# Evaluation of Deep Learning Models for Smoking Recognition with Smartwatch and Smartphone Sensors

Yasemin Akan, Sumeyye Agac and Ozlem Durmaz Incel

*Abstract*—**Smartwatches and smartphones are extensively used in human activity recognition, particularly for step counting and daily sports applications, thanks to the motion sensors integrated into these devices. Machine learning algorithms are often utilized to process sensor data and classify the activities. There are many studies that explore the use of traditional classification algorithms in activity recognition, however, recently, deep learning approaches are also receiving attention. In this paper, we utilize a dataset that particularly consists of smoking-related activities and explore the recognition performance of three deep learning architectures, namely *Long-Short Term Memory (LSTM)*, *Recurrent Neural Networks (RNN)* and *Convolutional Neural Networks (CNN)*. We evaluate their performances according to different hyperparameters, different sensor types and device types. The results exhibit that the performance of LSTM is much higher than that of CNN and RNN. Moreover, the use of magnetometer and gyroscope together with accelerometer data improves the performance. Use of data from smartphone sensors also enhances the performance results and the final accuracy with the best parameter combinations is observed to be 98.2%.**

*Index Terms*—**Activity recognition, deep learning, wearable computing, motion sensors.**

## I. Introduction

WEARABLE DEVICES such as smartwatches and smart glasses, that are integrated with a variety of sensors, are commonly used in human activity recognition studies [1,2]. Particularly, motion sensors available on these devices make it easy to collect and analyze data for sports and well-being applications. In some studies [2], classification of general activities is targeted for monitoring the activity levels, for example, for fitness purposes, while in some others, more specific activities are monitored, such as fall detection [3].

**YASEMİN AKAN**, is with Department of Computer Engineering Bogazici University, Istanbul, Turkey, (e-mail: yasemin.akan@boun.edu.tr)
https://orcid.org/0000-0003-3398-466X
**SUMEYYE AGAC**, is with Department of Computer Engineering of Bogazici University, Istanbul, Turkey, (e-mail: sumeyye.agac@boun.edu.tr).
https://orcid.org/0000-0001-5231-7008
**OZLEM DURMAZ INCEL**, is with Department of Computer Engineering of Galatasaray University, Istanbul, Turkey, (e-mail: odincel@gsu.edu.tr).
https://orcid.org/0000-0002-6229-7343

Manuscript received November 17, 2021; accepted August 03, 2021.
DOI: 10.17694/bajece.827342

Particularly, activities related to wrist and hand movements can easily be recognized with smartwatches. Smoking is one of the activities that users may be interested in tracking. Especially, for smoking cessation programs, it may be practical to automatically track the number of cigarettes smoked instead of self-reporting which puts a burden on the user [4].

In this paper, we utilize a dataset [5] that specifically consists of smoking-related activities. Smoking activity is more difficult to detect than simpler activities such as sitting since it can be done in a variety of postures (sitting, standing, and walking) and in parallel with other activities such as while chatting in a group or while drinking coffee. Other activities that include similar hand movements, such as eating and drinking, can easily be confused with smoking.

In human activity recognition studies, mostly, data from motion sensors is collected and processed with machine learning techniques to classify the activity types. Survey papers [1,2] provide a taxonomy on devices, sensors, activities and machine learning algorithms that are commonly used in human activity recognition with wearable devices. There exist many studies that focus on the use of simple machine learning techniques, such as Support Vector Machine (SVM), K-Nearest Neighbors, Decision Trees, etc. [2]. However, the use of deep learning approaches is recently receiving attention [6,7]. First of all, training and running deep learning models are costly on these resource-limited devices. However, a model can be trained offline, for example a cloud-based model would be a feasible solution. Moreover, unlike traditional machine learning algorithms, we do not have to deal with the feature extraction steps and can directly use the raw sensor data.

In this paper, we investigate the use of different deep learning algorithms, namely *Long-Short Term Memory (LSTM)*, *Recurrent Neural Networks (RNN)* and *Convolutional Neural Networks (CNN)* on activity recognition with mobile and wearable devices. As mentioned, we use a dataset that was collected in our previous work [5]. This dataset was analyzed using general machine learning models, particularly with Random Forest (RF), Decision Trees, SVM and Multilayer Perceptron (MLP) [4]. Moreover, CNN model was also applied to the same dataset [8] by other researchers. However, the performance analysis of LSTM and RNN models on this dataset was not investigated before. LSTM and RNN are mostly used with sequential data, such as text. In this paper, we also work on a dataset which consists of sequences of activities. Moreover, these three algorithms are the most widely used deep learning architectures in the literature in human activity recognition studies [7]. Hence, our contribution is to explore the performance of different deep learning architectures on a challenging dataset.

We perform different evaluations according to different hyperparameters for different architectures, two types of devices (smartwatch and smartphone) and three types of sensors (accelerometer, gyroscope, and magnetometer). The results show that we achieve the highest accuracy scores, 92.0% with the LSTM architecture when only the accelerometer data from a smartwatch is used. If smartphone data is also used, 96.9% accuracy is obtained, again only with the accelerometer data. If other sensors are used together with an accelerometer, we could achieve 95.6% accuracy with the only a smartwatch, and 98.2% accuracy when both devices and three sensors are used. We can summarize our contributions as follows:

- We explore the performance of different deep learning architectures on a challenging dataset, which contains smoking activities in different postures and other activities that are similar to smoking. We also explore different values for hyper- parameters for different architectures
- We investigate the use of different sensors and devices used alone and in combination with others. We conclude that when both devices, smartwatch, and smartphone, are used together, the recognition performance increases by 3 percent, however the use of only the smartwatch also performs well, 95% accuracy.
- In comparison to the related studies that utilize deep learning algorithms for human activity recognition, we observe that in most of the studies, the CNN architecture was used, a smaller number of activities were targeted and accelerometer and gyroscope sensors were used, however in this paper, we focus on three different architectures, more than ten different types of activities, three sensors, including magnetometer besides other two sensors, and two devices.

The rest of the paper is organized as follows: In Section II., we summarize the related studies, in Section III. we present our methodology, particularly the details of the applied models. In Section IV, we present the performance analysis of three different deep learning architectures and finally in Section V, we draw the conclusions.

## II. RELATED WORK

In our previous work [4], the same dataset was used and the effects of the parameters were analyzed using 4 different window sizes, 63 features which are calculated separately for each sensor, 4 different sensors, 2 different sensor combinations, 3 classification algorithms (SVM, RF, MLP)). The results showed that simple features including median, standard deviation, minimum, maximum, range, and mean can be useful in recognizing smoking activities. Besides, it was concluded that only time-based features provided better performance and using the only accelerometer is sufficient to recognize simple activities, whereas for complex activities the combination of accelerometer and gyroscope performs better. The performance of different classifiers was compared, and the results were similar when effective features were used. We observed that on average of 10 activities, the recognition performance achieved by using simple features extracted from accelerometer and gyroscope sensors was 83% (in terms of F1-score) with the RF classifier. In that study, we did not make use of the data collected from the phones, only the smartwatch data was used. Moreover, we did not evaluate the performance with deep learning algorithms. In this paper, we observe a much higher accuracy, between 92-98% in terms of F1-score with LSTM.

In [8], it was proposed to use CNN as a deep learning method, to recognize smoking and the other activities on the same dataset, as in our study. A comparison of using smartwatch versus smartphone data and accelerometer versus gyroscope data was performed. The dataset was divided into 3 parts which are 70% training, 15% validation and 15% testing. The F1-score was used as a performance metric of the model. Raw data and extracted features (4 time-domain features: maximum, minimum, skewness and kurtosis) were given into the system. Relu was selected as the activation function and SoftMax was selected to calculate the probability distribution. The model was trained to minimize the use of cross-entropy using the Adam gradient descent optimization with a logarithmic loss function. The CNN model produced significant results to differentiate smoking from concurrent activities. For concurrent activities, two inputs were compared; for feature input, it was observed that similar activities were confused, and raw input achieved the highest F1-score. Although the performance of the model is reduced when only smartphone data is used for the smoking activity, it is observed that this has little impact on classification performance. On the other hand, high F1-scores were obtained when smartwatches data were used alone. Using the accelerometer and gyroscope data, it was concluded that the performance of the CNN model was independent of the sensor. The CNN model surpassed previous studies and got a very high F1-score of 92-96% for recognition of smoking activity.

In [9] deep RNN and LSTM learning algorithms were used and models were integrated into an Android application for real-time predictions. The tri-axial accelerometer of smartphone is used to measure the acceleration. WISDM Dataset ([10]), which contains approximately 1.098.207 samples of data was used. 80% of the data was used for training and 20% for testing. They tried to build a deep network using 2 RNNs fully connected with 2 LSTM layers stacked on top of each other with 64 hidden units in total. To calculate the loss, they use the L2 standard loss function, also called the Least

TABLE I
COMPARATIVE ANALYSIS OF RELATED STUDIES

| Reference | Architecture | Activities | Performances |
|---|---|---|---|
| [4] | SVM, RF, MLP | $S_{ST}$, $S_{SD}$, $D_{ST}$, $D_{SD}$, E, ST, SD, $S_G$, $S_W$, W | 83% F1-score |
| [8] | CNN | $S_{ST}$, $S_{SD}$, $D_{ST}$, $D_{SD}$, E, ST, SD, $S_G$, $S_W$, W | 92-96% F1-score |
| [9] | RNN-LSTM | $W_{US}$, $W_{DS}$, $J_g$, SD, ST, W | 97% Accuracy |
| [12] | CNN | R, W, CW, CCW, $M_{UD}$, $M_{LR}$ | 82.8% Accuracy |
| [11] | RF, HMMs, CNN-MLP, CNN-LSTM | SD, ST, B, $W_{US}$, $W_{DS}$, W | 98.1% F-score |
| [13] | CNN | F, R, J, W, $W_S$, $W_Q$, $W_{US}$, $W_{DS}$ | 93.8% Accuracy |
| This Study | CNN, RNN, LSTM | $S_{ST}$, $S_{SD}$, $D_{ST}$, $D_{SD}$, E, ST, SD, $S_G$, $S_W$, W | 98.2% Accuracy |

Squared Error (LSE). For model optimization, the Adam Optimization algorithm, which is an extension of stochastic gradient descent, was used as the best choice for the deep learning model. As the learning model, the Tensorflow was used with 50 epochs and the batch size was 1024 (32 x 32). After training the model, the accuracy of the test was greater than 97%. RNN and LSTM achieved 97% overall test accuracy.

In [11], various techniques were used to develop a human activity recognition system using accelerometer sensors of smartwatches (LG G and Samsung Galaxy Gear) and smartphones (LG Nexus 4, Samsung Galaxy S+, Samsung Galaxy S3 and S3 mini). Six activities were targeted: sitting, standing, biking, walking, walking upstairs and walking downstairs. They evaluated two architectures for this study. The first one is an architecture composed of pre-processing, feature extraction and activity recognition/segmentation (RF and Hidden Markov Models (HMMs)). The second is an architecture with deep learning methods which are CNN-MLP and CNN-LSTM. For smartphones, modelling time sequences using HMM showed lower performance than the RF algorithm. The use of CNN-LSTM for modelling gave the best results for smartphones. In addition to the six activities, it was observed that the degradation for HMM or CNN-LSTM was lower when an additional "Null" class was included when the user switched between the two activities or did not perform any of these activities.

In [12] CNN was applied on a dataset collected from a smartphone. Firstly, they analyzed the performance of machine learning algorithms. The data collected from the accelerometer and the gyroscope sensors were first pre-processed with noise filters and then sampled with a fixed size window for 2.5 seconds for the extraction of features. 60% of the data was used for training and 40% for testing. A Support Vector Machine classifier with Linear kernel (LibSVM) and Fisher Linear Discriminants (FLD) were used to detect error percentages. It was observed that LibSVM was better than FLD in terms of classification accuracy. Besides, results show that the accelerometer data contributed more to the accuracy results than those of the gyroscope sensor. Secondly, the data was analyzed using CNN. According to the test results, the accuracy of the prediction using the accelerometer data was 82.8% and for the gyroscope data was 78%. As a result, it was concluded that the use of both could increase the accuracy of recognition.

In [13] acceleration-based human activity recognition using the CNN deep architecture was investigated. CNN is commonly used for image data, and they emphasized that the biggest difference between the three-axis accelerometer data and the image data is the size of the data and this considerably limits the construction of the CNN architecture. Therefore, it was determined that the size of the convolution core was 2 with the lowest error rate. The best parameters for activity recognition were determined and the model was trained. An Android application was developed for data collection. The data were collected from 100 healthy people (68 men and 32 women) in a natural environment and at different accelerometer positions. The activities were falling, running, jumping, walking, walking quickly, step walking, walking upstairs and walking downstairs. After pre-processing, 31.688 of labelled instances were obtained. 27.395 of them were used for the training and the rest 4.293 for testing phases. The CNN structure used in this study is composed of 3 convolution layers and 3 pooling. The CNN model performed better than the results of the manual extraction of time and frequency-based features (Fast Fourier transform and Discrete cosine transform coefficients) and SVM and an 8-layer deep belief network classification algorithm. As a result, based on the confusion matrix, it was determined that recognizing walking activity was the most difficult activity because it was often mixed with walking quickly, walking upstairs and downstairs. However, CNN achieved 93.8% accuracy on average.

A comparison of related studies in terms of architectures, activities and best performance results are summarized in Table I[1]. In most of the studies, the CNN architecture is used [8,11-13]. The studies that use a different dataset than ours focus on a smaller number of activities. As mentioned, in [8], the same dataset was used but only the CNN architecture was evaluated. In contrast, in this paper, we evaluate the impact of the three popular deep learning architectures, which are CNN, RNN and LSTM. Moreover, we investigate the use of magnetometer sensor in combination with accelerometer and gyroscope sensors whereas some of the studies only focus on the use of accelerometer [9,11,13] or accelerometer and gyroscope [4,8,12].

## III. METHODOLOGY

### A. Dataset Details

In this paper, we focus on a dataset which was collected in our previous work [5]. 11 participants were involved in the data collection stage performing 10 different activities. A smartwatch (LG Watch R, LG Watch Urbane and Sony Watch

[1] $S_{ST}$: smoking while sitting, $S_{SD}$: smoking while standing, $D_{ST}$: drinking while sitting, $D_{SD}$: drinking while standing, E: eating, ST: sitting, SD: standing, $S_G$: smoking while talking in a group, $S_W$: smoking while walking, W: walking, $W_{US}$: walking upstairs, $W_{DS}$: walking downstairs, $J_g$: jogging, R: running, CW: clockwise movement, CCW: counterclockwise movement, $M_{UD}$: moving up-down, $M_{LR}$: moving left-right, B: biking, F: falling, J: jumping, $W_S$: step walking, $W_Q$: walking quickly.

3 models) and a smartphone (Samsung Galaxy S2 and S3, placed in trousers' pocket) were used to collect the data. A logger application was used to collect time-stamp, accelerometer, gyroscope and magnetometer readings in three dimensions from a smartwatch and a smartphone. All data was sampled at 50 Hz. However, based on the results of previous studies [4,5], since 10 Hz gives a good enough accuracy to recognize such human activities, we used a down-sampled version of the dataset at 10 Hz.

TABLE II
TYPES OF SCENARIOS

| Scenario type | Sequence of activities in one cycle |
|---|---|
| T1 | $S_{SD}$, W, $S_{G}$, $D_{ST}$, SD, $S_{W}$, SD, E, ST, $S_{ST}$, SD, $D_{SD}$, W |
| T2 | $S_{SD}$, ST, $S_{G}$, ST, $D_{ST}$, SD, E, ST, $S_{ST}$, SD, $D_{SD}$ |
| T3 | $S_{SD}$ , ST, $D_{ST}$, SD, E, , ST, $S_{ST}$, SD, $D_{SD}$, SD |

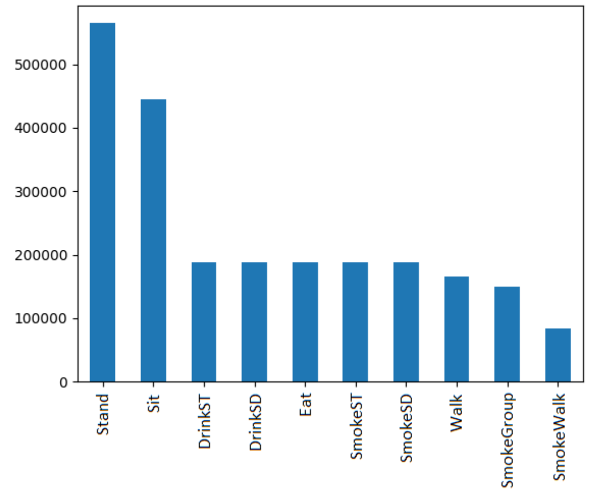


Fig. 1: The sample distribution according to the activities

The dataset contains 10 different activities, which are four types of smoking (while sitting, standing, walking and in a group conversation), three activities with similar wrist movements (drinking while sitting, drinking while standing and eating) and three simple activities (sitting, standing and walking) without smoking. Each activity was performed for 5 minutes by each participant and was repeated at least five times (not consecutively, at different times). The dataset contains 45 hours of data, in which 17 hours are related to smoking and 28 hours are related to the other activities. Smoking in a group conversation, smoking while walking and walking activities are performed by 8, 3 and 3 participants, respectively. The remaining 7 activities are performed by all 11 participants. Table II of [5] contains more detailed information about the dataset. Thus, the dataset consists of approximately 2.4 million lines (sensor readings), but the data was reshaped in the preprocessing step (or in other words we used windowing) and ts size was reduced to approximately 118000 by using timestep information. Figure 1 also shows the sample distribution according to the activities.

The dataset file is transformed into a scenario file with the aim of making the order of activity sessions more realistic and similar to everyday life patterns. A scenario (sequence of activities) was created for each participant, as presented in Table II. For example, in the second column of T2 row in Table II, for one scenario cycle of participant 4 (P4), we assume that the participant starts smoking while standing, then sits and smokes in a group conversation. After that, she/he sits again and drinks a cup of coffee, then she/he stands and so on. As mentioned before, all participants were not performed all activities. Therefore, there are some differences in scenarios (see Table II and III for more details of differences).

## B. *Deep Learning Architectures*

In this section, we briefly explain the deep learning architectures and the associated hyperparameters that are utilized in the paper. Further details about the deep learning architectures can be found in [14].

### 1) *CNN*

Convolutional neural networks, CNN, is simply a neural network that uses convolution instead of the general matrix product in at least one of its layers. In CNN terminology, the first convolution variable is usually called an entry and the second variable is called a kernel. Convolution is applied to the input data with kernels to determine the features of the input. The output is called a feature map. The CNN is composed of one or more convolution layers, pooling layers, then after, one or more fully connected layers, such as a standard multi-layer neural network. CNN is used widely in many fields such as natural language processing, biomedical, image, video and audio processing.

### 2) *RNN*

RNN evaluates its input not only instantaneously, but based on previous inputs, unlike the feed-forward neural networks. In other words, the decision made for the input at time *t-1* in RNN has an impact on the decision to be made at time *t*. Hence, in such networks, inputs combine current and previous information to generate the output. In other words, recurrent networks have a memory. Therefore, these networks are generally used to understand the structure of incoming data where it exists an order in data, such as in text, speech, time-dependent sensors or statistical data. In recurrent networks, in order to correctly classify the inputs, the backpropagation method is used to calculate the gradient descent on the network and the weight matrices are updated.

In this study, we use Vanilla RNN (VRNN). It has a simple architecture, but it may face problems related to gradient calculations. Gradient is an important metric used to adjust the weights of the network. However, in long connected networks, the effect of the error may decrease notably and therefore, the gradients may begin to be very close to zero. In other words, they begin to disappear. This is called the "vanishing gradient problem". The other problem is that due to the structure of the network, all derivatives are constantly multiplied with each other. Values obtained by multiplications can grow enormously and even explode. This is called "exploding gradient problem".

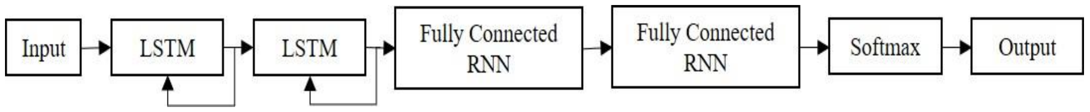


Fig.2. The utilized LSTM architecture

### 3) *LSTM*

LSTM that is invented to avoid the vanishing gradient problem is used in backpropagation to limit error values from layers. Instead of a single neural network layer, the repeating module in the LSTM structure has four gates that are specially interacted. This structure receives information outside. The cells decide what to store, when to read, write or delete information via the gates. These gates have a network structure and activation function. Just like in neurons, they allow to pass or stop the incoming information according to the weights. The cell learns to receive or release data thanks to the structure that uses these weights calculated during the learning process of the recurrent network.

In Figure 2, the architectural details of the used LSTM model are presented. Firstly, the global variables (hyperparameters, class number, number of features, time step, etc.) are defined. Different values are assigned to hyperparameters to investigate their impact as discussed in Section IV. The number of classes is 10 and the number of attributes is 9, including data from three sensors (accelerometer, gyroscope and magnetometer) and from three dimensions (x, y and z-axis). The data is divided into two subsets: 80% for training and the remaining 20% for testing. A deep network was built by stacking multiple layers of LSTM memory units with 2 fully connected RNN layers. LSE or L2-norm was used as the loss function to minimize the loss and AdamOptimizer is used as the optimization algorithm. The model is formed with the specified hyperparameters and the accuracy value for each epoch was computed, as explained in Section IV.

TABLE III
DETAILS OF THE DATA COLLECTED PER PARTICIPANT

| **Participant id** | **Type of Scenario** | **Number of scenario cycles** | **Total duration (minutes)** |
|---|---|---|---|
| 1 | T1 | 9 | 630 |
| 2 | T1 | 10 | 700 |
| 3 | T1 | 10 | 700 |
| 4 | T2 | 8 | 480 |
| 5 | T2 | 4 | 240 |
| 6 | T2 | 4 | 240 |
| 7 | T2 | 4 | 240 |
| 8 | T2 | 5 | 300 |
| 9 | T3 | 5 | 250 |
| 10 | T3 | 4 | 200 |
| 11 | T3 | 4 | 200 |

## C. *Hyperparameters*

Like other machine learning techniques, deep learning algorithms also include many hyperparameters that impact performance. Some of these hyperparameters affect the cost of time and memory when the algorithm is executed, while some others affect the ability of the trained model to achieve the correct results on test data. In this study, we investigated the following hyperparameters and their effect on success and loss rates:

- **Epoch number:** While creating a model, not all of the data is included in the training at the same time. The first part is trained, after the performance of the model is tested, backpropagation is updated according to the successful results. Then, the weights are updated again by retraining the model with the new training set. This process is repeated at each training step, called epoch, until the optimal weight values for the model are calculated. The epoch numbers used in this study are 30, 60 and 50.
- **Batch size:** In deep learning applications, learning by processing all the data in the dataset is a costly task in terms of time and memory because of the backpropagation. In this process, the gradient descent on the network is calculated and the weight values are updated accordingly. The larger the amount of data in this calculation, the longer the calculation takes. To solve this problem, the dataset is splitted into small groups and the learning process is performed on them. The value specified as a batch size means how much data the model can process at the same time. The batch sizes used in this study are 512, 1024 and 2048.
- **Number of hidden layers:** The most significant parameter that separates deep learning approaches from the other neural networks is the number of hidden layers. This is the concept of depth. As the number of hidden layers increases, the model learns better, at some point, however, as the number of hidden layers increases, the backpropagation effect is less than the first few layers. The number of hidden layers used in this study are 32 and 64.
- **Learning rate:** In deep learning methods, the parameters are updated by the backpropagation process. In the backpropagation process, this update is performed by calculating the new weight value by subtracting the result from the weight values and finding the difference by taking backward derivative called chain rule and multiplying the difference value by learning rate parameter. This parameter can be set as a fixed value, or as a step-by-step incremental value, depending on the momentum value or can be learned by adaptive algorithms during learning. The values of learning rates used in this study are 0.0001 and 0.0025.
- **Optimization algorithm:** In deep learning applications, the learning process is basically an optimization problem. To solve nonlinear problems by choosing the optimum value, optimization methods are used. Optimization algorithms such as stochastic gradient descent, adamax, adagrad, are generally used in deep learning applications. There are differences in performance and speed between these algorithms. In

this study, AdamOptimizer is used as the optimization algorithm.

- **Activation function:** In multilayer artificial neural networks, activation functions are used for nonlinear transformation operations. The output of the hidden layers is normalized by using different activation functions such as sigmoid, tanh, ReLu, and others to obtain back derivative in the hidden layers. In this study, Relu is selected as the activation function.
- **Time step:** RNNs have a recurring hidden state, the activation of which is determined by the previous activation. They can be utilized in sequential data because they share parameters for different time steps. The length of the input sequence is called time step. The time steps values used in this study are 10, 25, 50, 100 and 200.

We investigate the impact of these parameters in Section IV.

TABLE IV
IMPACT OF HYPERPARAMETERS ON THE PERFORMANCE OF LSTM (WITH ACCELEROMETER AND GYROSCOPE, SMARTWATCH)

| Epoch | Hidden Layer | Batch Size | Learning Rate | Time Steps | Accuracy | Loss | Run Time |
|---|---|---|---|---|---|---|---|
| 30 | 32 | 512 | 0.0001 | 200 | 0.85 | 0.90 | 19:28.1 |
| 30 | 32 | 512 | 0.0025 | 200 | 0.92 | 0.40 | 17:52.0 |
| 30 | 32 | 1024 | 0.0001 | 200 | 0.83 | 0.98 | 09:37.7 |
| 30 | 32 | 1024 | 0.0025 | 200 | 0.91 | 0.42 | 10:03.7 |
| 30 | 32 | 2048 | 0.0001 | 200 | 0.82 | 1.06 | 05:56.3 |
| 30 | 32 | 2048 | 0.0025 | 200 | 0.90 | 0.52 | 05:45.4 |
| 30 | 64 | 512 | 0.0001 | 200 | 0.90 | 1.09 | 18:46.0 |
| 30 | 64 | 512 | 0.0025 | 200 | 0.93 | 0.39 | 17:57.6 |
| 30 | 64 | 1024 | 0.0001 | 200 | 0.88 | 1.32 | 10:09.4 |
| 30 | 64 | 1024 | 0.0025 | 200 | 0.93 | 0.42 | 10:40.3 |
| 50 | 32 | 512 | 0.0001 | 200 | 0.87 | 0.76 | 28:52.2 |
| 50 | 32 | 512 | 0.0025 | 200 | 0.92 | 0.37 | 34:22.1 |
| 50 | 32 | 1024 | 0.0001 | 200 | 0.86 | 0.83 | 0:21:37 |
| 50 | 32 | 1024 | 0.0025 | 200 | 0.93 | 0.38 | 0:19:07 |
| 50 | 64 | 1024 | 0.0025 | 200 | 0.93 | 0.38 | 0:17:02 |
| 50 | 64 | 1024 | 0.0025 | 100 | 0.89 | 0.46 | 09:22.0 |
| 50 | 64 | 1024 | 0.0025 | 50 | 0.86 | 0.54 | 04:58.2 |
| 60 | 32 | 512 | 0.0001 | 200 | 0.88 | 0.70 | 35:27.2 |
| 60 | 32 | 512 | 0.0025 | 200 | 0.93 | 0.36 | 34:39.9 |
| 60 | 32 | 1024 | 0.0001 | 200 | 0.83 | 0.90 | 10:59.9 |
| 60 | 32 | 1024 | 0.0025 | 200 | 0.89 | 0.46 | 09:51.1 |
| 60 | 32 | 2048 | 0.0001 | 200 | 0.83 | 0.93 | 05:56.1 |
| 60 | 32 | 2048 | 0.0025 | 200 | 0.88 | 0.51 | 05:49.4 |
| 60 | 64 | 1024 | 0.0025 | 200 | 0.90 | 0.43 | 10:41.3 |

## IV. PERFORMANCE ANALYSIS

In this section, we present the performance of the deep learning algorithms in terms of different parameters. We use four performance metrics: i) accuracy, also known as the true positive rate, ii) F1-score which is the harmonic mean of precision and recall, iii) run time which is time spent to train and test an algorithm and iv) loss (LSE or L2-norm was used). As mentioned, we performed the experiments with scenario data to make the experiments similar to daily life patterns where activities are performed sequentially. But in our previous work [4], we applied different machine learning algorithms on the dataset without considering a scenario. For the experiments, we used Python programming language and TensorFlow [15] an open-source machine learning library developed by Google. Evaluations are performed on a computer that has four NVidia Tesla P100 GPUs, but only one GPU was reserved for the runs.

### A. *Performance of LSTM*

In this section, we present the results obtained with different combinations of hyperparameters using data from the accelerometer and the gyroscope sensors of smartwatches. Then, we examine the impact of also using data from the phone on the recognition performance of activities. Finally, we perform tests using accelerometer, gyroscope, and magnetometer sensors to better understand the impact of the sensor type when they are used alone or in combination.

#### 1) *Impact of hyperparameters*

In this section, we evaluate the performance of LSTM according to different hyperparameter values that were explained in Section III.C. In Table IV, we present the results: 3 different values for epoch, 2 values for the number of hidden layers, 3 values for batch size, 2 values for learning rate and 3 values for time steps are evaluated. The values are selected according to the values used in similar studies that were presented in Section II. Accuracy and loss results and also Run Time values, presented in hour:minute:second.millisecond format, are also shown in the table. In the following, we summarize our findings for different hyperparameters:

- **Epoch number:** In [9] the authors obtained an accuracy of 97% with an epoch number of 50 using an LSTM model. Taking this into account, we started with epoch number 30 and increased the value till 60. According to the results, if the other hyperparameters are kept constant and only the epoch numbers are changed, for example when the number of hidden layers is 32, the batch size is 512, the learning speed is 0.0001, the time step is 200 and the epoch number is 30, the accuracy value is obtained as 84%.[2] When the epoch number is increased to 50, 1 the accuracy is 87%. If we further increase the epoch number to 60, the accuracy is 87%. We observed that increasing the number of epochs generally increases success to a certain extent, but after a point, success remains constant or begins to decrease. According to this result and the other values presented in Table IV, the results with epoch number as 50 are better than 30. Since the success rates did not increase for the epoch number of 60 and the execution time was extended, we did not further increase the number of epochs.
- **Number of hidden layers:** Based on the results, increasing the number of hidden layers has a positive effect on the performance, up to a certain success rate, but does not affect that. For our experiments, this rate is 92% of accuracy. For example, when epoch number

[2] Although the accuracy values are presented between 0 and 1 in the table, we refer to the values in percentage in the text, for ease of reading and comparing with the results of the related work.

is 30, the number of hidden layers is 32, the batch size is 512, the learning rate is 0.0001, and the time step is 200, the accuracy is 84% and when the number of hidden layers is increased to 64, the accuracy increases to 89%. However, when the epoch number is 50, the batch size is 1024, the learning rate is 0.0025 and the time step is 200, the increase in the number of layers hidden from 32 to 64 does not change the accuracy result of 92%.

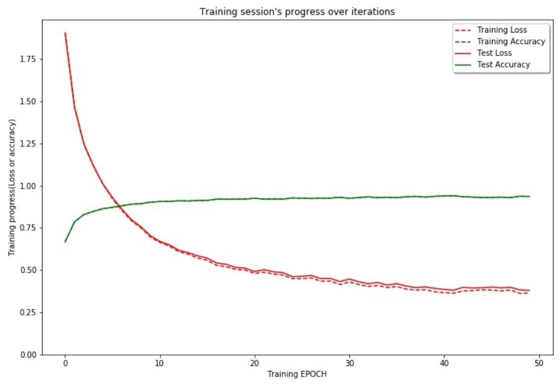
- **Batch size:** In our study, 3 different batch sizes, which are 512, 1024 and 2048, were used. Generally, when we increase the batch size, we found that the accuracy did not change or decrease. For the values of 512 and 1024, when we increase the batch size, accuracy decreases for learning rate 0.0001, accuracy sometimes decreases and sometimes remains constant for 0.0025. For example, when the batch size of 512 for the epoch number of 30, the number of hidden layers of 32, the learning rate of 0.0001 and the time step value of 200 the accuracy value was 84% while for a batch size of 1024 the accuracy was 82%. For the same values, there was no difference in accuracy results depending on the batch size of 1024 and 2048. Besides, increasing the batch size from 1024 to 2048 at learning rate 0.0001 and 0.0025 for epoch number of 60, hidden layer of 32, time step of 200 decreased 1% the accuracy value.
- **Learning rate:** We selected two different learning rate values which are 0.0001 and 0.0025. As can be seen in Table IV, for the LSTM model, better results are obtained for the learning rate of 0.0025.
- **Time step:** We used this hyperparameter at a constant value of 200 at the beginning. However, we also worked on the RNN model and observed that decreasing the time step in the RNN model had a positive effect. For this reason, also for LSTM, we tested the epoch number as 50, the number of hidden layers as 64, the batch size as 1024 and the learning rate as 0.0025 by decreasing the time step parameter. Consequently, we have determined that the best value of this hyperparameter is 200 for the LSTM model because we obtain smaller accuracy values compared to the RNN model by decreasing this value.

Based on the results of LSTM, we observe the best combination of hyperparameters as follows: the epoch number as 50, the number of hidden layer as 64, batch size as 1024, learning rate as 0.0025 and time step as 200 are best values which have approximately an accuracy of 0.92, or in other words 92%, and a loss value of 0.35. Run times are also often less than 30 minutes, only in three cases, it was observed to be more.

Figure 3a shows the accuracy and loss curves in the training and testing phases with the best hyperparameter combination. It was found that when the number of epochs increases, the accuracy increases and the loss decreases. Besides, the curves of the test data show that the model does not overfit. The actual and predicted values for each activity is shown in a confusion matrix in Figure 3b with the best performing parameter combination. It was observed that the most confusing activity couples are smoking in a group (smokeGroup) and smoking while standing (smokeSD), drinking while sitting (drinkST) and smoking while sitting (smokeST), drinking while sitting (drinkST) and drinking while standing (drinkSD), drinking while standing (drinkSD) and smoking while standing (smokeSD). This was also reported in our previous work [5].

### 2) *Impact of using watch and/or phone*

Considering the activities included in the dataset, it was important to distinguish smoking activity from the other activities which have similar wrist movements, such as drinking activities. That is why we mainly used the data collected from a smartwatch, but users also carried a smartphone in their



| | | **Predicted Class** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **smokeSD** | **smokeST** | **eat** | **drinkSD** | **drinkST** | **sit** | **stand** | **smokeWalk** | **walk** | **smokeGroup** |
| **Actual Class** | **smokeSD** | 1516 | 29 | 0 | 115 | 7 | 7 | 10 | 4 | 5 | 198 |
| | **smokeST** | 34 | 1640 | 11 | 30 | 145 | 12 | 8 | 0 | 0 | 12 |
| | **eat** | 0 | 1 | 1847 | 26 | 5 | 11 | 4 | 0 | 0 | 3 |
| | **drinkSD** | 18 | 8 | 22 | 1638 | 134 | 0 | 17 | 1 | 5 | 49 |
| | **drinkST** | 1 | 126 | 15 | 138 | 1546 | 49 | 11 | 0 | 0 | 6 |
| | **sit** | 0 | 6 | 5 | 1 | 12 | 4402 | 0 | 0 | 0 | 6 |
| | **stand** | 0 | 2 | 3 | 13 | 3 | 0 | 5609 | 2 | 0 | 1 |
| | **smokeWalk** | 0 | 0 | 0 | 4 | 0 | 0 | 1 | 803 | 6 | 11 |
| | **walk** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 21 | 1633 | 2 |
| | **smokeGroup** | 187 | 9 | 8 | 41 | 5 | 4 | 0 | 21 | 9 | 1209 |

(a) Accuracy and loss for LSTM
(b) Confusion matrix for LSTM

Fig. 3: Results with the best hyperparameter combination for LSTM (with accelerometer and gyroscope, smartwatch)

pockets, and in this section, we explore whether smartphone sensors also have an impact on the performance of activity recognition. Additionally, we investigate the performance with different sensor combinations here and also in Section IV.A.3.

TABLE V

EVALUATION RESULTS (%) WITH DIFFERENT SENSOR AND DEVICE COMBINATIONS USING LSTM (ACC: ACCELEROMETER, GYR: GYROSCOPE AND MAG: MAGNETOMETER)

| | Smartwatch | | | | Smartphone | | | | Smartwatch + Smartphone | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **Accuracy** | **F1- score** | **Precision** | **Recall** | **Accuracy** | **F1- score** | **Precision** | **Recall** | **Accuracy** | **F1- score** | **Precision** | **Recall** |
| **Acc** | 92.0 | 92.0 | 92.2 | 92.0 | 88.9 | 88.8 | 88.8 | 88.9 | 96.9 | 96.9 | 96.9 | 96.9 |
| **Acc+Gyr** | 92.7 | 92.7 | 92.7 | 92.7 | 90.6 | 90.5 | 90.5 | 90.6 | 96.9 | 96.9 | 96.9 | 96.9 |
| **Acc+Mag** | 94.7 | 94.7 | 94.7 | 94.7 | 96.1 | 96.1 | 96.3 | 96.1 | 97.6 | 97.6 | 97.7 | 97.6 |
| **Acc+Gyr+Mag** | 95.6 | 95.6 | 95.6 | 95.6 | 96.2 | 96.2 | 96.2 | 96.2 | 98.2 | 98.2 | 98.3 | 98.2 |

Table V shows the accuracy, precision, recall and F1-scores (in percentage) obtained by these devices and sensors. Device-based comparison shows that the success rate is higher when using smartwatch data as expected since most of the activities include wrist motions. Besides, the achieved success rates with smartphone data are observed to be the highest using accelerometer-magnetometer and accelerometer- gyroscope-magnetometer sensor data, which is about 95-96%. However, the success rates for the smartwatch with different sensor combinations are not very different and it is above 92%. When the data of the two devices are used together, there is an increase of at least 4% in F1-scores in case of using only accelerometer and accelerometer-gyroscope. There is a slight increase when accelerometer-magnetometer and accelerometer-gyroscope-magnetometer combinations are used. We observe that we can achieve slightly higher success rates when the two devices are used together.

### 3) *Comparison of sensors*

As mentioned, results according to the different sensor and device combinations are presented in Table V. If a smartwatch or a smartphone is used alone, the minimum success rates are observed with using only the accelerometer sensor. When two other sensors are used together with the accelerometer, we observe that the addition of magnetometer increases the success rates more than using gyroscope, especially when using the only smartphone. When all the sensors are used together, results are very similar to using accelerometer and magnetometer combination.

When two devices are used together, even with using only accelerometer data, we achieve more than 96.9% accuracy and F1-score. If the magnetometer data is fused with the accelerometer data, there is a slight increase, about 0.7%, both in accuracy and F1-scores. Using all the sensors brings a similar increase in the results which is about 0.6%.

Sampling more sensors may increase battery consumption, particularly for the smartwatch. In this case, rather than using other sensors, it may be a better option to use smartwatch and smartphone together, especially considering the fact that smartwatches are usually used in combination with a smartphone. If it is not preferred to use both devices due to resource consumption, then the only smartwatch can be used considering the target activities.

TABLE VI
IMPACT OF HYPERPARAMETERS ON THE PERFORMANCE OF RNN (WITH ACCELEROMETER AND GYROSCOPE, SMARTWATCH)

| Epoch | Hidden Layer | Batch Size | Learning Rate | Time Steps | Clip | Accuracy | Loss | Run Time |
|---|---|---|---|---|---|---|---|---|
| 30 | 32 | 512 | 0.0001 | 200 | --- | 0.549 | 1.604 | 0:06:38.128 |
| 30 | 32 | 512 | 0.0025 | 200 | --- | 0.24 | 2.311 | 0:06:31.41 |
| 30 | 32 | 1024 | 0.0001 | 200 | --- | 0.492 | 1.67 | 0:03:41.9 |
| 30 | 32 | 1024 | 0.0025 | 200 | --- | 0.49 | 1.514 | 0:03:53.9 |
| 30 | 32 | 2048 | 0.0001 | 200 | --- | 0.374 | 2.077 | 0:02:32.45 |
| 30 | 32 | 2048 | 0.0025 | 200 | --- | 0.576 | 1.405 | 0:02:13.6 |
| 30 | 64 | 512 | 0.0001 | 200 | --- | 0.473 | 2.208 | 0:06:55.286 |
| 30 | 64 | 512 | 0.0025 | 200 | --- | 0.311 | 2.079 | 0:06:57.55 |
| 30 | 64 | 1024 | 0.0001 | 200 | --- | 0.504 | 2.172 | 0:03:44.2 |
| 50 | 64 | 1024 | 0.0001 | 50 | --- | 0.757 | 1.421 | 0:01:38.724 |
| 50 | 64 | 1024 | 0.0025 | 100 | 5 | 0.789 | 0.768 | 0:03:33.774 |
| 50 | 64 | 1024 | 0.0025 | 100 | 10 | 0.832 | 0.622 | 0:03:25.44 |
| 50 | 64 | 1024 | 0.0025 | 200 | --- | 0.551 | 1.476 | 0:06:34.59 |
| 50 | 64 | 1024 | 0.0025 | 200 | 5 | 0.773 | 0.868 | 0:06:04.45 |
| 100 | 32 | 1024 | 0.0025 | 100 | -- | 0.72 | 0.88 | 0:06:29. |
| 100 | 64 | 512 | 0.0001 | 10 | --- | 0.769 | 1.079 | 0:01:42.704 |
| 100 | 64 | 512 | 0.0001 | 25 | --- | 0.774 | 1.049 | 0:03:30.680 |
| 100 | 64 | 512 | 0.0001 | 25 | 5 | 0.788 | 1.076 | 03:49.52332 |
| 100 | 64 | 512 | 0.0025 | 100 | --- | 0.827 | 0.572 | 0:11:34.07 |
| 100 | 64 | 1024 | 0.0001 | 25 | --- | 0.762 | 1.268 | 0:01:44.53 |
| 100 | 64 | 1024 | 0.0025 | 50 | 5 | 0.836 | 0.543 | 0:03:09.547 |
| 100 | 64 | 1024 | 0.0025 | 100 | -- | 0.769 | 0.844 | 0:06:00 |
| 100 | 64 | 1024 | 0.0025 | 100 | 5 | 0.84 | 0.542 | 0:06:50.76 |
| 300 | 64 | 256 | 0.0001 | 25 | 5 | 0.811 | 0.768 | 0:23:41.8066 |

## B. *Performance of RNN*

We present the results obtained using the VRNN model with different hyperparameter values in Table VI. The columns include the hyperparameters. Clip column includes the values used in gradient clipping which is a method to prevent exploding gradients deep architectures. As we will explain, RNN model experienced the gradient problem and these values are used to solve the problem.

To compare the results with the results of the LSTM model (presented in Table IV), for example, the accuracy of VRNN at 50 epochs, 64 hidden layers, 1024 batch sizes and 0.0025 learning rate and 50 time steps (as mentioned, these values were the best performing combination in LSTM), is about 77%, which was observed to be 92% accuracy and 0.38 loss for LSTM. In general, lower accuracy values and higher loss values are obtained. As mentioned, when we are dealing with large sets of sequential series of data, we frequently encounter the gradient disappearance problem in VRNN. While the weight matrix is being updated in the backtracking during the training phase, its effect on the result is slow because the gradient values are too small, and we think that the model could not be well trained.

In Figure 4 we present the learning curve for VRNN. Compared to Figure 3a, a smoother curve was obtained for LSTM. After each training phase, it was observed that certain values of the weight matrix were very small. To solve the problem of disappearing gradient and to optimize the results, we first reduced the parameter time step because VRNN may not follow the previous information. At the same time, we have increased the number of epochs. Thanks to these improvements, we obtained the best accuracy result (about 84%) and loss result (about 0.54) while using 100 epochs, 64 hidden layers, 1024 batch sizes, 0.0025 learning rate and 50 time steps. However, recognition results of the LSTM model are still higher than RNN because, as we mentioned above, it can be difficult to train RNN models due to the problems that require learning long-term dependencies with the gradient problems. As LSTM actually prevents problems that arise in RNNs, thanks to the cells in its structure, the trained model predicts the new test data more accurately. On the other hand, run times are often less than the run times observed with the LSTM architectures.

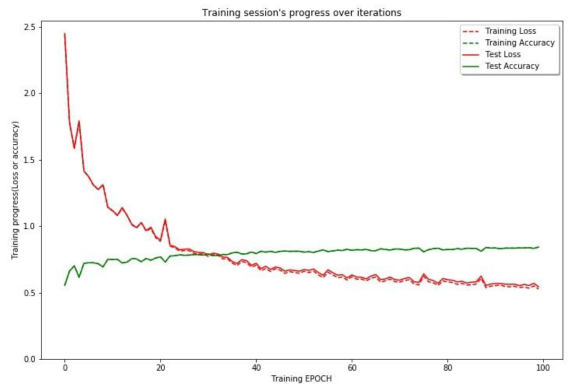


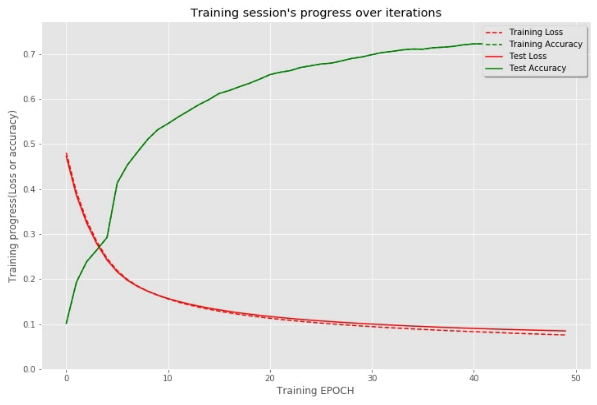
Fig. 4: Accuracy and loss for RNN with the best hyperparameter combination (with accelerometer and gyroscope, smartwatch)

## *C. Performance of CNN*

In [8], CNN was applied on the same dataset but the dataset was divided into 3 groups according to the types of activities unlike the organization of the scenarios in this study, as presented in Table II. In that study, an F1-score of 92-96% was reported. In this study, we achieved F1-scores close to these results and even higher with the LSTM model. However, we wanted to compare the performance of LSTM and CNN also on the scenario data.

CNNs have different structures than RNNs. For example, in CNNs, instead of the time step parameter, the window size parameter determines the size of the input. Besides, the convolution process in CNNs is performed using kernels. Therefore, kernel size and stride of this kernel are important parameters. Epoch number, hidden layer number, batch size, learning rate, activation function and optimization algorithm parameters are similar. The CNN architecture consists of a convolution layer with the Relu activation function, max pooling layer with a stride of 2 and a convolution layer. One fully connected layer and an output layer with Softmax complete the structure of the model.

In the analysis, when we used the whole scenario dataset, we encountered a resource shortage error. Accordingly, we reduced the data size and tested the model with only 509,170 rows (about 14.14 hours) from the dataset (original dataset has 2,349,970 rows, about 65.27 hours). We included less repetitions of activities in the dataset. As in RNN models, this dataset is divided into two groups: 80% training and 20% testing. The accuracy-loss graph of this model is given in Figure 5a and the confusion matrix is given in Figure 5b. The highest accuracy obtained for the CNN model was approximately 71% using 50 epochs, 1000 hidden layers, 16 batch sizes, 0.0001 learning rate, 300 window sizes and 60 kernel sizes. As it can be seen in the confusion matrix and the loss values, the prediction results are not bad except for smokeGroup, smokeSD and smokeWalk activities. Specifically, the number of false predictions for smokeGroup-smokeSD and smokeGroup-smokeWalk activities is greater than the number of correct predictions. Compared to the results reported in [8], there is a 20% difference in terms of accuracy. We think that the difference could be due to the fact that we used a reduced version of the scenario data while they used parts of the data organized differently. As future work, CNN experiments should be repeated considering the whole dataset after eliminating the resource shortage error.



| | | Predicted Label | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | smokeSD | smokeST | eat | drinkSD | drinkST | sit | stand | smokeWalk | walk | smokeGroup |
| True Label | smokeSD | 25 | 0 | 0 | 0 | 1 | 0 | 1 | 7 | 1 | 17 |
| | smokeST | 5 | 27 | 4 | 7 | 2 | 1 | 1 | 0 | 0 | 1 |
| | eat | 0 | 5 | 35 | 5 | 3 | 0 | 0 | 0 | 0 | 0 |
| | drinkSD | 1 | 8 | 3 | 22 | 9 | 1 | 0 | 1 | 2 | 1 |
| | drinkST | 4 | 2 | 0 | 5 | 36 | 1 | 0 | 2 | 0 | 2 |
| | sit | 1 | 0 | 0 | 1 | 6 | 82 | 1 | 2 | 2 | 0 |
| | stand | 0 | 1 | 0 | 0 | 0 | 0 | 141 | 0 | 0 | 1 |
| | smokeWalk | 10 | 2 | 1 | 1 | 1 | 0 | 0 | 18 | 4 | 12 |
| | walk | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 90 | 0 |
| | smokeGroup | 13 | 5 | 1 | 2 | 1 | 0 | 1 | 13 | 3 | 10 |

a) Accuracy and loss for CNN

(b) Confusion matrix for CNN

Fig. 5: Results with the best hyperparameter combination for CNN (with accelerometer and gyroscope, smartwatch)

### D. *Summary*

To summarize our findings, when we consider the three deep learning architectures, we observe that the recognition performance with LSTM is much higher than that of CNN and RNN. LSTM works better with sequential data and with the cell structure in the architecture, it models the data better than RNN. CNN is mostly used for image recognition problems or with sensor data it is used in the feature extraction step in some studies [7]. One could also use a combination of different algorithms or ensemble methods [16], however since LSTM achieved a high recognition accuracy, we did not further evaluate the performance with other algorithms. We also explored the performance with different hyper-parameters and analyzed how selecting different values for epoch number, batch size, number of hidden layers, learning rate, may affect the performance. We report the values where we achieved the highest recognition accuracies and did not consider training time or memory limitations. For example, using 64 hidden layers instead of 32 will increase the training time and if training time is an important parameter than this tradeoff should be analyzed. It may be argued that training and running deep learning models are costly on resource-limited wearable and mobile devices. However, recent mobile deep learning platforms, such as TensorFlow Lite, CoreML make it possible to train and run deep learning algorithms on mobile devices.

One of the questions that we focused on was the effect of using different sensors, accelerometer alone and together with gyroscope and magnetometer. Using other sensors besides accelerometer slightly increases the success rates, however sampling three sensors increases battery consumption. Hence, only accelerometer can be used if battery level is critical. Another question was whether to use phone sensors besides the smartwatch sensors. Again, we observe a slight increase in success rates when the two devices are used together. However, using smartwatch alone also exhibits good performance. As mentioned, rather than using the three sensors, it may be a better option to use a smartwatch and a smartphone together, since smartwatches are usually used in combination with a smartphone.

## V. CONCLUSION

In this paper, we studied the problem of sensor-based smoking recognition using three deep learning architectures (LSTM, RNN and CNN) with different hyperparameter settings. We use a dataset of 10 different activities collected from 11 participants with the accelerometer, gyroscope and magnetometer sensors embedded in smartphones and smartwatches. Experiment results show that the performance of LSTM is much higher than that of CNN and RNN. The best hyperparameters are observed as 50 epochs, 64 hidden layers, 1024 batch size, 0.0025 learning rate and 200 time step for LSTM. When the performance of sensors and devices are compared, the highest performance, 98.2% accuracy, is achieved using the fusion of accelerometer, gyroscope and magnetometer data of smartphone and smartwatch. It is observed that the magnetometer improves performance much more than the gyroscope when used with the accelerometer. The maximum F1-score for all types of devices and sensors is observed to be about 98.2% with a combination of three sensors from smartwatch and smartphone data.

In this study, we focused specifically on smoking activity recognition. However, this study can be extended to different types of activities and also real-time activity recognition can be considered. Moreover, we focused on the usage of deep learning architectures separately, however different combinations of the architectures or ensemble methods can also be evaluated.

## ACKNOWLEDGEMENT

This work is supported by Tubitak under Grant Number 117E761. We thank Beyza Cankuş for her contributions in the design of experiments.

## REFERENCES

[1] Bulling A, Blanke U, Schiele B. A tutorial on human activity recognition using body-worn inertial sensors. ACM Computing Surveys (CSUR) 2014; 46 (3): 33. doi: 10.1145/2499621

[2] Shoaib M, Bosch S, Incel OD, Scholten H, Havinga P. A survey of online activity recognition using mobile phones. Sensors 2015; 15 (1): 2059-2085. doi: 10.3390/s150102059

[3] Gjoreski H, Lustrek M, Gams M. Accelerometer placement for posture recognition and fall detection. In: Intelligent Environments (IE), 7th International Conference on Intelligent Environments; Nottingham, United Kingdom; 2011. pp. 47-54.

[4] Agac S, Shoaib M, Durmaz Incel O. Smoking recognition with smartwatch sensors in different postures and impact of user's height. Journal of Ambient Intelligence and Smart Environments. 2020(Preprint):1-23.

[5] Shoaib M, Scholten H, Havinga P, Incel O. A hierarchical lazy smoking detection algorithm using smartwatch sensors. In: 2016 IEEE 18th International Conference on e-Health Networking, Applications and Services; Munich, Germany; 2016. pp. 1-6.

[6] Ordóñez FJ, Roggen D. Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition. Sensors 2016; 16 (1): 115. doi: 10.3390/s16010115

[7] Wang J, Chen Y, Hao S, Peng X, Hu L. Deep learning for sensor-based activity recognition: A survey. Pattern Recognition Letters. 2019. 119: 3-11.

[8] Alharbi F, Farrahi K. A convolutional neural network for smoking activity recognition. In: 2018 IEEE 20$^{th}$ International Conference on e-Health Networking, Applications and Services; Ostrava, Czech Republic; 2018. pp. 1-6.

[9] Parameswarappa G. Human activity recognition using deep recurrent neural nets, lstm and tensorflow on smartphones. MS, University of Massachusetts Dartmouth, Dartmouth, Massachusetts, USA, 2017.

[10] Kwapisz J, Weiss GW, Moore SA. Activity recognition using cell phone accelerometers. ACM SigKDD Explorations Newsletter 12.2 (2011): 74-82.

[11] San-Segundo R, Blunck H, Moreno-Pimentel J, Stisen A, Gil-Martin M. Robust human activity recognition using smartwatches and smartphones. Engineering Applications of Artificial Intelligence 2018; 72: 190-202.

[12] Liu Q, Zhou Z, Shakya SR, Uduthalapally P, Qiao M et al. Smartphone sensor-based activity recognition by using machine learning and deep learning algorithms. International Journal of Machine Learning and Computing 2018; 8 (2): 121-6.

[13] Chen Y, Xue Y. A deep learning approach to human activity recognition based on single accelerometer. In: 2015 IEEE International Conference on Systems, Man, and Cybernetics; Hong Kong, China; 2015. pp. 1488-1492.

[14] Goodfellow I, Bengio Y, Courville A. (2016). Deep learning. MIT press.

[15] Abadi M, et al. Tensorow: A system for large-scale machine learning. In: 12th USENIX symposium on operating systems design and implementation (OSDI 16); Savannah, GA, USA; 2016. pp. 265-283.

[16] Mukherjee, D., Mondal, R., Singh, P. K., Sarkar, R., & Bhattacharjee, D. (2020). EnsemConvNet: a deep learning approach for human activity

recognition using smartphone sensors for healthcare applications. Multimedia Tools and Applications, 79(41), 31663-31690.

## BIOGRAPHIES

**YASEMIN AKAN, MSc.,** received her BSc. degree in computer engineering from Galatasaray University, Istanbul, Turkey, in 2019. Currently, she is pursuing the MSc degree with the department of computer engineering, Bogazici University, Istanbul, Turkey. She is also working as an Assistant Software Engineer in Insurance Application Development Team, Yapi Kredi Teknoloji A. S., Istanbul, Turkey.

**SUMEYYE AGAC, MSc.,** received her BSc. and MSc. degrees in computer engineering from Galatasaray University, Istanbul, Turkey, in 2016 and 2019, respectively. Currently, she is a Ph.D. student in Bogazici University, Istanbul, Turkey. Her current research interests are human activity recognition, machine learning and e-health.

**OZLEM DURMAZ INCEL, Assoc. Prof.,** received her Ph.D. in computer science from the University of Twente, Netherlands, in March 2009. Then, she worked as a postdoctoral researcher at Bogazici University in 2009-2012. She is currently working as an associate professor in the Computer Engineering Department of Galatasaray University. Her research interests include wearable computing, Internet of Things and applied machine learning.