Playing Web Cricket game using Emulator and IMU sensors on Arduino Nano 33 BLE

Divyansh Sharma

April 2024

1 Problem Statement

Figure 1: Cricket Game

In the sport of cricket, fast bowlers are capable of delivering balls at speeds up to 160 km/h, creating extremely challenging conditions for batsmen. At such speeds, the ball reaches the batsman in approximately 200 milliseconds, demanding extraordinarily rapid reaction times for both effective play and safety. The skill to react swiftly is not only crucial for a batsman's performance—enabling them to score runs against fast deliveries—but is also integral to their safety. Tragic incidents have occurred where batsmen suffered severe injuries, including fatal ones, due to being struck by the ball. While protective gear such as

helmets can mitigate some risk, players still face the possibility of concussions and other serious injuries.

Despite the advancements in protective equipment, the primary defense against fast deliveries remains the batsman's ability to react and execute the correct shot timely. The necessity for enhanced training methods to develop and sharpen batsmen's reflexes is evident, especially in an era where the pace of the game continues to accelerate. Traditional training methods may not sufficiently simulate the speed and intensity of facing world-class fast bowling in real match conditions.

To address this gap, the proposed system utilizes a wearable device integrated with a machine learning model and a game emulation setup. This system is designed to classify and respond to cricket shots in real-time, creating a dynamic and interactive training environment. By emulating real-world conditions in a controlled setting, this technology aims to improve batsmen's reaction times and shot accuracy, thereby enhancing their performance and safety against high-speed bowling. This training tool provides a novel approach by allowing batsmen to experience and adapt to the challenges posed by extreme pace, without the physical risks associated with practicing in the nets against elite fast bowlers.

Figure 2: Australian Fast Bowler Mitchell Johnson Bowling at Pace of 161km/h

2 Solution

To address the challenges faced by cricketers in responding effectively to fast bowlers, this project introduces an innovative training solution that combines machine learning, wearable technology, and game simulation. The core of this system is a wearable device—a cricket glove equipped with the Arduino Nano 33 BLE Sense, which includes built-in accelerometer and gyroscope sensors. These sensors capture the motion data of the batsman's hand during various cricket shots, providing real-time input for analysis.

The wearable device is programmed to classify five distinct cricket shots: the cover drive, pull shot, on drive, square cut, and an idle position. This classification is achieved through a lightweight machine learning model developed

and optimized specifically for the Arduino platform, ensuring rapid processing to match the speed of real-time cricket gameplay.

Once a shot is identified, the system interacts with a web-based cricket game, Stick Cricket, through an emulator built using Python and Selenium. This emulator translates the detected shot type into corresponding key presses in the game, effectively allowing the player to control the game with physical movements. The integration also uses a Bluetooth connection to facilitate wireless communication between the Arduino and the emulator, enhancing the system's usability and responsiveness.

This setup not only simulates the experience of facing fast bowlers but also does so in a safe, controlled environment. By practicing with this system, batsmen can develop quicker reflexes and improve their shot timing and selection without the risk of physical injury. The technology provides instant feedback and allows for repetitive practice, which is essential for mastering the skills needed to face high-speed deliveries effectively.

Overall, this solution TinyML to create a realistic and safe training environment, preparing batsmen to excel against some of the fastest bowlers in cricket.

2.1 Sensors

For this project, I have depended on 2 sensors. The Arduino Nano 33 BLE is equipped with an LSM9DS1 IMU (Inertial Measurement Unit), which combines a 3D digital linear acceleration sensor, a 3D digital angular rate sensor (gyroscope), and a 3D digital magnetic sensor (magnetometer) into a single compact unit. This integrated sensor module provides nine degrees of freedom (DoF) by capturing precise motion in three dimensions: acceleration, orientation, and magnetic field direction.

The accelerometer and gyroscope functionalities are particularly useful in bat motion detection. The accelerometer measures the acceleration forces acting on the object it is attached to, whether due to motion or gravity, while the gyroscope detects the rate of rotation around the object's three axes. This capability makes it an excellent choice for projects involving dynamic motion analysis, such as fitness trackers, drones, and, as in this case, sports training aids for monitoring and classifying specific movements like cricket shots. Magnetometer does not provide any useful readings as it changes as per the external devices present.

The Arduino Nano 33 BLE uses the I^2C (Inter-Integrated Circuit) protocol to communicate with the LSM9DS1 IMU sensor. This protocol allows the Arduino board to connect with the LSM9DS1 using a serial data line (SDA) and a serial clock line (SCL).

Figure 3: Figure showing the Arduino placement

Figure 4: Data Collection of Square Cut Shot

2.2 Product Description

3 Data Collection

The project relies on 2 aspects. The first feature of the project is a multi-classification neural network that is able to classify cricket bat movement into one of 4 shots or IDLE state. To do this, a dataset needs to be built for training.

The dataset collection followed a long-term data collection plan. In this training plan, the Arduino board is connected serially to the computer. A training script is executed that reads IMU data at frequency of 100Hz(every 10ms). The script stores the IMU data locally. Upon start, a human wears the Arduino-glove and starts playing the same cricket shot again and again for a continuation of 10 mins. Therefore, each cricket shot is recorded for 10 minutes.

The data collection script also simultaneously computes the angular velocity

in x, y, z directions. This is done using the Karman Filter.

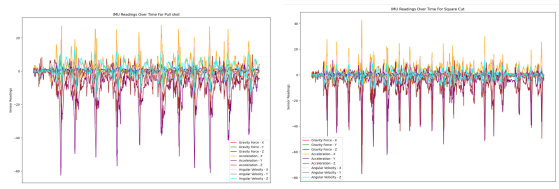

Figure 5: Pull Shot and Square Cut

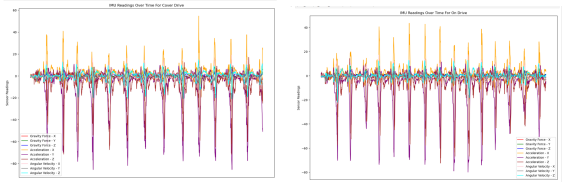

Figure 6: Cover Drive and On Drive

3.1 Data Preparation

3.1.1 Calculating average frequency of each shot

Within the 10 minutes of each shot, each shot was played with an average time period of 2.8s. This has been calculated using the discrete fourier transform and peak analysis in the python notebook. This makes the number of shots played to be about 24 shots every minute which translates to 240 shots collected in the entire 10 minutes.

3.1.2 Window Samples

Given that each shot is played at about every 2500ms, a window sampling algorithm is implemented to extract samples of each individual shot. More formally, a window of size 2500ms is passed through the 10 minute data. The window is shifted by 300ms to increase the number of samples taken of each window.

4 RNN: Time Series Data

Given that the data collected was a time-series data, the first intuition was to deploy an Long Term Short-Term Memory model(LSTM). LSTMs are an advanced form of Recurrent Neural Networks(RNN) that are designed to handle sequential data. The LSTM was connected to soft-max layer which classified LSTM inference to one of the shot categories.

```
model = Sequential([
    LSTM(64, input_shape=(X_train.shape[1], X_train.shape[2]), return_sequences=True),
    Dropout(0.5),
    LSTM(32),
    Dropout(0.5),
    Dense(y_train.shape[1], activation='softmax')
])
```

Figure 7: Model Structure

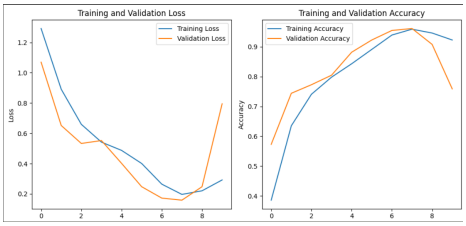

Figure 8: LSTM training Validation and Loss Curves Before Deployment

After training multiple epochs, a LSTM model was produced which had a high accuracy of 96%. However, when the model was deployed on the Arduino Board, it only produced an accuracy of only 78%. This was too low. Therefore, there is a

The performance was not too bad. However, after researching the machine learning models trained on IMU data, it was discovered that Feed Forward Neural Network Models that employed feature engineering produced better results. Typically this method involves extracting statistical measures from the accelerometer and gyroscope sensor data. Specifically, the notebook calculates statistics like mean, standard deviation, maximum, minimum, and signal magnitude area (SMA) from the time-series sensor data.

A very popular syntiant IMU deployed on edge impulse carries out signal processing to generate 1800 features. These models produce high accuracy as well. However, generating 1800 features quickly takes a lot of computing resources. Such an accuracy causes suffering in both time and space complexity.

Therefore, for the application of short detection a unique approach was taken to label the importance of features. The library **tsfresh** is a specially designed python library that automatically extracts relevant features like mean, variance, correlation and many ocmplex features, which are useful for sensor data analysis and time-series data analysis. Given a data sample, **tsfresh** can produce 700 features. However, not all features will be important. Therefore, in this project, the important features are evaluated and then used for model training.

5 Deployment of model into main Application

5.1 Stick Cricket and Selenium Automation

The aim of this project is to create a system capable of controlling the game play of the web-based cricket game "Stick Cricket" using data received from a

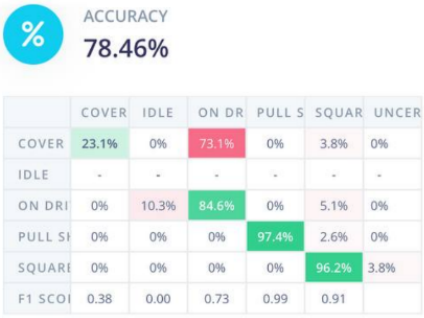

Figure 9: Confusion Matrix showcasing Model Performance on actual device

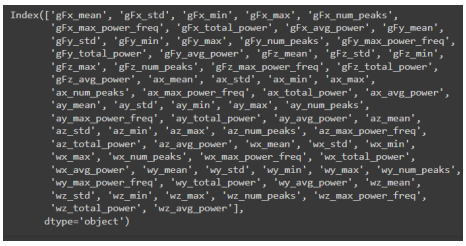

Figure 10: Important Features

serial port. By interfacing with a hardware device that provides input data, we enable real-time interaction with the game, allowing users to control the game play dynamically.

5.1.1 Implementation Details

The system interfaces with a hardware device connected to the host computer via a serial port (COM9). This device presumably captures relevant user input or sensor data, which is then transmitted to the computer for processing.

- **Serial Communication:** Python's serial module is utilized to establish communication with the hardware device. Data received from the serial port is continuously monitored for relevant information.
- **Web Automation:** The Selenium WebDriver library is employed to automate interactions with the Stick Cricket game running in a web browser. WebDriver allows programmatically simulating user inputs such as key presses to control the gameplay.
- **Inference and Decision Making:** Inference results received from the serial port, presumably related to shot classifications in cricket (e.g., cover drive, pull shot), are processed to determine the appropriate action to

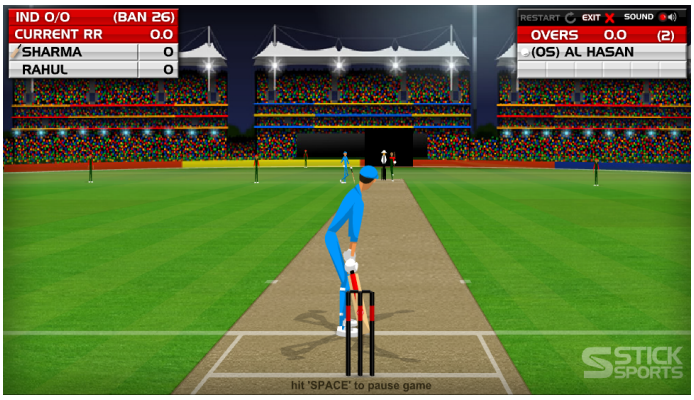

Figure 11: The cricketer must face the ball in 2.5s

take in the game. This decision-making process is implemented based on predefined rules mapping inference results to game controls.

- **Action Execution:** The determined actions are executed using Selenium's ActionChains, which simulate keyboard inputs corresponding to the desired gameplay actions (e.g., moving the player character left or right).
- **Real-Time Control** The system operates in real-time, continuously monitoring the serial port for incoming data. Upon receiving relevant information, such as shot classifications, it promptly translates this data into game actions, ensuring a responsive and interactive gaming experience.

6 Conclusion

While the classification model produced came out as very accurate. It was not able to fit into playing the game because of the delay in sampling. In stick cricket, the shot has to be played in 2s. However, it takes 2.8 seconds for sampling a shot. Because of this stick cricket could not be played using