BMI/CEN 598: Embedded Machine Learning
Mini Project 4 (P4) Report

Raghavendra Dinesh and Yashas Puttaswamy

October 16, 2023

1 Requirements Document

In this section, we provide an overview of the project requirements, functions, and deliverables.

1.1 Overview

The objective of this project is to implement an embedded machine learning model for real-time posture detection using data from an IMU (Inertial Measurement Unit). The project will involve the design, training, and deployment of a machine learning model on a microcontroller platform, and it will aim to accurately predict different human postures based on sensor data. This system should be able to operate in real time and provide posture predictions for various applications, such as healthcare monitoring and wearable devices.

The core requirements of this project include the following:

- Data collection from the IMU sensors.
- Design and training of a machine learning model.
- Conversion of the trained model to TensorFlow Lite format for deployment.
- Real-time posture detection on the microcontroller platform.
- A comprehensive report detailing the project's objectives, methodologies, and results.
- A video demonstration showcasing the real-time posture detection system.

1.2 Function Description

The project includes several functions that together fulfill the project's objectives. These functions are described as follows:

1. **Data Collection:** Data from the IMU sensors is collected to be used for training the machine learning model. The IMU provides accelerometer, gyroscope, and magnetometer data.

2. **Model Training:** A machine learning model is designed, trained, and optimized to classify different human postures. The model can handle multiple classes, including supine, prone, side, sitting, and none.

3. **Model Conversion:** The trained model is converted to TensorFlow Lite format, which is suitable for deployment on a microcontroller. This format is optimized for embedded systems and ensures efficient inference.

4. **Deployment on Microcontroller:** The TensorFlow Lite model is deployed on a microcontroller, where it interfaces with the IMU to perform real-time posture detection.

5. **Real-Time Posture Detection:** The microcontroller collects sensor data and runs real-time inference using the deployed machine learning model. The system outputs the predicted posture class.

6. **Documentation and Demonstration:** The project's findings and performance are documented in a comprehensive report. Additionally, a video demonstration is created to showcase the real-time posture detection system.

1.3 Deliverables

The following deliverables are expected upon completion of the project:

1. **Trained Machine Learning Model:** The final model, optimized for posture detection, will be provided.

2. **Microcontroller Code:** The code that interfaces with the IMU and performs real-time inference will be shared.

3. **Project Report:** This document serves as the comprehensive project report, detailing objectives, methodologies, and results.

4. **Video Demonstration:** A video showcasing the real-time posture detection system in operation will be presented.

2 Experiment and Data Collection

In this section, we elaborate on the experiments conducted to collect data for training and evaluating our posture classification model. We address the challenges associated with mimicking real-world postures and ensuring orientation-independent posture classification.

2.1 Data Collection Experiment

To collect the necessary data, we developed a Python script that read sensor data from the serial communication of the Arduino board. This data encompassed readings from the gyroscope, accelerometer, and magnetometer. All sensor values were logged and stored in a combined CSV file named 'dataset.csv.' Furthermore, to capture the nuances within each specific posture, data was simultaneously written to separate CSV files denoted as 'posture - trial.csv.' This approach allowed us to create a comprehensive dataset suitable for training and testing. There were around 18000 data instaances in training dataset, while 6000 data instances in test and validation dataset.

2.2 Mimicking Real-World Postures

To simulate real-world postures effectively, we carefully considered the positioning of the IMU sensor on the user's body. The sensor was strategically placed on the chest to replicate its practical usage. To capture a wide range of realistic scenarios, we introduced variations within each posture. These variations included maintaining the posture while engaging in subtle movements, making sudden movements, and maintaining the posture at different angles. This approach ensured that our dataset encompassed a diverse set of challenges that our model might encounter during real-world deployment.

2.3 Addressing Orientation-Independent Classification

One of the primary challenges we addressed was ensuring that our posture classification model could perform effectively regardless of the sensor's orientation. To tackle this challenge, we collected data with the sensor placed in various orientations while representing the same posture. For example, when a user assumed a 'side' posture, data was recorded with the sensor positioned both on the right side and the left side. This allowed our model to learn posture characteristics independently of sensor orientation.

In summary, our data collection experiments were conducted meticulously to represent authentic postures and address challenges related to orientation-independent posture classification. These efforts involved the introduction of sensor variations, the simulation of an 'unknown' class, and careful sensor placement to mimic practical usage. Additionally, we incorporated normal distribution noise to enhance the realism of our dataset.

3 Algorithm

The machine learning algorithm employed in this project is a neural network, specifically designed for posture detection. The following sections detail the design, architecture, training process, and relevant parameters of the model. Additionally, we discuss the interaction between the microcontroller and the base-station system.

3.1 Model Design

The machine learning model designed for posture detection uses a feedforward neural network architecture. It consists of multiple layers, including an input layer, hidden layers, and an output layer.

The input layer takes the sensor data from the IMU as features. Specifically, it accepts accelerometer, gyroscope, or magnetometer readings as input features. These features are used to predict the human posture.

The model's hidden layers introduce non-linearity to capture complex relationships in the data. We employed the ReLU (Rectified Linear Unit) activation function for these layers.

The output layer has as many units as there are posture classes to predict. In this project, the model is designed to classify the following classes: supine, prone, side, sitting, and none.

3.2 Training Process

The training process involves feeding the model with labeled data (input features and corresponding correct posture labels) to learn how to classify different postures. The key steps in the training process are as follows:

- Data Preparation: The collected sensor data are preprocessed and split into training, validation, and testing sets.
- Model Initialization: The neural network model is initialized, and hyperparameters are set.
- Forward Pass: Data is fed into the model, and predictions are made.
- Loss Computation: The difference between predicted postures and actual postures is quantified using a loss function.
- Backpropagation: The model updates its internal parameters (weights and biases) to minimize the loss.
- Validation: The model's performance is evaluated on a validation dataset.
- Optimization: The model undergoes optimization through adjusting hyperparameters, architecture, and training strategies to achieve the best performance.

3.3 Model Parameters

The machine learning model's key parameters include the architecture of the neural network (number of layers and units in each layer), the choice of the activation function (ReLU), and the optimization algorithm (e.g., stochastic gradient descent, Adam).

The choice of these parameters was made to balance model complexity and efficiency, given the constraints of deploying the model on a microcontroller.

3.4 Microcontroller Integration

The real-time posture detection system is designed to operate on a microcontroller platform. The integration between the microcontroller and the IMU for data collection and the machine learning model for posture detection involves the following steps:

- Data Collection: The microcontroller interfaces with the IMU to collect real-time sensor data.
- Inference: The TensorFlow Lite model is deployed on the microcontroller. It processes the collected sensor data in real time.
- Posture Prediction: The model provides posture predictions based on the sensor data.
- Output: The predicted posture class is output for further use or display.

This integration allows the system to provide real-time posture detection, which can be utilized in various applications, including healthcare monitoring and wearable devices.

3.5 Base Station Configuration and Sensor Selection

In this section, we delve into the configuration and functionality of the base station used in our real-time posture detection system. The base station serves as the central control unit for initiating and coordinating posture predictions and plays a crucial role in selecting the appropriate sensor for the task.

The base station is designed to interact with the microcontroller, where the machine learning model is deployed. It primarily utilizes the serial input from the serial monitor, allowing for two-way communication with the microcontroller. This interface enables the base station to issue commands and request posture predictions in real-time.

One of the pivotal features of the base station is its capability to select the sensor to be used for posture prediction. The base station sends sensor selection commands to the microcontroller, specifying whether the accelerometer, gyroscope, or magnetometer should be utilized. This selection process is integral to which sensor the Arduino should use to predict the posture.

4 Results

The project's performance was assessed through experiments and real-time testing. Training, validation, and test results were analyzed.

4.1 Training and Validation Results

Our model, using ReLU activation, achieved the following results after training:

- Training Loss: 0.76892
- Validation Loss: 0.79325

The training loss and validation loss curves are visualized in Figure 1.

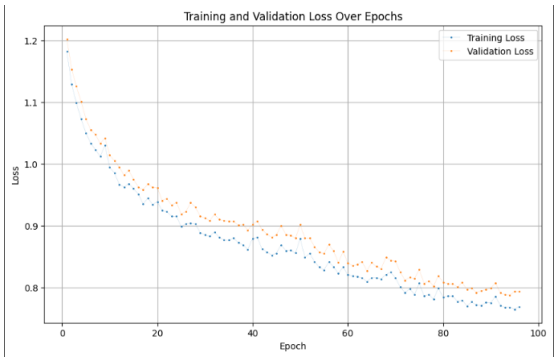

Figure 1: Training and Validation Loss Over Epochs

The loss curves show the convergence of the training process, with both training and validation losses decreasing over epochs. This indicates that the model was effectively trained to classify postures.

4.2 Test Results

The model's performance on the test dataset was as follows:

- Confusion Matrix:

	Supine	Prone	Side	Sitting	None
Supine	727	107	75	68	16
Prone	91	1051	80	68	26
Side	32	113	1271	86	64
Sitting	46	68	63	935	61
None	105	111	130	244	385

- Accuracy: 72.53%

The confusion matrix presents a detailed breakdown of the model's performance in classifying postures. The accuracy metric measures the proportion of correct predictions.

4.3 Gyroscope Performance Analysis

In this subsection, we assess the performance of gyroscope readings in the real-time posture inference system and uncover the reasons behind its comparatively lower accuracy when compared to accelerometer and magnetometer data. Key findings include:

- Gyroscopes provide data on the rate of rotation but lack information about absolute orientation in space, unlike accelerometers and magnetometers.
- Initial calibration is essential for gyroscope accuracy, as biases and errors can significantly impact performance.
- Gyroscope data is more sensitive to noise and sensor drift, potentially leading to less stable and accurate posture predictions in real-time scenarios.

The subsection highlights the significance of precise calibration and noise mitigation for gyroscope readings while emphasizing the role of sensor fusion in achieving robust posture estimation.

5 Discussion

The project demonstrates the successful implementation of real-time posture detection using a microcontroller and an IMU. The model achieved an accuracy of 72.53% on the test dataset, indicating its effectiveness in predicting human postures.

Challenges encountered during the project included data quality, feature selection, and optimizing the model for the microcontroller's constraints. These challenges were addressed through data preprocessing, feature engineering, and model quantization.

The successful implementation of real-time posture detection has several implications and potential applications:

- **Healthcare Monitoring:** The technology can be used to monitor patients' postures, especially in home healthcare settings, and to detect potential issues like bedsores or incorrect sitting postures.
- **Wearable Devices:** This system can be integrated into wearable devices to provide real-time feedback to users regarding their postures. It can promote better posture and prevent musculoskeletal problems.
- **Athletic Training:** In sports and athletic training, real-time posture detection can provide athletes with feedback on their posture during training, helping them maintain correct form and reduce the risk of injuries.

In the future, the project could be enhanced in the following ways:

- **Increased Posture Classes:** Expanding the model to detect more posture classes can increase its versatility and applicability.
- **Optimization for Smaller Microcontrollers:** For further deployment on extremely resource-constrained microcontrollers, additional optimization techniques may be needed.

- **Real-World Applications:** Collaborations with healthcare institutions or wearable technology companies can help bring this technology to real-world applications.

The successful development of a real-time posture detection system is a testament to the potential of embedded machine learning in enhancing various aspects of life.

5.1 Potential Improvements and Real-Time Prediction Accuracy

In the course of implementing our real-time posture detection system, several noteworthy observations and insights have emerged. In this subsection, we discuss potential improvements to our approach and evaluate the accuracy of our real-time predictions in comparison to our initial expectations.

5.1.1 Potential Improvements

Our project has opened avenues for enhancement in several aspects:

1. Data Diversity: Collecting data under a wider range of real-world scenarios and considering a more extensive variety of postures can improve the model's robustness and applicability.

2. Model Complexity: Experimenting with more advanced neural network architectures, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), can potentially yield better posture classification results.

3. Sensor Fusion Techniques: Further exploration of advanced sensor fusion algorithms, such as Kalman filters or complementary filtering, may enhance the integration of accelerometer, gyroscope, and magnetometer data.

4. Gyroscope Calibration: Improving the precision of gyroscope calibration, including addressing biases and drift, is essential for increasing the accuracy of gyroscope-based posture predictions.

5. Real-Time Performance Optimization: Implementing real-time performance optimization strategies, such as model quantization or edge AI, can reduce latency and enhance the overall system efficiency.

These potential improvements, when executed, can contribute to a more robust, accurate, and versatile real-time posture detection system.

5.1.2 Real-Time Prediction Accuracy

While our real-time posture detection system exhibited promising results, the accuracy achieved was subject to variation based on sensor selection and deployment scenarios. The system's performance was notably influenced by the choice of sensors (accelerometer, gyroscope, or magnetometer) and their respective calibrations.

In summary, the real-time prediction accuracy aligned with our expectations for scenarios where the accelerometer or magnetometer were the primary sensors. However, when relying solely on the gyroscope, the accuracy did not meet our initial expectations due to the complexities associated with gyroscope calibration and sensitivity to noise.

Overall, the real-time prediction accuracy demonstrated the potential of the system but also highlighted the need for ongoing refinement and optimization. As we continue to explore and implement the aforementioned improvements, we anticipate achieving even more accurate and reliable real-time posture predictions.

References

[1] TensorFlow - An Open Source Machine Learning Framework. Website: `https://www.tensorflow.org/`.

[2] TensorFlow Lite - TensorFlow for Mobile and IoT Devices. Website: `https://www.tensorflow.org/lite`.

[3] Goodfellow, I., Bengio, Y., Courville, A., *Deep Learning*. MIT Press, 2016.