# Monty MATLAB Group 9: Silly Walks Classifier

Group Members: Junqing Du, Mengtian Xu, Peilin Lu, Shuchen Xu
Technical University of Munich

12.07.2024

**Abstract**

This project aims to develop a MATLAB program that classifies accelerometer data collected from IMU sensors. The program distinguishes between ”Normal walk” and ”Silly walk” by analyzing motion patterns captured in the x, y, and z axes. The classification process encompasses data collection, extraction, training of the classification model, and visualization of results via an intuitive Graphical User Interface (GUI).
The procedures involve data collection, data extraction, model training and classification and GUI design.

**1. Data Collection**

The initial data only includes acceleration matrix, with the real time when the data was recorded and the x, y, and z-axis data. This data cannot be used directly. Running the MATLAB program yields the required relative time and the desired matrix format. To ensure there were no errors in the data collection process, the images corresponding to the data need to be checked. Finally, run the example data and compare two images. If there are no significant differences(The signs of the x, y, and z data are reversed), it indicates that the data collection process is correct.

**2. Data Extraction**

The process involves several key steps: resampling the data, segmenting it into windows, and labeling each window based on the source file name. To ensure consistency across different datasets, we resample the data to a target sampling rate (e.g., 50 Hz) using linear interpolation. This step aligns the data to a uniform time base, which is crucial for subsequent analysis. Another critical aspect of data preparation is windowing. Here, we configure the window shift to half of the window length, ensuring a 50% overlap between consecutive windows. This overlap is essential for capturing transitional patterns in the data, as illustrated in Figure 2. By following these steps, we effectively prepare time-series data for various analytical tasks, ensuring consistency, accurate labeling, and readiness for machine learning applications.

**(a)** Silly Walking Style 1

**(b)** Silly Walking Style 2

**(c)** Silly Walking Style 3

**Figure 1:** Three different silly walking style

**3. Model Training and Classification**

Next we need a program to train a classifier based on the existing data, this classifier will be used to identify whether the given input is normal walk or silly walk. classifyWalk and output the accuracy of classification.The flowchart is shown in Figure 3.

Let’s analyse this whole procedure, which consists of many sub-functions, they are: load dataset, normalizeData, specify network, trainSillyWalkClassifier function. Let’s go through them in order. load dataset function’s input is the file name of the training set and test set, and the function uses its built-in extractData to output the data and labels we need for the training set and test set, which are XTrain, YTrain, XTest, and YTest, respectively; and then we use the normalizeData function to normalise the training and test data so that each data sample has zero mean and unit variance, which helps the subsequent model training effect and prediction performance. specify network function

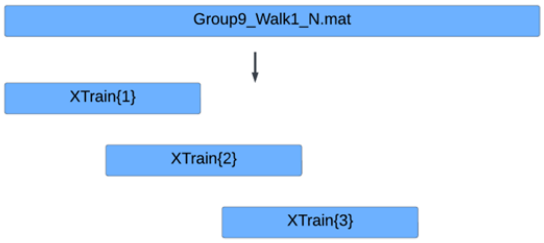


**Figure 2:** Windowing

defines the network structure and training options, and also defines parameters such as the maximum number of iterations for training, the batch size and the learning rate. Then the trainSillyWalkClassifier function, we take XTrain and YTrain as inputs, and use the previously defined network to train a model and store it. For the classifyWalk function, its inputs are our previously trained model and XTest, and its outputs are the nomal and silly Walk results judged by the model according to XTest. Comparing the model generated results with YTest, we can conclude the accuracy of the model's classification.

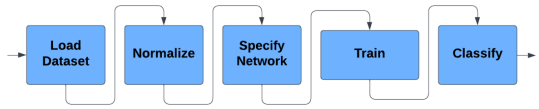


**Figure 3:** Train and classify procedures

We next focus on analysing the specify network function, because one of the influences on the classification effect of the classifier is the quality of the training set, and the other is the model, when training the network, we compared the two neural network structures of the LSTM and the GRU, and Figure 4 shows the LSTM layer structure and Figure 5 shows the GRU structure. After bringing it into the test set for testing, we found that the GRU showed a better classification effect, so we ended up using the GRU layer to train the neural network.

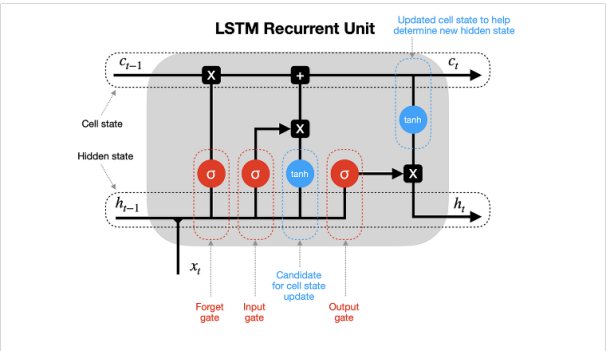


**Figure 4:** Long Short-Term Memory (LSTM)[1]

## 4. GUI

The first Tab in the GUI allows the training data and test data to be selected for import. Before the data is processed, the sampling rate and window length can be set by the user. The data will be imported when the button 'Data Load' is clicked. Meanwhile, to train the model later, the data will be normalized by clicking on the 'Data normalize' button. After clicking on the button 'Training and Classification' in the second Tab the accuracy will be displayed in the text box and also in the UI component Gauge, where we have set 80% as the threshold.The model used for training will be displayed in the text box at the bottom of Tab 2.

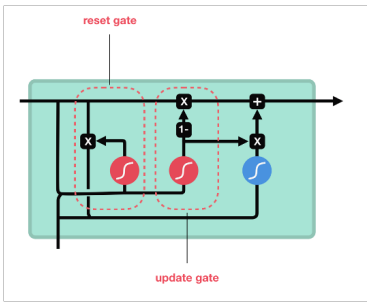


**Figure 5:** Gated Recurrent Unit (GRU)[2]

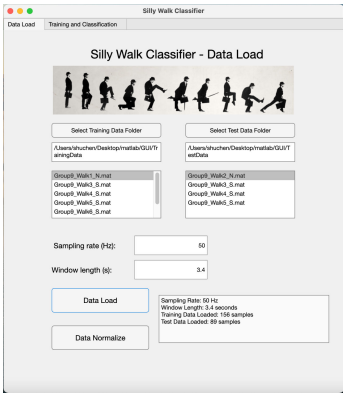


**Figure 6:** GUI Layout 1

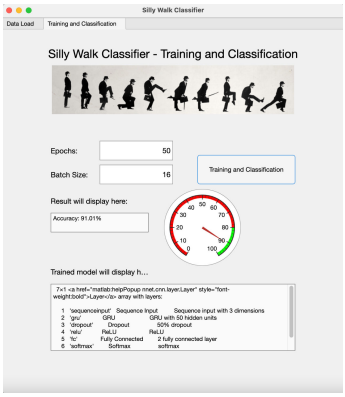


**Figure 7:** GUI Layout 2

## References

[1] Saul Dobilas. Lstm, 2022. Accessed: 2024-07-12.

[2] Gru. Accessed: 2024-07-12.