Using Recurrent Neural Networks to approximate orientation with Accelerometers and Magnetometers

Akil Hosang*, Nicholas Hosein†, Patrick Hosein*

*Department of Computer Science
The University of the West Indies
St. Augustine, Trinidad and Tobago
akil.hosang@my.uwi.edu, patrick.hosein@sta.uwi.edu

†Department of Computer Science
University of California
Davis, CA, USA
nhosein@ucdavis.edu

***Abstract*—Many smart devices possess sensors that enable them to collect data regarding their environment and their users' actions. For example, gyroscopes allow smart devices to determine the orientation. Functions such as exercise detection can then exploit this orientation data. However, gyroscopes are expensive and power-hungry. In contrast, accelerometers and magnetometers are cheap and relatively energy efficient. Hence, if we could use data accelerometers and magnetometers to approximate orientation through computations on the device, we can maintain the desired functionality at a lower cost. In this paper, we benchmark several Recurrent Neural Network (RNN) architectures that use accelerometer and magnetometer data to approximate orientation with reasonable accuracy.**

***Index Terms*—Recurrent Neural Network, Accelerometer, Body Sensor, Magnetometer, Smart Devices**

I. Introduction

Smart devices, such as smartphones, have become critical and ubiquitous fixtures of everyday life in our technologically driven society. These smart devices are often portable with their utility predicated on the availability of charge in their batteries. Users expect smart devices to facilitate many functions and tasks. These functions and their variety influence the rate at which a smart device's battery is depleted [1]. Every advance in functionality elevates the expectations of consumers of their smart devices, thereby increasing the load placed on the battery of devices. This problem is unavoidable, but we can endeavour to minimize it through efforts in both software and hardware. As such, many efforts are being made to increase the effective battery life of smart devices.

We aim to aid this effort by looking at a hardware and software approach. Smart devices that utilize orientation and motion-based features can do this by using MEMS (micro-electromechanical systems) inertial measurement units (IMUs) onboard the device, such as accelerometers, magnetometers and gyroscopes. MEMS are described as miniature machines with both electrical and mechanical components. These machines range greatly in size from several millimeters to one micrometer and are composed of parts such as micro-sensors, microprocessors, micro-actuators, units for data processing and parts that can interact with exterior pieces. MEMS are the machines responsible for tasks like deploying air bags, controlling building heating and cooling systems and more relevantly, controlling the orientation of smart devices. These three sensors constantly complete and correct each other to provide the smart device with a proper understanding of its orientation. By doing this, we can use IMUs in wearables and smart devices to provide real-time analysis of movement and feedback for a user but we need to be aware of battery life [2].

One approach to increasing the battery lifetime of a device is to use larger batteries. However, the compact design of said devices often precludes larger batteries as a sensible design alternative. Moreover, users are likely to grow frustrated with having to recharge their devices multiple times per week. An alternative solution is to reduce the amount of power that a component of the wearable device, such as the IMUs, uses to achieve its job. Of these MEMS IMUs, gyroscopes are known to be the most energy-hungry and expensive, as opposed to accelerometers and magnetometers which are energy efficient and cheap, as we see in Figures 1 and 2.

If we can derive orientation vectors using the accelerometer and magnetometer, we can obviate the use of a gyroscope, thereby eliminating a source of energy consumption. We implement Recurrent Neural Networks (RNN) that map sequences of accelerometer and magnetometer data to sequences of orientation vectors. In the future, after the neural network has been heavily trained and fine-tuned, the weights of this model can be transferred onto a small embedded device in place of the gyroscopic sensor and then used to mimic the behaviour at a lower energy cost. As mentioned earlier, for smart devices and wearables, especially those used for commercial health monitoring, it is essential to the consumer's quality of life that the battery efficiency of these devices continue to be optimized. Theoretically, omitting the used of the gyroscope within the device and using our embedded neural network in

978-1-6654-2180-5/21/$31.00 ©2021 IEEE

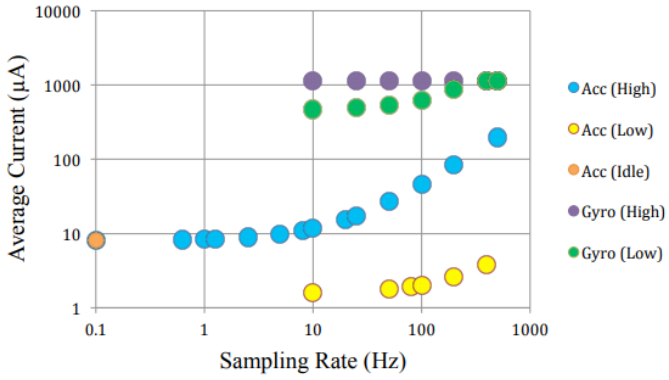

Fig. 1. Power consumption of MEMS IMUs

Representation	Pre-Processing				
Statistical	✗	✗	✗	✗	✓
Accelerometer	✗	✗	✓	✓	✓
Gyroscope	✗	✗	✓	✓	✓
Accel + Gyro	✗	✗	✓	✓	✓
Euler Angle	✗	✓	✓	✓	✓
Temporal Vector	✗	✓	✓	✓	✓
Spatial Vector	✓	✓	✓	✓	✓
	Space Domain Conversion	Orientation Alignment	Per-Axis Mean/STD Normalization	Principle Component Analysis Reduction	Per-Feature Mean/STD Normalization

Fig. 2. Data Representations and their Applied Pre-Processing Steps

its place maximises the usage of these devices making them more practical to use and improves the user experience.

Telehealth is one area in which the energy-efficient derivation of orientation vectors is likely to be a great boon. The commercialization of smart devices like smartphones and watches has given more people access to their health, such as blood pressure, heart rate, and other measurements of physiological function. Motion is another facet of health that can be tracked using smart devices. The sensors in these devices can track and record motion and orientation during physical activities. Access to this information can prove very useful in medical service such as physical rehabilitation, especially during the current global pandemic. Various settings and situations (e.g., due to COVID-19) can restrict access to medical practitioners or doctors. With people opting or requiring remote access to goods and services, fitness information can be electronically communicated to a physio-therapist who can then assist a patient from the comfort of their own home. We next provide related work followed by a description of the data collection process. We then describe the Recurrent Neural Network in detail and the methodology used to clean the data. Finally we provide our results and conclude.

II. Related Work

Cloud computing has become very popular by giving users a high performance, low-cost environment to perform computing tasks. In 2018, Akbar et al. [1] developed a mobile cloud hybrid application to leverage cloud computing. In their method, Akbar et al. considered two mobile cloud hybrid Android applications then executed them using a simple mobile cloud application framework while measuring the power and bandwidth consumption. From the results of this study, Akbar et al. found that by using their technique, the hybrid applications consumed approximately 63% less energy at the cost of 1MB of network bandwidth.

Also, in 2018, Patonis et al. [3] aimed to provide a fusion method for combining outputs from low-cost sensors for use in mobile devices. This method takes data from sensors such as an inertial accelerometer, a gyroscope and a non-inertial magnetometer. It then combines them into an optimal three-dimensional orientation vector output in real-time. It produces this value through the combination of Euler angles and a rotation matrix. This study considered the computational cost required to run on mobile devices, and they evaluated their fusion method's efficiency in an augmented reality application that was executed on an Android mobile device.

III. Collected Data

The data used in this project was collected by an android phone mounted on the left bicep using an elastic armband. From this accelerometer, magnetometer gyroscope and orientation data were collected at 50Hz. Participants executed ten exercises in addition to non-exercise activities, which worked as the source of the motion data in the IMUs [4]. The orientation data is the information we use to determine a position in 3D space and was in the format of Euler angles. Euler angles are three angles used to describe the orientation of a rigid body with respect to a fixed coordinate system [5]. Other critical numbering systems for this project include quaternions and rotation matrices.

Quaternions are a number system that extends complex numbers. They are applied to mechanics in three-dimensional space, typically to convey rotation. Since quaternions extend complex numbers, this means it implements real and imaginary numbers. A rotation matrix is a matrix used to perform a rotation in Euclidean space, such as a rotation in two-dimensional space through an angle. These numbering systems become important because using Euler angles to train the model is not the best approach to teaching the model.

IV. Neural Network Model

After making transformations and using the corrected data, a neural network accepts the input from the sensors and produces the orientation vector that the smart device will interpret. In this paper, we considered a RNN architecture. The intuition behind an RNN is that the neural network accepts input, applies weights to the input, produces an output and then uses values from processing in the hidden layer of the neural network for the next step of execution. This

basically gives context to the next step from the previous step, which is why this model is typically used for problems that involve time because the values from the sensors are taken at a specific frequency [6]. Aside from vanilla RNNs, other RNN models possess mechanisms that allow them to recall information over longer periods as well as to discard information as needed. To get the most accurate values from the model, we experimented with different hyper-parameters and architectures, altering training factors of the model and implementing different RNN architectures such as Long Short Term Memory Networks (LSTMs) and Gated Recurrent Unit Networks (GRUs)

V. Data Preprocessing

After parsing raw data files exported from Android and the action boundary file containing the start and stop times of each labeled action, a window size is determined based on the longest average action length. The time-domain window size is calculated to be 167 samples, which at a 50Hz sampling rate, results in a 3.3 seconds window. Space domain sampling is configured for $\theta = 5$, and the window size calculated to be 51 samples or 255of rotational movement. Non-action data is generated by random sampling windows which share no common elements with known action boundaries. In order to get a comprehensive non-action set, K-means clustering is used to remove similar non-actions, thereby increasing the variety of non-action samples presented to the classifiers. The total number of non-actions collected is equal to the average number of actions within an action class, and hence all classes have an approximately equal number of samples.

A. Orientation Alignment

In the context of action classification, the direction action performed has no bearing on its class. An action performed North or West are equivalent if performed identically. An orientation alignment is performed to the Euler angle, temporal vector and spatial vector representations. For the Euler angle representation, the window is rotated around the world vertical axis such that the yaw of the center sample is equal to 180. For temporal and spatial vector representations, the window is rotated around the vertical world axis such that the projection of the center sample y-vector onto the horizontal world plane is parallel to the world north vector. More intuitively, the window is centered such that the arm is pointed northward at the window halfway point.

B. Per-Axis Normalization

Before Principal Component Analysis (PCA) can be applied, input data needs to be appropriately normalized. Different sensor data, such as accelerometer or gyroscope data, are scaled differently because they are recorded in different units. Since PCA ranks based on input variance, having one sensor with large variance and another with low variance will result in one sensor dominating the other. Each data representation component is independently normalized to zero mean and unit standard deviation using statistics collected from the training/validation data files. For Euler angle, temporal and spatial vector representations, orientation alignment is first performed and random windows over all training/validation data files are collected from which Mean and Standard Deviation values are calculated. All other representations use the Mean and Standard Deviation over all samples of the training/validation data files.

TABLE I
Representation Window Dimension and Variance after PCA

Representation	Width	Depth	%Variance
Accelerometer	167	3	90.0
Gyroscope	167	3	61.3
Accelerator+Gyroscope	167	6	87.3
Euler Angle	167	3	90.4
Temporal Vector	167	6	88.0
Spatial Vector	51	6	89.7

C. Principal Component Analysis

Though there are many methods to perform dimension reduction. PCA is chosen for its wide use, not just in action classification but also in other machine learning applications. PCA uses an orthogonal transformation to map a set of input observations to a set of linearly uncorrelated variables called principal components. The largest principal components account for the largest variance in the original data and hence contains the most information. A transformation matrix is calculated per representation by performing dimension reduction over a collection of randomly selected windows across the training/validation files. In addition to reducing the dimension of the window data, PCA standardizes all representations to a fixed 12 dimensions. This allows for an even comparison since all classifiers are trained using the same number of features for all data representations. Figure I summarizes the window width, the number of components in a representation (depth) and retained variance after PCA for each representation.

D. Per-Feature Normalization

Following dimension reduction, a final normalization is performed such that all features across the training and validation sets have zero mean and unit standard deviation. For most classifiers, per-feature normalization increase learning performance by preventing uneven contribution of features. The per-feature normalization is calculated across all examples in the training/validation data and later applied to the test set.

VI. Methodology

A. Data Collection and Cleaning

Firstly, sensory and orientation data was collected from a dataset of 11 persons performing ten repetitions of 10 different exercises recorded by a bicep mounted smartphone over a recorded period of time. The aim of the project was to predict accurate orientation vector values only using accelerometer and magnetometer data, and so the values from these two

sensors and the Euler angle output, which is responsible for describing orientation, were extracted from the dataset. Using this orientation representation for training the model would not have been optimal for learning, so the Euler angle data was first converted into a quaternion by use of trigonometric functions [7]. Due to the fact that quaternions utilize a complex number system, this would have also produced inaccurate results, so the data was then converted to a rotation matrix [8]. After transforming the orientation data into a useful format, the magnetometer data was calibrated. This is due to the hard and soft iron distortion the magnetometer sensor may have experienced when the readings were taking place [9].

B. Neural Network Model

Once all the data was in a usable state, the six value vector was fed into a simple RNN to begin training. This architecture of neural networks was chosen because it is important to include context for the sequence of readings from the IMUs, which describe motions that happen in temporal space. Without accounting for time, the network will not fully understand the sequence of movements that make up the motions and will not learn as expected. With varying batch sizes, learning rates and dataset sizes, the model was trained, and the mean squared error loss along with the values used to obtain that loss value was recorded for further analysis.

Aside from changes in model values, different approaches were taken to be as accurate as possible, utilizing more complex RNN architectures such as the LSTM or GRU. Another method used to increase accuracy was partitioning the data in such a way that the model received data in a window slider fashion. Initially, the data was given in batches with one batch of a set size containing values from one example to another and the following batches of the same size continuing from after the last example of the previous batch. Using the window slider method, the following batch takes its first example from the second example of the previous batch. After processing the training samples, it then applies the learned weights to foreign testing set to compare how well the model works. This value is then compared against the training accuracy over every epoch to evaluate how well the model is learning.

VII. Results

After testing multiple values and architectures, the loss for the RNN is shown in TableII and for the Gated Recurrent Unit in Table III. Note that loss was calculated using mean squared error against the true values. A batch size of 20 was used for all tests and the data was fed in discrete batches where no two batches in an epoch contained the same values. The loss as a function of number of epochs is provided in Figures 3 and 4.

After running these tests, the window slider approach to inputting the data in the model was implemented with the losses provided in Tables IV and V. The corresponding values for loss as a function of number of epochs is provided in Figures 5 and 6.

TABLE II
Loss from Simple Recurrent Neural Network

		Learning Rate		
		0.01	0.05	0.1
Batch Size	50	0.01172036	0.00376974	0.0027762
	100	0.02107552	0.01304974	0.00651864

TABLE III
Loss from Gated Recurrent Unit

		Learning Rate		
		0.01	0.05	0.1
Batch Size	50	0.00766621	0.00664519	0.00408696
	100	0.00710135	0.00751824	0.00313041

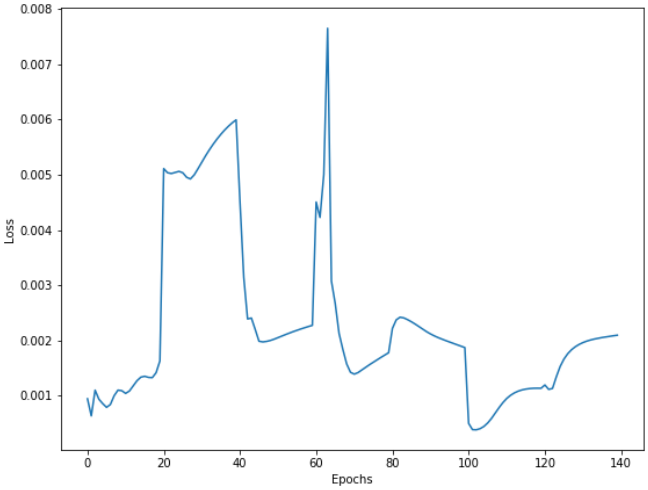

Fig. 3. Loss from RNN using Optimal Parameters

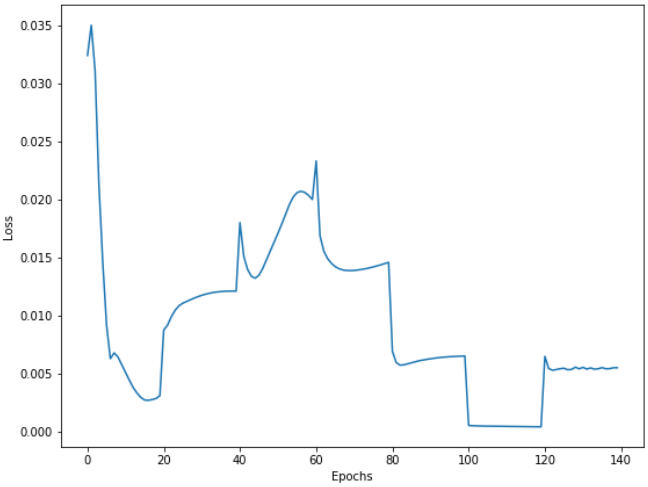

Fig. 4. Loss from GRU using Optimal Parameters

VIII. Discussion

GRU generally outperforms the LSTM and so values for LSTM were omitted from the paper. Looking at the results of the RNN and GRU from the first tests, using discrete

TABLE IV
LOSS FROM SIMPLE RNN WITH WINDOW SLIDER

		Learning Rate		
		0.01	0.05	0.1
Batch Size	50	0.00039917	0.00040634	0.00029641
	100	0.00027974	0.00035158	0.00033473

TABLE V
LOSS FROM GATED RECURRENT UNIT WITH WINDOW SLIDER

		Learning Rate		
		0.01	0.05	0.1
Batch Size	50	0.00049255	0.00036113	0.00044662
	100	0.00031506	0.00024084	0.00028324

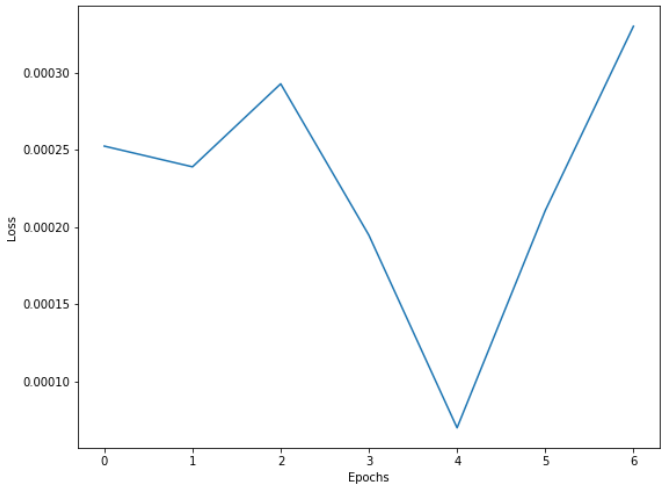

Fig. 5. Loss from RNN using Optimal Parameters and Window Slider

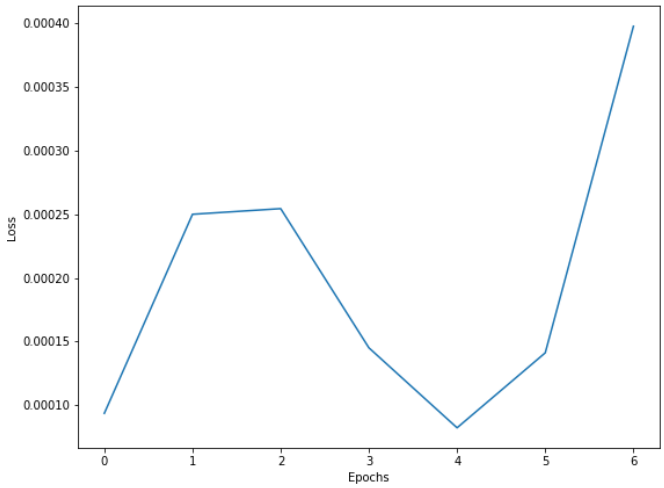

Fig. 6. Loss from GRU using Optimal Parameters and Window Slider

batches, it was expected that the GRU would outperform the RNN. This was assumed because the GRU is a more complex architecture built on the base RNN architecture, but the results obtained showed that the models did not seem to differ by a significant amount. Furthermore, after implementing the window slider method, there was a significant increase in accuracy, decreasing the loss by approximately 90%. In the case of processing and categorizing a motion, we observe the motion in a continuous motion taking context from each small movement as opposed to observing it in discrete or distinct batches of movement. Furthermore, from the graphs showing a loss, it was observed that there was a drastic increase in loss. To address this, the model may have to be prematurely ended to avoid the loss inaccuracy. Alternatively, the model could be set to train longer, and the results can be observed if additional training could be of benefit.

IX. CONCLUSION

We investigated whether one can decrease energy consumption in a smart device by using fewer sensors together with a Recurrent Neural Network. We showed that we could in fact get similar performance with fewer sensors together with a RNN. Given these results, it is shown that the model has learned to some degree of accuracy, more so using the window slider method. Theoretically this means that we can obtain a model that can mimic the behaviour of the gyroscopic sensor within a range of accuracy and thus implement an embedded neural network that can perform the gyroscope's function while operating at a reduced energy cost. Once implemented as a physical solution, the battery usage of the embedded device should be recorded. Given that the usage is significantly lower than the gyroscopic sensor, the model can then be retrained with different methods to attain a more accurate model so the device can perform its tasks efficiently. Some possible avenues to continue the work of this project can be to implement the Long Short Term Memory model architecture and measure its accuracy against the Gated Recurrent Network model.

REFERENCES

[1] A. Akbar and P. R. Lewis, "The importance of granularity in multiobjective optimization of mobile cloud hybrid applications," *Transactions on Emerging Telecommunications Technologies*, vol. 30, no. 8, p. e3526, 2019.
[2] Q. Liu, J. Williamson, K. Li, W. Mohrman, Q. Lv, R. P. Dick, and L. Shang, "Gazelle: Energy-efficient wearable analysis for running," *IEEE Transactions on Mobile Computing*, vol. 16, no. 9, pp. 2531–2544, 2017.
[3] P. Patonis, P. Patias, I. N. Tziavos, D. Rossikopoulos, and K. G. Margaritis, "A fusion method for combining low-cost imu/magnetometer outputs for use in applications on mobile devices," *Sensors*, vol. 18, no. 8, p. 2616, 2018.
[4] N. Hosein and S. Ghiasi, "Wearable sensor selection, motion representation and their effect on exercise classification," in *2016 IEEE First International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)*, 2016, pp. 370–379.
[5] L. C. Biedenharn and J. D. Louck, "Angular momentum in quantum physics: theory and application," *Addison-Wesley*, 1981.
[6] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing*. Ieee, 2013, pp. 6645–6649.
[7] J. Diebel, "Representing attitude: Euler angles, unit quaternions, and rotation vectors," *Matrix*, vol. 58, no. 15-16, pp. 1–35, 2006.
[8] C. W. Wong, *Introduction to mathematical physics: Methods & concepts*. OUP Oxford, 2013.
[9] K. Winer, "kriswiner," 2021. [Online]. Available: https://github.com/kriswiner/MPU6050/wiki/Simple-and-Effective-Magnetometer-Calibration