CLASSIFICATION OF THE WALK

Yiming Shan[1], Ou Tang[1], Taiyu Liu,[1]

[1]Chair of Data Processing, Technical University of Munich, Germany

1. INTRODUCTION

A silly walk is a term that was popularized by British comedian John Cleese in a sketch from the television show "Monty Python's Flying Circus." In the sketch titled "The Ministry of Silly Walks," Cleese plays a civil servant who demonstrates a series of exaggerated and absurd walks in a government office.

The concept of the silly walk is to perform ordinary actions, such as walking, in a deliberately ridiculous and exaggerated manner. It involves unconventional movements, strange gestures, and exaggerated leg and arm swings that defy normal walking patterns. The result is a humorous and entertaining display of physical comedy. The recent research from British Medical Journal says Silly walk from Monty Python keeps a person fit.Anyone who walks like British actor John Cleese in the famous Monty Python sketch "Ministry of Silly Walks" uses 2.5 times as much energy as someone who walks normally. And in order to distinguish silly walk and normal walk more accurately, we intend to design a Matlab program to do so.

2. DATA PREPARATION

2.1 Data Collection and Preprocessing

We utilize the Matlab Mobile App for recording walking data. By enabling the accelerometer and setting the sampling rate to 60Hz, we place the phone in the right back pocket with the Y-axis facing downwards. After 50 seconds of walking, the sensor is deactivated. In terms of walking posture, we replicate some walks from YouTube videos while also capturing data from our own playful movements.

We collect a total of 132 sets of walking data. Considering that some silly walks only involve exaggerated hand movements and the leg actions are similar to normal walking, we remove 8 sets of poor-quality data. Among the remaining 124 sets of data, 12 sets are randomly chosen as test data, while the other 112 sets are used for training.

After obtaining the raw data, we perform the following processing steps: First, we remove the initial and final 10 seconds of data to ensure that all data points were recorded while in motion. Then we modify the data format. The data collected by MATLAB Mobile is in an N×3 matrix format, with a timestamp recording the real-time. We stored the data as a 3×N variable and generated a new timestamp starting from 0 using the $linspace$ function. After processing, the data was named according to the standard, with either "$_S$" or "$_N$" indicating the type. The preprocessing is done by the function $preprocessing.m$.

2.2 Data Extraction

The $extractData$ function extracts data from a specified folder by using the given samplingRateHz and windowWidthSeconds. It consists of three steps.

The first step is resampling. We use interpolation to supplement the data. Then selects data points based on the new sampling rate. The $interp1$ function with default settings is used for resampling the acceleration in the x, y, and z directions, which are saved in the variable "resampled_data." This portion is handled by the $resampleData.m$ function.

The second step is performing windowing. Based on window -WidthSeconds, the number of windows is calculated using the $floor$ function. Each window is shifted by 50 percent and saved in the variable "X". This step is completed by the $performWindowing.m$ function.

Finally, based on the data file names, class labels are assigned using the $categorical$ function. This part is written at the end of the $DataExtraction.m$. The extracted data will be used for later training and testing.

3. ALGORITHM

The GRU Algorithm. Since the data collected through the sensors is time-dependent, here we use a time sequence classification algorithm. The commonly used algorithms are LSTM and GRU, and SVM, etc. Here we choose the GRU model. GRU has fewer parameters and thus converges more easily(Wang et al., 2017), but the LSTM expression performs better for large datasets. Structurally, GRU has only two gates (update and reset), while LSTM has three gates (forget, input, output). GRU directly passes the hidden state to the next cell, while LSTM wraps the hidden state with a memory cell. Here, we choose the more lightweight GRU model.

The input-output structure of GRU is the same as that of a normal RNN. There is a current input x^t, and a hidden state h^{t-1} delivered from the previous node, which contains the information about the previous node. Combining x^t and h^{t-1} , GRU will obtain the output of the current hidden node y^t and the hidden state h^t passed to the next node.

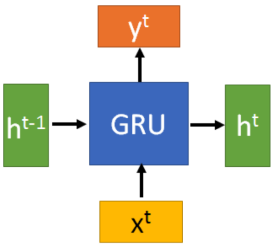

Figure 1. The input and output structure of GRU.

Next, we obtain the two gating states from the last transmitted state h^{t-1}, and the input x^t from the current node, as in Fig.2, where r is the gating control for reset and z is the gating control for update. σ is a sigma function that transforms data to a value in the range 0-1 to act as a gating signal. The detail process is shown in Figure 2. where $\odot$ is the Hadamard Product, i.e., the multiplication of the corresponding elements in the operation

matrix. $\oplus$ then represents a matrix addition operation.

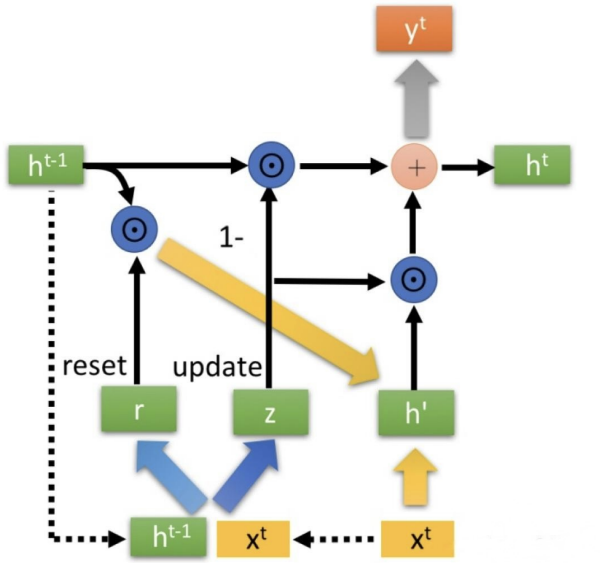

Figure 2. The structure of GRU.

4. GRAPHICAL USER INTERFACE

We have developed a graphical user interface (GUI) as an alternative to calling functions in the command line of Matlab, as illustrated in Figure 2. This GUI provides a more intuitive way for users to interact with the program. It allows them to choose the data folder, specify the values of sampling rate (samplingRateHz) and window width (windowWidthSeconds), call the training and test functions, and view the test accuracy.

In the GUI, users can easily select the training and test data folders by clicking on the corresponding buttons. If the user does not specify a folder, the default names for the training and test data folders are "TrainingData" and "TestData," respectively. These folders are created in the current directory. Additionally, users have the flexibility to modify the default values of the sampling rate and window width by entering any positive number.

Once the user clicks the "Train" button, a new pop-up menu displays the training process, allowing them to monitor its progress. Simultaneously, the trained model named "Model.mat" is automatically generated and saved.

To evaluate the performance of the model, users can click the "Test" button. The test accuracy is quickly calculated and displayed. With the Graphic button, you can get a graph of the test data.The classified results will be displayed in the result interface.Moreover, users can employ the model selection button to test a different model named "Model.mat" on the test dataset, facilitating the comparison of different models.

Overall, this GUI provides an efficient and user-friendly interface, enhancing the ease of use and enabling users to interact with the program in a more visual and intuitive manner.

5. CONCLUSION

After several tests, the accuracy of our algorithm is maximum 97.5% and above 96% in most cases. However, due to the fact that the amount of data is not sufficient, in rare cases, it may be over-fitting. If we can increase the amount of data and introduce more types of data, the algorithm will be more stable and reliable, and the robustness will be stronger.

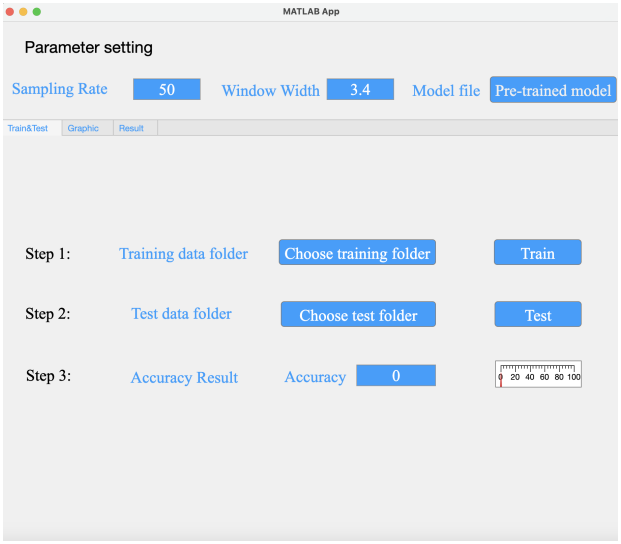

Figure 3. GUI

REFERENCES

Wang, W., Yang, N., Wei, F., Chang, B., Zhou, M., 2017. Gated self-matching networks for reading comprehension and question answering. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 189–198.