2019 16th International Conference on Ubiquitous Robots (UR)
Jeju, Korea, June 24-27, 2019

Deep Learning Based Real-time Daily Human Activity Recognition and Its Implementation in a Smartphone

Tsige Tadesse Alemayoh, Jae Hoon Lee, *Member, IEEE,* and Shingo Okamoto

Abstract— **A novel method for classifying and identifying human activities in real-time is needed in various human-machine interaction fields. In this paper, a multi-channel motion data collected from a smartphone is structured in a new way and converted to a virtual image. An iOS application software was developed to record and stream motion data and to recognize real-time activities. The time series data of an accelerometer and gyroscope motion sensors are structured into 14x60 virtual image. Similarly, their respective amplitudes of 1 dimensional DFT (Discrete Fourier Transformation) are organized into 14x60 image format. The resultant data was given to the designed CNN (Convolutional Neural Network) for classification. Both data structuring methods were analyzed and compared yet the time series data structuring showed a better result and attained an accuracy of 99.5%. Additionally, the model was tested for real-time activity recognition in a computer and smartphone and achieved an excellent result.**

I. Introduction

Human activity recognition (HAR) is a broad area of study mainly concerned with identifying specific movement or action of a person based on given input data. The activities can be typical ones, like walking, sitting, running or more focused ones like eating, reading or industrial operations. With the advancement of smart devices, wireless communication, and deep learning, daily human activity classification and recognition has got an increasing interest in areas like context-aware computing, smart assistive technologies for industries, where manual work remains dominant and in rehabilitation centers where human motion monitoring is essential. Mostly, input data signals are obtained from videos, where video frames are taken for analysis or multi-axis time-series IMU (Inertial Measurement Unit) devices (dedicated IMU sensors or smartphones). Comparably, wearable IMU sensors became a popular and convenient way of data collection mechanisms without an extensive installation of equipment and privacy issues. Particularly, smartphones are ubiquitous devices with reliable wireless communication, which made them a better choice than others.

The traditional human activity classification methods were creating domain-specific, sensor-specific, or signal processing-specific features of raw data from a sliding sampling window. The processed version of the data was used to fit a statistical and machine learning models such as SVM (Support Vector Machine) as in Anguita et al. [1] and a decision tree as in Bao et al. [2]. However, only shallow features can be learned resulting in an underperformance, especially in unsupervised learning. With the large variability of human activity way of performing, faster and deep discriminative feature extraction methodologies are essential. Deep learning methods have shown the capability and even achieve state-of-the-art results by automatically learning high-level and meaningful features from raw data. RNN (Recurrent Neural Network) by Singh D et al. [3], Hidden Markov Model by Yamato et al. [4] and CNN by Ha et al. [5], Jiang et al [6], Um et al [7] and Rueda et al [8] are among the most common deep learning methods used.

In large-scale data classification, CNN is competent to extract features from signals and it has demonstrated excellent performance in image classification, speech recognition, and sentences classification. Due to the local dependency and scale invariance advantages over other models, CNN has been the best candidate for time series related classification problems. To further improve the performance, CNN can be combined with other neural networks such as LSTM (Long Short-Term Memory), SVM at the expense of computational cost.

In this paper, a new way of time series data arrangement with CNN based classifier is presented. The 6-axis time-series data, 3-axis Acc (accelerometer) and 3-axis Gyro (gyroscope), and their corresponding Fourier transformed amplitudes are transformed into a virtual image. Due to the fact that most human activities are periodic, a CNN based on the frequency response of input signals is also an important topic to study. Finally, the deep learning model was converted to a model of Core ML [13], to be used in the iOS development IDE (Integrated Development Environment), Xcode. The proposed CNN has an outstanding capability to learn the salient features and distinguish the activities.

II. Data Structuring

The smartphone used was attached tightly on the waist of the subjects. Out of the various motion-related sensors of a smartphone, the 3-axis Acc and 3-axis Gyro were chosen for a better result.

An iOS application software was developed using Swift programming language on Xcode IDE. It has three functions: data recording in *CSV* format used for training the neural network parameters; real-time data streaming for instantaneous activity recognition on a computer; and activity recognition on the smartphone itself. Motion data of eight activities were collected. The activities are: walking, jumping, running,

T. T. Alemayoh, J. H. Lee and S. Okamoto are with Department of Mechanical Engineering, Graduate School of Science and Engineering, Ehime University, Ehime, Japan (corresponding author to provide email: lee.jaehoon.mc@ehime-u.ac.jp).

bicycle riding, stairs ascending, and descending, laying down, and still. The application software collects data at a sampling frequency of 50 Hz (50 readings per second) for a certain time.

Owing to the fact that most human activities are a sequence of a repetition of a single motion (e.g. walking is the combination of steps), cutting time-series data using a sliding window of a certain time interval has been a practice. The original sequence data is segmented in 1.2 seconds (60 readings) to form a 6x60 virtual image, where the top 3x60 Acc and the bottom 3x60 are filled with the Gyro readings.

In BenAbdelkader et al [9], it was indicated that the cadence of a normal walking person is 90 steps/min (1.5 steps per second). Considering all the activities, almost double of it, 1.2-second window size with 50% overlap, is considered so that activities with a slower period could benefit from the wider range.

To prevent fast vanishing of the features and to exploit deeper features from the spatial dependency of the time signals, the sensors data of each window are structured as shown in Fig. 1 in a 14x60 image. As it is depicted in the figure each 1-dimensional time signal is repeated once in the vertical stack. Due to the parameters of the neural network selected, the x-axis part of both sensors is represented thrice. This creates an equal representation of signals in the succeeding layers. Similarly, by carrying out the amplitudes of the 1D DFT of each 6-time series signals along their temporal axis, another 14x60 frequency-domain virtual image is created [10].

III. Neural Network Design

For its popularity in image recognition, a CNN with the architecture summarized in Fig. 2 is applied to the activities' one-dimensional virtual images prepared.

Each convolutional layer performs a 2D convolution on its inputs followed by a non-linear activation function, ReLU (Rectifier Linear Unit). To reduce the effect of internal covariance shift of activations, batch normalization was utilized, which forces each mini-batch input of a layer to have similar distribution throughout the hidden layers, as described in the original paper Ioffe et al [11]. Besides, it allows the use of larger learning rates to speed-up the optimization process.

The fact that the activity images do not have extreme features, made average pooling the preferable subsampling method. It is a smoother way which can take in to account the effect of all the elements within the pooling window on its output map.

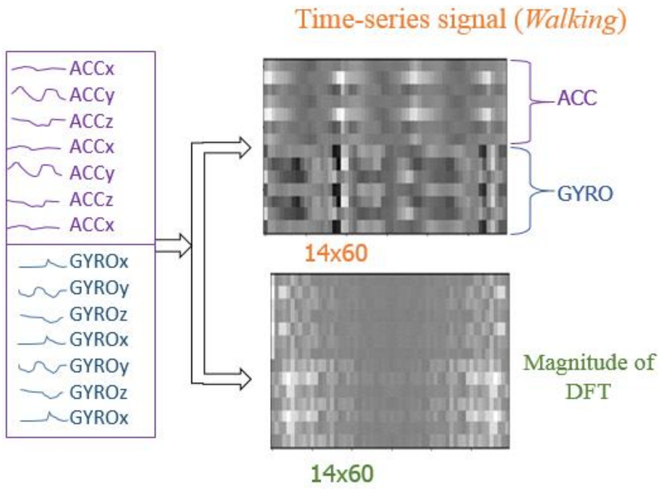


Figure 2. Sensor data structuring

After the output of the second pooling is flattened to form a long 1D feature map vector, the classification is decided by the probability distribution of an eight-class softmax layer.

All the parameters of the network are updated by Adam optimizer using the back-propagation principle according to the original work of Kingma et al [12].

IV. Conversion To core ML Model

Core ML is a framework used to integrate machine learning models into an iOS application. It is the foundation for domain-specific frameworks and functionality. Especially, it supports *Vision* for image analysis, *Natural Language* for natural language processing, and *GameplayKit* for evaluating learned decision trees (Developer.apple.com, 2019), [13]. Depending on the deep learning library a previously trained model can be converted to a core ML model using built-in or third-party converters. In our case, TensorFlow, a third-party library is adopted for conversion [14]. The converted model is like a system box, composed of the neural network structure and the learned weights.

The converted model can only be accessed through its input and output nodes. Firstly, the relevant sensor data are structured and prepared using the *MLMultiArray* Swift programming language class. The prepared multidimensional data is given to the input node of the model for further processing.

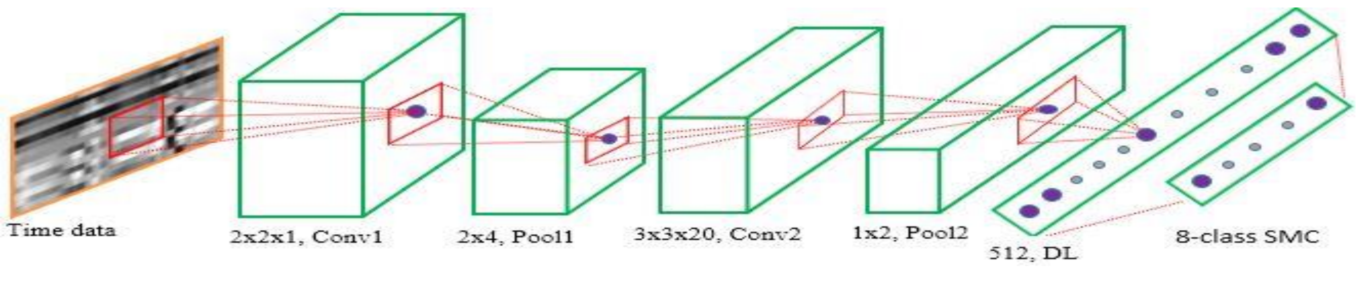


Figure 1. CNN architecture of time data. Conv, Pool, DL, and SMC denote the outputs of convolution, pooling, dense layer, and softmax classifier respectively

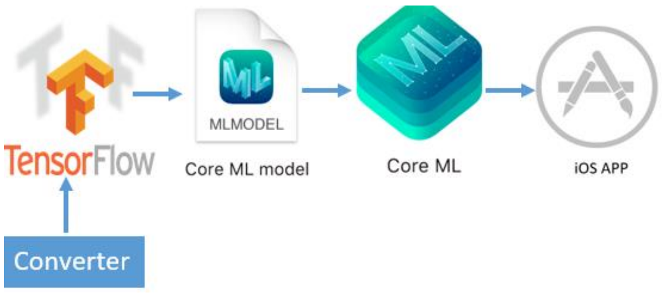


Figure 3. TensorFlow model to ML model conversion

After the ML model did the mathematical calculations, an output, class label, is given as a result, which is finally displayed on the screen. The main elements used in converting and integrating the model with the iOS development environment are summarized in Fig. 3.

V. Experimental Results

The deep-learning experiments were carried out with a 16GB RAM, 3.6 GHz Core-i7 processor with NVIDIA GeForce GTX GPU computer running TensorFlow. Already built-in functions and optimizers such as Adam optimizer, batch normalization, softmax, and others were employed for the neural network learning.

Eight subjects have participated in data collection with an average of 18.7 minutes per person to accomplish the 8 activities. The total number of dataset recorded is 14962, which include 11962 training sets (79.9%), 2,000 testing sets (13.4%), and the rest 1,000 (6.7%) development sets. The neural network was trained with a mini-batch size of 50 and an exponentially decaying learning rate to speed up the training.

Fig. 4 shows the performance of time-based, frequency-based with the same model and the same datasets. The time data model has outperformed its counterpart, the frequency data model as it is revealed in the figure. The time data model quickly learns essential features and attains its maximum accuracy earlier, while the other took time to settle. The final accuracies are 99.5% and 99.0% for time-based and DFT-based models respectively.

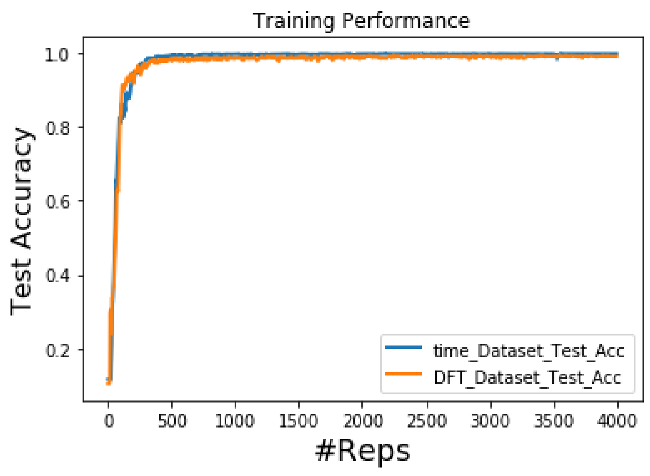


Figure 4. Test accuracies of both time and frequency datasets

Table. I shows the confusion matrix implemented over the development dataset, using the already learned weights. It is shown that a good classification result was achieved, as can be seen from the diagonal cells, which represent the matching of the predicted with actual activity.

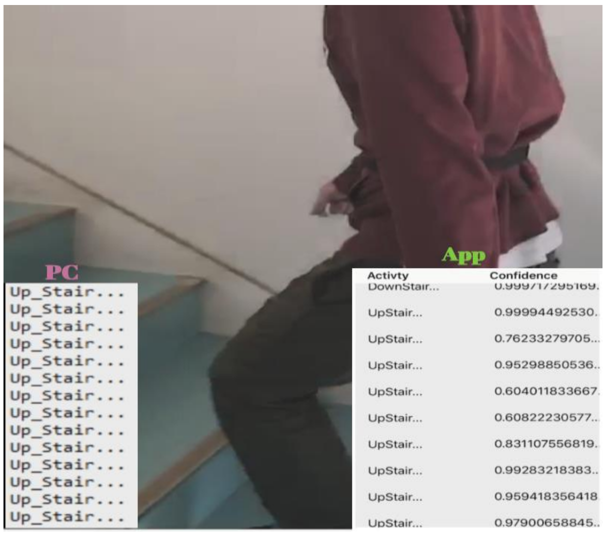
An iPhone 7 was used throughout the experiments. Before exporting the model to the iOS application, it was tested in a computer for real-time recognition, using the data streaming functionality of the iOS application developed. The phone sends the required data over TCP (Transmission Control Protocol) and the computer uses the pre-trained model to classify the activities in real-time. Snapshots of the recognition are displayed on the bottom left of Fig. 5 and Fig. 6.

Finally, the deep learning model is converted to an ML model, convenient for the iOS IDE using the best currently available third-party converter library, *tfcoreml*. However, the converter supports an only limited number of TensorFlow operations. This affects the accuracy of the iOS application's real-time activity recognition of some of the activities to some extent. For new subjects, who haven't participated in the data collection, the results were satisfactory for most of the classes. Some of the iOS application recognition results in real-time are shown on the bottom right of Fig. 5 and Fig. 6.

TABLE I. CONFUSION MATRIX

Actual Activities	**wk**	191	0	0	0	0	1	0	0
	Jp	0	92	0	0	0	0	0	0
	us	3	0	126	0	0	2	0	0
	ds	2	0	0	135	0	0	0	0
	rn	0	0	0	0	103	0	0	0
	br	0	0	1	0	0	142	0	0
	st	0	0	0	0	0	1	120	0
	ly	0	0	0	0	0	0	0	81
		wk	jp	us	ds	rn	br	st	ly
	Predicted Activities								

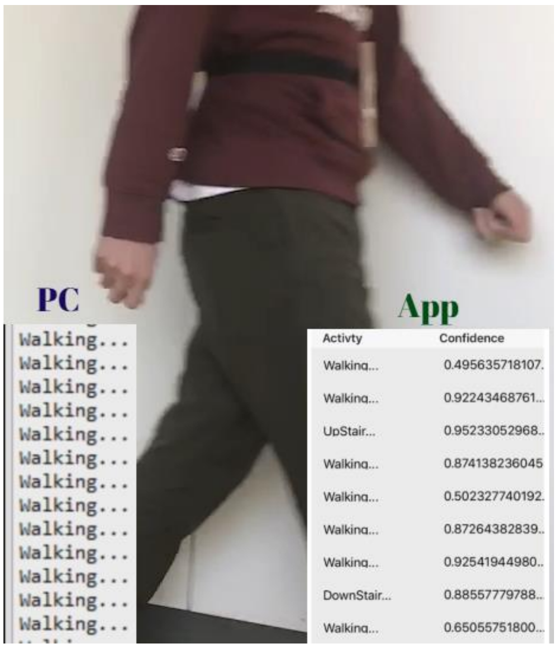
wk: walking, jp: jumping, us: upstairs, ds: downstairs, rn: running, br: bike riding, st: still, ly: lying down



```
Up_Stair...
Up_Stair...
Up_Stair...
Up_Stair...
Up_Stair...
Up_Stair...
Up_Stair...
Up_Stair...
Up_Stair...
Up_Stair...
Up_Stair...
Up_Stair...
Up_Stair...
Up_Stair...
```

Activty	Confidence
DownStair...	0.999717295169.
UpStair...	0.99994492530.
UpStair...	0.76233279705...
UpStair...	0.95298850536..
UpStair...	0.604011833667.
UpStair...	0.60822230577...
UpStair...	0.831107556819.
UpStair...	0.99283218383..
UpStair...	0.959418356418
UpStair...	0.97900658845..

Figure 5. Real-time recognition snapshot of upstairs walking



```
Walking...
Walking...
Walking...
Walking...
Walking...
Walking...
Walking...
Walking...
Walking...
Walking...
Walking...
Walking...
Walking...
```

Activty	Confidence
Walking...	0.495635718107.
Walking...	0.92243468761...
UpStair...	0.95233052968..
Walking...	0.874138236045
Walking...	0.502327740192.
Walking...	0.87264382839..
Walking...	0.92541944980..
DownStair...	0.88557779788..
Walking...	0.65055751800...

Figure 6. Real-time recognition snapshot of walking

VI. Conclusion

Both the time and frequency domain signals of the activities have comparable results but time signals are better in terms of representing motion data for deep learning recognition and classification. Considering the experimental results of the time series datasets, the system can be employed in different fields where human motion is important, such as safety, healthcare, and human-robot interaction by adding extra functionalities. Especially, it can be extended to be utilized in transportation systems, where driving while using a phone is a common scenario, to draw drivers' attention for their safety.

Generally, the overall system has less complicated and low energy consuming iOS Application and program codes written in the most famous programming language, Python. Summing up all the functions into a single device in the hands of many, an iPhone, is the best way to give users the capability of control over their data. In a nutshell, the popularity of the language, IDE and libraries used among the software development society and the ubiquitousness of smartphones makes this product a simpler yet effective choice. There are few if any, such an iOS application developed so far.

With the increasingly released deep learning libraries, we are looking forward to improving the application overall performance.

References

[1] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "A Public Domain Dataset for Human Activity Recognition Using Smartphones," *ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*. Bruges, Belgium, Apr. 24-26, 2013.

[2] L. Bao, S.S. Intille, "Activity Recognition from User-Annotated Acceleration Data," *2nd Int. Conf. on Pervasive Computing*, Vienna, Austria, pp. 1-17, Apr. 21-23, 2004.

[3] D. Singh, E. Merdivan, I. Psychoula, "Human Activity Recognition using Recurrent Neural Networks," *Int. Cross-Domain Conf. for Machine Learning and Knowledge Extraction*, Reggio, Italy, pp. 267-274, Aug. 29-Sep. 1, 2017.

[4] J. Yamato, J. Ohya, K. Ishii, "Recognizing human action in time-sequential images using hidden Markov model," *Proceedings IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, Champaign, IL, USA, pp. 379-385, Jun. 15-18, 1992.

[5] S. Ha, S. Choi, "Convolutional neural networks for human activity recognition using multiple accelerometer and gyroscope sensors," *2016 Int. Joint Conf. on Neural Networks*, Vancouver, BC, Canada, pp. 381-388, Jul. 24-29, 2016.

[6] W. Jiang, Z. Yin, "Human Activity Recognition using Wearable Sensors by Deep Convolutional Neural Networks," *Proceedings of the 23rd ACM Int. Conf. on Multimedia*, Brisbane, Australia, pp. 1307-1310, Oct. 26-30, 2015.

[7] T. T. Um, V. Babakeshizadeh, D. Kulić, "Exercise Motion Classification from Large-Scale Wearable Sensor Data Using Convolutional Neural Networks," *Int. Conf. on Intelligent Robots and Systems (IROS 2017),* Vancouver, BC, Canada, pp. 2385-2390, Sept. 24-28, 2017.

[8] F. M. Rueda, R. Grzeszick, G. A. Fink, S. Feldhorst, and M. T. Hompel, "Convolutional Neural Networks for Human Activity Recognition Using Body-Worn Sensors," *Informatics*, Vol. 5, No. 2, pp. 26, May 2018.

[9] C. BenAbdelkader, R. Cutler, and L. Davis, "Stride and cadence as a biometric in automatic person identification and verification," *Proceedings of Fifth IEEE Int. Conf. on Automatic Face Gesture Recognition,* Washington, DC, USA, pp. 372-377, May 21, 2002.

[10] T. T. Alemayoh, J. H. Lee, and S. Okamoto, "Implementation of convolutional neural network for classification of daily human activities," *The 24th Int. Symposium on Artificial Life and Robotics,* Beppu, Japan, pp. 43-48, Jan. 23-25, 2019.

[11] S. Ioffe, C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *Proceedings of the 32nd Int. Conf. on Machine Learning*, Lille, France, pp. 448-456, Jul. 06-11, 2015.

[12] D. P. Kingma, J. L. Ba, "Adam: A Method for Stochastic Optimization," *3rd Int. Conf. on Learning Representations*, San Diego, USA, May 7-9, 2015.

[13] Developer.apple.com. (2019). Core ML | Apple Developer Documentation. [online] Available at: https://developer.apple.com/documentation/coreml [Accessed on: Jan. 21, 2019].

[14] A. Shkarlinska and A. Wadhwa, (2004). TensorFlow Core ML v2.0. [Online]. Available: https://github.com/tf-coreml/tf-coreml, [Accessed on: Jan. 22, 2014].