Progress Check #2: Deep Learning-Based Gait Recognition Using Smartphones in the Wild

Aniruddha Anand Damle[1] Prakriti Biswas[1] Aditya Shrikant Kaduskar[1]
[1]Ira A. Fulton Schools of Engineering, Arizona State University

***Abstract* – We have made use of an application named 'AndroSensor', from the Google Playstore. This application provides a text file or sends an email to the user with the accelerometer and gyroscope data from the phone. Our peers used the aforementioned application and provided us with their gait data. We have further used this new dataset on the codes made available by the authors of the original paper. We have studied the paper in detail and implemented the methods proposed by the authors of the original paper, for gait recognition using smartphones. In this report, we discuss the implementation of the various methods, the experimental results, and any difficulties we have faced during the course of this project.**

I. Problem statement

The method proposed by the authors of the paper[1], to identify and authenticate a person by the way they walk, comprises three steps: gait data extraction, identification, and authentication. As a result of the extraction part of the proposed method, the inertial data collected by smartphones is partitioned into walking and non-walking sessions, gait features are then extracted from the walking data. The person identification and authentication models are constructed based on the results from the feature extraction step. Compared to other biometrics, gait is difficult to conceal and has the advantage of being unobtrusive. Inertial sensors, such as accelerometers and gyroscopes, are often used to capture gait dynamics. These inertial sensors are commonly integrated into smartphones and are widely used by the average person, which makes gait data convenient and inexpensive to collect. In this paper, we study gait recognition using smartphones in the wild. In contrast to traditional methods, which often require a person to walk along a specified road and/or at a normal walking speed, the proposed method collects inertial gait data under unconstrained conditions without knowing when, where, and how the user walks. To obtain good person identification and authentication performance, deep-learning techniques are presented to learn and model the gait biometrics based on walking data. Specifically, a hybrid deep neural network is proposed for robust gait feature representation, where features in the space and time domains are successively abstracted by a convolutional neural network and a recurrent neural network.

II. Methods

A. Data Extraction

1. Introduction

We have no idea when, where, or how cellphones will be utilized to capture inertial data in the wild. As a result, the collected data includes both walking and non-walking sessions, although gait feature extraction and person identification are only interested in walking data. As a result, the continuous inertial sequence gathered in the wild by smartphones must be partitioned.

2. Architecture

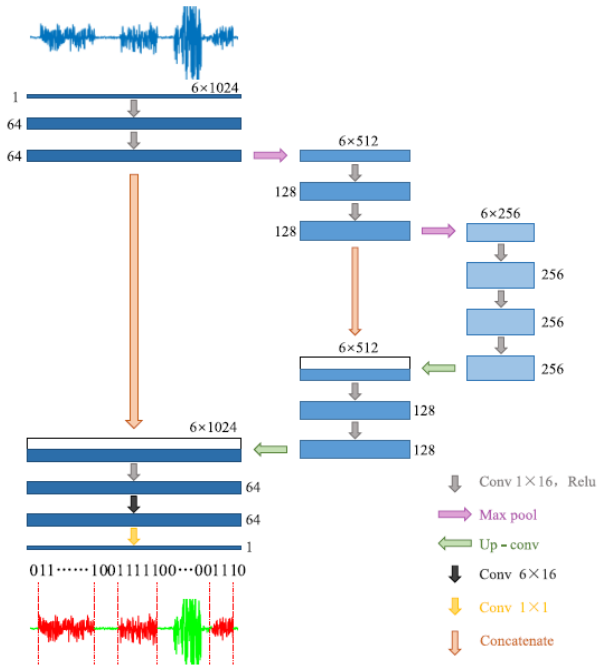

Fig. 1:Architecture of the network for gait data extraction.

Layer Name	Kernel Size	Kernel Num.	Stride	Feature Map
conv1_1	1x16	64	1	6x1024x64
conv1_2	1x16	64	1	6x1024x64
pool1	1x2	/	2	6x512x64
conv2_1	1x16	128	1	6x512x128
conv2_2	1x16	128	1	6x512x128
pool2	1x2	/	2	6x256x128
conv3_1	1x16	256	1	6x256x256
conv3_2	1x16	256	1	6x256x256
conv3_3	1x16	256	1	6x256x256
upconv1	1x2	128	I	6x512x128

concat1	/	/	1	6x512x256
conv4_1	1x16	128	1	6x512x128
conv4_2	1x16	128	1	6x512x128
upconv2	1x2	64	1	6x1024x64
concat2	/	/	/	6x1024x128
conv5_1	1x16	64	1	6x1024x64
conv5_2	6x16	64	1	1x1024x64
conv5_3	1x1	1	1	1X1024X1

Table 1: Details of the Data Extraction Network Structure

Considering that walking data and non-walking data are semantically different and that inertial time series are continuous in both the space and time domains, we model the partitioning problem as a time-series segmentation problem. Without loss of generality, we select accelerometer data as the basis for partitioning. As the smartphone can be moved in random directions by the user, a single axis of acceleration or gyroscope cannot stably reflect the fluctuations for the gait curve. Meanwhile, the absolute acceleration along the axis perpendicular to the ground is the largest among the three values. To remove the influence of the phone's orientation, we process the triaxial acceleration data to obtain ACC_o as the basis for gait cycle segmentation. ACC_o is calculated by $ACC_o = \sqrt{ACC_x^2 + ACC_y^2 + ACC_z^2}$, where ACC_x, ACC_y and ACC_z denote the values of acceleration in the X, Y, and Z directions,respectively.

Dataset Name	Number of Subjects	Samples for Training	Samples for Test
Dataset #7	10	519	58
Dataset #8	118	1022	332

Table 2. Detail Information of the Gait-Data Extraction Datasets

Two datasets are constructed for evaluation of the proposed gait-data-extraction method. Basic information of the two datasets have been shown in Table 2, and the details are given as below:

a) **Dataset #7:** it contains 577 samples of 10 subjects, with data shaped as 6 × 1.024. Among these samples, 519 are used for training and 58 are for test. Both the training and test samples are from the 10 subjects.

b) **Dataset #8:** It contains 1,354 samples of 118 subjects, with data shaped as 6×1.024. Among these data, 1,022 samples from 20 subjects are used for training, and 332 samples from the other 98 subjects are used for test. For both datasets, each sample is attached with a label file, which contains 1,024 binary values, with '1' as the walking data, and '0' as the non-walking data. The labels are manually annotated.

B. Identification

1. Introduction

Given an inertial gait curve **X** with a sampling length of **T** , **X** can be expressed as:

$$\mathbf{X} = (x_1, x_2, \ldots, x_t)$$

where $X_t = (ACC_x, ACC_y ACC_z, GYR_x, GYR_y, GYR_z)$, (1) where (ACC_x, ACC_y, ACC_z) and $(GYR_x, GYR_y GYR_z)$ denote the accelerometer and gyroscope components along the X, Y and Z axes at time t, respectively. Then, the problem is how to recognize the identity of a subject based on input data x. To formulate this problem, suppose $S = (S_1, S_2 \ldots, S_N)$ is the classes allotted to n number of candidates, where S_i is the i^{th} subject. Then, the output can be represented as an n-dimensional vector,

$$\mathbf{O = (O_1, O_2, \ldots, O_n),}$$

where $\mathbf{O_i = P(S_i|x)}$, i.e., the possibility that x belongs to si . Further suppose that s is the identity of the input data x; then, it is formulated as

$$\mathbf{S = argmax\ S_i\ \{O_i \mid 1 \le i \le n\}.}$$

Thus, to solve the problem of gait identification, we have to associate the maximum possibility values with the corresponding subjects.

2. Architecture

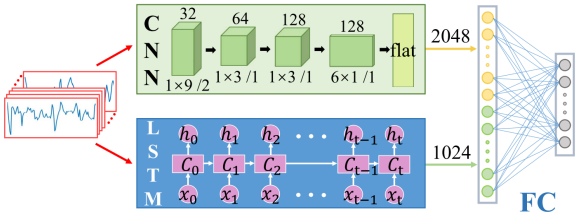

Fig. 2:The network architecture for gait identification.

As illustrated in Figure 2, the gait identification network consists of a CNN and an LSTM in parallel. The CNN and LSTM networks are feature extractors that obtain the corresponding features – $Feat_{CNN}$ and $Featl_{LSTM}$. The fully connected layer works as a classifier and uses the feature vector obtained by concatenating $Feat_{CNN}$ and $Featl_{LSTM}$ as the input.

C. Authentication

1. Introduction

After receiving the output of the Identification segment of the project, which are the identities of all the subjects involved, the output of the Identification model is fed into the Authentication segment of the project, to authenticate their identities. This segment utilizes an architecture that is a hybrid of a fixed-parameter CNN and LSTM.

2. Architecture

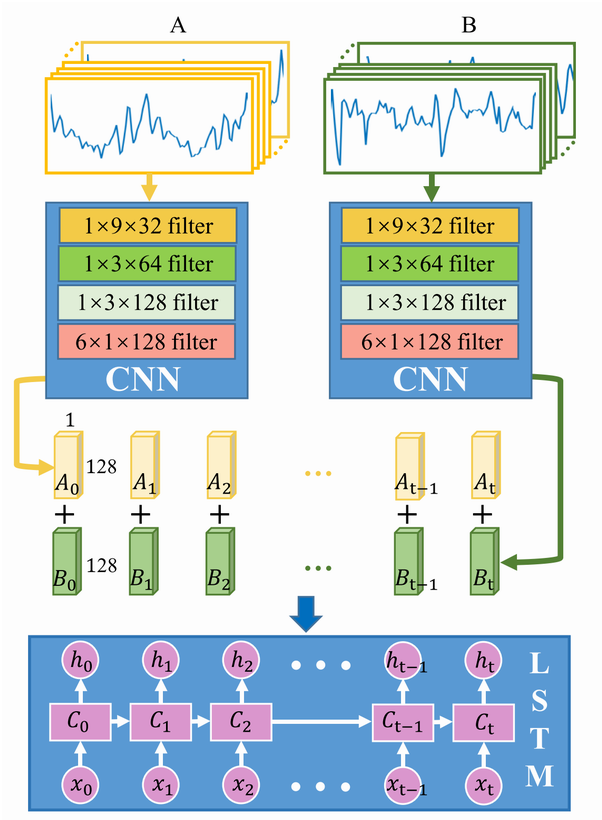

Fig. 3: Architecture for the network for Authentication

Let two sequences of gait data x_a and x_b be the input of the authentication network,

$$x_a = (x_{a,1}, x_{a,2}, x_{a,3}, \ldots, x_{a,T})$$

$$x_b = (x_{b,1}, x_{b,2}, x_{b,3}, \ldots, x_{b,T})$$

Where,

$$x_t = (ACC^t{}_x, ACC^t{}_y, ACC^t{}_z, GYR^t{}_x, GYR^t{}_y, GYR^t{}_z)$$

And T is the length of the sequence. authentication is formulated as a binary classification problem. The output of the network is set as two dimensions. The authors of the paper have used 'True' and 'False' to denote that the input data are from the same subject and different subjects, respectively. To fully utilize the advantages of the CNN and RNN, the CNN is used as a feature extractor to map the input inertial signals into lower-dimensional abstractions. The CNN portion of the network uses 98 subjects of the 118 total subjects, for training purposes. The remaining 20 subjects are used to test the CNN architecture. The 98 subjects have nothing in common with the remaining 20 subjects to prevent overfitting. The input to the CNN is the gait signal with dimensions of 6 rows and 128 columns. The 6 rows represent the x,y, and z signals of the accelerometer and the gyroscope, and the 128 columns represent the number of samples. The output of the CNN is based on the dimension requirements of the LSTM. The output of the CNN has the dimensions 1x16x128, that is, 1 layer of a 16x128 matrix. The output comprises 16 features along the time axis for the 128 samples.

The CNN features are rearranged into 16 blocks of 256 features, and fed into the LSTM block for training and prediction.

III. Implementation and Simulation

A. Data Extraction

To collect our own dataset, we installed the "AndroSensor" android app on our smartphones. We gave this app to our friends and gathered their walking data. The dataset is being compiled, and once it is complete, it will be posted to our GitHub account. We wrote a Python script. to convert this data to the format of the neural network's input.The accelerometer and gyroscope have a 50Hz frequency, and the data from these two sensors is captured in real time. The time stamp, the triaxial values of the acceleration sensor, and the triaxial values of the gyroscope are among the seven dimensions of the collected data.

B. Identification

A number of network structures, including LSTM-based, CNN-based, and CNN+LSTM-based, are designed for gait classification, and their performances are compared in terms of accuracy. For LSTM-based methods, each hidden layer in the LSTM has N = 64 hidden nodes, the learning rate is set to 0.0025, and the number of epochs for training is 200. For the hybrid method, N = 1024 is set for LSTM. For CNN-based methods, the six-axis interpolation data are used as the input, with the data shaped as 6 × 128. The classification experiments are conducted on the first four datasets introduced in Section V-A. When training the CNN, the learning rate is 0.0025, and the number of epochs for training is 200.

C. Authentication

The original paper compares the performance of 4 models i.e. CNN, LSTM, CNN_{fix}+LSTM, and CNN+$LSTM_{fix}$. Since the CNN_{fix}+LSTM performs better for authentication, we have decided to use this model. As discussed in the original paper, we have implemented a model where the parameters of the CNN are fixed and the LSTM network, which comprises a hidden layer with 64-nodes, is trained with a learning rate of 0.0025, 300 epoch, and a batch size of 1500. The authentication step of this project uses datasets 5 and 6 for training and testing the networks. The data samples from dataset 5 and 6, comprise 2-step accelerometer and gyroscope data, that are interpolated into a fixed length of 128 samples using a linear interpolating function. In the 5th dataset, the samples are horizontally aligned to create an authentication sample, as opposed to the 6th dataset where the samples are vertically aligned.

IV. Results

A. Data Extraction

1. Data Collection: The following table shows a snippet of the data collected using the android application.

ACC_x	ACC_y	ACC_z	GYR_x	GYR_y	GYR_z
5.4718	-4.9038	8.5472	-2.1584	-2.8953	0.3885
7.6155	-5.0602	9.4259	-2.2623	-1.9091	1.2344
9.8375	-2.2454	9.6667	-1.3594	-0.4714	1.0227
10.8174	-0.7675	8.2726	-0.9212	0.3715	0.8147
13.8278	1.5915	1.7554	0.1788	-0.7287	0.5966

Table 3: Snippet of collected data

	ACCELEROMETER X (m/s²)	ACCELEROMETER Y (m/s²)	ACCELEROMETER Z (m/s²)	LINEAR ACCELERATION X (m/s²)	LINEAR ACCELERATION Y (m/s²)	LINEAR ACCELERATION Z (m/s²)	GYROSCOPE X (rad/s)	GYROSCOPE Y (rad/s)	GYROSCOPE Z (rad/s)
0	3.5450	-1.6642	6.8910	-0.8329	-1.5699	-0.4815	-1.2360	-3.1367	0.0426
1	5.4718	-4.9038	8.5472	-2.3322	-3.7456	-0.2938	-2.1584	-2.8953	0.3885
2	7.6155	-5.0602	9.4259	-2.3534	-3.2655	3.3445	-2.2623	-1.9091	1.2344
3	9.8375	-2.2454	9.6667	2.2775	0.8591	6.2060	-1.3594	-0.4714	1.0227
4	10.8174	-0.7675	8.2726	2.2049	2.4813	4.9761	-0.9212	0.3715	0.8147
...	...	...	...	...	...	...	...	...	...
13	1.1219	5.6100	8.0653	0.1660	0.3625	-0.4662	-0.2442	0.5722	-0.0157
14	-2.5146	6.8683	8.0342	1.2787	-0.4727	1.1521	0.2611	0.4783	-0.1368
15	-1.9029	6.6882	6.8862	-0.3601	0.1902	1.0755	0.7428	0.7242	0.0564
16	-2.4227	6.9826	6.6096	0.4953	-0.2253	0.2477	0.5174	0.5340	-0.0079
17	-2.3372	7.0301	5.7855	0.4783	-0.4549	0.7433	0.6076	0.5635	-0.0266

8 rows × 11 columns

Fig. 4: Snippet of Data collected from android application

The following is the code written for data formatting:

```
In [5]:
for i in range(3):
    data.drop(data.columns[3], axis=1, inplace=True)

In [6]:
for i in range(2):
    data.drop(data.columns[6], axis=1, inplace=True)

In [7]:
new_header = data.iloc[0] #grab the first row for the header
data = data[1:] #take the data less the header row
data.columns = new_header #set the header row as the df header

In [8]:
data.drop(data.index[1024:data.shape[0]])
```

Out[8]:

	3.5450	-1.6642	6.8910	-1.2360	-3.1367	0.0426
1	5.4718	-4.9038	8.5472	-2.1584	-2.8953	0.3885
2	7.6155	-5.0602	9.4259	-2.2623	-1.9091	1.2344
3	9.8375	-2.2454	9.6667	-1.3594	-0.4714	1.0227
4	10.8174	-0.7675	8.2726	-0.9212	0.3715	0.8147
5	13.8278	1.5915	1.7554	0.1788	-0.7287	0.5966
...	...	...	...	...	...	...
1020	1.8182	5.3504	7.8646	0.1735	-0.0852	-0.3283
1021	-0.0144	4.5921	8.4072	-0.2297	0.5159	-0.2307
1022	0.7735	5.5484	8.8288	0.3744	-0.4273	-0.0681
1023	0.6778	4.6463	9.7457	-0.0224	0.1958	-0.0837
1024	0.0640	4.5473	8.5412	-0.0484	0.0809	0.0808

1024 rows × 6 columns

```
In [9]:
np.savetxt(r'E:\Arizona\Assignments\SEM-2\EEE-511\Datasets\Dataset #9\001_Aniruddha\1.txt', data.values, fmt='%f')
```

Fig. 5: Snippet of code written for data formatting

2. Data Segmentation

We train the CNN network proposed in Section II-A-2 for gait data extraction. For Dataset #7 and Dataset #8, the learning rate is set to 0.0001, and the number of training epochs is set to 150. Figure 6 shows four sample results obtained by the proposed network. Most of the data are correctly classified: a small portion of walking data are extracted as non-walking (red on blue) and a small portion of non-walking data are extracted as walking (red on green). The misclassification occurs at the transition area between walking and non-walking, which is reasonable since there are uncertainties for those points in the transition area. Specifically, on Dataset #7, where the training data and test data have no overlap but are all from the same 10 subjects, the proposed method achieves an accuracy of 90.22%, which shows the effectiveness of the proposed method in separating walking data from non walking data. On Dataset #8, where the training data and the test data are from different subjects, an accuracy of 85.57% is obtained, which indicates that the proposed method has high generalization ability.

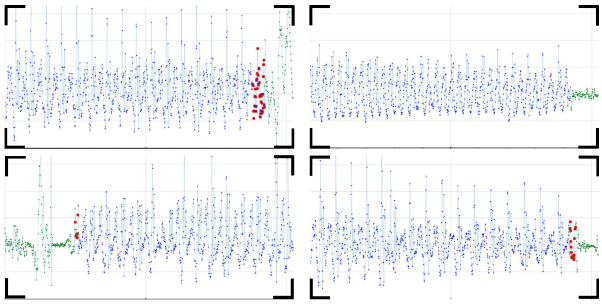

Fig. 6: Four examples of walking data extraction using the proposed method. Note that, the blue points denote the walking data, green points denote the non-walking data, and the red denotes the false classified.

B. Identification

The two Dataset #1, Dataset #2, are used to evaluate the seven deep learning-based methods: CNN, CNN_{fix}+LSTM, and CNN+$LSTM_{fix}$. The classification results are shown in Table. All the methods achieve greater than 91.8% accuracy on Dataset #1 which contains 118 subjects, and greater than 96.7% accuracy on Dataset #2 which consists of the gait data of 20 subjects.

Classification Methods	Dataset #1	Dataset #2
CNN	92.54%	95.10%
CNN+LSTM	91.42%	95.67%
CNN+$LSTM_{fix}$	**93.43%**	**98.43%**

Table 4: Classification Performance Of Deep Learning-Based Methods

C. Authentication

On Implementing the CNN and CNN_{fix}+LSTM architectures on dataset 5 and 6, we have received the following performance table (testing accuracies):According to these results, CNN_{fix}+LSTM performs more effectively on the vertical dataset (Dataset #6).

Authentication Methods	Dataset #5 (Horizontal)	Dataset #6 (Vertical)
CNN	78.3%	86.96%
CNN_{fix}+LSTM	85.48%	**93.7%**

Table 5: Authentication Performance based on Training Accuracy of models

V. Discussions and Conclusions

A. Discussion

It is assumed that the three axis accelerometer data and three-axis gyroscope data and their respective time-stamps are related to each other. Hence the tri-axis accelerometer and gyroscope gait data into six-axis inertial gait data. One dimensional convolution operations are applied to process the information among different axes. For the first three layers, one-dimensional convolutional kernels of sizes 1×9, 1×3, 1×3, where the signals are separately processed along the time zone.The proposed one-dimensional convolution kernels can guarantee the temporal property of the final convolution results. As the convolution is a local operation, the output features are supposed to also hold the property of time series. A 6 × 1 kernel is applied to the previous convolution results and spatially correlates the signals on different axes, and produce 1 × 16 × 128 data. Note that, 16 is the number of the features along the time axis. After rearranging it into 128 × 16, the required time-series is obtained.

While experimenting with the two models on datasets 5 and 6, for authentication, we have realized that the authentication task may be challenging, since the 98 subjects training CNN_{fix} is the same for datasets 5 and 6.

B. Conclusion

For robust inertial gait feature representation, a hybrid technique was presented that smoothly combined the DCNN and DRNN. The smartphones were used in unrestricted conditions during gait data collection, and no information regarding when, where, or how the user walks was available. The suggested hybrid network outperformed the standalone networks by a wide margin.

We got the opportunity to dive deep into machine learning frameworks such as Tensorflow, and development environments such as Jupyter Notebook and Google Colab. We were able to better understand many ideas related to the CNN and LSTM models. We discovered the many hyperparameters that have a direct impact on our model's accuracy.

We have yet to create and compile a method for performance evaluation of our datasets.

VI. Bibliography

[1] Q. Zou, Y. Wang, Q. Wang, Y. Zhao, and Q. Li, “Deep Learning-Based Gait Recognition Using Smartphones in the Wild,” *IEEE Transactions on Information Forensics and Security*, vol. 15. pp. 3197–3212, 2020 [Online]. Available: http://dx.doi.org/10.1109/tifs.2020.2985628