Documentation For Monty Matlab Project

Group 3: Mingcong Li, Tianyuan Kong, Wen Bing, Yueqiu Wang

July 13, 2023

1 Introduction

The Monty MATLAB project aims to develop a classification application for detecting silly walks. This report provides an overview of the project's development, focusing on differentiating between "normal walk" and "silly walk" using smartphone-collected data. We employ two algorithms, the k-nearest neighbors algorithm (k-NN) and the long short-term memory algorithm (LSTM), to train the classification model. Through rigorous experimentation and result comparisons, we select the more effective algorithm for integration into graphical user interface (GUI).

This report is structured as follows: Section 2 elaborates on the methodology used to design the application. Section 3 presents the results obtained from various experiments and provides a comparison analysis. Finally, in Section 4, we draw the conclusion and discuss the potential future work.

2 Methodology

2.1 Data Collection

An iPhone XR running the MATLAB Mobile App was used to record acceleration data during the data collecting process. Acceleration data in the X, Y, and Z directions was collected for training the classification model. To simulate the scenario of a smartphone placed in the right back pocket with the Y-axis turned towards the ground, we held the smartphone on the right side of our bodies, behind the waist, due to limitations of clothing.

A total of 15 runs of group members walking normally and 20 runs of group members performing Monty Python's Silly Walks were conducted. The data was sampled at a rate of 50 or 60Hz to ensure accurate representation of the movements with multiple data points recorded per second.

2.2 Data Preprocessing

During the data preprocessing process, for each collected walking data, we cut off pre- and post walks, ensuring that only effective walks were contained in the data set. This step was essential to obtain accurate results of our classification models. The processed data, consisting of acceleration measurements in three directions, was organized into a 3×M matrix. Additionally, a 1×M array representing the true time point for each data point was incorporated. Both the data matrix and the time array were saved in a .mat file.

2.3 Data Extraction

The data extraction function is implemented in the MATLAB script "extractData.m". Each sample in a MAT-File is divided into smaller parts of data by shifting a window of 3.4 seconds with a 50% overlap. This function generates an abundant amount of training data, with a size of 3×170.

2.4 Training Algorithms

2.4.1 K-NN

K-NN is a commonly used non-parametric supervised learning algorithm. It classifies objects based on a vote from its nearest neighbors, assigning the object to the class that is most common among its k nearest neighbors. One of the advantages of k-NN is its shorter training time compared to LSTM. However, as a machine learning algorithm, feature extraction is an essential preprocessing step. The classification results highly depends on what kind of features are chosen to train the model.

2.4.2 LSTM Neural Network

LSTM networks are particularly suitable for modelling temporal dynamics in activity time-series and addressing the vanishing gradient problem. They perform well in scenarios where capturing and utilizing long-term dependencies are necessary for making predictions or classifications based on sequential data. An LSTM unit consists of essential components, including a cell, an input gate, an output gate, and a forget gate. These components enable the LSTM to capture relevant information from previous inputs and use them to mod-

ify the current output. The function "trainSillyWalkClassifier.m" is applied to specify the neural network architecture and train the model.

2.5 Classification and Evaluation

The files "classifyWalk.m" and "evaluate.m" are used for testing and evaluating the model, respectively. The predictions obtained from "classifyWalk.m" are compared with the correct labels, and the accuracy of the models is calculated in "evaluate.m". The evaluation process generates a more intuitive confusion chart.

2.6 Graphical User Interface (GUI)

In our GUI section, we implemented a startup interface called "Silly Walk Detection Lab". The main functionality is to classify user's gait. Click "Import my gait!" to launch the other GUI for analyzing walking posture and performing silly walk detection. To accomplish this, walking data needs to be imported, which will be further displayed as a curve in the GUI. In addition, if the user want to choose another data, simply clicking "Choose another Data". There is also a secondary functionality by clicking "How does Silly Walk look like?", for those, who want to know how the Silly Walk looks like, clicking "I want to learn silly walk!".

3 Experiments and Results

3.1 k-NN

For the k-NN model, we selected the mean value, standard deviation, sum of magnitude, range of magnitude and maximum magnitude of data as features in the directions of X, Y, and Z. We set the number of nearest neighbors k equal to 10, in order to obtain the best result with this model.
After testing the model, a confusion chart was generated, as shown in Fig. 1. Using the test data, the k-NN model's accuracy was 88.6%.

3.2 LSTM Neural Network

The neural network architecture for the LSTM model consists of the following layers:
i) Input layer: Processing sequences of data. ii) LSTM layer: Learns long-term dependencies between time steps in sequences. iii) Dropout layer: Reducing overfitting by randomly dropping 50% of the outputs. iv) Rectified Linear Unit (ReLU) layer: Introducing non-linearity. v) Fully connected layer: Interpreting the learned features. vi) Softmax layer: Generating probabilities for each class. vii) Classification layer: Determining the most probable output class.
After fine-tuning the number of hidden units in the LSTM layer and the mini-batch size, we obtained an accuracy of 98%, as shown in Fig. 2.

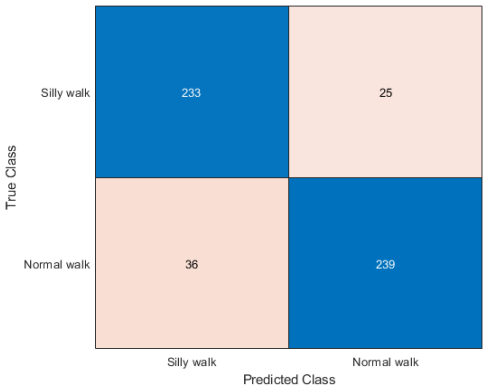

Figure 1: Confusion chart (k-NN)

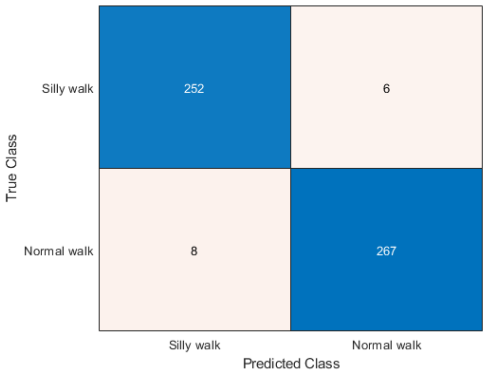

Figure 2: Confusion chart (LSTM)

4 Discussion and Conclusion

The results clearly demonstrate that the accuracy of the LSTM model is significantly better than that of the k-NN model with the currently chosen features. Although the training time for the LSTM model is slightly longer, LSTM model offers advantages in our situation, particularly as it eliminates the need for feature extraction. Based on the comparison of results, our group has decided to use the LSTM model for the development of our application.
For future work, there are several areas we can explore. For instance, we can further improve the k-NN model by finding better features. Additionally, we can also extend our classification system to differentiate between different types of silly walks.