GESTURE RECOGNIZATION VIA TINYML

A (Thesis/Project/Dissertation)

Presented to the

Faculty of

University of North Carolina,

Greensboro

In Partial Fulfillment

of the Requirements for the Degree

Master of (Arts/Science)

in

Computer Science

by

(Amulya Yadagani)

(December 5th, 2020)

ABSTRACT

Powerful computations involving machine learning and deep learning with minimum cost and power usage along with maintaining data privacy is made achievable through microcontrollers. In this research, the Arduino nano 33 BLE sense is used as a microcontroller to predict human gestures by developing an end to end edge computing application. The Arduino board is used to capture gestures data, rapidly with a sampling rate of 119hz and then using the trained TensorFlow Lite model on the board to predict the gestures. TensorFlow Lite model is developed using Keras deep Learning sequence model having 96% accuracy. Along with the embedded sensors on the board, the Arduino IDE eased programming on the board to capture and classify gestures. Transfer of the data from the board to the system is achieved through Bluetooth connectivity enabled on the Arduino board and by using an open-source GATT library called BLEAK. Microcontrollers like Arduino boards can be an alternative to machines with high computing power to run complex machine learning and deep learning models without sacrificing accuracy and with a minimum cost as it is independent of expensive hardware or reliable internet connections. Additionally, since the form factor of these devices is tiny and with embedded sensors, it can be used as a wearable device and therefore used in building multiple real-world applications.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my research Professor Dr. Somya Mohanty for giving me the opportunity to work on this wonderful project of Arduino Board and for his guidance throughout the project. During my research for the project I have learned lot of interesting new things which I believe will be very helpful in my future endeavors.

TABLE OF CONTENTS

CHAPTER ONE

INTRODUCTION

1. Introduction

The wide popularity of machine learning algorithms has increased due to its ability to do cheap computations on large amounts of data and one might think this requires big computers with fast CPUs and GPUs, big RAM size, or running algorithms on the cloud. However, imagine performing all these computations on a microcontroller powered by a single coin cell battery with only a few kilobytes of memory. This is made possible by some of the microcontrollers like Arduino Boards, Raspberry Pi, NVIDIA. Some of the advantages of using microcontrollers are low energy consumption, low cost, flexibility, and privacy. As part of my research, I have chosen the Arduino BLE nano sense board to collect gestures data using the sensors on the board and finally to classify gestures by using the trained deep learning TensorFlow Lite model that is loaded on to the board.

1.1 Arduino nano 33 BLE sense

The name of the board gives us an abundant description and a lot of special features the board can provide. The "nano" represents the board's tiny form factor having 45x18mm dimensions, "33" represents that board is operating on 3.3V. The "BLE" indicates the unique feature among microcontrollers which the board provides us Bluetooth pairing via NFC. The "sense" indicates that it has 9 on-board sensors like accelerometer, gyroscope, magnetometer. Finally, the main feature of this board is it is AI-enabled and gives us the possibility of running Edge Computing applications (AI) on it using TinyML.

Board Hardware has a powerful Nordic nRF52840 processor that contains a 32-bit ARM Cortex M4 CPU and NINA B306 module for Bluetooth communication. It includes sensors, LED lights, pushbuttons, micro-USB connector.

Board Software provides us with an IDE, which makes the programmers easy to set up, write, and compile the code on to the board. Sensors on the board can be used by installing libraries into the IDE such as "LSM9DSI" provides functionality related to Accelerometer, Gyroscope, Magnetometer sensors.

As part of my research to collect data for human gestures like "Squat", "Jump", "Walk", "Run", "Other", the "LSM9DSI" library is used to collect accelerometer and gyroscope data for x, y, z-axis. Also, the "ArduinoBLE" library is used to send and receive data through Bluetooth from the board to the laptop.

1.2 BLEAK Library

BLEAK acronym for Bluetooth Low Energy platform Agnostic Klient is a python library that is used in discovering, connecting, reading, writing, and getting notifications from BLE devices. It is an open-source, asynchronous, cross-platform GATT client software where the BLE devices act as GATT servers. The connection is established by discovering the BLE device through its Bluetooth address or UUID in the case of Mac OS, its GATT services, and its characteristics. In this research work, the BLEAK library acts as a peripheral device by connecting to the Arduino board that acts as a central device. On establishing the connection, the training data captured on board is sent to the peripheral through notifications and also it is further used in reading the classified data.

1.3 Google Collaboratory

It is an interactive environment to execute python code as well as the rich-text in the same document (notebook). It requires zero configuration to set up an environment and also provides free GPU. Therefore, one doesn't have to worry about having a specially equipped computer. As part of this research work, the google colab notebooks are used to execute python code to

preprocess the training data, visualize, train the deep learning Keras model, prune and quantize the model, and finally convert the Keras model to the TensorFlow Lite model to load onto the board.

1.4 Research Overview

The Arduino BLE sense nano board is used to capture the human gestures accelerometer and gyroscope data using the sensors with 119Hz as sample rate. The BLEAK library is used to read the data from the board through Bluetooth. The captured training data is used in training the Keras sequential model. This model is converted to the TensorFlow Lite model, has an accuracy of 96% after pruning and quantization. Finally, the model is loaded onto the board and is used for the classification of gestures. The recognized gesture having the highest probability is identified by the LED Color on the board and also the result is sent through Bluetooth.

CHAPTER TWO

METHODS

2. Methods

The deep learning sequential model is used for the classification of 5 gestures. Model is built using neural networks, where the input fed into the neural network is processed in hidden layers using weights that are adjusted during training. The trained model is used for predictions.

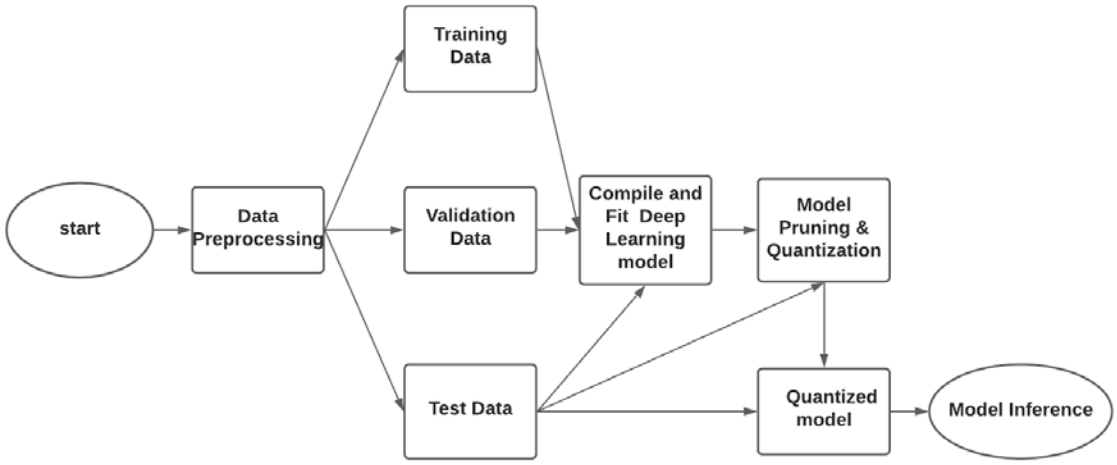

Figure 1. Deep Learning Workflow

2.1 Data Collection

The data for 5 gestures - "Squat", "Jump", "Walk", "Run", "Other", is collected using the Arduino Board. The category "Other" is used to classify any other gesture that is not "Squat", "Jump", "Walk" or "Run". Including "Other" increased the accuracy of the model. The board is programmed to capture the 3-axis accelerometer and a 3-axis gyroscope data using the inertial sensors (IMU) provided by the board. Also, the board is programmed to act as a central device where it sends the captured data via Bluetooth to the execution environment of the Bleak library that acts as a peripheral device using 16-bit String Characteristic UUID. The programmed board is then worn as a wearable just above the knee sideways. After establishing a connection between

central and peripheral, the board emits an orange LED. Each gesture is performed 100 times with a sampling rate of 119Hz. The board only captures the data if it has a minimum acceleration threshold of 1.5.

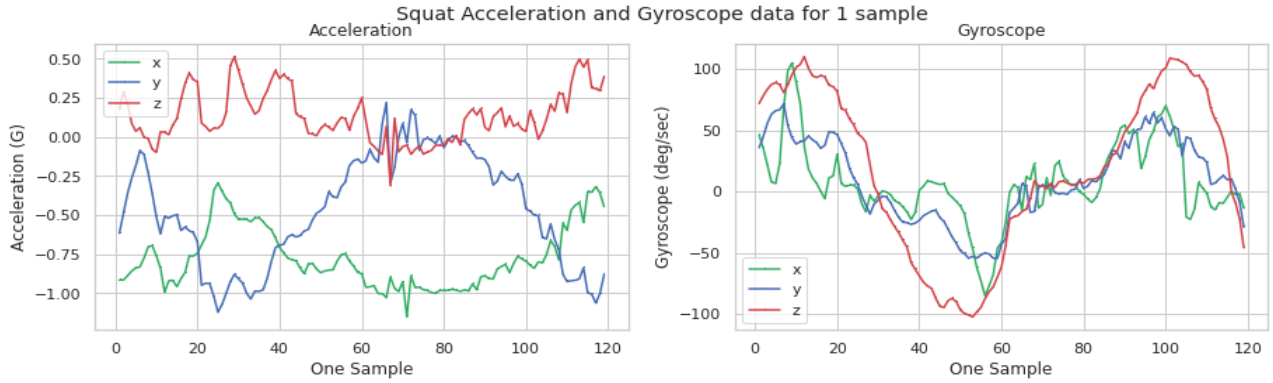

Figure 2. Squat Acceleration and Gyroscope data for 1 sample

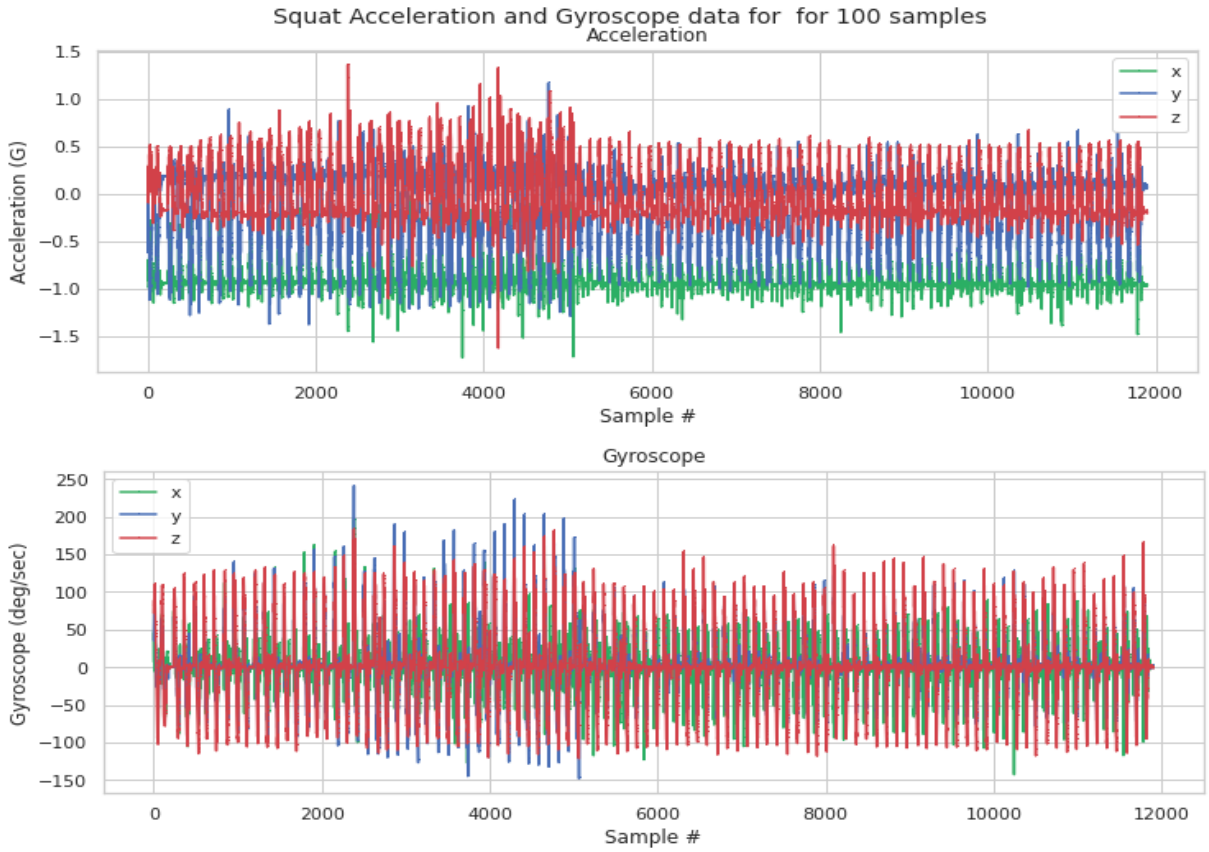

Figure 3. Squat Acceleration and Gyroscope data for 100 samples

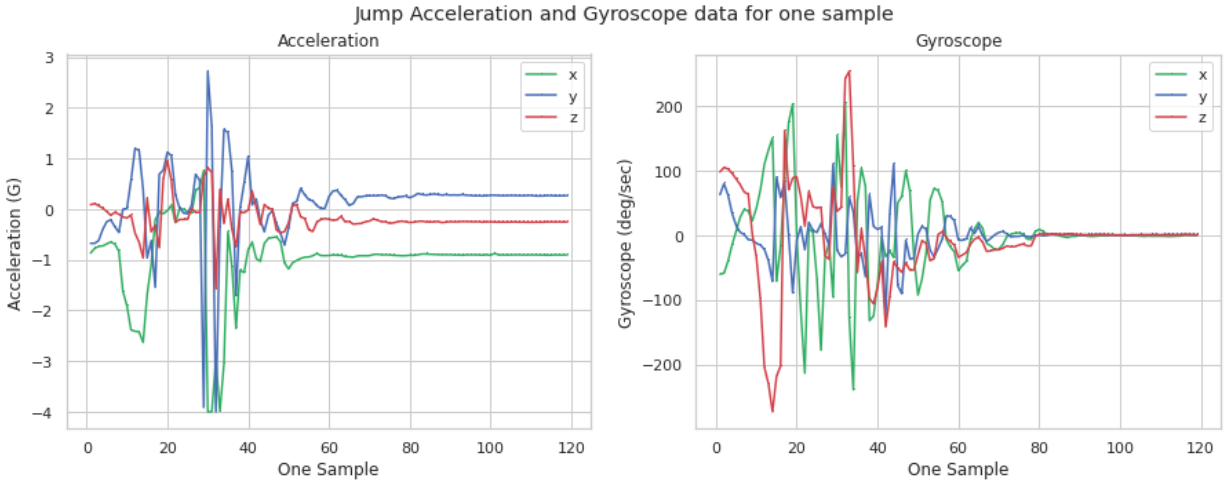

Figure 4. Jump Acceleration and Gyroscope data for one sample

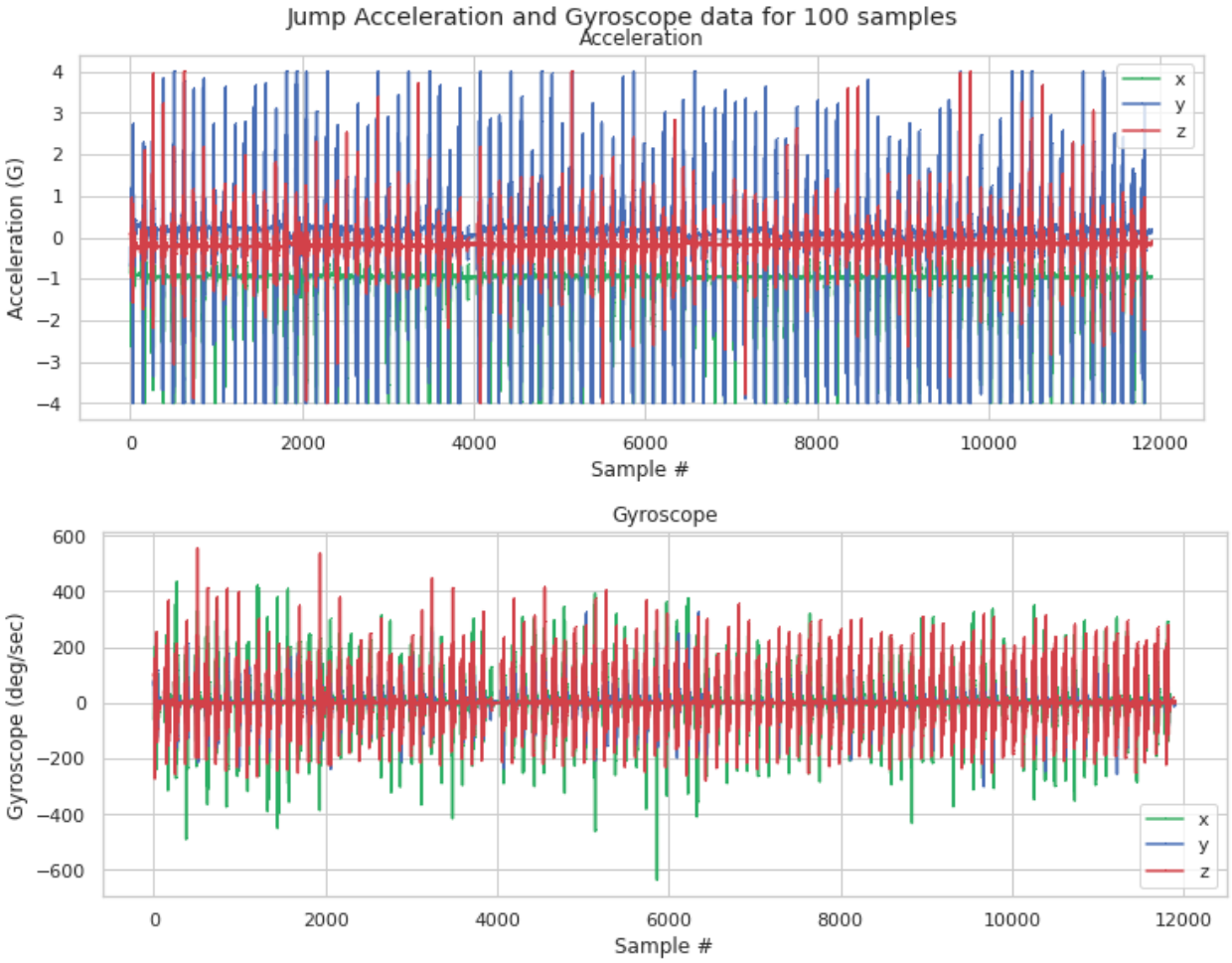

Figure 5. Jump Acceleration and Gyroscope data for 100 samples

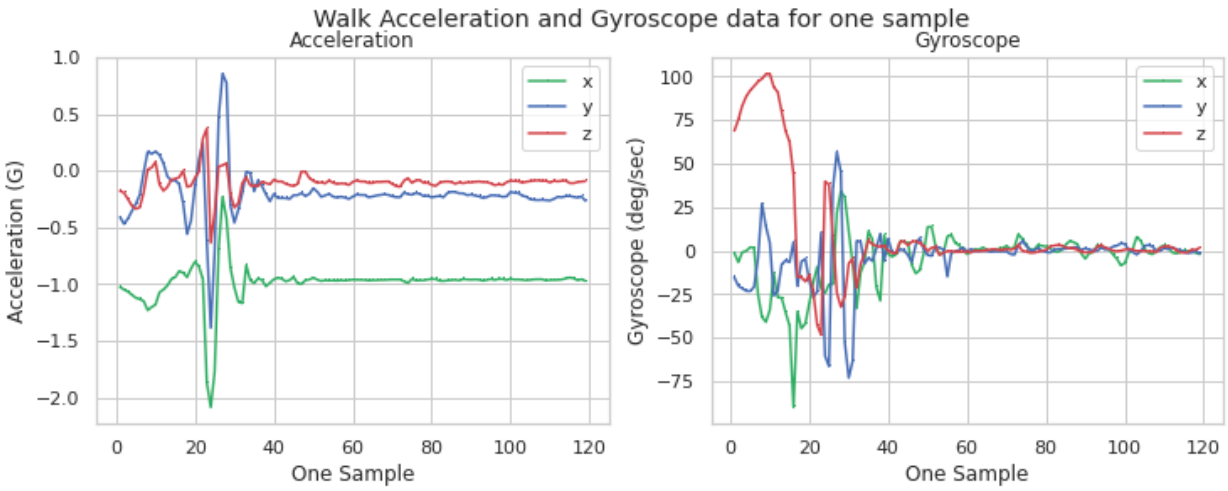

Figure 6. Walk Acceleration and Gyroscope data for one sample

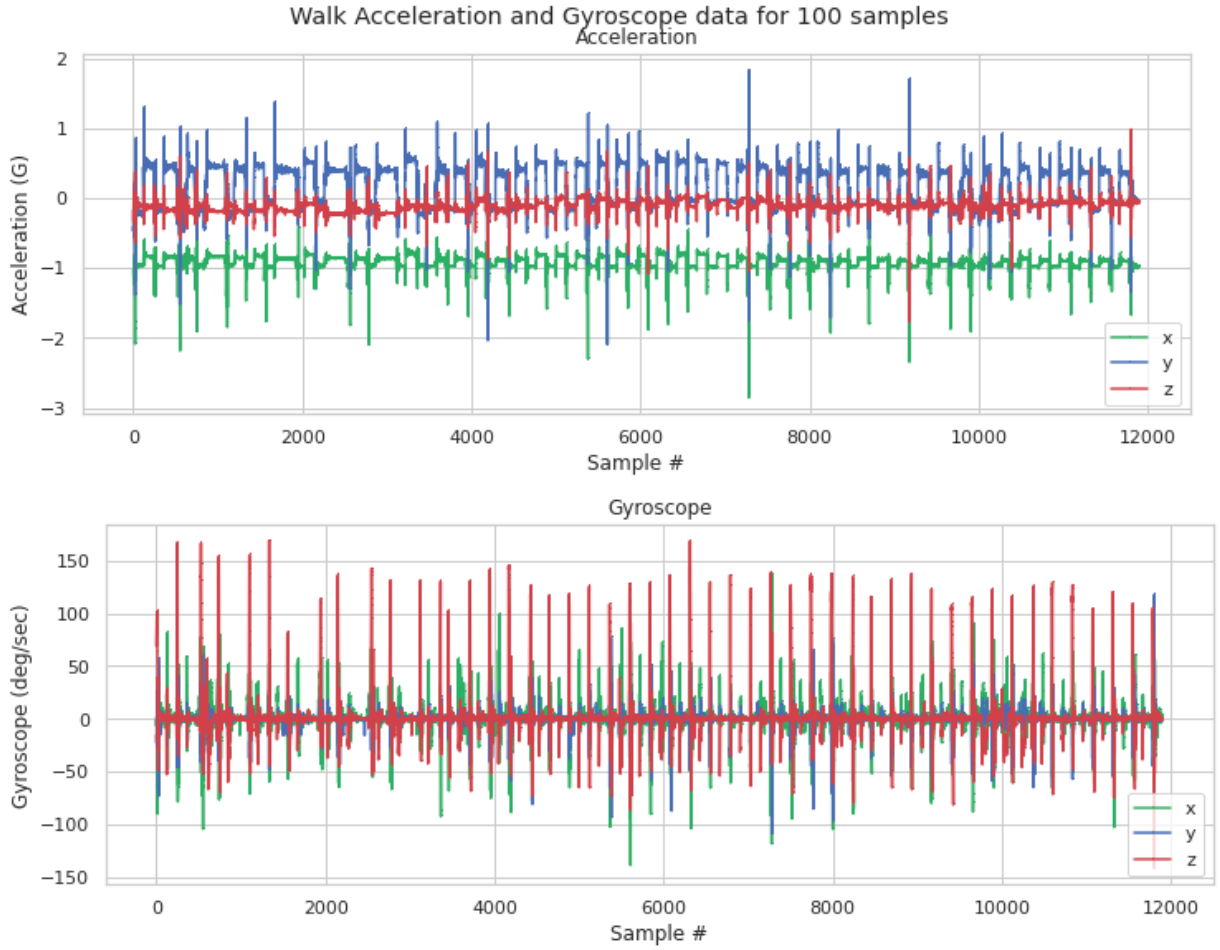

Figure 7. Walk Acceleration and Gyroscope data for 100 samples

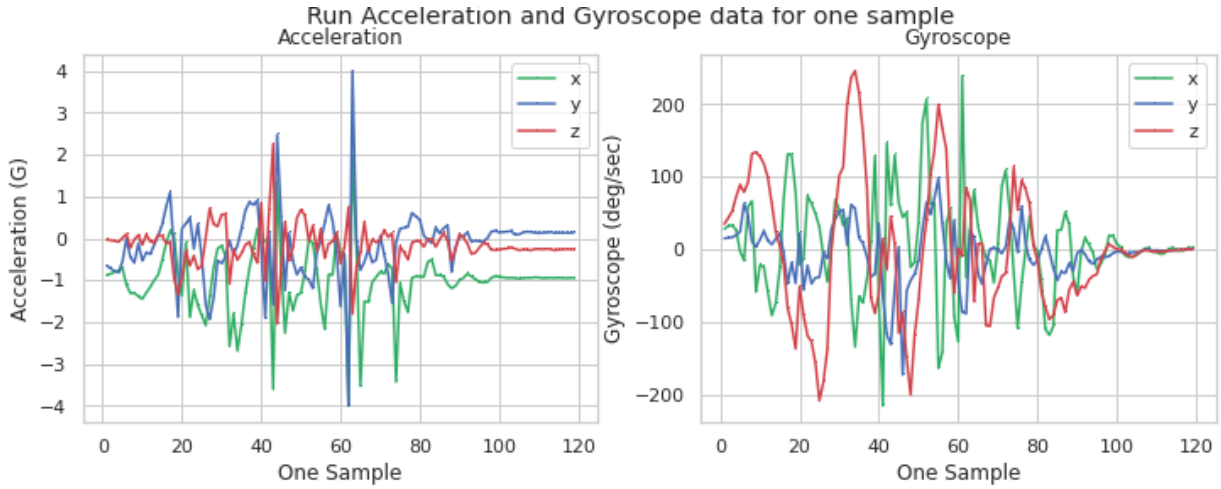

Figure 8. Run Acceleration and Gyroscope data for one sample

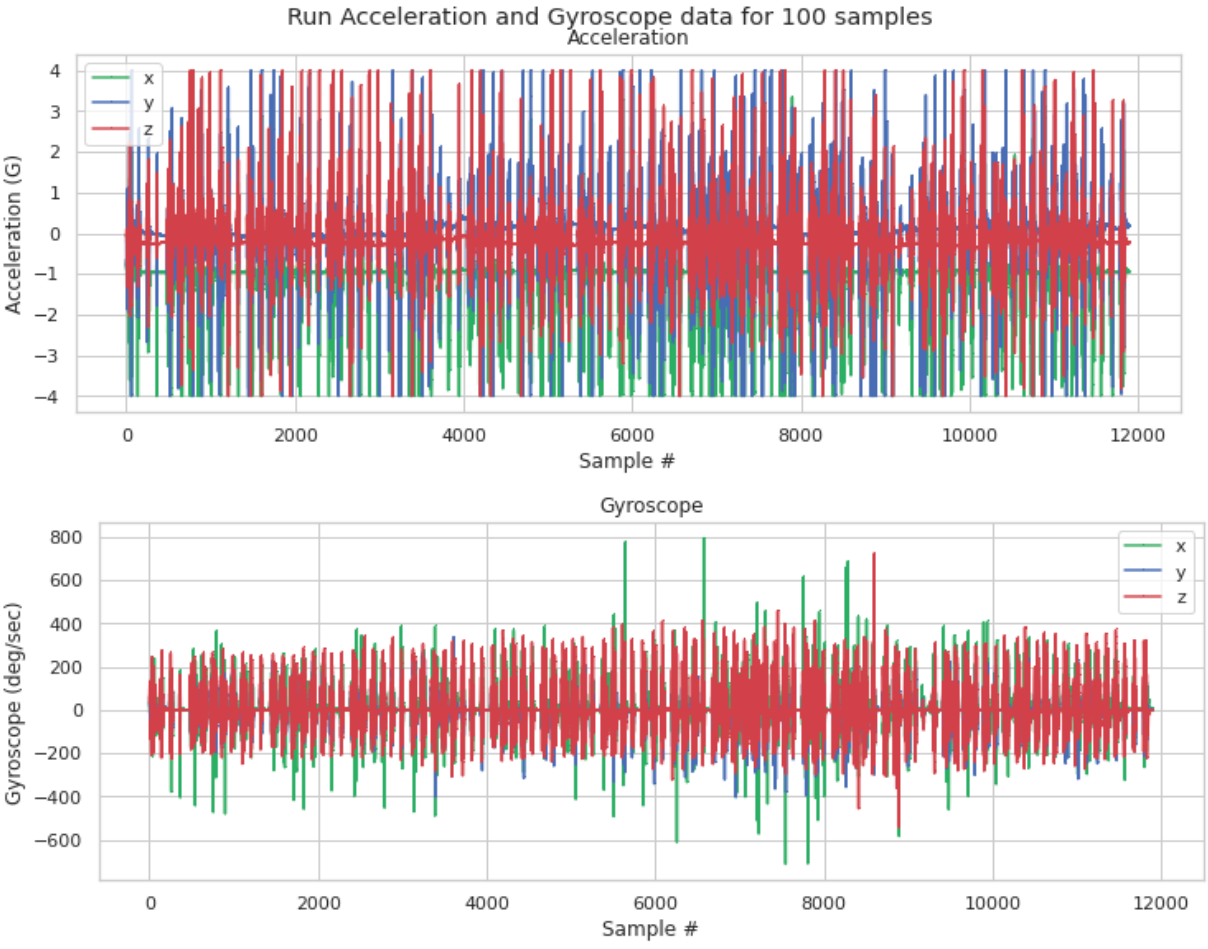

Figure 9. Run Acceleration and Gyroscope data for 100 samples

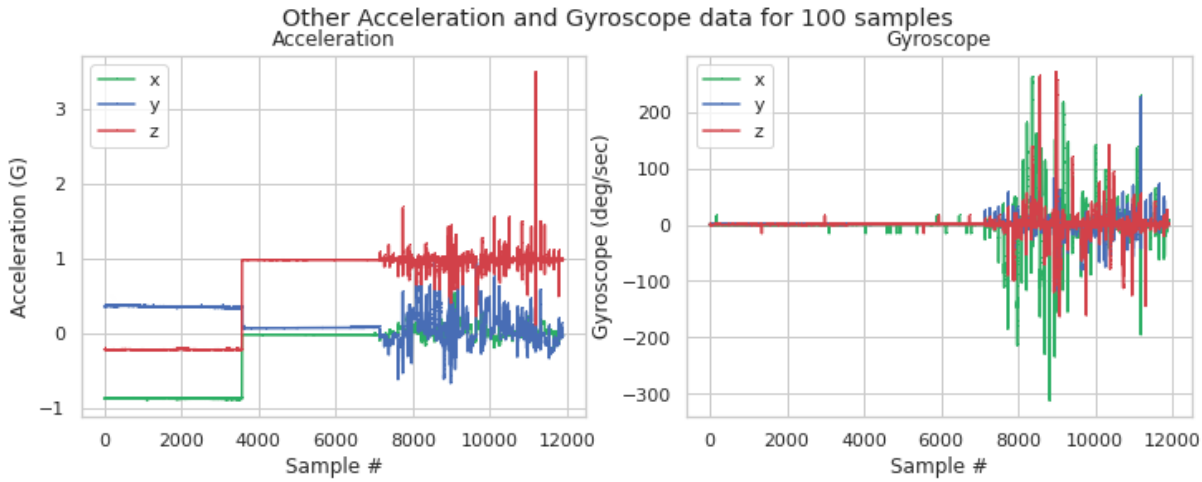

Figure 10. Other Acceleration and Gyroscope data for 100 samples

2.2 Data Preprocessing

Deep learning models accept input and generate output in the form of tensors. A tensor is a list that can contain either numbers or other tensors similar to an array. To transform the input into tensors, a multi-dimensional tensor is created having a shape (500, 716). 500 indicates 100 samples for each of 5 gestures and 716 represents the 3-accelerometer (ax, ay, az) and 3-gyroscope data with a sampling rate of 119 per gesture that is 119 x 6 = 716. The output tensor has a shape (500, 5) where 500 represents 100 samples for each of 5 gestures and 5 indicates the one-hot encoded matrix for each of 5 gestures. The input data is normalized between 0 to 1.

2.3 Train, Test and Validation Sets

The input and output data is shuffled into random order so that all the gestures data is evenly distributed into training, testing, and validation sets. The shuffled data is split into 60% training, 20% test, and 20% validation data.

2.4 Building Sequential Model

The Deep Learning sequential model is built layer by layer in Keras. Each layer has weights that correspond to the layer that follows it. The ‘add()’ function is used to add 7 layers to our model, among them, 4 are "Dense" layers and 3 are "Dropout" layers following the first 3 dense layers. Dense is a layer type that connects all nodes in the previous layer to the nodes in the current

layer. There are 100 nodes in the first input layer, 75 nodes in the second input layer, 75 nodes in the third input layer, and 5 nodes for each gesture in the output layer. The Dropout layer is used to prevent the model from overfitting. An activation function is responsible for transforming the summed weighted input from the node into the activation of the node or output for that input. Here, the rectified linear activation function (ReLU) is used that outputs the input directly if it is positive or it will output zero.

2.5 Compiling and Fitting the model

Compiling the model takes 3 parameters - optimizer, loss, metrics. The optimizer controls the learning rate. While training the model, the "Adam" optimizer with learning rate=0.0001 was well suited in giving good accuracy and minimum loss. The mean_squared_error as a loss function which is an average squared difference between the predicted and actual values where the value closer to 0 means the better the model performed. To calculate the accuracy of the model it is given as a parameter in metrics.

The fit function of the model takes 6 parameters - train input, train output, epochs as 700, batch size as 10, callbacks, validation data. The "epochs" is the number of times the model will cycle through the data. The batch size is the number of samples to work through before updating the internal model parameters. To reduce the learning rate when a metric has stopped improving, "ReduceLROnPlateau" is used as a callback to monitor validation loss with a lower bound on the learning rate as 0.0001.

2.6 Model Pruning and Quantization

Model optimization techniques such as pruning and quantization are used to reduce the size of the model with minimal loss of accuracy. Pruning eliminates unnecessary weights by setting those values to zero by removing unnecessary connections between the layers of a neural network. The pre-trained model is pruned with an initial sparsity of 65% and ends with 90% sparsity for

450 epochs. The strip_pruning removes every "tf.Variable" that pruning only needs during training. The accuracy after pruning is 92%. The size of the pruned model is further reduced by quantizing the weights to int8 using the representative_data_gen method. By adding headers to the quantized model it is converted to the TensorFlow lite model to load it onto the Arduino board. The accuracy of the quantized model is 96%. The size of the model is reduced from 2.1Mb to 0.5Mb.

2.7 Classification

The Arduino board is programmed to use the trained TensorFlow Lite model for the classification of gestures. The board is programmed with static tensor size (40x1024), string characteristic UUID to transfer the output (highest probability gesture) to the bleak execution environment, and an interpreter to run the model. The input data is normalized between 0 to 1. After the establishment of a connection between the central (Arduino Board) and peripheral (BLEAK), an orange LED light is shown on the board. Once the board starts receiving gesture's accelerometer and gyroscope data with a minimum threshold of 1.5, an interpreter is invoked to run the inference. The output generates probability for each gesture, based on the highest probability among 5 gestures (Squat, Run, Walk, Jump, Other) a corresponding LED light (Red, Green, Blue, Magenta, White) is shown on the board and also transmitted over the Bluetooth to the peripheral.

CHAPTER THREE

RESULTS AND DISCUSSION

3. Results and Discussions

3.1 Deep Learning Model Results

Training the deep learning model initially with "Adam Optimizer" alone did not give good validation accuracy, the model was overfitting. Therefore, "ReduceLROnPlateau" callback is used to reduce the learning rate when a metric has stopped improving. Also, with the addition of the 5th gesture "Other" which contains random data, the model finally achieved good accuracy of 96% and validation accuracy of 91% with minimum mean-squared error loss for both training and validation data. These results also show that the model is not overfitting.

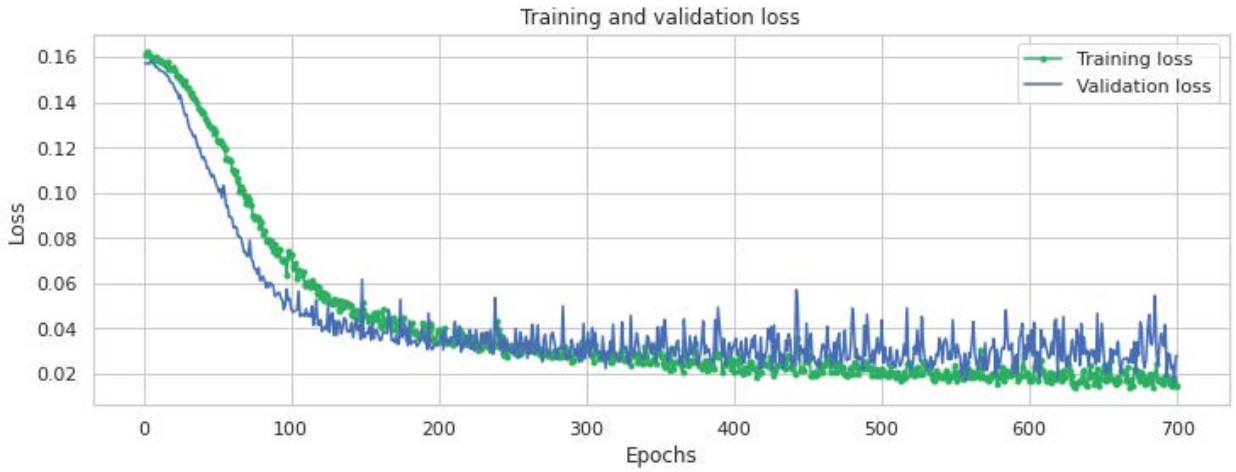

Figure 11. Training and validation loss of the trained model

From Figure 1, we can observe that the mean-squared error loss gradually decreases with value close to zero until 100 epochs and remains stable after training for 200 epochs. From Figure 2, we can see that the accuracy increased gradually with value close to 90% until 100

epochs and remains stable after that. Both the training and validation loss achieve the same results during the model training which shows that the model is not overfitting.

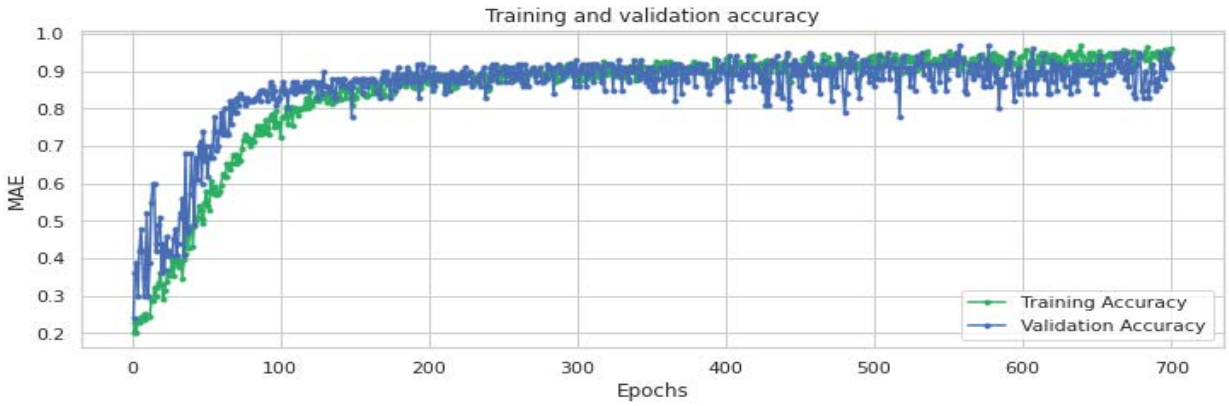

Figure 12. Training and validation Accuracy of the trained model

Figure 13. Confusion Matrix of the trained model

From Figure 3, The confusion matrix for the trained model explains that for gestures such as "Jump" and "Walk" there are few misclassifications. In total 8 gestures are misclassified during training. From Figure 4, The confusion matrix for the pruned model explains that for gestures such as "Jump" and "Walk" there are few misclassifications similar to the model before

pruning. In total 5 gestures are misclassified after pruning, which shows that the model did not lose its accuracy after pruning.

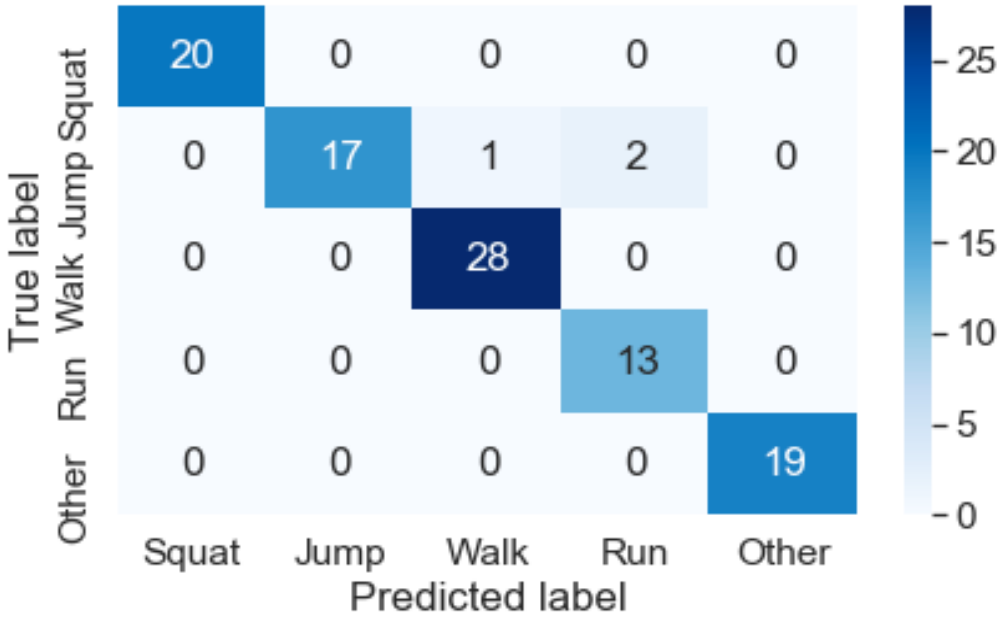

Figure 14. Confusion Matrix of the pruned model

3.2 Classification Results

The Arduino Board loaded with the TensorFlow Lite model classifies the gestures by giving the probabilities of all the 5 gestures. The gesture with the highest probability is identified based on the LED color on the board. Table 1 describes the corresponding LED color for each gesture.

Table 1. Gestures LED color

Gesture	LED Color
Squat	Red
Jump	Magenta
Walk	Blue
Run	Green
Other	White

CHAPTER FOUR

CONCLUSION

4. Conclusion

Using Microcontrollers for Deep Learning has several benefits over big computers with high CPU and power resources such as 'Low Energy Consumption' where it requires very little energy for processing power, memory, and storage due to its small size, its "cheap" to buy, "flexible" to use, and most importantly maintaining "privacy" of data. Along with these benefits that every microcontroller provides, the usage of Arduino BLE nano sense board gave us extra advantages such as the ability to use the sensors, establishing Bluetooth connectivity, and the possibility of running deep learning network models using TinyML. The use of Arduino IDE made it easier to program the board and visualize the captured data.

This research work shows us how simple it is to achieve an end to end application involving deep learning to predict human gesture with real data - The sensors on the board allowed us to capture training data with a sampling rate of 119Hz, TinyML for training deep learning sequential model that achieved 96% accuracy, and finally prediction of gestures by the board with the output in the form of LED color specifying particular gesture.

Based on this research work and the many benefits of the Arduino board, it can be concluded that though it possesses limited memory, it can replace machines with high computing power to perform complex machine learning and deep learning without sacrificing accuracy.

REFERENCES

1. https://www.arduino.cc/en/Guide/NANO33BLESense

2. https://blog.arduino.cc/2019/10/15/get-started-with-machine-learning-on-arduino

3. https://bleak.readthedocs.io/en/latest/

4. https://github.com/hbldh/bleak

5. https://keras.io/api/

6. https://www.tensorflow.org/model_optimization/guide/pruning/pruning_with_keras

7. https://www.tensorflow.org/model_optimization/guide/quantization/post_training