Media Engineering and Technology Faculty
German University in Cairo

Deep learning for mobile sensed data:Activity Recognition

Bachelor Thesis

Author:	Lama Ihab
Supervisors:	Dr. Mervat AbuelKheir
Submission Date:	30 July, 2020

This is to certify that:

(i) the thesis comprises only my original work toward the Bachelor Degree

(ii) due acknowledgement has been made in the text to all other material used

---

Lama Ihab
30 July, 2020

Acknowledgements

I would like to express my sincere gratitude to my supervisor,Dr. Mervat, for her guidance throughout the project. I would also like to thank my family and friends for their support.

Abstract

Human activity recognition (HAR) is a field that gained a lot of attention in recent years due to its usage in various applications such as healthcare monitoring and fitness tracking. Also, the HAR problem became more easier and straightforward to solve due to the widespread of various sensors embedded in smartphones such as accelerometers and gyroscopes. Since using state-of-the-art machine learning techniques to solve the HAR task requires manual feature engineering and a very high computational cost, researchers began to use deep learning techniques as it requires very low computational cost and it performs automatic feature extraction. Moreover, deep learning models are more fit to run on mobile devices due to its small size so it will be easier to obtain the needed sensor data from smartphones directly and feed it to the deep learning model on the device to predict activities at real time. In this paper, I will be using deep learning techniques to address the HAR problem where time-series data from smartphone sensors will be used as an input to the model. A convolutional neural network will be built for feature extraction and prediction of activities. Moreover, the CNN model will be trained and tested using the UCI HAR dataset. Furthermore, on-device machine intelligence is a recent breakthrough that enables machine intelligent features to run on users' personal devices, so accordingly, the model will be loaded on a smartphone emulator to test the model's prediction of activities at real time and measure the performance of embedding a CNN model on a mobile device.

Contents

Acknowledgements **iii**

1 Introduction **2**

2 Background **4**

3 Methodology **10**
3.1 Analysis Architecture 10
3.1.1 Model Training 11
3.2 Experimental Results 11
3.2.1 UCI Dataset 11
3.2.2 Smartphone Emulator 12

4 Conclusion & Future Work **15**
4.1 Conclusion 15
4.2 Future Work 15

Appendix **15**
List of Figures 17
List of Tables 18

References **20**

Chapter 1

Introduction

With the widespread of various sensors embedded in mobile devices, the analysis of human daily activities becomes more common and straightforward. This task now arises in a range of applications such as healthcare monitoring, fitness tracking or user-adaptive systems, where a general model capable of instantaneous activity recognition is needed. Deep Learning models are being used lately for Human Activity Recognition (HAR) using input data from sensors embedded in smartphones like accelerometers and gyroscopes. Moreover, deep learning models are being embedded in mobile devices to make predictions at real time in order to preserve the users' privacy by not sending their information over a network but have it all run on the device. However, running a deep learning model on devices faces a couple of challenges like demand on memory, computation and energy.

In this paper, I purpose a Convolutional Neural Network (CNN) model to address the HAR problem using feature extraction and class classification where raw sensor data from accelerometers and gyroscopes (which are embedded in smartphones) are used as input to the model to predict a given activity from 6 different activity classes (Walking, Walking Upstairs, Walking Downstairs, Standing, Sitting, Laying). The model is trained and tested using an online dataset and experiments were made several times to achieve a high accuracy of activity prediction. Moreover, the pre-trained model was loaded on an android emulator to test its performance and accuracy when embedded on a smartphone device and to test its ability to predict activities at real time without sending the user's sensor data over a network.

This paper is organized as follows:

- Chapter 2 includes summary of related papers and background research.

- Chapter 3 includes the model's architecture in section 3.1 and includes the experimental results in section 3.2.
- Chapter 4 includes the conclusion which sums up the work and results in section 4.1 and includes the future work in section 4.2.

Chapter 2

Background

This chapter includes an introduction to on-device machine intelligence ,introduction to some deep learning techniques, and summary to research papers that used deep learning models to solve HAR problem.

On-device machine intelligence is a recent breakthrough that enables machine intelligent features to run on users' personal devices. Deep Learning models can be embedded and run on mobile devices to make predictions after getting the needed inputs directly from the device. However, the adoption of deep learning within mobile devices face significant barriers due to the resource requirements of these algorithms. Demands on memory, computation and energy make it impractical for most models to directly execute on mobile devices. As a result, deep learning seen on phones are largely cloud-assisted where data are collected from smartphones and sent through a network to be trained on the cloud. This introduces important negative side-effects: first, it exposes users to privacy dangers as sensitive data is processed off-device by a third party; and second, the inference execution becomes coupled to fluctuating and unpredictable network quality (e.g., latency, throughput). Fortunately, researchers designed some tools to help deep learning models to run on devices directly.

The DeepX toolkit (DXTK) was proposed [3] which is an open source collection of software components for simplifying the execution of deep models on mobile devices . DXTK is a series of model optimizers that implement various techniques to reduce the computational and memory demands of deep models at the expense of a small loss in accuracy but it is tunable by the user. DXTK is compromised by 3 types of software components : 1) low resource models which are set of models already prepared and can perform classification 2) Model optimizers which are used to reduce execution-time

memory consumption, the storage/memory footprint of the model itself, and the amount of computation required and it includes 5 different optimizers (weight factorization, sparse factorization, convolution separation, precision scaling and parameter cleaning 3) Runtime Support where DXTK makes it easy for Torch to run within Android and Linux environments. Experiments were carried out and Results show that DXTK can significantly reduce the memory footprint of a deep model with little loss in inference accuracy.

Deep Learning models are being used to predict human activities after getting inputs from wearable sensors or sensors embedded in smartphones. Different deep learning techniques are being used for the HAR problem.

Since the problem of HAR mostly rely on inputs coming from wearable sensors or from sensors embedded in smartphones which are in the form of signals, Convolutional Neural Networks (CNN) has been used for feature extraction and HAR classification. A CNN is another type of feed forward deep neural networks which is mostly used for feature extraction where features can be learned automatically through the network instead of being manually designed like in traditional machine learning algorithms. A CNN is composed of two parts : a feature learner and a classifier. The feature learner is made up of convolutional layers placed sequentially and we determine the number of those layers based on the problem. It consists of filters (kernels) that are applied to the input to produce feature maps which are used to extract features. Between these convolutional layers, pooling layers are found to decrease the size of feature maps which are then passed to the classifier to detect the activity. The classifier can be a fully connected network like the traditional multilayer perceptron (MLP) or in some cases it can be a recurrent neural network. Researches have been made to determine different human activities using CNN based on the fact that we get the input in the form of signals and basic continuous movements correspond to smooth signals and the sudden transition among those basic movements can cause a significant change in those signals, these properties of signals require feature extraction to capture the nature of these movements.

Another deep learning network that researchers use for HAR is Recurrent Neural Network (RNN). A RNN is not considered a feed forward network as it contains loops where a unit in a layer receives two inputs : a vector of states from the previous layer and a vector of states from the previous time step. LSTMs (long short-term memory) extend RNN with memory cells, instead of recurrent units, to store and output information, easing the learning of temporal relationships on long time scales. It is mainly used to replace

some units of the RNN to solve the problems of an input/output weight conflict which is the conflicts between the input from the previous layer and the recurrent value, and vanishing/exploding gradient problem where a delta vanishes or explodes by the deep backward propagation.

A CNN design for HAR using smartphone sensors (accelerometer) was designed by researchers. The architecture [11] consisted of only one pair of a convolutional layer and a max pooling layer and two normal fully connected neural networks and they used back propagation and gradient descent to train the model. Also, for regularization and to avoid overfitting problem, they used momentum, weight decay and dropout techniques. The model was tested and proved to give more accurate results than ML algorithms.

Researchers also investigated the varying of number of layers and feature maps [7] on the performance and accuracy of the system. Raw sensor data from accelerometer and gyroscope were segmented using a sliding window and passed to the architecture consisting of convolutional layers and max pooling layers and a very simple softmax classifier was used. The system was trained using stochastic gradient descent (SGD) to minimize cost function. Experiments were conducted with ReLU as activation function. They kept on increasing the number of filter maps by 10 and increasing the convolutional layers until the 3rd layer. It was shown that increasing the number of feature maps improved the performance but as we keep on increasing, it will not improve further. Also, it was found that the complexity of derived features indeed increases with increasing convolutional layers, but the difference in complexity between adjacent layers decrease with each additional layer.

A CNN model was built [6] for human activity recognition from smartphone sensors as well (accelerometer and gyroscope). The architecture consists of 3 convolutional layers for the feature extraction and ReLU for the activation function and MLP and a softmax layer for the classification part as well as a dropout with probability of 0.8 to avoid overfitting and the data was trained with maximum of 2000 epochs with SGD method for optimization. Comparisons were made between this system and other state-of-the-art ML algorithms and the CNN system achieved accuracy of 94.79% with raw sensor data and 95.75% accuracy with additional information of temporal fast Fourier transform of the HAR data set which proved to be more accurate than other ML algorithms.

Researchers investigated how the time series length affects the recognition accuracy as well. They used CNN for HAR [1] using quasi-periodic ac-

celerometer readings from smartphones. The CNN architecture used in this paper consisted of 196 filters and ReLU was used as an activation function and max pooling layer. The output of the max pooling layer is flattened and stacked together with additional statistical features : mean, variance, sum of absolute values and histogram of each data input channel to eliminate the problem of the network not being able to capture the global properties of the signal. The output is then passed to a fully connected network with 1024 neurons and then passed to a softmax classifier. They used a dropout of rate 0.05 to avoid overfitting and gradient descent method for training and minimizing the cross-entropy loss function. Experiments were carried out and it was concluded that segmentation of signal data using short intervals of 1 second gave better results and the proposed algorithm is more accurate than state-of-the-art methods and had a small running time and can be efficiently executed on mobile devices in real time.

Moreover, Authors designed a fast and robust CNN (aka: FR- DCNN) [5] for HAR using smartphone sensors (accelerometer , gyroscope, magnetometer) to improve the computational speed and classification accuracy. The raw signals went through 3 phases before the CNN network which are :1) signal processing to improve the identification ability where signal processing algorithms were used to extend dimensions of raw signals. 2) Data Compression to delete the similar segments for fast computation. 3) Signal selection to choose the useful signals for high accuracy classification. The designed DCNN consisted of four convolutional layers, the first three deep convolutional modules consist of a 2D convolution layer, a batch normalization layer and a rectified linear units (ReLU) as an activation function and used to avoid problem of exploding and vanishing gradient while training, the last deep convolutional module add a max-pooling layer then there is the full connected network for classification and a softmax layer to calculate probability. They also added a dropout layer of percentage 0.5 to avoid overfitting. The system was tested and compared with other ML algorithms, the traditional CNN and LSTM and proved to have less computational time.

Researches were also made by obtaining sensor readings from wearable sensors.Authors designed a CNN model on multichannel time series [9] where input was obtained from a set of worn inertial sensors and system outputs predefined human activities. They start by segmentation of the input data using a sliding window where they used 30 or 32 as sampling rate and 3 for step size of sliding window. The architecture consists of 5 sections. First two sections consist of convolutional layer that convolves the input with a set of kernels, a rectified linear unit (ReLU) layer as the activation function, max

pooling layer ,a normalization layer that normalizes the values of different feature maps in the previous layer. The third section consists of only a convolutional layer , ReLU layer and a normalization layer. The fourth section is used to unify feature maps from all sensors using a fully connected layer. The fifth section is the classifier which is a standard MLP where the output is governed by a softmax function. The system is then trained by learning the parameters to minimize cost function using back propagation method and they also used dropout method for regularization to avoid problem of overfitting. Experiments have shown that this system gave more accurate results in HAR than other systems where traditional ML algorithms were used.

Furthermore, authors investigated the performance of CNN after changing the number of convolutional layers and changing the kernel size. Data was collected from body worn sensors [10] on five different parts of the body, the sensors used were inertial sensors (accelerometers and gyroscopes). The data from sensors were segmented first then sent to the network. The architecture used 3 convolutional layers , 3 max pooling layers with ReLU as activation function and a softmax classifier as well as back propagation (gradient descent) method for optimizing until maximum epoch was reached. Comparisons regarding accuracy and computational cost were made between this system and SVM and MLP. It was shown that deep CNN had the most accuracy and the least computational cost. Moreover, tuning the filter and pooling sizes revealed that wider kernels and lower pooling size improves the recognition performance and it was shown that increasing the number of convolutional layers increased the computational cost and decreased the complexity of derived features.

Researchers also proposed a RNN architecture for HAR [2] with high throughput using input from mobile devices. The system consisted of multiple layers and LSTM cells and they used a softmax function to calculate probability. The mini-batch SGD method was used for optimization, gradient clipping technique was used for solving exploding gradient problem and dropout method was used for regularization to prevent overfitting. The model was trained and tested and experiments were carried out and it was proven that the following parameters gave the best performance: 3 internal layers, 60 units in one layer, 5 for gradient clipping parameter and 0.5 for dropout rate. The throughput was also measured and this system proved to have better throughput than other ML methods.

Authors also analyzed the performance of two networks: CNN with LSTM (ConvLSTM) and RNN with LSTM (RNNLSTM) [8] using raw data from

smartphone sensors (tri-axial accelerometer, gyroscope, compass sensor). These two models were trained and compared with other ML algorithms (K-NN, Regression Tree, SVM, Random Forest, Extra Trees, Gradient Boosting). RNNLSTM gave the highest accuracy of 93.89% and ConvLSTM gave the second highest accuracy of 92.24%.

Researchers proposed a model for the HAR problem using CNN and LSTM RNN together (DeepConvLSTM) [4]. The input to DeepConvLSTM network is a short time series extracted from wearable sensors composed of several sensor channels after segmentation using a sliding window. The network consists of 4 convolutional layers for feature extraction with ReLU as the activation function and no pooling layers followed by 2 dense recurrent layers with hyperbolic tangent function as the activation function. The output is obtained from a softmax layer and model was trained using mini-batch gradient descent to minimize cross-entropy loss function and they used dropout with 0.5 rate to as a form of regularization.

Chapter 3

Methodology

3.1 Analysis Architecture

To address the HAR problem, I built a one dimensional Convolutional Neural Network to predict a given activity that is presented in figure 3.1. The network was implemented using PyTorch framework. The input to the network was sensor data from triaxial accelerometer and gyroscope sensors embedded in smartphones.

The architecture of the network is as follows:

- Convolutional layer: a one dimensional convolutional layer which consists of number of kernels (filters) will be used for feature extraction. Six input channels will pass through this layer representing the triaxial (x, y, z) signals from each sensor. The output to this layer (feature maps) will be passed through a non-linear activation function. I used ReLU for this implementation.

- Pooling layer : this layer follows the convolutional layer to summarize the obtained output. I used maximum pooling layer in this architecture.

- Fully connected layer: after the feature extraction phase, the output is flattened and passed through a fully connected layer for classification. The flattened vector will also pass by a ReLU activation function.

- Soft max layer: Finally, the output of the last layer is passed to a softmax layer that computes probability distribution over the predicted classes.

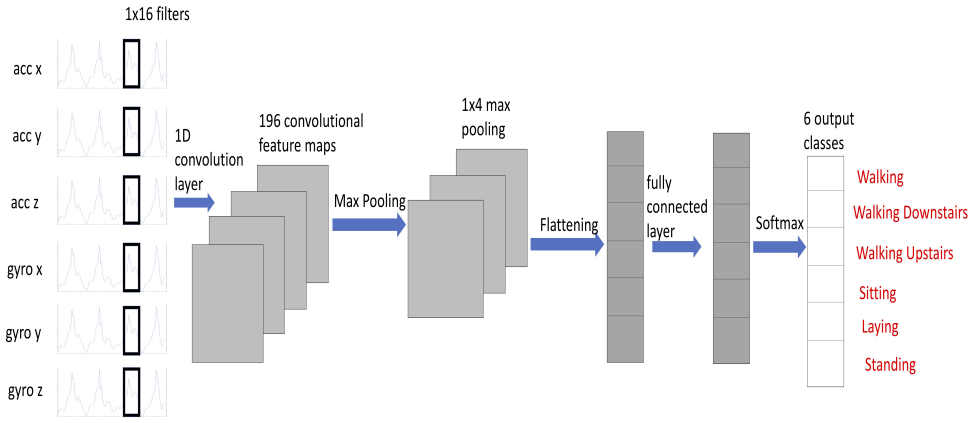

Figure 3.1: proposed CNN

The advantage of the purposed architecture is that it is shallow, which makes it suitable to run on mobile devices with a small running and computation time to be able to predict activities in real time.

3.1.1 Model Training

After building the network, the model was trained on CPU (local machine with processor of 1.6 GHz Intel Core i5) without usage of GPU, to minimize cross-entropy loss function using stochastic gradient descent technique with back propagation algorithm for optimization and for tuning the parameters of the network. Moreover, to avoid overfitting of the network, The loss function was augmented with l2-norm regularization of CNN weights and a dropout technique with 0.05 rate was used in the fully connected layer.

3.2 Experimental Results

3.2.1 UCI Dataset

For training and testing the model, I used the online UCI HAR dataset[1] Ex-

[1] `http://archive.ics.uci.edu/ml/datasets/Smartphone-Based+Recognition+of+Human+Activities+and+Postural+Transitions`.

periments were carried out with a group of 30 volunteers while performing six basic activities (standing, sitting , laying, walking , walking upstairs, walking downstairs) and raw sensor readings from both sensors (accelerometer and gyroscope) were recorded and saved in text files as sequential numbers.The raw sensor data were pre-processed by dividing them into segments of equal length (recognition interval of 128) and saving the corresponding activity label with each segment which resulted in a total of 5409 segments. The obtained segments were also divided into training and testing data with 70% of the segments saved for training and 30% saved for testing.

The model was trained and tested for the first experiment with 100 epochs, dropout rate of 0.05 and resulted in 58% accuracy. To improve the accuracy, another experiment was conducted after increasing the number of epochs to 200 epochs and same dropout rate of 0.05 and then tested which resulted in 78% accuracy. Another experiment was conducted after adding the l2-norm regularization with a factor of (5e-4) to the loss function, dropout of 0.05 and 200 epochs which resulted in 90% accuracy. I wanted to increase the accuracy further so I increased the dropout rate from 0.05 to 0.5, with 200 epochs as well and l2 regularization with 5e-4 factor but it made no improvements to the model's accuracy. Moreover, I have added a second convolutional layer to the network with the same hyper parameter values(dropout rate:0.05,l2 regularization:5e-4,200 epochs) which resulted in more training time than the original architecture and it did not improve the accuracy. Another experiment was conducted after normalizing the input sensor data with dropout rate:0.05,l2 regularization:5e-4 and 200 epochs but the accuracy decreased to 65% so normalization was excluded. The final experiments have been conducted by just increasing the number of epochs to 5000 epochs while keeping the same hyper parameter values as the previous experiment which resulted in no accuracy improvements. The model remained at 90% and no further experiments were conducted. Results for each class separately are shown in table 3.1

3.2.2 Smartphone Emulator

To test the performance of the proposed model on a smartphone device, the model was trained on the cloud using the UCI dataset where 70% of the volunteers' data was used for training (21 users).The pre-trained model was serialized, loaded and saved on an android smartphone emulator.

I used two ways to test the accuracy of the model to predict activities on the emulator which are as follows:

Table 3.1: Classification results on UCI Dataset

Activity	**Accuracy**
Walking	94.8%
Walking Upstairs	91.8%
Walking Downstairs	95.6%
Sitting	72.5%
Standing	80.5%
Laying	99.6%
Overall	90%

- Data from the accelerometer and gyroscope sensors (x, y, z from each sensor) were obtained from the emulator itself and pre-processed by dividing them into equal sized segments(interval size of 128) and each segment, after being collected, acts as an input to the pre-trained model. The on-device model was tested by providing it with the needed sensor data and it returned the activity "Laying" at real time which is correct since the emulator can only be tested at rest.

- The second way to test the model on the emulator is to feed the pre-trained model with the remaining users' data from the UCI dataset that was not used for training. I chose 5 users for testing where each user performed the 6 activities provided in figure 3.1 and fed their corresponding sensor data to the model. This resulted in 30 experiments being performed and overall accuracy of 76.7%. The results for each activity class can be shown in table 3.2.

Table 3.2: Classification results on Emulator

Activity	**Accuracy**
Walking	0%
Walking Upstairs	100%
Walking Downstairs	100%
Sitting	96.6%
Standing	96.6%
Laying	100%
Overall	76.7%

In order to measure the performance of the model on the emulator, I will

evaluate the model according to the challenges that deep learning models usually face when embedded on mobile devices which are as follows:

- Privacy: Since the model was loaded on the emulator without sending the user's sensor data over a network, the privacy of users was not compromised.
- Memory: The model's size is 217 KB which is considered small and can be loaded on a mobile device without a big demand on memory.
- Computation: Since the model was pre-trained before being exported and loaded on the emulator, it did not demand a lot of computations and energy to be done on the device.
- Performance at real time: The model returned the activity being performed immediately at real time with no delays after the needed sensor data was sent to it as input. This is considered as an advantage of having the model on the device instead of sending the sensor data over a network and being exposed to an unpredictable network quality.

Limitations on emulator:

I faced some limitations while testing the pre-trained model on the emulator. First, I could not test all the activities on the emulator using the device's sensor data because the emulator can not mimic real movement. Therefore, I used sensor data from the UCI dataset to test the model and see the results for all 6 activities. Second, the input to the model on the emulator had to be floating numbers but the model was trained on double numbers so I had to typecast the sensor data from double to float (which leads to a decrease in the numbers' accuracy) and that's why the accuracy of the model dropped from 90% to 76.7% on the emulator and the walking activity was not recognized at all as shown in table 3.2.

Chapter 4

Conclusion & Future Work

4.1 Conclusion

In this paper, I purposed a deep learning model to predict activities and solve the HAR problem after receiving sensor data from smartphone sensors (accelerometer and gyroscope). The model was based on a One Dimensional Convolutional Neural Network for feature extraction and classification with one convolutional layer, max pooling layer and one fully connected layer. The model was trained and tested using UCI dataset to evaluate the performance which resulted in 90% accuracy and the results for each activity alone can be seen in table 3.1. Moreover, due to the model's shallow architecture, it has a small running time and that's why it can easily run on mobile devices in real time. Therefore, The pre-trained model was loaded on an android emulator and after feeding it with the needed sensor data while performing an activity, the output was obtained immediately at real time with no delays and resulted in 76.7% accuracy after testing it using UCI dataset. Results of each activity on emulator can be seen in table 3.2. Also, the on-device model was evaluated according to memory, computation, performance at real time and privacy of users' data and it showed good results according to those challenges mentioned.

4.2 Future Work

Future works will include trying to improve the purposed model's accuracy by training it using a larger set of data and investigating the increase of filter and pooling sizes on the model's accuracy. Also, I will test the purposed model on a real device instead of an android emulator to measure its performance of predicting activities at real time.

Appendix

List of Figures

3.1 proposed CNN 11

List of Tables

3.1 Classification results on UCI Dataset 13
3.2 Classification results on Emulator 13

Bibliography

[1] Andrey Ignatov. Real-time human activity recognition from accelerometer data using convolutional neural networks. *Applied Soft Computing*, 62:915–922, 2018.

[2] Masaya Inoue, Sozo Inoue, and Takeshi Nishida. Deep recurrent neural network for mobile human activity recognition with high throughput. *Artificial Life and Robotics*, 23(2):173–185, 2018.

[3] Nicholas D Lane, Sourav Bhattacharya, Akhil Mathur, Claudio Forlivesi, and Fahim Kawsar. Dxtk: Enabling resource-efficient deep learning on mobile and embedded devices with the deepx toolkit. In *MobiCASE*, pages 98–107, 2016.

[4] Francisco Javier Ordóñez and Daniel Roggen. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16(1):115, 2016.

[5] Wen Qi, Hang Su, Chenguang Yang, Giancarlo Ferrigno, Elena De Momi, and Andrea Aliverti. A fast and robust deep convolutional neural networks for complex human activity recognition using smartphone. *Sensors*, 19(17):3731, 2019.

[6] Charissa Ann Ronao and Sung-Bae Cho. Human activity recognition with smartphone sensors using deep learning neural networks. *Expert systems with applications*, 59:235–244, 2016.

[7] Charissa Ann Ronaoo and Sung-Bae Cho. Evaluation of deep convolutional neural network architectures for human activity recognition with smartphone sensors. *Academic Journal of Korean Information Science Society*, pages 858–860, 2015.

[8] Ms S Roobini and Ms J Fenila Naomi. Smartphone sensor based human activity recognition using deep learning models.

[9] Jianbo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiao Li Li, and Shonali Krishnaswamy. Deep convolutional neural networks on multichannel time series for human activity recognition. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[10] Tahmina Zebin, Patricia J Scully, and Krikor B Ozanyan. Human activity recognition with inertial sensors using a deep learning approach. In *2016 IEEE SENSORS*, pages 1–3. IEEE, 2016.

[11] Ming Zeng, Le T Nguyen, Bo Yu, Ole J Mengshoel, Jiang Zhu, Pang Wu, and Joy Zhang. Convolutional neural networks for human activity recognition using mobile sensors. In *6th International Conference on Mobile Computing, Applications and Services*, pages 197–205. IEEE, 2014.