EMBEDDED MACHINE LEARNING

PROJECT 4

PRANIT SEHGAL

ASU : 1225456193

Google Collab Links:

1. **Machine Learning :** https://colab.research.google.com/drive/1IsssGuMVhRPL9l4skXLkncZtRjIxfMSa?usp=sharing
2. **Data Cleaning : https://colab.research.google.com/drive/1Dx1xko5obV4vSt65snU3PE_1QR-485VS?usp=sharing**

A: Requirements document

1. **Overview**:
    - **Objective**: The primary aim of this project was to design and develop a posture detection system that can distinguish between various postures by leveraging sensor data from an Arduino.
    - **Process**: We initiated the project by collecting data from Arduino sensors. This data was then processed, cleaned, and used to train a machine learning model for posture detection. Lastly, the model was converted into a suitable format for real-time posture prediction on an Arduino setup.
    - **Interaction with Existing Systems**: The machine learning aspect of this project was built upon TensorFlow and was integrated into a microcontroller environment via the Edge Impulse platform.
2. **Function Description**:
    - **Prototype**: The developed system is based on an Arduino equipped with sensors. Data from these sensors is processed and fed to a machine learning model, allowing it to predict the current posture.
    - **Performance**: The system achieves an impressive accuracy of approximately 88.65% on the test data, indicating its reliability.
    - **Usability**: The system can be deployed in real-time scenarios and has the potential for integration into health monitoring systems, gaming controls, or ergonomic equipment.

3. **Deliverables**:
    - A built prototype (Arduino with integrated sensors).
    - A machine learning model for posture detection.
    - A comprehensive report detailing the design, experiment, results, and future recommendations.

B: Experiment:

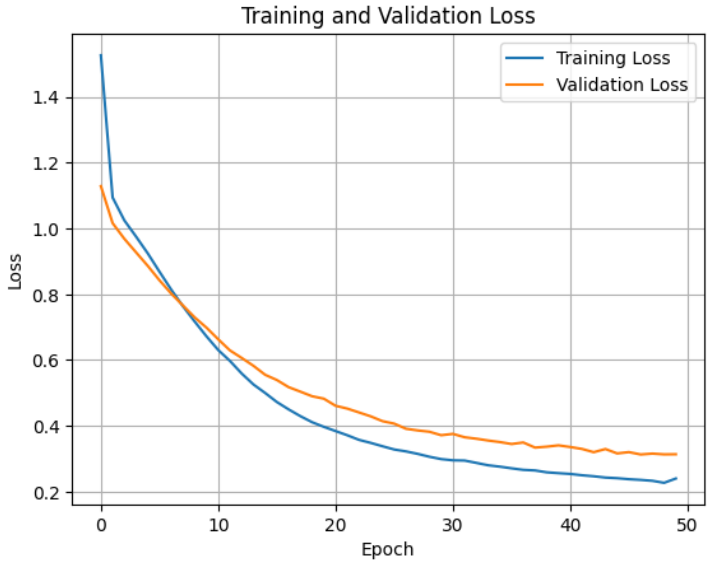
- **Data Collection**: Sensor data from the Arduino (including Accelerometer, Gyroscope, and Magnetometer) was collected and stored in various CSV files. These files represented data for different postures.
    - **Challenges**: Designing an experiment that mimics real-world postures is intricate. One significant difficulty was the inconsistent readings from the Magnetometer, potentially due to a faulty sensor. This inconsistency sometimes introduced noise into the data, which could impact model accuracy.
- **Data Cleaning and Aggregation**: Python scripts were employed using the pandas library to clean and merge the data. Specifically, data was read, unnecessary metadata was stripped, and values from different sensors were combined to create a unified dataset.

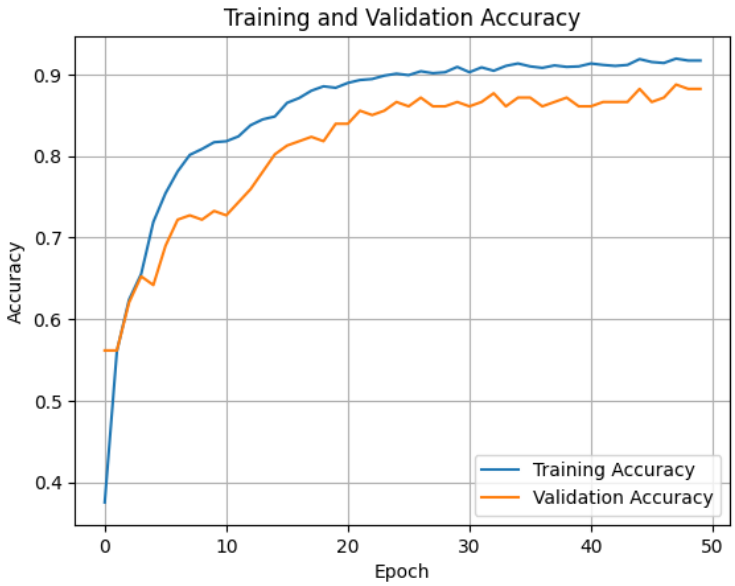
C: Algorithm:

- **Data Preprocessing**:
    - After aggregating the data, class labels were assigned, missing values were addressed, and the dataset was split into training, validation, and test sets.
- **Machine Learning Model**:
    - The posture detection system employs a Convolutional Neural Network (CNN) architecture. This choice was motivated by CNN's proficiency in identifying spatial hierarchies in data, making it well-suited for time-series data like ours.
    - The model structure comprises Conv1D layers, followed by MaxPooling, Flattening, and Dense layers. Activation functions `relu` and `softmax` are utilized for non-linearity and multi-class classification, respectively.
    - The model was trained using the `adam` optimizer, and the loss function used was `sparse_categorical_crossentropy`.
- **Coordination between Microcontroller and Base-Station**:
    - After training, the model was converted into TFLite format and uploaded to Edge Impulse, which facilitated its usage on Arduino. The Edge Impulse platform also provided testing capabilities to ensure the model's functionality with custom inputs.

D: Results:

- **Model Performance**:
  - Training Performance: The model was trained for 50 epochs, and its performance was monitored on both training and validation sets.
  - Test Performance: The model achieved a test accuracy of approximately 88.65% with a loss of 0.3164.





On-device performance

MCUs

		EON COMPILER		TFLITE	
DEVICE	LATENCY	RAM	ROM	RAM	ROM
Low-end MCU	84 ms.	2.3K	29.3K	4.7K +2.4K	48.0K +18.7K
High-end MCU	2 ms.	2.1K	32.5K	4.5K +2.4K	59.5K +27.0K
+ AI accelerator	2 ms.	2.1K	32.5K	4.5K +2.4K	59.5K +27.0K

Microprocessors

DEVICE	LATENCY	MODEL SIZE
MPU	1 ms.	13.6K
GPU or accelerator	1 ms.	13.6K

- **Real-time Prediction**:
  - The model, once integrated with the Arduino, could make real-time predictions based on live sensor data. The Arduino code interprets these predictions and determines the posture with the highest confidence score.

```
Select a sensor: (A)ccelerometer, (G)yroscope, (M)agnetometer
Read from Sensor: -0.01, -0.02, 0.98
Sending to Model: -0.01, -0.02, 0.98
1 : 0.98
2 : 0.00
3 : 0.00
4 : 0.00
5 : 0.01
Read from Sensor: -0.37, 11.23, -3.11
Sending to Model: -0.37, 11.23, -3.11
1 : 0.00
2 : 0.32
3 : 0.00
4 : 0.34
5 : 0.34
Read from Sensor: 0.67, -4.33, -2.99
Sending to Model: 0.67, -4.33, -2.99
1 : 0.00
2 : 0.06
3 : 0.00
4 : 0.93
5 : 0.01
Read from Sensor: 31.13, 7.04, 21.77
Sending to Model: 31.13, 7.04, 21.77
1 : 0.00
2 : 0.99
3 : 0.00
4 : 0.00
5 : 0.01
```

Comparison between Test Performance and Real-time Prediction:

- While the model displayed an accuracy of 88% on the test dataset, real-time prediction was accurate around 70% of the time. This discrepancy can be attributed to the inconsistency of the magnetometer sensor in the real-world application. The magnetometer, which played a crucial role in determining posture, showed occasional erratic behavior, thereby impacting the real-time prediction accuracy. It emphasizes the difference between controlled test environments and dynamic real-world scenarios.

E: Discussions:

- **Summary of Results**: The developed system can detect postures with a high degree of accuracy (around 88.65%). It's a testament to the robustness of the approach, even in the face of challenges like inconsistent magnetometer readings.
- **Difficulties**:
    - **Orientation-Insensitivity and Sensor-Agnosticism**: Designing a posture detection system that's orientation-insensitive is challenging. The inconsistency in magnetometer readings added another layer of complexity.
- **Future Recommendations**:
    - Addressing the issue of the inconsistent magnetometer, possibly by replacing the sensor or employing advanced noise-reduction techniques.
    - Exploring deeper or alternative neural network architectures for potential accuracy improvements.
    - Integrating more sensors or data sources to refine and expand the system's capabilities.
- **Project Challenges**: The most strenuous aspect of the project was managing the magnetometer's inconsistent readings and ensuring that the ML model remained robust despite potential noise in the data.
- **Improvements and Takeaways**: While the real-time prediction was quite accurate, there's always room for improvement. Enhanced data preprocessing, anomaly detection, and more advanced ML techniques could be explored in future iterations of this project.