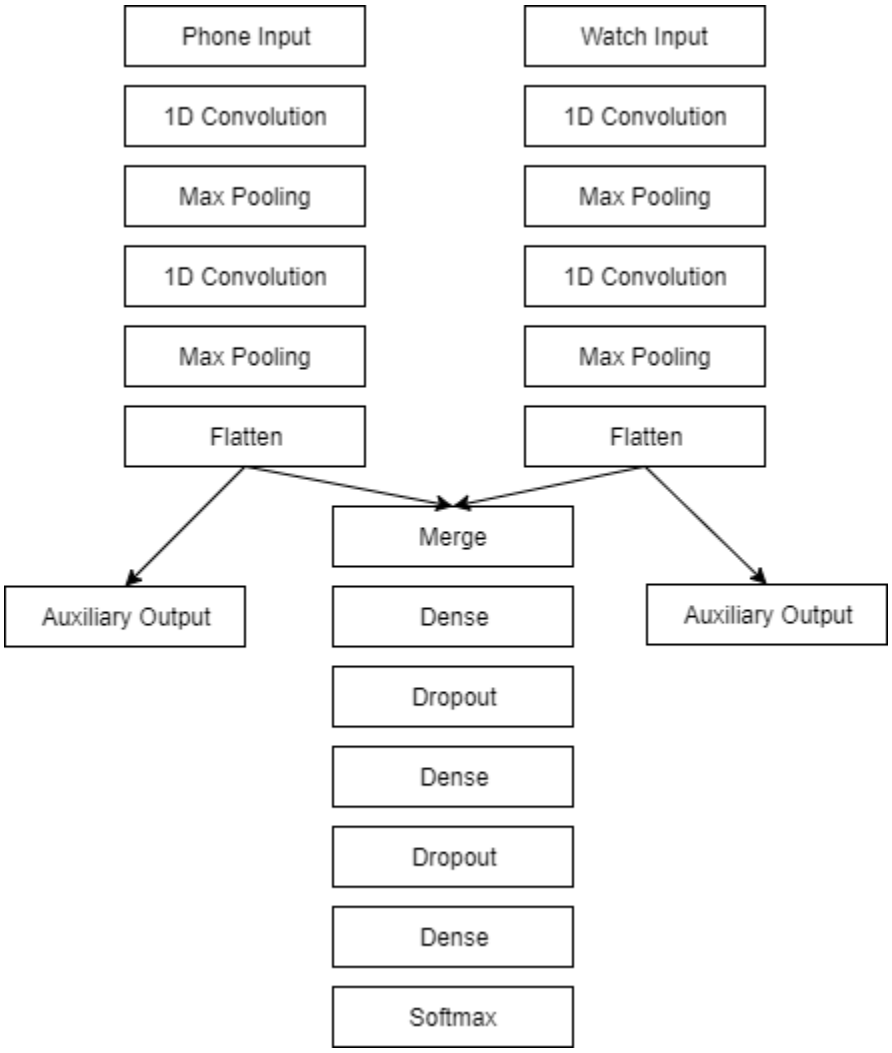

Figure 18: A diagram of our neural network architecture.

Our neural network architecture features two inputs that both feed into individual convolutional neural networks, which serve to extract features. After the standard layers of convolution and max pooling, the input is then flattened to resolve their output dimensions for the following softmax auxiliary output and merge layers. The purpose of the auxiliary output layer is so that the model can compute and minimize the loss calculated while each neural network is trained alone. This allowed the separate feature extractors for the phone and watch to reach maximum accuracy without confounding variables introduced when the layers are merged. The merge layer concatenates the now flattened inputs and outputs the extracted features to a feedforward multilayer perceptron classifier with dropout layers for regularization. Dropout layers help to reduce overfitting and enhances a model's ability to generalize by

randomly deactivating or dropping out nodes in layers, forcing the model to adapt to the data without relying on specific neurons which could memorize data (Goodfellow et al., 2016). The following dense layers act as a standard classifier using the features extracted from the convolutional layers as inputs. Finally, the final layer of the neural network is a softmax layer that outputs the computer probabilities for each class.

This architecture is the result of recommendations from our advisor, Professor Agu, for research involving multiple input models. The specific implementation was reasoned through Keras' documentation (Keras), which shows example implementations which lent insight on how to design our model based on our needs. A similar dual input model for a CNN can be found in Valkonen et al.'s 2017 study which used computer vision for cancerous tissue detection (Valkonen et al., 2017).

We use two separate inputs because the phone and watch are meant to extract features unique to their position on the body. As mentioned before, changes in arm swing and hip sway are believed to be indicators of intoxication, but are only measurable by the watch and phone respectively. For this reason, we implemented two separate feature extractors. In addition, it solved an issue caused by the different sampling frequencies of the phone and the watch. As the phone samples at about 100 Hz and the watch at about 20 Hz, they nearly always have a drastically different number of data points per window. As the neural network requires fixed dimension inputs and we wanted to avoid padding the data or cropping it significantly, we divided it into two inputs, which could logically utilize data inputs at different sampling rates.

As our dataset is slightly imbalanced, containing about 24% samples of participants with no goggles compared to about 19% of other classes, we set their class weights to be roughly inversely proportional to their frequency in the dataset. This increases the loss for the model when predicting common classes based on their frequency in order to balance our dataset without removing entries. The technique is described in detail in Chapter 7 of Goodfellow's book Deep Learning, titled Regularization (Goodfellow et al., 2016). Without this weighting, our model would sometimes decide to classify all new samples as the most frequently occurring class. This was especially true when using a larger number of hidden layers. Reducing the complexity of our architecture and implementing class weights solved this issue.

4.4 Tuning and Hyperparameter Optimization

4.4.1 Hyperopt

```
def get_space():
    return {
        'Conv1D': hp.choice('Conv1D', [30, 60,90,120]),
        'kernel_size': hp.choice('kernel_size', [5,15]),
        'strides': hp.choice('strides', [1,2,3]),
        'pool_size': hp.choice('pool_size', [10,20]),
        'strides_1': hp.choice('strides_1', [1,2,3]),
        'Conv1D_1': hp.choice('Conv1D_1', [30,60,90,120]),
        'kernel_size_1': hp.choice('kernel_size_1', [5,15]),
        'strides_2': hp.choice('strides_2', [1,2,3]),
        'pool_size_1': hp.choice('pool_size_1', [5, 10]),
        'strides_3': hp.choice('strides_3', [1,2,3]),
        'Dense': hp.choice('Dense', [200,500,1000]),
        'activation': hp.choice('activation', ['relu','sigmoid']),
        'Dense_1': hp.choice('Dense_1', [200,500,1000]),
        'activation_1': hp.choice('activation_1', ['relu','sigmoid']),
        'Dense_2': hp.choice('Dense_2', [200,500,10000]),
        'activation_2': hp.choice('activation_2', ['relu','sigmoid']),
        'batch_size': hp.choice('batch_size', [8,32]),
    }
```

Figure 19: Hyperparameter search space.

We tuned the hyperparameters of our neural network using Hyperopt 0.1, which is a library that provides the Tree of Parzen Estimators (TPE) hyperparameter search algorithm that can outperform competitors such as grid search and random search. The TPE algorithm is a complex process that is described in Bergstra et al.'s 2011 paper titled *Algorithms for Hyper-Parameter Optimization (Bergstra, Bardenet, Bengio, & Kégl, Nov 20, 2011)*. Given the capabilities of our hardware and the ample time in between development sessions, we explored our search space comprehensively. We evaluated various filter sizes, pooling sizes, kernel sizes, batch sizes, number of epochs, dropout ratios, strides, activation functions, thickness of hidden layers, and number of hidden layers. We chose to limit our parameter search to 200 permutations, which took approximately eight hours.

4.4.2. Manual Hyperparameter Tuning

The Hyperopt library was very helpful in gaining an easy preliminary understanding of a viable specific architecture for our model. However, it was not feasible to fully explore the hyperparameter space in this way due to constraints on the output dimensions of the convolutional layers, which depend on other hyperparameters. Hyperparameters such as kernel size and pooling size change the output dimensions of their layers and cause incompatibility with certain configurations. Due to this and also the desire to produce concrete evaluations and

visualizations of our methods, which are not afforded by Hyperopt, we elected to perform some hyperparameter searching of our own. To this end, we wrote a script that would vary the hyperparameters and store the results for further analysis, which can be found in Section 5.4.

4.5 Data Gathering App

In order to record accelerometer and gyroscope data on the smartphone and smartwatch during our test studies, we used an Android application that was developed by a previous MQP. The app features only one screen. On this screen, there is a start/stop button and five radio buttons, each of which corresponds to which pair of drunk goggles our test study participants were wearing. Once the start button is pressed, the app will begin to log gyroscope and accelerometer data along the *x*, *y*, and *z*-axes. Once the stop button is pressed, the app will save both the accelerometer and gyroscope data to separate CSV files for both the phone and watch. All CSV files have a label that identifies which set of drunk goggles the participant was wearing or none if the participant was not wearing any of the goggles during that trial. Figure 20 shows the app being ran on an Android phone.

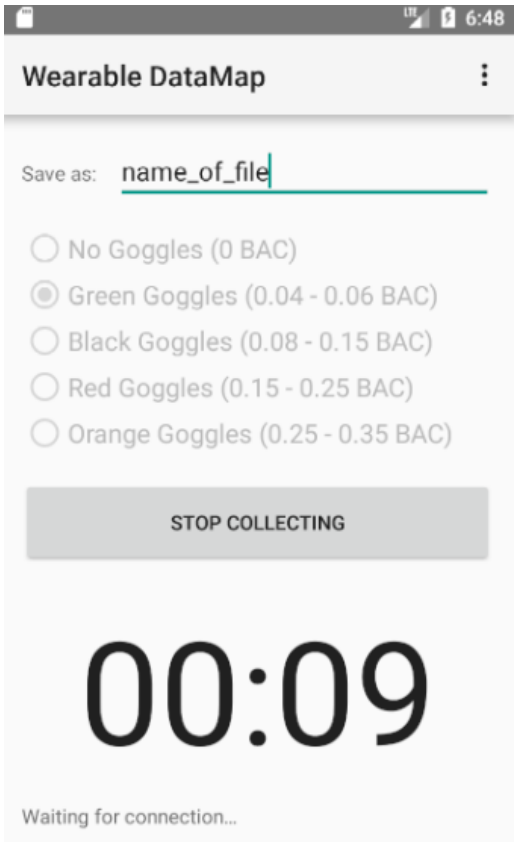

Figure 20: A screenshot of the Android data collection app while data is being collected.

When we first retrieved the data gathering app, we tested it by graphing its sensor measurements in the outputted CSV files for the smartphone and smartwatch. While the app was running, we held both devices in hand and kept them stationary for a few seconds. We then moved both hands in the same motions. We then took the measurements and created four line graphs, one for the accelerometer and gyroscope on both devices. After graphing, we found that the line graphs for the smartwatch sensors seemed a bit unstable and noisy, even during the window in which the watch was stationary. To address this, we changed the API used for communication between the two devices. While the previous MQP used the Data Layer API, we decided to use the Message API. After changing the API, we used the same procedure as before to test the app. As seen in Figure 21, we found that the line graphs for the smartwatch appeared more stable after swapping over to the Message API. Using the Data Layer API, the previous MQP group buffered data on the watch, which resulted in the timestamps for each data recording to be slightly off because timestamps were recorded when data is read from the buffer as opposed to when it is put into it. However, using the Message API we avoid this buffering of the data entirely.

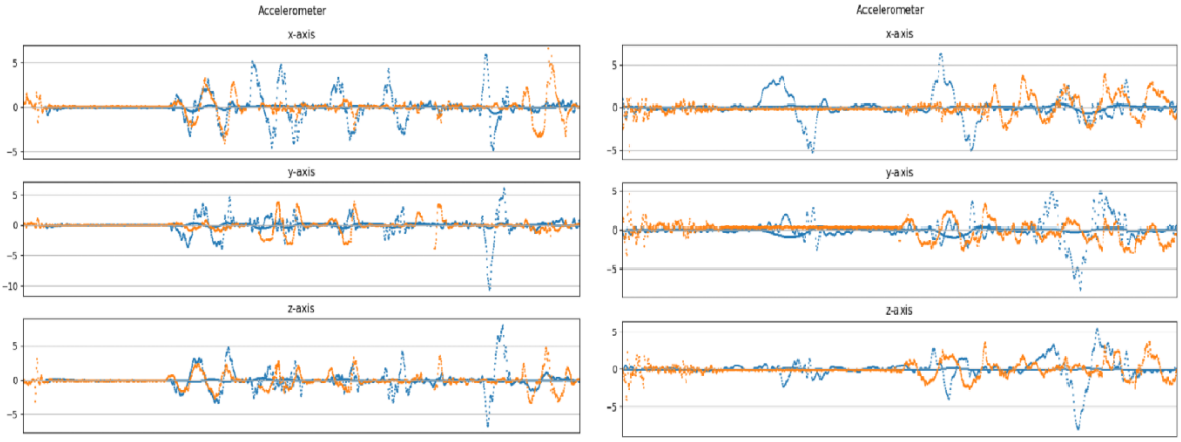

Figure 21: A graph of the accelerometer measurements along the x, y, and z axes as while the smartwatch and smartphone is held stationary in each hand and then moved in the same motion. The orange lines are measurements from the smartwatch and the blue lines are from the smartphone.

5. Results

In this section, we discuss the results we have achieved using our gathered data using neural net architecture.

5.1 Data Gathering Results

Over 12 weeks, we have gathered data from 45 participants. However, various errors in data collection invalidated seven of these trials, and as a result, we currently have viable data from 38 participants. From these participants, approximately two hours of viable readings were recorded.

5.2 Neural Network Results

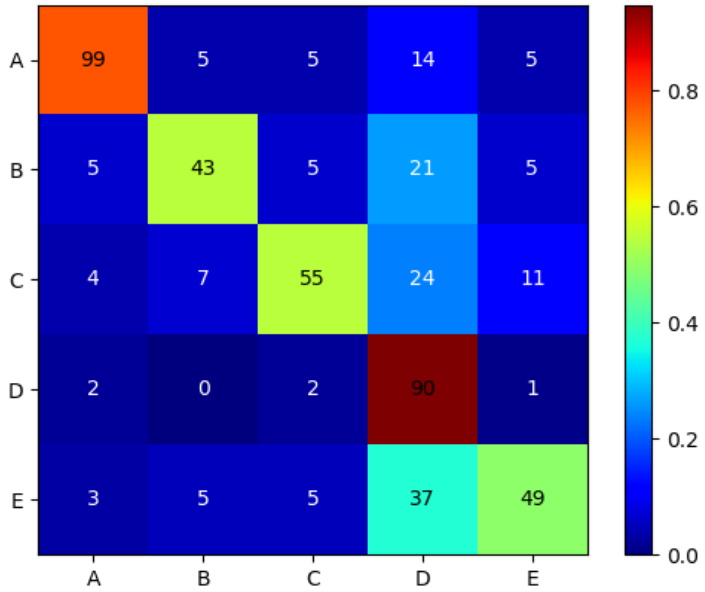

Figure 22: The confusion matrix for main output.

Our neural network's accuracy is roughly 64% on the training set and 69% on the test set into 5 bins. We use a test-train-validation split of ratios 60:20:20. The watch's convolutional neural network alone predicts correctly 26% of test samples and the phone's network alone predicts 51% (without any dense classifier layers). Note that due to the random initialization of the neural network's edge weights, the classifier will have slightly varying accuracy and loss rates each time it is trained. The ranges for accuracy observed for our classifier can be seen from about 61% to 67% without changes to parameters. This is important for all further evaluations of

our neural network with respect to different variables. With some evaluations, we provide multiple trials with the same parameters to mitigate random chance as a factor.

Initially, our neural network was overfitted to our data, as shown in the loss curve in Figure 23a below. After introducing more aggressive dropout rates (from 20% to 60%) for regularization, the loss curve in Figure 23b shows more consistent loss between the test and validation sets, showing that our model is further able to generalize on inputs that it has never seen.

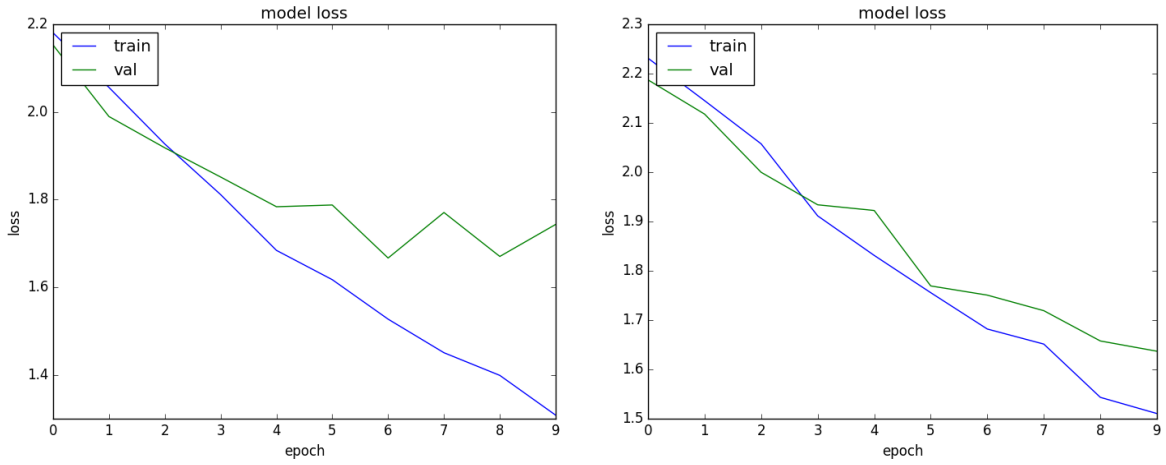

Figure 23a (left) and 23b (right): Loss curves before and after aggressive dropout.

In addition to these results for our combined watch and phone classifier, we also present an accuracy for solely the phone's data of 72% on the training set and 62% on the test set into five bins. With only data from the watch, the accuracy is 34% for training and 28% for testing. From this, we gather that the watch is not very helpful in collecting useful features for classification, as the phone alone performs nearly just as well. We also observed an 85.1% accuracy among two bins for every two second segment of data using both the phone and watch. Using the phone data alone on the convolutional and dense classification layers yielded an accuracy of 80% over two bins, and using the watch alone yielded an accuracy of 57.1%. The phone's improvement is natural, but the drastic improvement of the watch into two bins means that the device is capable of collecting useful data that can determine intoxication level, but the intricacies between bins are lost. We discuss the shortcomings of the watch's data in Section 6.2.2 Hardware Drawbacks.

5.3 Data Augmentation Results

The data augmentation provided us with simulated data generated from that which was gathered from subjects. After testing each augmentation method individually, we found that the results varied. An overview of our results can be found in Figure 24 below.

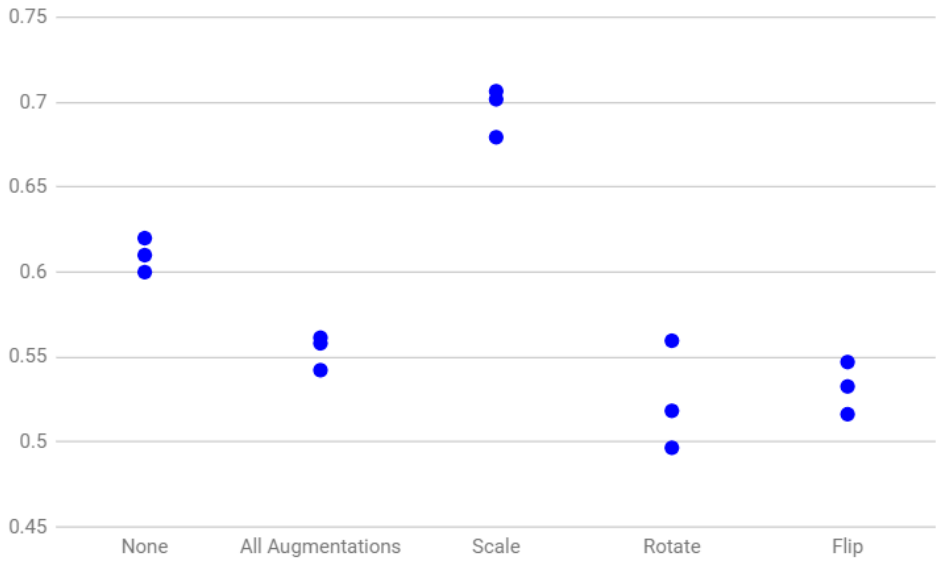

Figure 24: Accuracy of different augmentations compared to base accuracy.

Out of the three augmentations, the scale augmentation performed the best, consistently raising the accuracy of the model by whenever a large amount of scaled data was introduced. However, both rotate and flip seemed to reduce the accuracy by a significant margin, with a drop of about 5% each. This is reasonable as signals from sensors on Android phones are consistent and never appear similar to the signals produced by those augmentations.

As scale was the only augmentation to show promise, we conducted a deeper investigation, training our model 162 times to better observe the trend. Figure 25 shows the loss as we add trials containing scaled data.

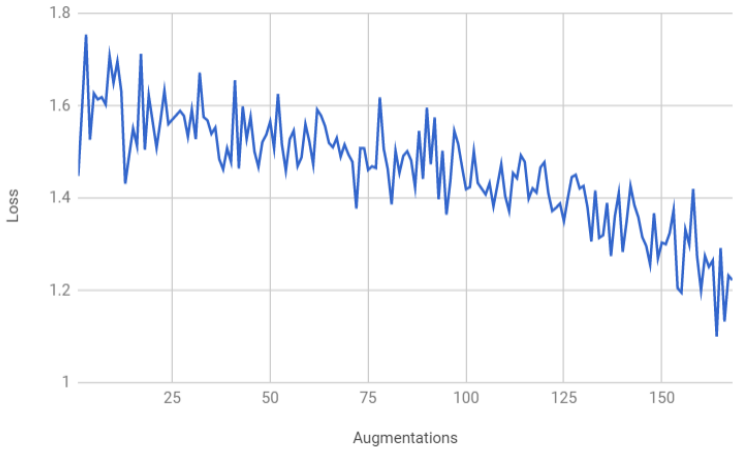

Figure 25: Loss vs number of trials of scaled data introduced

The trend shows that adding scaled data can decrease the loss by a huge factor unseen by previous attempts to improve the accuracy of our neural network. The accuracy corresponding to a loss value of less than 1.2 is consistently greater than 70%, sometimes even reaching 78%. We also observe that only adding a small number of scale augmentations causes the loss to be unstable and generally increase, only yielding positive results when the ratio of augmented to real data is greater than 1:2.

Due to this behavior, we suspect that the data scaled by 5% is not sufficiently dissimilar to the original data and our result may be due to the neural network having been trained on nearly the exact same inputs as it sees in the test and validation sets. This is because we do not have enough data to exclude the augmented data's unscaled counterpart when adding a ratio of augmentations greater than 1:2, causing the model to be trained on scaled trials nearly identical to those it would see in the validation and test sets. Nevertheless, the issue of using augmented data too similar to the original data causing overfitting is difficult to evaluate, requiring a sufficiently large set of validation data without an augmented counterpart. Due to our conjecture that the augmentations do not improve our model's ability to generalize new input, we choose to forego all data augmentations.

5.4 Hyperparameter Tuning Results

The hyperparameter search results (optimal values) are as follows:

Hyperparameter	Value
Batch size	16
Epochs	10

Activation function	ReLU
Number of nodes in dense layers	256
Phone convolutional layer 1 number of filters	15
Phone convolutional layer 2 number of filters	135
Phone convolutional layer 1 kernel size	15
Phone convolutional layer 2 kernel size	15
Phone max pooling layer pool size	20
Phone max pooling layer stride length	2
Phone convolutional layer 1 stride length	3
Phone convolutional layer 2 stride length	2
Phone dropout probability	.3
Watch dropout probability	.3
Dense layer 1 dropout probability	.65
Dense layer 2 dropout probability	.5
Watch convolutional layer 1 number of filters	30
Watch convolutional layer 2 number of filters	60
Watch convolutional layer 1 kernel size	10
Watch convolutional layer 2 kernel size	5
Watch max pooling layer pool size	5
Watch max pooling layer stride length	2
Watch convolutional layer 1 stride length	3
Watch convolutional layer 2 stride length	2

Table 3: Chosen hyperparameters.

Our chosen hyperparameters can be seen in Table 3. The following graphs show our manual hyperparameter search based on our initial result returned by Hyperopt. The first

hyperparameter shown is kernel size for our phone's first convolutional layer, which we chose to be 18 as it shows the least loss and the trend shows that higher kernel sizes work better.

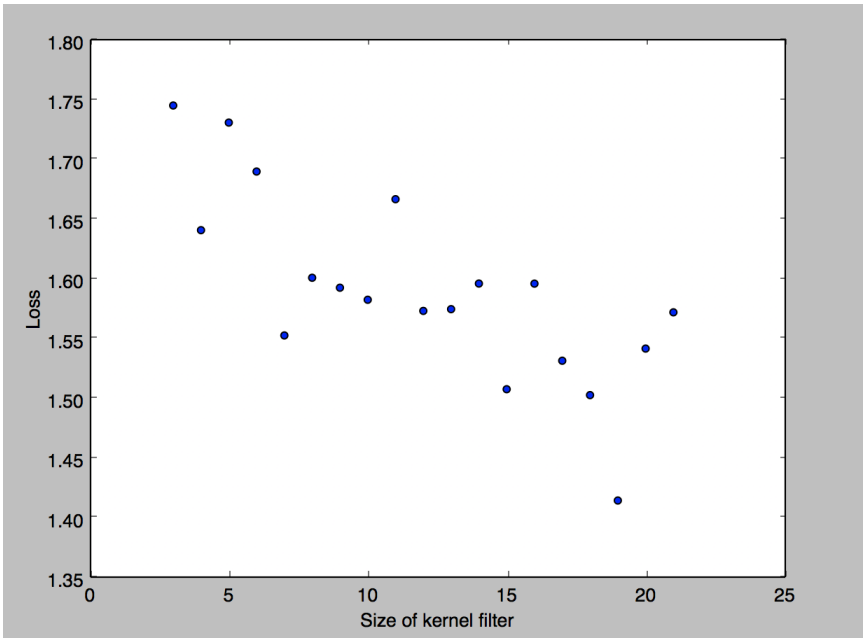

Figure 26: Kernel filter size versus loss.

The next hyperparameter is pool size for the phone's first max pooling layer. It shows another clear trend where the loss decreases as pool size increases, until around 20, when the loss goes up again. Therefore, we use 20 as our pool size.

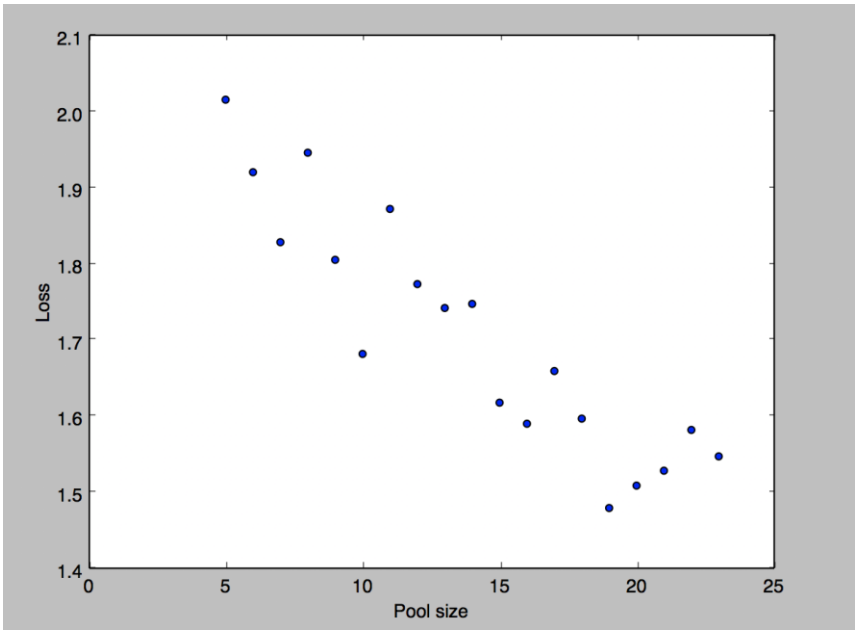

Figure 27: Pool size versus loss.

The number of hidden layers is was also heavily correlated with loss, as shown in Figure 28. We find that given the trend, 256 is a good value for our architecture.

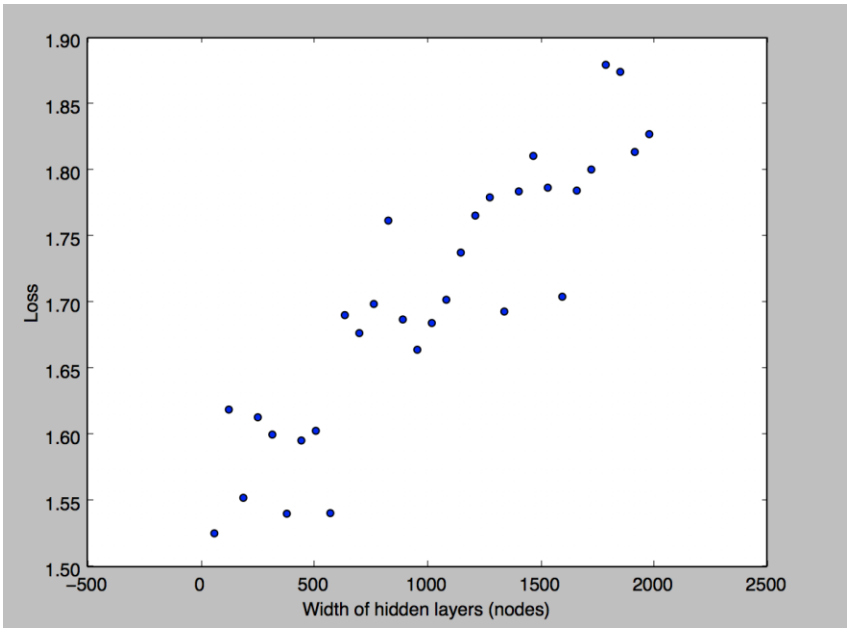

Figure 28: Width of hidden layers versus loss.

Next was batch size, which we chose to be 16 because the trend shows the least loss around that value and due to GPUs performing more efficiently on batches whose size are a power of two (Ponnuru, Pookalangara, & Nidamarty, 2017).

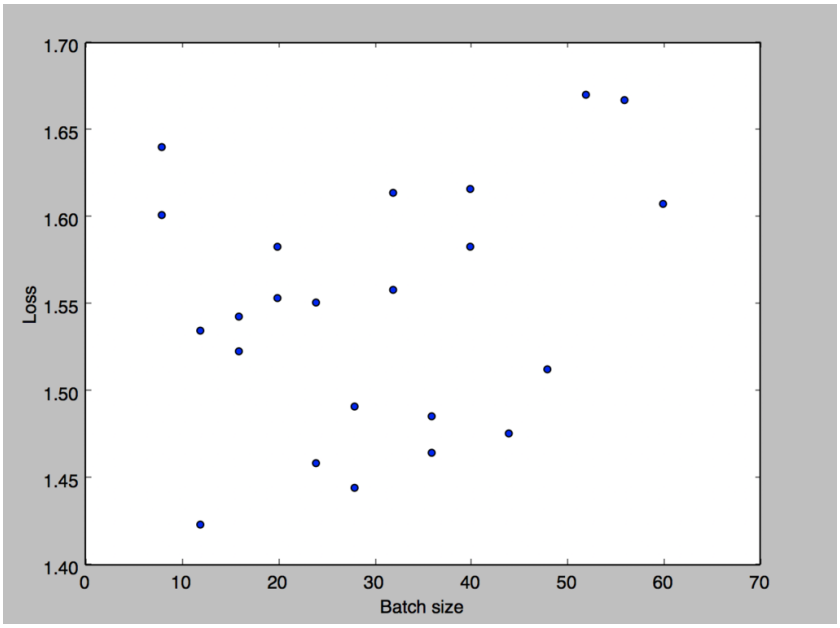

Figure 29: Batch size versus loss.

Figure 30 shows the number of output filters in the phone's first convolutional layer. The trend is not as clear but the loss appears to generally increase with the number of filters. Therefore, we chose 15 as the value for this hyperparameter.

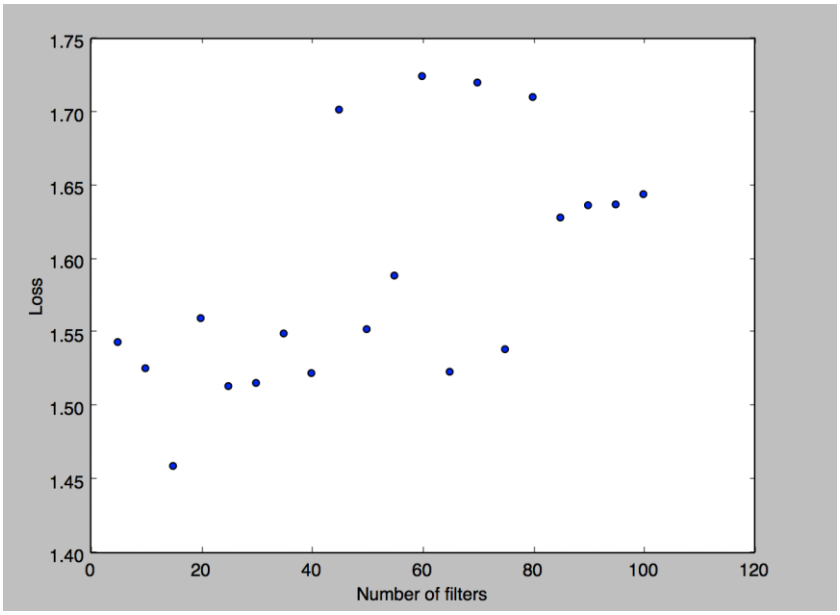

Figure 30: Number of filters versus loss.

The final plot, Figure 31, shows an artificial hyperparameter which was the size of our window. This determined how much data was contained in one input for the model. Higher values for windows gives more data when classifying and also reduces the likelihood that a person who walks slowly would have their cycle truncated. However, smaller windows increases the amount of samples that the model is trained upon. We initially planned for our window duration to correspond with one walk cycle and we find that the value of two seconds is a viable for conceptual and quantifiable reasons.

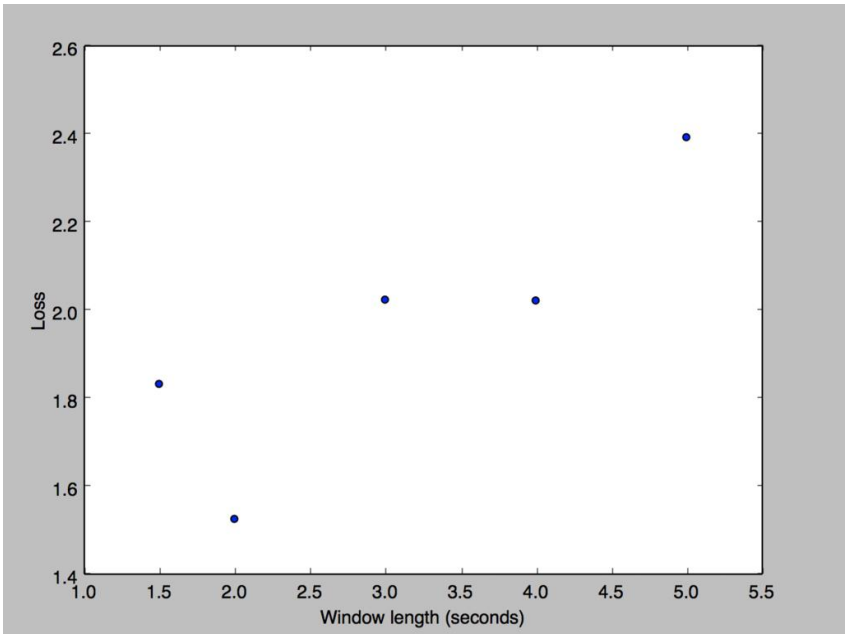

Figure 31: Window length versus loss

Changing the hyperparameters to other values can result in accuracy falling to as low as 25% due to their role in defining the architecture of our model. The configuration shown above is the result of eight hours of using hyperparameter searching and an additional twelve hours of constant training of our model to produce the plots and fine tune the results of the initial search. We find that these chosen values yield the minimum loss for our model.

5.5 Final IntoxiGait App

One of the primary goals of this project was to develop an Android application that could accurately predict the intoxication level of a user. The application was created using Android Studio, as this is the official IDE used for Android application development. When the user opens the app on their smartphone, they are presented with the home screen. This screen contains a status text that corresponds to the current state of the app. The statuses are as follows: “Connected to Watch”, “Disconnected from Watch”, “Recording”, “Done Recording Data”, “Sending Data to Server” , and “Receiving Data from Server”.

On the home screen, the user can press the start button to begin data collection if their phone is connected to their smartwatch. Once the user clicks the start button, a countdown of 3 seconds begins to allow the user time to put their phone in their pocket and start walking. Then, both the smartphone and smartwatch will update their screen to display a “recording” status and the app will record accelerometer and gyroscope data for both devices. After 12.2 seconds

(about six classification windows of 2 seconds with some tolerance), the app stops collecting sensor data and signals to the user via watch vibration that it has completed. The app sends the sensor readings via POST request to the server, which classifies them and returns a response. The user will be brought to the results screen, which displays their inferred intoxication level. Once the user hits the "Done" button on this page, they will be brought back to the home screen. Figure 32 displays a flow chart of what the smartphone and smartwatch looks like through the cycle of this application.

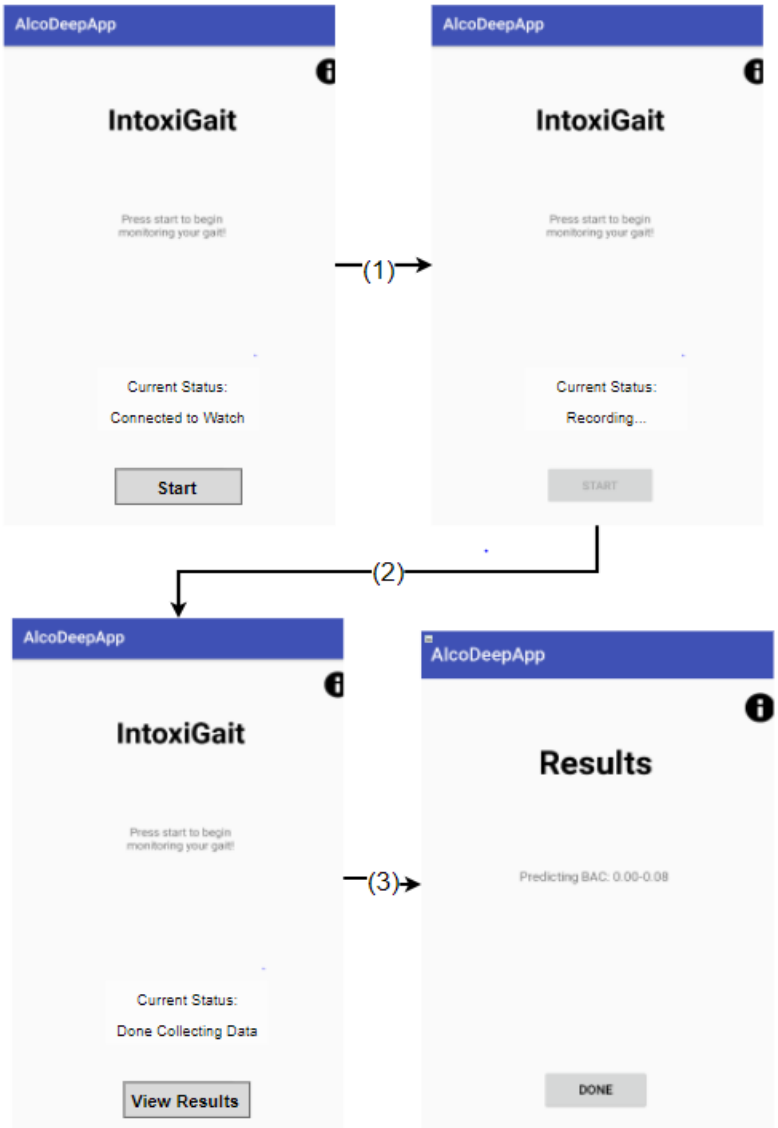

Figure 32: A flow chart of one cycle of our Drunk Detection Android App. The arrows show: (1) user presses the start button and App begins recording data, (2) App finishes collecting 12.2 seconds of data, and (3) User hits the view results button and is brought to the results page.

5.6 Server-Side Classification

To avoid expensive client-side data pre-processing and classification, we elected to host a server (alcogait.wpi.edu) that can perform the computations for the app. This improved device compatibility and left our app lightweight, scalable, and unencumbered by deep learning libraries. Though the server that we used had neither the computational power nor the GPU support of the machine we used to train the network (specified in 4.1), classification was still rather fast. However, iterating through the data to resolve the sampling rate and input dimensions was very costly, contributing to most of the five second response time. The computation time for each operation was measured using the system's clock and appears in Table 4 below. Note that the time required to upload the data from the phone varies with network conditions and is not shown.

Operation (in sequence)	**Time taken (ms)**
Loading model from disk	246 ms
Read data from disk	8 ms
Phone poll rate resolution	4170 ms
Data segmentation and integrity check	25 ms
Classification	27 ms

Table 4: Server computation time breakdown

The server code is written in Node.js with an Express router and takes POST requests with multipart/form-data encoded two file uploads, which are the phone and watch CSV files. Figure 33 depicts our client-server flow. Upon receiving these files, the server stores them and spawns a Python subprocess which launches our preprocessing scripts, imports our model which is stored on disk, and then performs the classification. The preprocessing scripts, as mentioned before, break the data into windows of two seconds, and so the result of the classification is 6 sets of probabilities, one for each two second window collected. The probabilities are summed by bin and the most likely class is returned as a JSON response to the client. After returning the response, the files are deleted to regain storage space for future files.

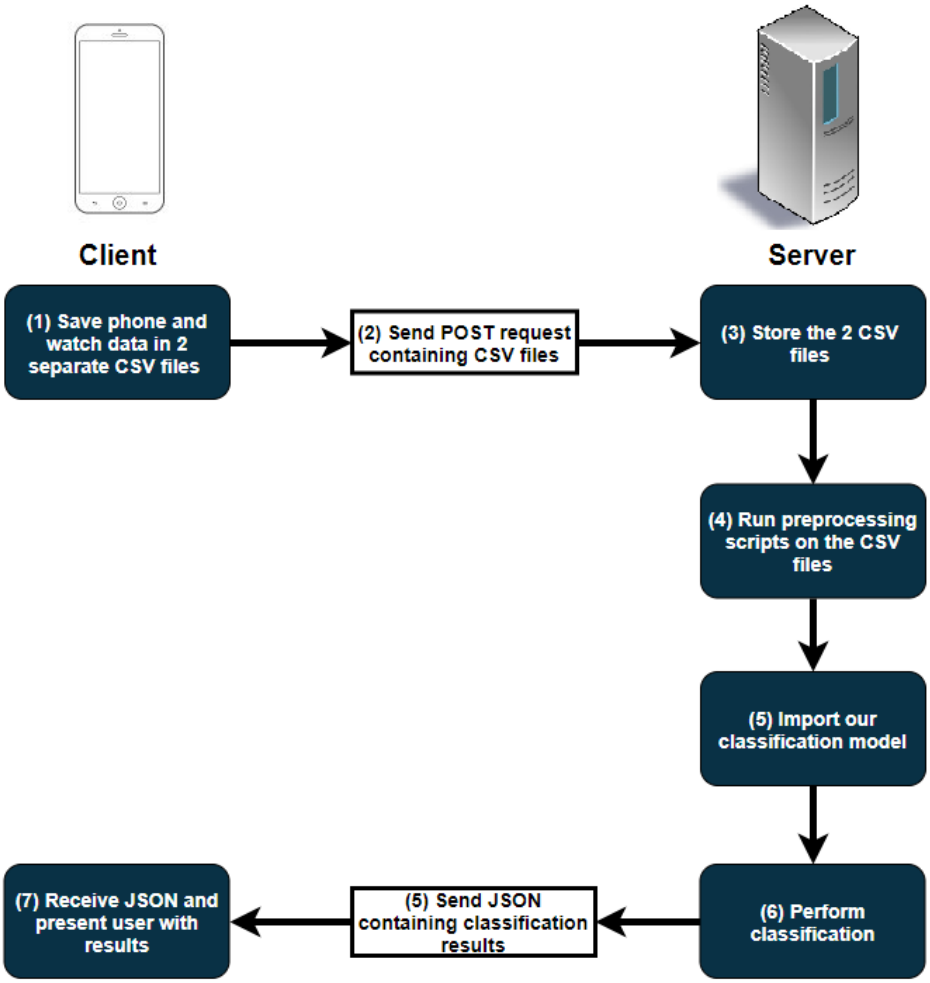

Figure 33: A diagram depicting the client-server flow of the IntoxiGait app.

By having the server able to classify data of arbitrary length, we increased the accuracy of the prediction. The neural network splits the data into two-second windows and accuracy is measured for each of the windows. The neural network outputs probabilities for each class. So, given multiple two-second windows, each individual probability can be added together to produce a more accurate response. In other words, the server has multiple opportunities to classify the data. For example, even if the neural network incorrectly classifies one two-second window of a walk into the wrong category, it is likely that other windows would be correctly classified. As a result, the probability of the highest sum of the probabilities being correct should increase given more and more windows. Each trial is not independent but having many inputs representing two seconds of data will increase the overall accuracy.

6.0 Discussion

In this section, we summarize our main findings and challenges, and discuss the significance of our findings while executing our project.

6.1 Successes and Findings

6.1.1 Investigated Neural Network

As previously stated, we were able to achieve a roughly 64% accuracy across five bins for every two second segment, with one corresponding to each set of goggles, along with one specifying no goggles. This is an improvement over the results found by the AlcoWatch team, who found a 40.69% success rate in the same five bins for every input. We find this accuracy to be reasonable and useful for our application as people often walk for longer than two seconds. We discuss our recommended interval for an accurate reading in Section 6.1.4. Based on our research, we are confident that our methodology and use of deep learning, as opposed to basic machine learning methods, is responsible for this increased success. Into two bins, the AlcoWatch team achieved an accuracy of 75% compared to our 85% in this category. However, they used only data collected from a smartwatch. Using only smartwatch data, our model is only able to correctly predict 57% of the time. Thus, we believe that the noise associated with the smartwatch sensors have a very tangible effect on the ability for our convolutional neural network to extract useful features. We recommend introducing more data to address this problem.

6.1.2 Collected Data

Throughout the entirety of this project, we consistently gathered at least a few participants for our study each week. Ultimately, we successfully gathered data from 38 participants. A large portion of our data came from soliciting our friends and WPI students on campus. Often, WPI students we approached were willing to help us with our study. Another good source of data came from WPI's SONA system. Using SONA, we collected data from 20 amount of people. Ultimately, we determined that SONA was an extremely useful method for recruiting participants, as we received a steady flow of subjects after we began posting our study on the tool.

6.1.3 Explored Data Manipulation

We also found success in manipulating our data in different ways prior to classification. Firstly, trimming the data was a crucial first step, as each and every data file we worked with had some noise at the beginning and the end, due to pauses before and after the participants were walking. Additionally, due to Bluetooth connectivity issues (which we expand upon in 6.2), there were empty holes in much of the data, which it was necessary for us to remove manually. The benefits of trimming our raw data can be visualized in Figure 34.

Additionally, we noted an increase in accuracy rate after running our script on the data to simulate a fixed sampling rate. While the neural network's accuracy only increased by roughly 2%, we believe that performing this manipulation was a necessary step nonetheless, as running the script on the previously trimmed data yielded a significant average decrease in loss, as shown below in Figure 34.

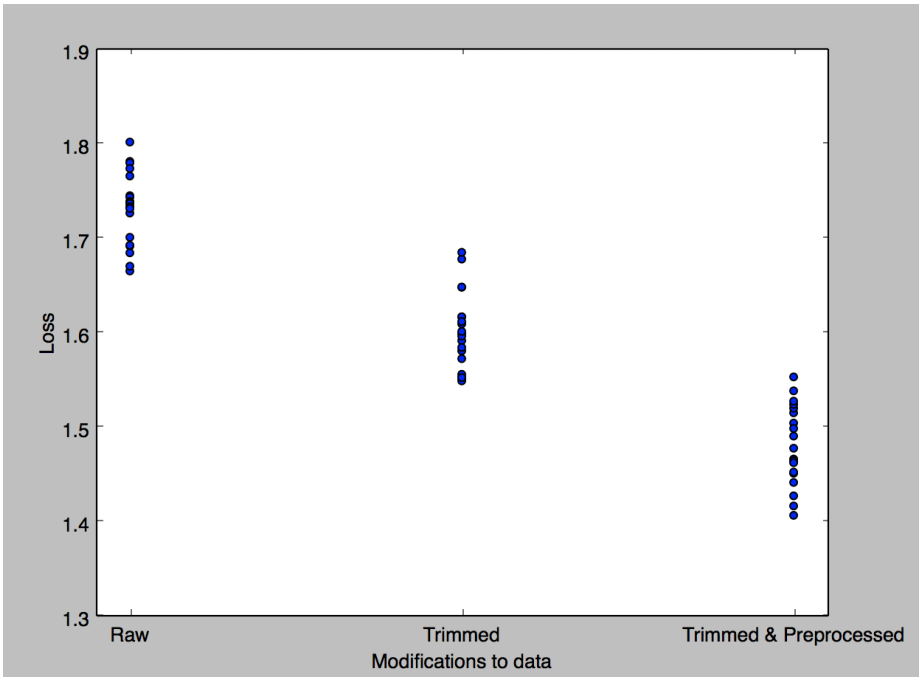

Figure 34: A comparison of loss values observed on raw data, trimmed data, and trimmed/pre-processed data, when the neural net is trained 20 times on each category. Here, 'pre-processed' refers to the simulation of a fixed poll rate on the phone data, rather than segmentation into windows. One can note the increased success, corresponding with a decrease in loss, as the data is trimmed and then pre-processed.

We also used data augmentation to replicate more data while trying to avoid any overfitting. The different augmentations yielded different individual accuracies, with the scaling augmentation performing the best in terms of accuracy, with a 74% accuracy with a 1:1 ratio of

augmentations to gathered data. However, using such a high concentration of augmented data can create over fitting. After graphing loss over total augmentations, we found a downward trend in loss, as shown in figure 35. As the data is inverse to accuracy, the possibility there is some level of overfitting is high.

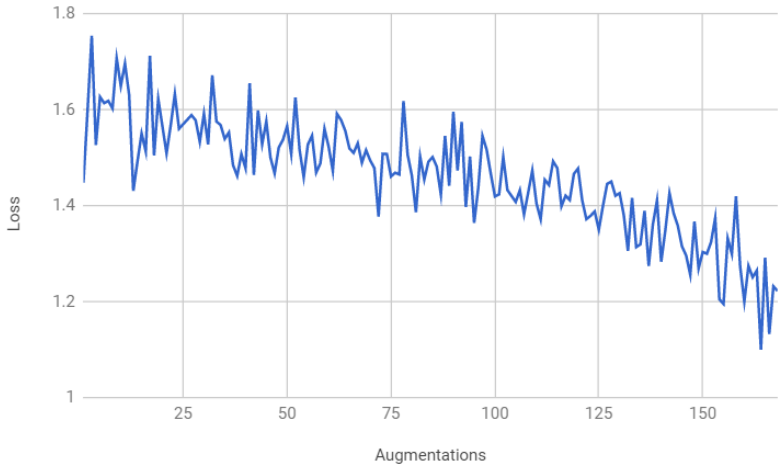

Figure 35: Graph of Loss with Scale Data

6.1.4 Prototyped Mobile Application

We successfully created an application that worked as we initially envisioned. Setting out on this project, we wanted to create an end user application that can be used to predict levels of intoxication. Users can run our app to classify a 12-second segment of their walking. The interface is minimal and fairly intuitive to use; if there are any confusions, there is a button users can press to have an info box pop up and inform the user on how to use our app.

6.2 Limitations and Drawbacks

6.2.1 Data Collection Drawbacks

One of the key limitations on any neural net is the total amount of data that can be used. For this project, the majority of the data used to train the neural net was gathered by this group within the span of a few months. As such, there were large limitations on data gathering. An element of human error also impeded our success in collecting data: on several occasions, we thought we had posted our study on the WPI SONA system but actually hadn't due to issues we had with the interface. On other occasions, we simply forgot to post our study for the week. We also believe we missed out on potential participants by posting our study later on in B-term than

would have been ideal, as students tend to sign up to participate in studies at the beginning and the end of the term more so than in the middle of the term.

Additionally, it was common for participants to behave strangely during data collection in ways that could introduce noise to the data. Being acutely aware of the watch and seeming to try to keep it as steady as possible, perhaps in an effort to not damage it against walls, talking and making jokes with hand gestures, and walking with arms outstretched to avoid unlikely collisions were all behaviors that we noted might introduce error. However, it is possible that real intoxicated people do these things and so we defer this research to a future study with access to alcohol and intoxicated participants.

Another drawback was that we were not able to confirm the accuracy of the goggles. When we asked participants about the distortion to their vision after using each of the 4 goggles, a majority of the respondents indicated that the "Green Goggles" were the most disorientating, despite the green goggles representing the lowest intoxication levels. Additionally, the effects of the goggles produced a warped view with the black, red, and orange goggles causing the bottom part of a wearer's view to be magnified while the green goggles cause the top to appear stretched instead.

6.2.2 Hardware Drawbacks

Throughout our project, the main pieces of hardware we used were the Asus ZenWatch and the Samsung Galaxy S6. Unfortunately, we did encounter several issues with the hardware that impeded our ability to collect data. For one, we regularly encountered Bluetooth connectivity issues between the phone and the watch while collecting data. Occasionally, during a test study, the phone and the watch would briefly disconnect. The data collection app that we used was equipped to handle this type of situation, and would reconnect as soon as possible, but we would be left with holes in the data. This occurred in roughly a third of our collected data.

Another issue that we encountered throughout our project was characterized by the nature of the watch data. While the outputted phone data was consistent at roughly a 100 Hz sampling rate and easy to work with, our raw watch data was much more inconsistent. The watch data's polling interval had a mean of 52.7 ms but a standard deviation of 122 ms. This is partly caused by its ability to poll at rates close to the phones, but somehow not all the time. The other problem with the watch data is illustrated in Figure 35, in which many points of data are clearly associated with the same timestamp, thus preventing us from being able to discern the order in which much of the data was collected. While the figure only shows a small excerpt of watch data, errors of this type were ubiquitous throughout the data. While we wrote a script to

preprocess the raw phone data to simulate a completely fixed poll rate, we were unable to work the script on the watch data as a result of this issue. Had we been able to use more coherent and consistent watch data, we suspect we would have been able to produce a higher accuracy rate than we were actually able to.

Additionally, many timestamps in the watch data were outputted out of order, another common error that was not present in the phone data. We are unsure as to whether or not these issues resulted from hardware or software issues, however, we were not able to identify any standout issues in the data collection app code that would justify the discrepancies in the quality of the watch data versus the phone data. As a result, we tentatively attribute our watch data issues to sensor and other hardware (processing power/connectivity) related limitations, as well as software limitations. We note that the watch's input response time, even for simple actions such as opening the menu, is sometimes more than three seconds despite not running any discernible background processes and never having been used for any other purpose than conducting this study.

```
7602205787943,0.25946522,1.139811,-1.2123481,-0.15978967,0.2727077,-0.14913702,no_goggles
7602215675818,0.26711845,4.0283027,-2.2773042,-0.4346279,0.15126756,-0.1736381,no_goggles
7602225653152,0.13037872,8.696793,-2.0945554,-1.1121361,-0.022370555,-0.2545982,no_goggles
7602235623110,0.1518445,6.6869054,1.2965115,-1.7864485,-0.06498113,-0.38242993,no_goggles
7602246349568,0.7471781,3.1480174,1.4361392,-1.8311896,0.3174488,-0.5134575,no_goggles
7602255552193,1.7675574,2.3461647,0.4211468,-1.4775218,0.8404937,-0.75101143,no_goggles
7602265786402,3.4673378,2.7622628,-0.10926062,-1.0791129,1.0492855,-1.2506205,no_goggles
7602276098610,2.6275024,1.3348818,-0.84115505,-0.8277105,0.93423694,-1.8663434,no_goggles
7602285046860,1.7300925,0.09293938,-1.5004016,-0.70627034,0.6562029,-2.1955101,no_goggles
7602294751818,1.3537543,-0.742671,-2.3912282,-0.6679208,0.33129725,-2.3691483,no_goggles
7602305712402,1.0012841,-1.9520664,-2.263209,-0.6040049,-0.18535602,-2.4298682,no_goggles
7602314583568,0.0817759,-3.4335861,-1.5868793,-0.5550028,-0.62104917,-2.4064324,no_goggles
7602324647068,-0.8024864,-4.2786837,-1.1170459,-0.5400891,-0.94595486,-2.2487733,no_goggles
7602335429402,-2.2303412,-5.2094116,-0.18559858,-0.45060685,-1.1994878,-2.0090888,no_goggles
7602345235443,-2.7811396,-5.116152,-0.19111645,-0.33768883,-1.071656,-1.6160061,no_goggles
7602355223360,-3.469398,-4.8130627,0.111622065,-0.3323625,-0.79042625,-1.2080098,no_goggles
```

```
1511126880852.999936,-0.315137,3.606865,-7.670274,1.51414,-0.513189,0.638651,no_goggles
1511126880852.999936,-0.6192,3.580529,-7.505074,1.57273,-0.384292,0.767548,no_goggles
1511126880852.999936,-0.858621,3.393781,-7.155521,1.612145,-0.266048,0.908163,no_goggles
1511126880852.999936,-1.141136,3.319561,-6.975955,1.632385,-0.158456,1.059431,no_goggles
1511126880852.999936,-1.244087,3.238158,-6.643161,1.628124,-0.055126,1.216025,no_goggles
1511126880852.999936,-1.32549,3.223793,-6.485144,1.610014,0.036487,1.37688,no_goggles
1511126880852.999936,-1.385345,3.29083,-6.307973,1.573795,0.117447,1.5388,no_goggles
1511126880852.999936,-1.323096,3.391387,-6.094889,1.538642,0.179233,1.697524,no_goggles
1511126880896,-0.803704,4.160823,-5.962011,1.513075,0.220778,1.849857,no_goggles
1511126880896,-0.698359,4.330812,-5.85906,1.494966,0.243148,1.991537,no_goggles
1511126880896,-0.57386,4.553473,-5.672313,1.487509,0.252736,2.120434,no_goggles
1511126880896,-0.554707,4.788104,-5.432893,1.499227,0.245279,2.234417,no_goggles
1511126880896,-0.542736,4.994006,-5.279664,1.521597,0.230365,2.329226,no_goggles
1511126880896,-0.628927,5.132869,-5.133617,1.548229,0.214386,2.395272,no_goggles
1511126880896,-0.786944,5.271733,-5.006725,1.597231,0.190951,2.437883,no_goggles
1511126880896,-0.913837,5.317223,-4.73618,1.65369,0.176037,2.458123,no_goggles
```

Figure 36: 16 lines of raw accelerometer and gyroscope data collected from the smartphone, then from the smartwatch, respectively. The first "column" of the data denotes timestamps used to keep track of the exact time each point of data was collected. Note that each timestamp in the phone data is different, but for the watch data, multiple different points of data are associated with the same timestamp.

The clear solution to this class of issues would be to have a device that polls at a fixed rate, such as a dedicated accelerometer and gyroscope. The Android API allows only for interrupt based sensor readings that trigger whenever new sensor data is received. Rectifying this inconsistent factor would greatly simplify data collection, solve most of the data preprocessing, and perfectly resolve the input dimensions for the neural network without padding, truncation, or inference. We believe that this will result in a far more effective classifier.

6.2.3 Mobile Application Limitations

Throughout this project, we were not able to implement everything we wanted to and encountered various obstacles during the development process of this app. The main cause of these issues was our limited time.

One limitation of our mobile application is that it is not a continuously running app that classifies the user's data in real time. Having our app like this would allow users to leave it running in the background and be continuously notified of what our neural net classifies their current walk.

When initially designing this application we wanted to personalize the application by having user made accounts that store a history of their use of our app. A log of the user's application use would allow them to track their drinking habits. This component of the app was not considered vital to its main functionality and was ultimately not implemented due to a lack of time.

The app currently requires a server to perform its classification. It is reasonable to port the classifier to TensorFlow and export it to mobile, or to construct the model using an Android Java library such as deeplearning4j and load our model's weights. As the classification is not exceedingly computationally intensive and the mobile phones often have powerful processors, this should not take too long. The data preprocessing will be expensive to compute onboard, but could be minimized with the design changes outlined previously. If performance is not important, a direct port of the data manipulation routines used on the server can run on the phone as well for a wholly client side version of our implementation.

6.2.4. Neural Network Limitations

Our neural network is based on the concept of deep learning, which requires large volumes of data to replace feature extraction. However, the small amount of data we have is a severe limitation and so we conjecture that the techniques we use to optimize our classifier may show better results on a model trained using more data. In Section 2.7, Ramaiah et al.'s evaluation of deep learning against standard machine learning methods showed that without hundreds of hours of data to train their classifier, it would not surpass a random forest classifier . The importance of data can be visualized in Figure 37, which shows increased loss values as our neural net is trained on less and less data. A clear trend can be seen that would presumably continue to yield less loss and more accuracy with more data.

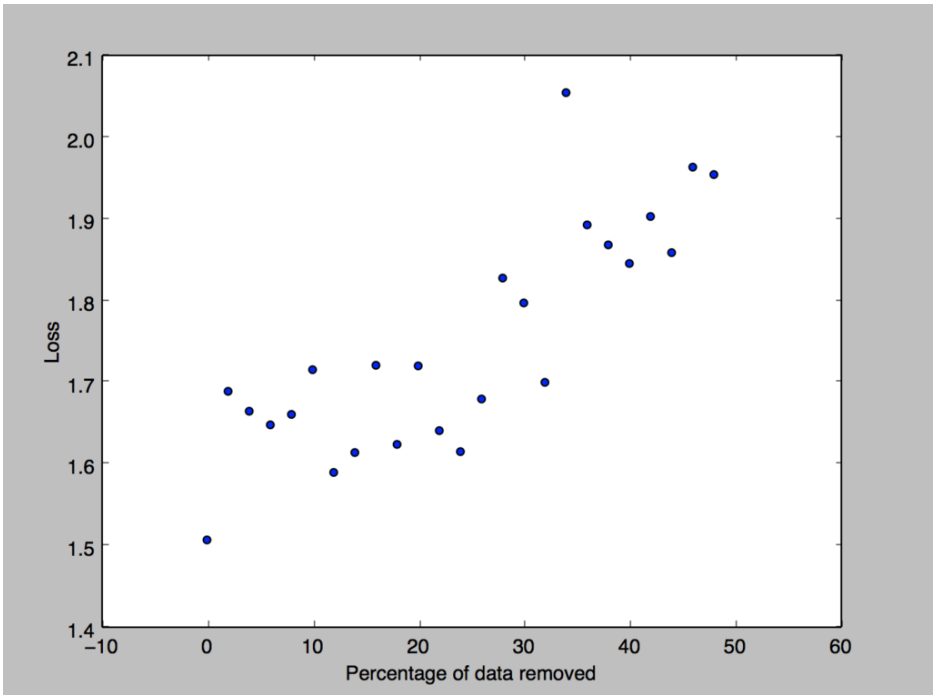

Figure 37: Loss graphed against percentage of data removed. We repeatedly trained our neural net on less and less of our data, and observed a trend in which the loss was higher and higher as more data was removed.

Additionally, our convolutional architecture is difficult to optimize using hyperparameter search algorithms because while we want to apply the widest possible search space to find the best configuration, some configurations are invalid and will cause the execution of the algorithm to stop due to model compilation error. This is because the convolution operation reduces the output dimension by a variable amount depending on the filter size and stride length, two important hyperparameters. If the output dimension exceeds the next layer's minimum input dimension, the model will fail to compile. Therefore, we limited our search to only viable combinations and fine-tuned by manually searching. A more robust algorithm that will calculate output dimensions and reject invalid permutations could search a wider space by considering multiple variables at a time, which our manual search could not achieve.

7. Conclusion and Future Work

7.1 Conclusion

Alcohol abuse is a significant problem in the United States, resulting in nearly 88,000 annual deaths (National Highway Traffic Safety Administration, 2016). In an attempt to alleviate this issue, past research has resulted in the AlcoGait app, which used machine learning to classify BAC levels in users. Further research resulted in the AlcoWatch app, which classified BAC levels in users based on both smartphone and smartwatch data. Our project built on these past solutions by utilizing deep learning techniques to classify smartphone and smartwatch accelerometer and gyroscope data.

Our team was able to develop a new version of the AlcoGait system, which we renamed IntoxiGait, using deep learning. IntoxiGait instructs the user to walk for twelve seconds, while the smartphone and smartwatch gather accelerometer and gyroscope data in conjunction with one another. After this data is collected, it is sent to the server via a POST request, where it is preprocessed and classified server-side. The classification results are then sent back to the phone as JSON, and the user is informed of their BAC level. Using a dual-input convolutional neural network architecture trained on data gathered from 40 participants, IntoxiGait can classify user data into five bins: [0.00], [0.00-0.08), [0.08-0.15), [0.15-0.25) and [0.25-0.35), with an accuracy of 65%.

This is an improvement over the AlcoWatch team's results, who achieved a 40.69% success rate among five bins. While our success rate is not directly an improvement over Christina Aiello's AlcoGait smartphone app, which achieved an 89.45% success rate using the J48 classifier using a 99:1 test-train split, we note that our result using our test-training-validation split of 60:20:20 is comparable to her result of 72.66% using a 66:33 test-train split with the Random Forest classifier. Additionally, Aiello does not specify how much data was inputted to her classifier to yield this result. Because of this, we unfortunately cannot directly compare our result, which classifies inputs of length two seconds, to hers.

7.2 Future Work

To expand upon our study, future researchers can add a gait personalization feature, where a user can input an initial walk recording while sober which will provide a baseline for their specific gait. This baseline can be inputted into the feature extracting layers of our model to obtain the model's representation of the walk, and then save this output and supply it as a

feature vector for a third input in our main neural network. This will allow the model to consider an individual's baseline gait whenever classifying other walks.

Additionally, we recommend conducting test studies on actual drunk people, if at all possible. It is possible that the Drunk Busters goggles were not an accurate representation of different ranges of intoxication. Additionally, it would be beneficial to write a robust preprocessing script that can resolve the sampling frequency and noise introduced by having different models of phones and watches, each of which have different hardware specifications. Ideally, this preprocessing script would also automatically trim out any noise within the data. While we were able to reliably do this manually, automating this task would reduce a significant amount of overhead.

In the future, we believe that this could be turned into an application that runs in the background of a cell phone and constantly informs users of their states. Possible applications could include, but are not limited to, contacting friends and family when one is not fit to drive, informing one that they are drunk as they move into a car, or recommending one find someone else to help them get home as they leave a drinking location. The one concern that may arise from creating such an application is power consumption. Further research may need to be done in order to determine if having this application constantly runs in the background will consume a significant amount of battery life.

Additionally, if future researchers use the same data gathering app as us, we recommend that they add a feature which vibrates the devices when the phone and watch are disconnected during the time that the app is recording data. We encountered several instances in which the phone and watch would disconnect as we were recording data from participants, we were not aware of it happening until we plotted the data. Having this vibrate feature could have potentially save us a lot of time.

Rather than following our method of manually trimming the data, we recommend that future researchers automate this trimming process. This process of manually trimming the data took a decent amount of time.

References

Aiello, C. (2015). *Investigating gyroscope sway features, normalization and personalization in detecting intoxication in smartphone users*

Alsheikh, M. A., Selim, A., Niyato, D., Doyle, L., Lin, S., & Tan, H. (2015). Deep activity recognition models with triaxial accelerometers. Retrieved from http://arxiv.org/abs/1511.04664

Arnold, Z., & LaRose, D. (2015). *Smartphone gait inference*.

Bianchi, B., McAfee, A., & Watson, J. (2017). *Estimating BAC levels via accelerometer and gyroscope data with smartwatches*. Worcester Polytechnic Institute:

Chen, Y., & Xue, Y. (2015). A deep learning approach to human activity recognition based on single accelerometer. 1488-1492. doi:10.1109/SMC.2015.263

Choi, P., Chou, Y., Su, F., & Lin, T. (2003). Normal gait of children. *Biomedical Engineering Applications Basis and Communications, 15*(4) Retrieved from https://www.researchgate.net/publication/247914496_Normal_gait_of_children

Dey, A. K., Wac, K., Ferreira, D., Tassini, K., Hong, J., & Ramos, J. (2011). Getting closer. *Proceedings of the 13th International Conference on Ubiquitous Computing - UbiComp '11,*

Dodge, S., & Karam, L. (2017). A study and comparison of human and deep learning recognition performance under visual distortions. Paper presented at the 1-7. doi:10.1109/ICCCN.2017.8038465

François Fleuret, Ting Li, Charles Dubout, Emma K. Wampler, Steven Yantis, & Donald Geman. (2011). Comparing machines and humans on a visual categorization test. *Proceedings of the National Academy of Sciences of the United States of America, 108*(43), 17621-17625. doi:10.1073/pnas.1109168108

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. London, England: The MIT Press.

Hammerla, N. Y., Halloran, S., & Ploetz, T. (2016). Deep, convolutional, and recurrent models for human activity recognition using wearables. Retrieved from http://arxiv.org/abs/1604.08880

Jiang, W., & Yin, Z. (2015). *Human activity recognition using wearable sensors by deep convolutional neural networks*.

Jiang, W., & Yin, Z. (Oct 13, 2015). Human activity recognition using wearable sensors by deep convolutional neural networks. 1307-1310. doi:10.1145/2733373.2806333

Kanny, D., Liu, Y., Brewer, R. D., & Lu, H. (2013). *Binge drinking - united states, 2011*. National Center for Chronic Disease Prevention and Health Promotion.

Karch, MD, FFFLM, Steven B. (2016). *Forensic issues in alcohol testing* (1st ed.). Baton Rouge: CRC Press.

Lane, N., Bhattacharya, S., Georgiev, P., Forlivesi, C., & Kawsar, F. (Nov 1, 2015). An early resource characterization of deep learning on wearables, smartphones and internet-of-things devices. 7-12. doi:10.1145/2820975.2820980

Mokdad, A. H., Marks, J. S., Stroup, D. F., & Gerberding, J. L. (2004). Actual causes of death in the United States; *JAMA: Journal of the American Medical Association,* (291(10)), 1238–1245.

Moore, S. C., Wood, A. M., Moore, L., Shepherd, J., Murphy, S., & Brown, G. D. A. (2016). A rank based social norms model of how people judge their levels of drunkenness whilst intoxicated. *BMC Public Health, 16*(1), 798. doi:10.1186/s12889-016-3469-z

Moore, S., Wood, A. M., Moore, L., Shepherd, J., Murphy, S., & Brown, G. D. A. (2016). *A rank based social norms model of how people judge their levels of drunkenness whilst intoxicated.*

Munro, D. (2015, Apr 27,). Inside the $35 billion addiction treatment industry. *Forbes,*

National Highway Traffic Safety Administration. (2015). *The economic and societal impact of motor vehicle crashes, 2010*.

National Highway Traffic Safety Administration. (2016). *Alcohol-impaired driving: Traffic safety facts, 2015 data*. Washington, DC: National Highway Traffic Safety Administration.

NIDA. (2012). Principles of drug addiction treatment: A research-based guide (third edition). *Principles of Drug Addiction Treatment: A Research-Based Guide (Third Edition),* Retrieved from https://www.drugabuse.gov/publications/principles-drug-addiction-treatment-research-based-guide-third-edition/drug-addiction-treatment-in-united-states

Nieschalk, M., Ortmann, C., West, A., Schmäl, F., Stoll, W., & Fechner, G. (1999). Effects of alcohol on body-sway patterns in human subjects. *International Journal of Legal Medicine, 112*(4), 253-260. doi:10.1007/s004140050245

Ordóñez, F. J., & Roggen, D. (2016). Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition. *Sensors (Basel, Switzerland), 16*(1), 115. doi:10.3390/s16010115

Ramaiah, C., Tran, A. R., Cox, E. F., & Moehler, G. O. (2016). Deep learning for driving detection on mobile phones. 22nd ACM SIGKDD. doi: 10.1145/1235

Ravi, D., Wong, C., Lo, B., & Yang, G. (2017). A deep learning approach to on-node sensor data analytics for mobile or wearable devices. *IEEE Journal of Biomedical and Health Informatics, 21*(1), 56-64. doi:10.1109/JBHI.2016.2633287

Ronao, C. A., & Cho, S. (2016). Human activity recognition with smartphone sensors using deep learning neural networks. *Expert Systems with Applications, 59*, 235-244. doi:10.1016/j.eswa.2016.04.032

Sacks, J. J., Gonzales, K. R., Bouchery, E. E., Tomedi, L. E., & Brewer, R. D. (2015). *2010 national and state costs of excessive alcohol consumption*.

SAMHSA. (2013). National survey of substance abuse treatment services (N-SSATS), 2011. Retrieved from http://www.dx.doi.org/10.3886/ICPSR34539.v1

Sathyanarayana, A., Joty, S., Fernandez-Luque, L., Ofli, F., Srivastava, J., Elmagarmid, A., Taheri, S. (2016). *Sleep quality prediction from wearable data using deep learning*. JMIR Mhealth Uhealth.

Semwal, V., Mondal, K., & Nandi, G. (2017). Robust and accurate feature selection for humanoid push recovery and classification: Deep learning approach. *Neural Computing and Applications, 28*(3), 565-574. doi:10.1007/s00521-015-2089-3

Smith, J. (2016). Binge drinking map of America shows worst affected states where up to a quarter of the population are drinking too much. Retrieved from http://www.dailymail.co.uk/news/article-4059084/Binge-drinking-map-America-shows-worst-affected-states-quarter-population-drinking-much.html

Song-Mi Lee, Sang Min Yoon, & Heeryon Cho. (Jan 1, 2017). Human activity recognition from accelerometer data using convolutional neural network. 131.

Tahmina Zebin, Patricia J Scully, & Krikor B. Ozanyan. (Jan 1, 2016). Human activity recognition with inertial sensors using a deep learning approach. 1.

University of Wisconsin. A basic introduction to neural networks. Retrieved from

http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html

Valkonen, M., Kartasalo, K., Liimatainen, K., Nykter, M., Latonen, L., & Ruusuvuori, P. (2017). Dual structured convolutional neural network with feature augmentation for quantitative characterization of tissue histology. 27-35. doi:10.1109/ICCVW.2017.10

Yao, S., Hu, S., Zhao, Y., Zhang, A., & Abdelzaher, T. (2016). DeepSense: A unified deep learning framework for time-series mobile sensing data processing. Retrieved from http://arxiv.org/abs/1611.01942

Zheng, Y., Liu, Q., Chen, E., Ge Y., Zhao, J.L. (2016). Exploiting multi-channels deep convolutional neural networks for multivariate time series classification. *中国计算机科学前沿：英文版, 10*(1), 96-112. doi:10.1007/s11704-015-4478-2

Zhao, Y., Yang, R., Chevalier, G., & Gong, M. (2017). Deep residual bidir-LSTM for human activity recognition using wearable sensors. Retrieved from http://arxiv.org/abs/1708.08989