Determining Smartphone Use While In-Vehicle from IMU Sensors

Noah Curran
University of Michigan, Ann Arbor
ntcurran@umich.edu
Submitted: 14 December 2020

***Abstract*—The pervasiveness of smartphones is indisputable, and as such, they lend themselves as useful assistants to other technologies, such as fall detection, physical activity prediction, and vehicular dynamics estimations. However, person-on-phone interactions can meddle with the accuracy of these technologies. In this paper, we propose a technique for determining whether or not there is a person-on-phone interaction occurring while in-vehicle. We accomplish this via machine learning (ML) classification of inertial measurement unit (IMU) sensor readings. In pursuit of investigating this technique, we develop an application for collecting labeled IMU sensor information, and call it CarStudy. Following data collection, we evaluate this technique against five different classification algorithms: k-nearest neighbor, naive-Bayes, logistic regression, rule-based, and decision tree. The results show that training a decision tree classifier with IMU gyroscope readings sliced into 0.5s frames yields the best accuracy of 99.7969%. The high accuracy is likely due to biological vibrations picked up by the gyroscope during person-on-phone interactions.**

I. Introduction

Within modern smartphones, a number of sensors are embedded in order to enable the functionality of several features. For instance, there exists a GPS that enables map applications to provide a quick route from your current location to a desired location. In another case, an embedded gyroscope allows the phone to automatically rotate the screens orientation. The gyroscope is included in a complex of sensors called the inertial measurement unit (IMU). The other sensors included in the IMU are an accelerometer and sometimes a magnetometer.

As smartphones have become commonplace, innovation has leveraged them to act as an assistant to solving many different tasks. Much of this innovation utilizes sensors to sense the environment surrounding a smartphone to obtain some level of context awareness, such as how the smartphone is being used or activities the smartphone holder is performing [1]–[3]. These kinds of problems are actively pursued due to their practical applications. In the case of a fall detection application [4]–[7], researchers hope to be able to create accurate predictions of when this occurs in order to track when someone who suffers from an ailment has fallen. Other work attempts to accurately determine physical activities (e.g., walking, running, or climbing stairs) performed by the owner of the smartphone [8]–[11]. This is coined as activity recognition technology. In our case, we wish to sense and predict vehicular dynamics information—such as the speed or steering wheel angle—while the smartphone is within a vehicle. However, while attempting to sense context information of the smartphone's surroundings, it is critical to first interpret the context of how the smartphone itself is being used via activity recognition technology. Otherwise, the environmental sensing algorithms may draw incorrect conclusions due to person-on-phone interactions.

Previous work for determining person-on-phone usage in-vehicle is motivated by safety concerns, focusing efforts on driver smartphone usage. For instance, in [12]–[14] the main objective is to determine if the smartphone is within close proximity of the driver. However, these papers do not address whether or not the phone is currently in use. In [15]–[17] it is explored how to determine if a smartphone is distracting the driver, but extra cameras are utilized in [16], [17] and infrared sensors are utilized in [15]. These methods also implement computationally expensive deep learning algorithms. We wish to devise a method of determining person-on-phone usage in-vehicle without these constraints.

In order to address the concern of person-on-phone interaction within the context of the problem we are trying to solve (i.e., predicting vehicular dynamics information), we propose a system for determining whether a smartphone is in-hand or out-of-hand while within a vehicle, which can be found in Figure 1. Our solution leverages machine learning (ML) to classify the state of a smartphone based on its IMU sensor readings. In order to determine which ML classifier is most useful for our application, we test the framework against five different classifiers: k-nearest neighbor, naive-Bayes, logistic regression, rule-based, and decision tree.

En route to accomplishing this, we design a tool called CarStudy. CarStudy collects and records the accelerometer, gyroscope, and magnetometer sensor readings in the x-, y-, and z-axes. Before collection commences, a particular position in the car is selected, and this is automatically applied to these readings as their label. This enables the large-scale collection of labeled IMU sensor readings while in-vehicle.

The following summarizes the contributions of this work:

- We develop a tool, CarStudy, for collecting smartphone IMU sensor data while within a vehicle.
- We propose a framework for training a classifier for determining if a smartphone is in-hand or out-of-hand while within a vehicle.
- We evaluate the framework against several types of classifiers to determine which yields the best accuracy.

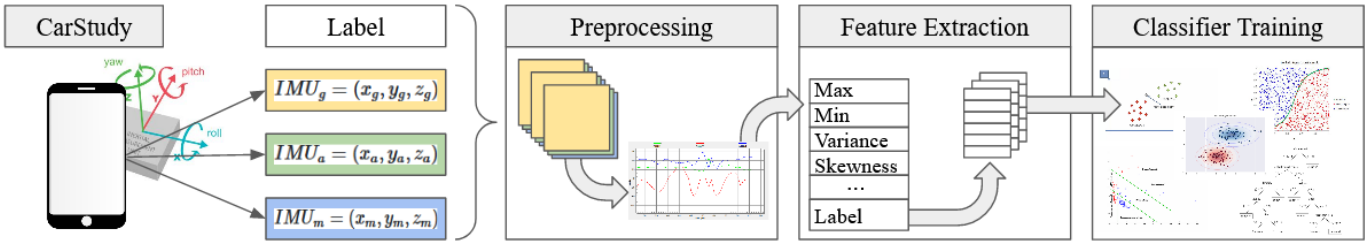

Fig. 1. The end-to-end framework for training a classifier to determine if the phone is held in-hand or out-of-hand. IMU data is collected and labeled using CarStudy. Then, the data is grouped into 0.5s, 1.0s, and 5.0s slices. Features are extracted from these slices before they are utilized to train each classifier.

II. Background

Smartphones lend themselves as a useful tool for solving complex problems in a creative way. In this vein, we hypothesize that the sensors within smartphones can be utilized to make accurate predictions of vehicular dynamics. However, in order to make these predictions, we first must determine whether or not the phone is in a person's hand. This is because when the phone is in-hand, person-on-phone interactions will cause noise within the IMU sensors. To describe why this is a problem in detail, we first discuss what IMU sensors are and what their values look like. Then, we describe how these can be utilized to estimate vehicular dynamics. To bring the problem into perspective, we introduce what it means for a sensor to be noisy and how this can be detected.

A. IMU Sensors

In order to provide smartphones with functions such as automatic screen orientation rotation, they are built with an embedded IMU. The IMU contains an accelerometer, gyroscope, and sometimes a magnetometer, which respectively report information about the smartphone's linear acceleration, angular acceleration, and orientation of the body in relation to magnetic north. Collectively, these sensors are called the IMU sensors.

Because the physical world is in three dimensions, the IMU sensors will report their readings in the form of

$$IMU_s = (x_s, y_s, z_s), \tag{1}$$

where x, y, and z represent the measurement along each of the axes. The values the IMU sensors report are due to changes within each of the IMUs axes, which are known as the pitch, roll, and yaw. A graphical representation of this can be seen in Figure 2. The pitch, roll, and yaw are relative to the orientation of the smartphone.

B. Estimating Vehicle Dynamics

Utilizing the IMU sensors, we can draw estimations of how the smartphone's surroundings interact with it. Following this, we can leverage the smartphone's IMU sensors to determine how a vehicle is moving. In essence, if the smartphone is resting within the vehicle, the IMU sensors will report information as the entire combined body of the smartphone and the vehicle. For example, we can use an accelerometer to determine how fast the vehicle is moving. This velocity can also help estimate the engine's RPM, the steering wheel angle, and the gear. The gyroscope is also used when estimating the steering wheel angle. The magnetometer helps to realign the IMU orientation.

These estimations could be useful for a variety of smartphone-based vehicle-assistant applications. For our future work, we wish to integrate these estimations within a smartphone anomaly detection application, which would distinguish if there are differences in vehicular dynamics reported within the in-vehicle network called the Controller Area Network (CAN).

C. Sensor Noise & Activity Recognition

Ideally, the smartphone will be able to perform these estimations at any given moment; however, in the real-world these ideals do not always hold. Sometimes, the driver or a passenger may interact with the smartphone, causing the IMU sensors to become noisy from the extra input. This will unquestionably throw off the estimations that the smartphone is able to make, and for applications that require accurate estimations, this is unacceptable.

To give a demonstration, suppose the smartphone is mounted on the dashboard of the vehicle. Passively, it is measuring the IMU sensor information and making estimations of the velocity of the vehicle. But then a person sitting in the passenger seat reaches over and grabs the smartphone to find a route to a restaurant for the driver. While assisting

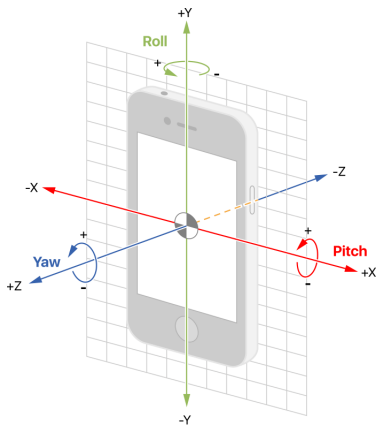

Fig. 2. The pitch, roll, and yaw denote the change in the x-, y-, and z-axes respectively.

the driver, the passenger has a conversation about which restaurant they should go to. During the entire conversation, the passenger makes gestures with the phone in-hand. Once decided, the passenger returns the smartphone to the mount. While the conversation was occurring, the smartphone was waved around due to the passenger's gestures, causing the velocity estimations to be thrown off.

Therefore, in order to know when it is appropriate to estimate the vehicular information, we propose utilizing activity recognition techniques to determine when the smartphone is in-hand while within a vehicle. Similar to the vehicular dynamics estimations, activity recognition utilizes the IMU sensors to determine which activity is occurring around the smartphone. This typically involves making use of ML classification or statistical modelling to distinguish between various actions imposed on the IMU sensors. We opt to implement a classification model to analyze the IMU sensors to determine if the smartphone is in-hand or out-of-hand.

III. Data Collection

In order to train a ML classification model, we require a verbose data set. In our case, we should train the classifier on IMU sensor data that represents all the common locations smartphones are kept in-vehicle. To easily obtain this data we implement CarStudy, a tool for collecting and storing labeled IMU sensor data. In the following subsections, we describe how this tool works and how we use it to collect vehicle data for this study.

A. CarStudy

To easily collect IMU sensor data, we implement a tool, CarStudy. This was developed for Android smartphones with an easy-to-use interface, shown in Figure 3. To use the application, the user simply presses the drop-down menu, selects the position the phone will reside while recording information, and taps the button labeled START. When the user is done collecting data, they press STOP.

The drop-down menu lists several positions the smartphone can reside while within a vehicle. These are specifically chosen to have broad coverage of the places smartphone owners typically put their phone according to a survey found in [18]. To summarize, 97% of surveyed individuals put their phone in their pants pocket, a bag or purse, their jacket pocket, or somewhere out in the car such as the cup holder, seat, car door, or mounted on the dashboard. The remaining 3% of individuals reported keeping the phone in their hand. Therefore, we include the following locations as choices in order to have good coverage: mounted, cup holder, seat, bag, shirt pocket, pant pocket, or held.

Once data collection begins, sensor values from the accelerometer, gyroscope, and magnetometer are obtained around every 10ms. These values are obtained in the form of Equation 1. Once data collection is halted, you can either choose to discard the collected data by pressing RESET, or record the collected data by pressing RECORD. When recording the data, it is saved locally to an SQLite database for future retrieval. The selected position and the sensor values are grouped together, as shown in Figure 1.

B. Collecting Data

When using CarStudy, we drive a 2020 Jeep Compass. The model of the phone we use for the data collection is a Samsung Galaxy S6. The routes selected include a variety of different roads, including neighborhoods, suburban streets, backroads, and highways. We collect around 15 minutes of IMU sensor data for each of the seven locations in the vehicle previously listed. Since we record sensor values every 10ms, this amounts to around 90,000 sets of values for each IMU sensor for each position. We collect all data during days where there is no precipitation and the roads are dry.

C. Challenges

A few challenges arose during data collection. First, bugs not caught during manual testing of the tool arose. For instance, if the smartphone entered sleep mode in the middle of collecting or saving the data, all of the recorded values are destroyed. Therefore, during early data collection, many retrials were conducted to obtain accurate and complete data despite these hiccups.

Furthermore, when saving the data to the SQLite database, it would take around twice as long to save the data as it took to collect it. For instance, for a single collection of 15 minutes of data, it would take around 45 minutes total between collecting and saving. Because of this, data collection took significantly longer than expected, preventing more trials from being conducted.

If more data is collected in the future, these challenges should be kept in mind, and a second version of the tool should attempt to mitigate them.

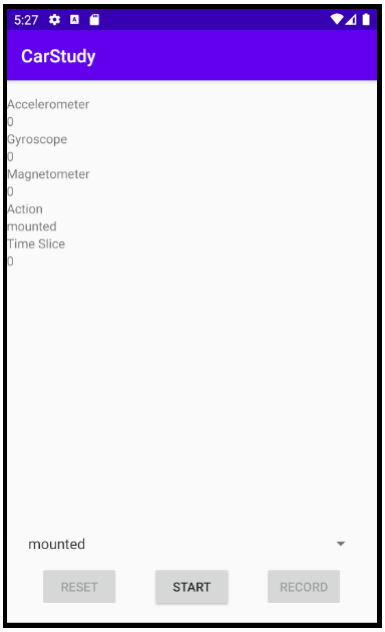

Fig. 3. The user interface for CarStudy.

TABLE I
FEATURES EXTRACTED DURING PREPROCESSING

Feature	Notation	Equation
Max	s_{max}	$s_{max} = \max s(t)$
Min	s_{min}	$s_{min} = \min s(t)$
Mean	μ	$\mu = \frac{1}{N}\sum s(t)$
Variance	σ^2	$\sigma^2 = \frac{1}{N}\sum (s(t)-\mu)^2$
Kurtosis	K	$K = m_4/m_2^2$
Skewness	S	$S = m_3/m_3^{3/2}$
Peak-to-peak signal	s_{pp}	$s_{pp} = s_{max} - s_{min}$
Peak-to-peak time	t_{pp}	$t_{pp} = t_{s_{max}} + t_{s_{min}}$
Peak-to-peak slope	m_{pp}	$m_{pp} = s_{pp}/t_{pp}$
Absolute-latency-to-amplitude ratio	$ALAR$	$ALAR = \lvert t_{s_{max}}/s_{max}\rvert$
Energy	$E(S(f))$	$E(S(f)) = \sum \lvert S(f)\rvert^2$
Entropy	$H(S(f))$	$H(S(f)) = -\sum_{i=1}^{N} p_i(S(f)) \log_2 p_i(S(f))$

IV. EXPERIMENTATION

After the data is collected, we can begin training a ML classifier to determine whether or not a person is holding a phone in-vehicle. However, there are multiple classifiers we can choose from that are actively used in activity recognition research, including: k-nearest neighbor, naive-Bayes, logistic regression, rule-based, and decision tree. In order to determine which classifier suits this problem the best, we should train each one on the data set collected with CarStudy and compare the results.

Furthermore, when preprocessing the data set, the sensor values need to be grouped together in slices before they are analyzed for features. It has been shown that for activity recognition, 5.0s slices are sufficient [2], [19] for classification, but we will also consider 0.5s and 1.0s slices since the activity recognition we are classifying is less exaggerated than other activity recognition problems.

Finally, since some IMU sensor information may not be as useful for identifying smartphone usage, we look at every combination of IMU sensors. For instance, only utilizing the gyroscope data, both accelerometer and magnetometer data, or all three sensor data together.

Therefore, following this process, the experimentation looks at every combination of classifier, data slicing window size, and IMU sensors. We examine the accuracy of each classifier in order to determine the best combination of choices.

A. Preprocessing

During the preprocessing phase, we transform the data into usable feature vectors for training the ML classifiers. Because the data on its own is for a particular orientation of the smartphone, we first calculate the magnitude of each vector of data collected with

$$Mag_s = \sqrt{x_s^2 + y_s^2 + z_s^2} \tag{2}$$

since the magnitude does not vary with the orientation of the smartphone. Furthermore, the data for each sensor is sliced into even groups of sequential values that do not overlap. We create three separate groups of slices, where their frame sizes are either 0.5s, 1.0s, or 5.0s. A particular slice for an arbitrary sensor axis is represented by the function $s(t)$.

Then, features are extracted from each individual slice. Previous work provides good values to be extracted for activity recognition [2], [19]–[21], and these are outlined in Table I. Alongside the label for the particular data being processed, the features are kept in a vector for future training and testing. In total, the feature vector for each slice has 144 available features. There are 12 features for each axis and the magnitude for each IMU sensor.[1]

B. Training ML Classifiers

Once the collected data has been preprocessed into the respective feature vectors, it is ready to be trained on a classifier. We test the data sets on five ML classifiers commonly found in activity recognition research: k-nearest neighbor, naive-Bayes, logistic regression, rule-based, and decision tree. Since we are testing 3 data frame slice sizes, 7 combinations of sensors (e.g., only gyroscope, both accelerometer and magnetometer, or all three), and 5 classifiers, there will be a total of 105 trials.

Due to the large number of trails, we require a way to perform these tests quickly. Therefore, we chose to use Weka [22], a ML and data mining software suite. Embedded within Weka is Weka Explorer, which allows for us to select a data set, omit features of the data—like when we wish to exclude certain sensors—and train it on a selected classifier. When training a classifier, we have the opportunity to fine tune some settings. For instance, we can choose to exclude a percentage of the data from the training data set so it can be used later

[1]Preprocessing code repo: github.com/noah-curran/SensorDataCleaning.

TABLE II
ACCURACY OF CLASSIFIER USING ONLY ACCELEROMETER

Classifier \ Frame Slice	0.5s	1.0s	5.0s
Naive-Bayes	70.4961%	71.4701%	76.3158%
Logistic Regression	95.5614%	94.9448%	89.7661%
k-Nearest Neighbor	91.6739%	92.2719%	90.9357%
Rule-Based	95.8225%	95.7002%	95.614%
Decision Tree	**96.4317%**	95.8745%	95.0292%

TABLE III
ACCURACY OF CLASSIFIER USING ONLY GYROSCOPE

Classifier \ Frame Slice	0.5s	1.0s	5.0s
Naive-Bayes	81.8393%	81.871%	82.7485%
Logistic Regression	97.0119%	96.688%	90.0585%
k-Nearest Neighbor	96.5767%	94.3056%	92.3977%
Rule-Based	99.1587%	98.373%	99.7076%
Decision Tree	**99.7969%**	99.3608%	98.538%

TABLE IV
ACCURACY OF CLASSIFIER USING ONLY MAGNETOMETER

Classifier \ Frame Slice	0.5s	1.0s	5.0s
Naive-Bayes	77.2556%	76.8739%	78.0702%
Logistic Regression	87.1773%	86.9262%	85.6725%
k-Nearest Neighbor	74.0064%	76.7577%	77.4854%
Rule-Based	**85.8718%**	85.2992%	80.9942%
Decision Tree	85.8138%	85.0087%	82.7485%

TABLE V
ACCURACY OF CLASSIFIER USING ACCELEROMETER & GYROSCOPE

Classifier \ Frame Slice	0.5s	1.0s	5.0s
Naive-Bayes	93.5596%	93.4340%	92.3977%
Logistic Regression	98.7525%	97.7339%	90.0585%
k-Nearest Neighbor	97.4761%	96.5718%	96.1988%
Rule-Based	99.2457%	98.3730%	99.4152%
Decision Tree	**99.7679%**	99.3608%	98.2456%

TABLE VI
ACCURACY OF CLASSIFIER USING ACCELEROMETER & MAGNETOMETER

Classifier \ Frame Slice	0.5s	1.0s	5.0s
Naive-Bayes	77.0525%	76.8158%	76.6082%
Logistic Regression	98.0853%	97.0366%	91.2281%
k-Nearest Neighbor	93.5596%	93.4922%	92.6901%
Rule-Based	97.7372%	97.0366%	95.3216%
Decision Tree	**98.9266%**	98.1406%	95.0292%

TABLE VII
ACCURACY OF CLASSIFIER USING GYROSCOPE & MAGNETOMETER

Classifier \ Frame Slice	0.5s	1.0s	5.0s
Naive-Bayes	94.9811%	95.0029%	96.7836%
Logistic Regression	97.9402%	96.9204%	92.9825%
k-Nearest Neighbor	97.0699%	95.7583%	92.9825%
Rule-Based	99.4778%	99.1865%	99.7076%
Decision Tree	**99.7099%**	99.5352%	99.1228%

TABLE VIII
ACCURACY OF CLASSIFIER USING ALL THREE

Classifier \ Frame Slice	0.5s	1.0s	5.0s
Naive-Bayes	96.7218%	96.3393%	96.4912%
Logistic Regression	99.1877%	98.0244%	93.5673%
k-Nearest-Neighbor	97.3020%	96.4555%	95.0292%
Rule-Based	99.4198%	99.1865%	99.4152%
Decision Tree	**99.7099%**	99.4770%	99.1228%

on for testing the accuracy of the classifier. We can also perform a k-fold cross-validation to ensure the classifier is well-performing.

To ensure the classifiers are properly trained, a few precautions are taken. First, the labels attached to the feature vectors are set to either be "in-hand" or "out-of-hand". The "out-of-hand" label encompasses all non-held positions the smartphone can reside while in the vehicle. Due to the disparity in data collected, the final training data set is filtered to have the same number of entries for each label. This is to prevent over training one label over the other. Furthermore, in order to mitigate overfitting in the classifiers, a k-fold cross-validation is used with $k = 10$. In the following section we summarize the results.

V. Results & Discussion

In this section we summarize the results for training ML classifiers to detect if a smartphone is in-hand or out-of-hand while in a vehicle. Table II to Table VIII contain the results of each classifier, frame slice size, and IMU sensor combination's accuracy. The best result for each table is in bold.

In general, the decision tree classifier for the 0.5s slice size data set almost always yields the best result for each sensor combination. The only exception is seen in the table of results from training classifiers with only the magnetometer's data, which achieves its best result from a rule-based classifier trained with 0.5s slices. However, the accuracy was nearly the same as that of the decision tree classifier trained with 0.5s slices. Therefore, this suggests that a decision tree might be the ideal classifier for recognizing whether or not a smartphone is being used within a vehicle. Moreover, the decision tree is a fast classifier, so it would not compromise the real-time requirements of applications that would integrate this framework. The results for the decision tree classifiers are compared side-by-side in Figure 4.

Furthermore, it is observed that training classifiers on exclusively the gyroscope data tends to yield the best accuracy. There are several results with above 99% classification accuracy, with the highest yielding 99.7969% accuracy. In fact, all classification groups that use the gyroscope data within the feature vector have good and very similar results across the board. We believe the reason for this is due to the gyroscope's ability to pick up on biological vibrations applied on the smartphone from being held by a person. This minute difference in variability enables accurate classification. Because a smaller feature vector yields quicker classification, we conclude that utilizing only the gyroscope data is sufficient and optimal.

The results are incredibly promising, indicating that activity recognition of in-vehicle smartphone usage is achievable with high accuracy.

VI. Future Work

Because this work is designed with the intention of being integrated into other applications that make use of in-vehicle activity recognition, there is much to be done for the future work. Furthermore, there are improvements that can likely be made on these results.

First, there is reason to be concerned that the results are misleading. They do feel almost too good to be true, and as such they should be verified with further real-world testing before they are accepted as strong evidence. Without this verification, we remain skeptical. However, we are optimistic, as we hypothesize the reason for the strong results are due to the biological vibrations present when a person holds the phone. The gyroscope detects these vibrations, and as a result additional noise is present in the sensor readings. Some of our measured features—such as the variance and kurtosis—will catch this additional noise and aid in distinguishing between in-hand and out-of-hand interactions.

We also believe additional weather scenarios should be covered in future work. For instance, rain or snow cause the road to become slippery, changing the dynamics of the vehicle and imposing abnormal readings on the IMU sensors. A classifier will not account for these abnormal readings if it is not trained to detect them. By adding this data to a future training set, we believe this issue will become negligible; however, it is left to the future work to confirm this.

Furthermore, we wish to utilize this classification in future work for vehicular dynamics estimations. When the phone is out-of-hand, the sensors are less noisy, making this moment the suitable time to make estimations. This classifier will be integrated in this vehicular dynamics estimation framework to help make this distinction.

On the note of utilizing this classifier in determining the proper time to make estimations, we hypothesize that false results may actually not impact the results all that much. Further testing is required, but we believe that if the classifier is indicating that the smartphone is in-hand when it is not, then the IMU sensors may be noisy enough that they would not be usable for accurate results anyway. On the flip side, if the classification indicates the smartphone is out-of-hand, then the IMU sensors might be stable enough to make accurate estimations.

VII. Conclusion

In this paper we present a framework for determining whether or not a smartphone is in a person's hand while within a vehicle. In pursuit of evaluating this framework, we create a tool for easily collecting and labeling smartphone IMU sensor information for various parts of the vehicle, which we call CarStudy. We utilize this tool to collect 15 minutes of data (obtained every 10ms) for the 7 most common locations.

After collecting data, we use it to evaluate our proposed framework. We first preprocess the data into set slices of 0.5s, 1.0s, and 5.0s. We then calculate the magnitude of the sensor readings. Finally, we calculate 12 features for each slice created for each axis and the magnitude. Since there are three sensors in the IMU, there are 144 features obtained in total.

Finally, with the data preprocessed, we use Weka Explorer to test the feature vector data set against 5 different classifiers commonly used in activity recognition. Our results show that

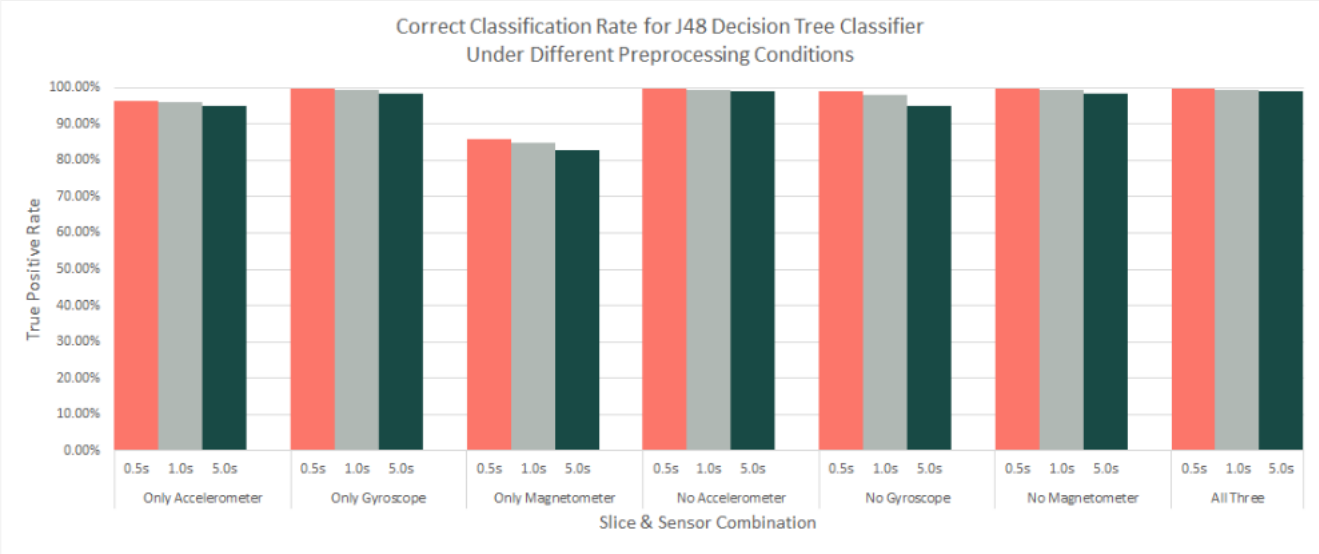

Fig. 4. The accuracy of the Decision Tree classifier. Results are grouped by their IMU sensor combination, and listed by increasing order of slice size.

the gyroscope data is able to accurately determine whether or not the phone is in-hand or out-of-hand. Training a decision tree classifier on exclusively the gyroscope features obtained from 0.5s slices of data obtained the best accuracy.

Future work will integrate this framework into an application for estimating vehicular dynamics information in order to detect anomalies reported within the in-vehicle network. This will help obtain more reliable results so that person-on-phone interactions do not impact the application's ability to detect anomalies.

References

[1] M. Ehatisham-ul Haq, M. A. Azam, J. Loo, K. Shuang, S. Islam, U. Naeem, and Y. Amin, "Authentication of smartphone users based on activity recognition and mobile sensing," *Sensors*, vol. 17, no. 9, p. 2043, 2017.

[2] M. Shoaib, H. Scholten, and P. J. Havinga, "Towards physical activity recognition using smartphone sensors," in *2013 IEEE 10th international conference on ubiquitous intelligence and computing and 2013 IEEE 10th international conference on autonomic and trusted computing*. IEEE, 2013, pp. 80–87.

[3] O. D. Incel, M. Kose, and C. Ersoy, "A review and taxonomy of activity recognition on mobile phones," *BioNanoScience*, vol. 3, no. 2, pp. 145–171, 2013.

[4] R. Igual, C. Medrano, and I. Plaza, "Challenges, issues and trends in fall detection systems," *Biomedical engineering online*, vol. 12, no. 1, p. 66, 2013.

[5] A. Z. Rakhman, L. E. Nugroho *et al.*, "Fall detection system using accelerometer and gyroscope based on smartphone," in *2014 The 1st International Conference on Information Technology, Computer, and Electrical Engineering*. IEEE, 2014, pp. 99–104.

[6] J. Santiago, E. Cotto, L. G. Jaimes, and I. Vergara-Laurens, "Fall detection system for the elderly," in *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2017, pp. 1–4.

[7] B. Aguiar, T. Rocha, J. Silva, and I. Sousa, "Accelerometer-based fall detection for smartphones," in *2014 IEEE International Symposium on Medical Measurements and Applications (MeMeA)*. IEEE, 2014, pp. 1–6.

[8] J. Wannenburg and R. Malekian, "Physical activity recognition from smartphone accelerometer data for user context awareness sensing," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 12, pp. 3142–3149, 2016.

[9] D. Coskun, O. D. Incel, and A. Ozgovde, "Phone position/placement detection using accelerometer: Impact on activity recognition," in *2015 IEEE Tenth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*. IEEE, 2015, pp. 1–6.

[10] S. Yao, S. Hu, Y. Zhao, A. Zhang, and T. Abdelzaher, "Deepsense: A unified deep learning framework for time-series mobile sensing data processing," in *Proceedings of the 26th International Conference on World Wide Web*, 2017, pp. 351–360.

[11] R.-A. Voicu, C. Dobre, L. Bajenaru, and R.-I. Ciobanu, "Human physical activity recognition using smartphone sensors," *Sensors*, vol. 19, no. 3, p. 458, 2019.

[12] J. Yang, S. Sidhom, G. Chandrasekaran, T. Vu, H. Liu, N. Cecan, Y. Chen, M. Gruteser, and R. P. Martin, "Detecting driver phone use leveraging car speakers," in *Proceedings of the 17th annual international conference on Mobile computing and networking*, 2011, pp. 97–108.

[13] Y. Wang, Y. J. Chen, J. Yang, M. Gruteser, R. P. Martin, H. Liu, L. Liu, and C. Karatas, "Determining driver phone use by exploiting smartphone integrated sensors," *IEEE Transactions on Mobile Computing*, vol. 15, no. 8, pp. 1965–1981, 2015.

[14] H. Park, D. Ahn, T. Park, and K. G. Shin, "Automatic identification of driver's smartphone exploiting common vehicle-riding actions," *IEEE Transactions on Mobile Computing*, vol. 17, no. 2, pp. 265–278, 2017.

[15] S. K. Leem, F. Khan, and S. H. Cho, "Vital sign monitoring and mobile phone usage detection using ir-uwb radar for intended use in car crash prevention," *Sensors*, vol. 17, no. 6, p. 1240, 2017.

[16] R. Khurana and M. Goel, "Eyes on the road: Detecting phone usage by drivers using on-device cameras," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020, pp. 1–11.

[17] T. Hoang Ngan Le, Y. Zheng, C. Zhu, K. Luu, and M. Savvides, "Multiple scale faster-rcnn approach to driver's cell-phone usage and hands on steering wheel detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2016, pp. 46–53.

[18] J. Wiese, T. S. Saponas, and A. B. Brush, "Phoneprioception: enabling mobile phones to infer where they are kept," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2013, pp. 2157–2166.

[19] A. Anjum and M. U. Ilyas, "Activity recognition using smartphone sensors," in *2013 ieee 10th consumer communications and networking conference (ccnc)*. IEEE, 2013, pp. 914–919.

[20] M. Shoaib, S. Bosch, O. D. Incel, H. Scholten, and P. J. Havinga, "Fusion of smartphone motion sensors for physical activity recognition," *Sensors*, vol. 14, no. 6, pp. 10 146–10 176, 2014.

[21] X. Su, H. Tong, and P. Ji, "Activity recognition with smartphone sensors," *Tsinghua science and technology*, vol. 19, no. 3, pp. 235–249, 2014.

[22] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.