**BMI/CEN 598: Embedded Machine Learning**

**Mini Project 4 (P4)**

**Submitted by: Group 18**

**Debanjalee Roy (1225660787) and Dheeraj Kumar (1225939028)**

# **A: System Design**

## **Motivation:**

The motivation behind this project lies in the importance of accurate posture detection, particularly in healthcare applications. Tracking lying postures can be crucial for monitoring patients' health, and this project aims to develop a system using IMU sensor (accelerometer, gyroscope, and magnetometer) data for efficient posture detection. This project aims to enhance the lying posture tracking system, achieving sensor-agnostic real-time posture detection using an Arduino board with an embedded IMU sensor. The model is designed to make predictions based on 3-channel input, irrespective of the sensor type (accelerometer, gyroscope, or magnetometer).Also, it aims to predict the data based on base-station command i.e., command from user through serial communication.

## **High-level Design:**

The IMU sensor, accelerometer, magnetometer, and gyroscope sensor data, embedded in Arduino Nano BLE Sense board, was chosen for data collection. Placed on a particular orientation emulating the chest position, it depicts various postures: supine, prone, side (right or left), sitting, and an 'unknown' posture. The data is collected through serial communication via USB plugged into computer (using Putty) and stored as datasets in our system in CSV files. Those CSV files have been used to train the model for further training on the different sensors using 3-inout channels x,y and z-axis.

## **Observations and Difficulties:**

During the design phase, challenges included:

* Ensuring the model's incorrect predictions in sensor orientations due to lesser data input.
* Assumptions were made about possible ways the sensor can be worn were crucial, and noise needed to be added to simulate real-world conditions.
* Assuming directions for “Unknown” Posture and gathering data for it was challenging as these directions can be very random.
* Since the data collected is huge, it is difficult to train the model on large dataset as it increases the RAM space and gives linker error, which needed significant cleaning and trimming of data.
* Coming up with the right activation function to achieve the highest performance was key to develop an efficient ML model.

## **Sampling Frequency:**

A suitable sampling frequency was chosen to capture relevant information from the IMU sensor data. The choice is aimed at balancing data richness and computational efficiency. In the provided code, the sampling frequency for gathering sensor data is established at 2 Hz, which translates to 2 samples taken per second. This frequency has been chosen to enable effective real-time posture identification while also conserving battery power. Below screenshot shows the frequency printed out while executing the code for sample rate.

A white screen with black text

Description automatically generated

## **Deep Learning Model Architecture:**

The chosen model is a neural network with an input layer of three nodes (for X, Y, and Z accelerometer values), followed by two fully connected layers with ReLu activation. A dropout layer is included for regularization, and the output layer uses softmax activation for multiclass classification.

## **Training Parameters/Algorithms:**

The model is compiled using the Adam optimizer and sparse categorical cross-entropy loss. Training involves 10 epochs with a batch size of 32. Model evaluation includes assessing accuracy on a test dataset.

## **Process:**

The project involves code development for sensor data reading, model implementation, data collection for various postures, dataset construction, and training. Additionally, an interactive interface for real-time posture detection and communication with a base station is created using command line interface of serial communication.

## **Interaction with Existing Systems:**

The system provides accurate predictions regardless of sensor orientation, with an interactive interface facilitating communication between the microcontroller and the base station using Serial Communication.

## **Function Description:**

**Prototype:** The prototype includes Arduino code, TensorFlow Lite model integration, including a command line interface for posture detection. Three separate models for accelerometer, gyroscope, and magnetometer are utilized, each with its dedicated interpreter and tensor arenas.

**Performance:** The system's performance is evaluated during the design phase, encompassing validation and test evaluations, and found that more accuracy is observed with accelerometer data as compared to magnetometer and Gyroscope data.

**Usability:** The system will offer a user-friendly interface for selecting sensor types and initiating real-time predictions.

**Deliverables:**

***Built Prototype:*** Complete Arduino code, TensorFlow Lite models, and data collection code is attached.

***Report:*** Comprehensive documentation covering experiments, algorithm design, results, and discussions are provided.

# **B: Experiment**

## **Experimental Design:**

Data collected for various scenarios mimicking real-world postures using accelerometer, gyroscope, and magnetometer. Sufficient data for each unique posture scenario, including the 'unknown' class is collected and consideration of multiple rounds of data collection for each scenario is done. Experiments involved collecting sufficiently diverse data for each posture scenario as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Posture** | **No. of Experimental Sample without Noise for each orientation** | **No. of trials included** | **% of Total No. of Experimental Samples for data training** | **% of Total No, of Experimental samples for Test/Validation purpose** |
| Supine | 500 | 2 | 80% | 20%,10% |
| Prone | 500 | 2 | 80% | 20%,10% |
| Side (Left and Right) | 500 total for each | 4 (2 for right,2 for left) | 80% | 20%,10% |
| Sitting | 500 | 2 | 80% | 20%,10% |
| Unknown | 500 each | 2 | 80% | 20%,10% |

**Note:** More details on the data is provided in the “final\_eml\_project4.ipynb – Colaboratory” documentation attached with this project folder.

The sensor data was labeled and stored in separate csv files named as Lateral\_Left\_Posture\_Trial1, Lateral\_Right\_Posture\_Trial1, Supine\_Posture\_Trial1, Prone\_Posture\_Trial1, Unknown\_\_X\_Posture\_Trial1 (X=1,2),Sitting\_Posture\_Trial1 for without noise (or minimal) data, all the respective “\_Trial2” csv files are for data with different orientation. The dataset construction utilized a sliding window approach for every 500 samples for each posture to create input segments for the neural network. The left and right posture data is combined into the dataset as “side” class. Supine as “Supine” class, similarly “Prone”, “Sitting”, and “Unknown” classes. This is shown as below:

A graph of blue and white bars

Description automatically generated

* In the test dataset, 20% of the data is kept as Test data, 80% as Training data in which 10% is validation data.
* Utilization of three separate TensorFlow Lite models for accelerometer, gyroscope, and magnetometer.
* Coordination algorithm between the microcontroller and the base station for real-time predictions.
* **Training:** Offline training of models, with model performance assessments and adjustments to prevent overfitting or underfitting.
* The data is organized into three classes: accelerometer, gyroscope, and magnetometer. The distribution of CSV files across these classes can be visualized using a bar plot, providing insights into the dataset's composition.
* Data preprocessing involves concatenating sensor readings from each file within a class, resulting in comprehensive datasets for accelerometer, gyroscope, and magnetometer. The datasets are then split into training and testing sets.
* The models trained for each sensor type have been saved as .h5 files and converted into TensorFlow Lite format (.tflite).
* Finally, predictions are showcased on new data, demonstrating the model's ability to classify sleeping postures based on sensor readings.

## **Model Parameters:**

**For Accelerometer:**

A screenshot of a computer

Description automatically generated

**For Gyroscope:**

A screenshot of a computer

Description automatically generated

**For Magnetometer:**

A screenshot of a computer program

Description automatically generated

## **Challenges:**

Challenges included:

* Emulating real-world postures and addressing orientation-independent classification.
* Assumptions about how the sensor would be worn were made to ensure robustness.
* Lateral side postures direction had to be assumed.
* For Sitting posture, the direction and posture had to be assumed.
* The machine learning model was evaluated and tested repeatedly to get highest accuracy.
* Compared different activation functions such as Sigmoid, Tanh, and ReLU to find the best outcome.
* Size of model created a big concern for this project as including three models for accelerometer, gyroscope, and magnetometer increased the data set by large amount exceeding the RAM size.

# **C: Algorithm**

## **Machine Learning Algorithm Design:**

The algorithm involved preprocessing data by extracting acceleration values. The neural network architecture comprised layers with tanh activation, and dropout layers were added for regularization.

## **Architecture and Training Process:**

The model's architecture included two hidden layers and an output layer. Training involved 30 epochs with a batch size of 32, and the model was compiled using the Adam optimizer.

## **Activation Functions:**

Sigmoid, Tanh, and ReLU activation functions were compared. Relu was chosen for its performance, considering the limitations of embedded devices during our experimentation, we find the analysis for the activation functions- Relu, tanh, sigmoid activation. However, after overall analysis, the loss appeared almost same for all.

## **D: Results**

## **Test Data Performance:**

The model achieved a test accuracy of 100% on the test dataset for magnetometer,accelerometer and little less for gyoscope, demonstrating its ability to generalize to new, unseen data.

A screenshot of a computer

Description automatically generated

Arduino Serial monitor output for various postures and sensor readings.

A screenshot of a computer

Description automatically generated

A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

Description automatically generatedA graph of a graph of a graph

Description automatically generated with medium confidenceA graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

Description automatically generated

# **E: Discussions**

## **Results Summary:**

The model successfully detected lying postures with good(100 % for accelerometer and magnetometer) and almost 70%(for gyroscope)) accuracy. It showed promise in real-world scenarios, as demonstrated by its performance on diverse test data.

## **Difficulties and Solutions:**

Challenges in designing the system included ensuring orientation independence and simulating real-world conditions without actual sensor wear. Future improvements could focus on refining these aspects. More layers can be added to make better accuracy if more sets of data is taken.

## **Project Challenges:**

The most challenging part of the project was collecting huge sets of data for each posture and axis. That constitutes a huge spectrum of possibilities and hence the dataset is also large, therefore it is challenging to gather the right data. Improvements could involve data in different angles for more accurate lying posture detection.

# **Conclusion:**

In conclusion, the developed posture detection system using IMU sensor data and a neural network show promise for real-world applications. Continued refinement and data collection could enhance its accuracy and robustness. User input triggers the data collection and prediction for the different sensor data.

**References:**

* Arduino Sample examples- Sample\_Accelerometer, Sample\_Gyroscope,Sample\_Magnetometer, Hello\_World(TensorFlowLite)
* Class notes