

Hardware Implementation of ML Algorithms

Divyam Patel (B20EE082) and Dhruv Viradiya (B20CS079)

Demonstration: Gesture Recognition | Weather Prediction

Abstract

This project focuses on the implementation of machine learning (ML) algorithms on Arduino Nano 33 BLE Sense board for weather prediction and gesture recognition. The project involved collecting external data and extracting relevant features, training ML and deep learning models, and deploying the models on the Arduino board using relevant libraries. The results were obtained and displayed on the serial monitor. Challenges faced during the project included library issues and hardware limitations. Overall, this project demonstrates the feasibility of implementing ML algorithms on low-power hardware devices like Arduino for real-time applications.

I. INTRODUCTION

Machine learning (ML) has become an increasingly popular technique for solving a wide range of real-world problems, including image and speech recognition, natural language processing, and predictive analytics. ML models can now be deployed directly on hardware devices, such as microcontrollers and sensors, thanks to the expanding availability of low-cost and low-power hardware platforms. This enables an entirely new set of machine-learning applications for peripheral computing and real-time decision-making. Implementing ML on hardware devices provides numerous benefits. The data can be processed directly on the device, eliminating the need for cloud-based processing. Second, it facilitates real-time decision-making because the device can respond rapidly to changes in the surrounding environment without requiring a network connection. Finally, it can enhance data privacy and security by processing sensitive data locally rather than transmitting it to a remote server.

On hardware devices, multiple varieties of ML algorithms can be implemented, including supervised learning, unsupervised learning, and reinforcement learning. Each of these algorithms has its own advantages and disadvantages, and the algorithm chosen will depend on the nature of the problem and the limitations of the hardware platform. On the Arduino Nano 33 BLE Sense board, we will implement two ML tasks in this project: Weather Prediction and Gesture Recognition. The Arduino Nano 33 BLE Sense board is a small and low-power device that incorporates an accelerometer, gyroscope, magnetometer, humidity sensor, and temperature sensor. These sensors will capture data for the machine learning models that will be trained and deployed directly on the board using the TensorFlow Lite library.

II. HARDWARE AND SOFTWARE COMPONENTS USED

A. Hardware Components:

The main hardware component used in this project is the Arduino Nano 33 BLE Sense board. The board includes a variety of sensors, including a 9-axis IMU (accelerometer, gyroscope, and magnetometer), a humidity sensor, pressure sensor and a temperature sensor. The board also has Bluetooth Low Energy (BLE) connectivity, which can be used to transmit data to other devices. In addition to the Arduino Nano 33 BLE Sense board, we also used a USB cable for programming and debugging the board.



Fig. 1: Arduino Nano 33 BLE Rev2 Sensor

B. Software Components:

The Arduino Integrated Development Environment (IDE) was utilised for this project to write and upload code to the Arduino Nano 33 BLE Sense board. The Arduino IDE is a free, open-source software that provides an intuitive interface for writing code in the C++ programming language.

In addition to the Arduino IDE, we also used several libraries to interface with the sensors and to implement machine learning algorithms. The main libraries used in this project include:

- **TensorFlow Lite for Microcontrollers:** A lightweight version of the TensorFlow machine learning library that is designed to run on microcontrollers.
- **ArduinoBLE:** A library that provides Bluetooth Low Energy (BLE) connectivity for the Arduino Nano 33 BLE Sense board.
- **Arduino_BMI270_BMM150:** This library provides support for accelerometer and gyroscope sensors.
- **Arduino_HS300x:** This library is for measuring temperature and humidity of the surrounding.
- **Arduino_LPS22HB:** This library is for measuring pressure sensor readings.
- **Visual Studio Code:** We used VS code IDE for training the ML models. We used Jupiter Notebook for training ML models.

III. DATA COLLECTION AND FEATURE EXTRACTION

The Data collection is an essential step in any project involving machine learning since it serves as the basis for both the training and the evaluation of the models. In order for us to successfully fulfil the objective of weather prediction, we utilised an external weather dataset that was collected from a public source. The data set included observations taken from the weather forecasting history of a particular place, including those for temperature, humidity, and barometric pressure. Following the data's preprocessing, we extracted statistical properties such as mean, standard deviation, and variance to use as inputs for the machine learning models. These properties included mean, standard deviation, and variance. In this instance, we have focused on training ML models with three primary features: temperature, pressure, and humidity.

For the gesture recognition task, an external dataset containing motion data for various gestures, such as punch, slash, stab, and upper was utilized. We are using accelerometer and gyroscope reading in all 3 directions (x, y, z) as features. We labeled the dataset with the corresponding gestures and used it to train and evaluate our machine-learning models.

IV. MODEL TRAINING

After collecting and pre-processing the data, we applied various machine learning algorithms to the datasets. We began by visualizing the data and exploring the features using Python and the Matplotlib library. We then used the Scikit-learn library to split the dataset into training and testing sets.

For the gesture recognition task, we applied several machine learning algorithms to the dataset, including Random Forest, XGBoost, LightGBM, and Linear Regression. We also experimented with Naive Bayes, which is a simple probabilistic algorithm.

Several machine learning algorithms and deep learning models were implemented on the dataset for the weather prediction task. Scikit-learn was used to train and evaluate Random Forest, XGBoost, and LightGBM models, along with a fundamental Multilayer Perceptron (MLP) neural network. TensorFlow and Keras were also used to build and train Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) models for this task.

The models were trained on the training set and evaluated on the testing set. The classification report was printed for each model, including metrics such as accuracy, precision, recall, and F1 score. The model with the highest accuracy was selected and saved in the form of a file named *"model.h"*. This file can be loaded into the Arduino IDE for deployment on the Arduino Nano 33 BLE Sense board.

Overall, we used a combination of traditional machine learning algorithms and deep learning models to tackle the weather prediction and gesture recognition tasks. We selected the best performing model for each task and saved it in a format that could be used for deployment on the Arduino Nano 33 BLE Sense board.

V. DEPLOYMENT ON HARDWARE

We used the models trained on the dataset to make predictions in real-time on the Arduino Nano 33 BLE Sense board. We deployed the models on the board using the *"model.h"* file, which we had saved earlier in the training phase.

For the weather forecasting task, we used the following libraries to interface with the sensors and deploy the trained model:

- **Arduino_HS300x Library:** This library is used to interface with the HS300x temperature and humidity sensor. It provides an easy-to-use API for reading the sensor data and has built-in error checking for validating the data.
- **Arduino_LPS22HB Library:** This library is used to interface with the LPS22HB pressure sensor. It provides an easy-to-use API for reading the sensor data and has built-in error checking for validating the data.

- **Eloquent TinyML Library:** This library is used to load the trained model onto the Arduino board. It is specifically designed for deploying machine learning models on microcontrollers and provides a minimalistic API for loading and invoking the model. It supports a variety of model formats, including TensorFlow Lite.

We used the respective Arduino libraries to interface with these sensors and obtain the sensor readings. We then fed these sensor readings as input to the trained model using the *"predict()"* function provided by the *"eloquent_tinyML"* library. We obtained the predicted values for temperature, humidity, and pressure in real-time, which were displayed on the Serial Monitor.

Gesture Recognition For the gesture recognition task, we used the following libraries to interface with the sensors and deploy the trained model:

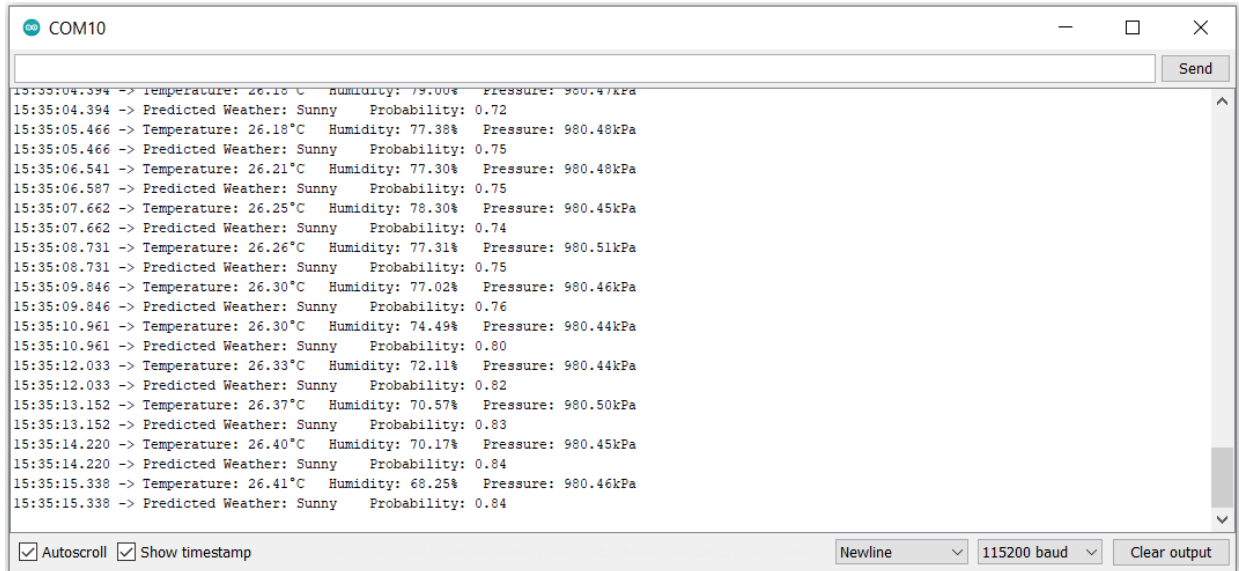
- **Arduino_BMI270_BMM150 Library:** This library is used to interface with the BMI270 accelerometer and BMM150 gyroscope sensors. It provides an easy-to-use API for reading the sensor data and has built-in error checking for validating the data.
- **TensorFlow Lite for Microcontrollers Library:** This library is used to deploy the trained model on the Arduino board. It is a lightweight implementation of TensorFlow designed for microcontrollers and provides a minimalistic API for loading and invoking the model. It supports a variety of model formats, including TensorFlow and Keras.

We used the respective Arduino libraries to interface with these sensors and obtain the sensor readings. We then fed these sensor readings as input to the trained model using the *"invoke()"* function provided by the TensorFlow Lite library. We obtained the predicted class for the input gesture in real-time, which was displayed on the Serial Monitor.

In both tasks, we measured the output on the Serial Monitor, which provided a convenient way to visualize and evaluate the performance of the deployed models in real-time.

VI. RESULTS

The results of the implemented weather forecasting and gesture recognition tasks were observed through the serial monitor. In the weather forecasting task, the system utilized the external dataset that was previously used to train the machine learning model. The real-time readings of temperature, pressure, and humidity were obtained from the Arduino Nano 33 BLE Sense's sensors, which were then used as inputs for the machine learning model. The model then predicted the weather condition and its probability based on the trained data. In the weather forecasting task, the serial monitor displayed the real-time readings of temperature, pressure, and humidity in Celsius, kilopascals, and percentage, respectively. Based on these readings, the system predicted the weather condition as Sunny, Partly Cloudy, Cloudy, Overcast, or Patchy Rain Possible, along with the probability of the predicted weather condition.



The screenshot shows a Serial Monitor window titled 'COM10'. The output displays a series of real-time sensor readings and weather predictions. Each line starts with a timestamp, followed by an arrow and three sensor values (Temperature, Humidity, Pressure), another arrow, and the predicted weather condition with its probability. The data is as follows:

| Timestamp | Temperature (°C) | Humidity (%) | Pressure (kPa) | Predicted Weather | Probability |
|--------------|------------------|--------------|----------------|-------------------|-------------|
| 15:35:04.394 | 26.18 | 79.00 | 980.47 | Sunny | 0.72 |
| 15:35:05.466 | 26.18 | 77.38 | 980.48 | Sunny | 0.75 |
| 15:35:06.541 | 26.21 | 77.30 | 980.48 | Sunny | 0.75 |
| 15:35:07.662 | 26.25 | 78.30 | 980.45 | Sunny | 0.74 |
| 15:35:08.731 | 26.26 | 77.31 | 980.51 | Sunny | 0.75 |
| 15:35:09.846 | 26.30 | 77.02 | 980.46 | Sunny | 0.76 |
| 15:35:10.961 | 26.30 | 74.49 | 980.44 | Sunny | 0.80 |
| 15:35:12.033 | 26.33 | 72.11 | 980.44 | Sunny | 0.82 |
| 15:35:13.152 | 26.37 | 70.57 | 980.50 | Sunny | 0.83 |
| 15:35:14.220 | 26.40 | 70.17 | 980.45 | Sunny | 0.84 |
| 15:35:15.338 | 26.41 | 68.25 | 980.46 | Sunny | 0.84 |

At the bottom of the window, there are checkboxes for 'Autoscroll' and 'Show timestamp', both of which are checked. On the right side, there are dropdown menus for 'Newline' and '115200 baud', and a 'Clear output' button.

Fig. 2: Serial Monitor Output for Weather Prediction

In the gesture recognition task, the system utilized the accelerometer and gyroscope readings from the Arduino Nano 33 BLE Sense's sensors to recognize different hand gestures. The machine learning model was trained on the external dataset, which consisted of different hand gestures performed by users. As the user performed different hand gestures on the Arduino board, the system recognized the gesture and displayed it on the serial monitor in real-time, along with the probability of the recognized gesture. For instance, if the user performed a "Punch" gesture, the system might recognize it with a probability of 0.9. The probability indicated the level of confidence that the system had in the recognized gesture based on the input data.

Overall, the results demonstrate the successful deployment of machine learning models on the Arduino Nano 33 BLE Sense, allowing for real-time prediction and recognition tasks. The accuracy of the predictions and recognitions depends on the quality of the input data and the complexity of the machine learning models used.

```

COM4
16:12:23.643 -> punch ████████:0.947557
16:12:23.643 ->
16:12:25.721 -> punch ████████:0.995257
16:12:25.721 ->
16:12:27.525 -> stab ████████:0.797632
16:12:27.525 ->
16:12:29.610 -> punch ████████:0.947501
16:12:29.610 ->
16:12:31.666 -> slash ████████:0.999963
16:12:31.666 ->
16:12:34.692 -> stab ████████:0.944586
16:12:34.692 ->
16:12:36.658 -> stab ████████:0.986688
16:12:36.658 ->
16:12:38.571 -> stab ████████:0.921195
16:12:38.571 ->
16:12:41.070 -> slash ████████:0.999989
16:12:41.070 ->
16:12:43.473 -> slash ████████:0.830332
16:12:43.473 ->
☒ Autoscroll ☒ Show timestamp
Newline 115200 baud Clear output

```

Fig. 3: Serial Monitor Output for Gesture Recognition

VII. CHALLENGES FACED DURING THE PROJECT

During the course of the project, we encountered several challenges that needed to be addressed. Some of the significant challenges we faced during the project are as follows:

- **Hardware compatibility:** Initially, we planned to use Arduino Mega 2560 for the project. However, this board required external sensors for temperature, humidity, pressure, and gyroscope measurements. We realized that it would require complex circuitry, and the deployment would become challenging. Therefore, we switched to Arduino Nano BLE Sense, which has all these sensors built-in, and it made our task easier.
- **Library compatibility:** While using the Arduino Nano BLE Sense Rev2, we faced issues with library compatibility. For instance, we had to change the library for temperature and humidity measurements from HTS221 to HS300x, and for gyroscope and accelerometer measurements from LSM9DS1 to BMI270_BMI150. Due to this, we had to modify the code and resolve errors that occurred while deploying the models.
- **Memory constraints:** Arduino Nano BLE Sense has limited memory, which posed a challenge while deploying deep learning models. We had to optimize the code and reduce the model's size to fit within the device's memory constraints.
- **Real-time prediction:** Since we were deploying machine learning models on the Arduino board, we had to ensure that the prediction was made in real-time. It required us to optimize the code and reduce the inference time to provide accurate predictions.

Overall, we were able to overcome these challenges by optimizing the code, modifying the libraries, and choosing the appropriate hardware.

VIII. CONCLUSION

In conclusion, this project aimed to implement machine learning on Arduino Nano 33 BLE Sense for weather prediction and gesture recognition tasks. We utilized external data for training and applied various machine learning algorithms such as random forest, XGBoost, LGBM, and naive bayes. We also implemented deep learning models and saved the one with the highest accuracy for deployment on the hardware. For weather forecasting, we used the Arduino_HS300x and Arduino_LPS22HB libraries to measure temperature, pressure, and humidity. We utilized eloquent_tinyML for deployment and predicted weather conditions in real-time. For gesture recognition, we used the Arduino_BMI270_BMM150 library to measure real-time accelerometer and gyroscope readings and deployed the model using TensorFlow Lite. We faced challenges such as library changes in the Arduino Nano 33 BLE Sense, which required modifications in the code. However, we were able to overcome them and successfully implemented machine learning on the hardware. The project shows the potential of machine learning on low-power devices such as Arduino, enabling real-time processing of data and efficient decision-making.

IX. REFERENCES

- 1) <https://forum.arduino.cc/t/diagnostics-for-nano-33-ble-sense/1089740/13>
- 2) <https://docs.arduino.cc/hardware/nano-33-ble-sense>
- 3) <https://scikit-learn.org/stable/index.html>
- 4) <https://docs.arduino.cc/tutorials/nano-33-ble-sense/get-started-with-machine-learning>
- 5) <https://www.kaggle.com/datasets/vonline9/weather-istanbul-data-20092019>