Journal of Ambient Intelligence and Humanized Computing
https://doi.org/10.1007/s12652-020-02865-4

ORIGINAL RESEARCH

Enhancing human activity recognition using deep learning and time series augmented data

Luay Alawneh[1] · Tamam Alsarhan[1,2] · Mohammad Al-Zinati[1] · Mahmoud Al-Ayyoub[1] · Yaser Jararweh[1] · Hongtao Lu[2]

Received: 23 April 2020 / Accepted: 16 December 2020

Abstract
Human activity recognition is concerned with detecting different types of human movements and actions using data gathered from various types of sensors. Deep learning approaches, when applied on time series data, offer promising results over intensive handcrafted feature extraction techniques that are highly reliant on the quality of defined domain parameters. In this paper, we investigate the benefits of time series data augmentation in improving the accuracy of several deep learning models on human activity data gathered from mobile phone accelerometers. More specifically, we compare the performance of the Vanilla, Long-Short Term Memory, and Gated Recurrent Units neural network models on three open-source datasets. We use two time series data augmentation techniques and study their impact on the accuracy of the target models. The experiments show that using gated recurrent units achieves the best results in terms of accuracy and training time followed by the long-short term memory technique. Furthermore, the results show that using data augmentation significantly enhances recognition quality.

Keywords Accelerometer · Time series data · Vanilla RNN · Long-short term memory (LSTM) · Gated recurrent units (GRU) · Moving average smoothing · Exponential smoothing

1 Introduction

Human activity recognition (HAR) is a classification problem that is concerned with identifying human movements and activities for monitoring purposes as well as the detection of anomalies in behavior (Osmani et al 2008). Human activity data could be collected remotely using different types of sensors such as cameras, radars, and wireless remote sensors. However, these methods are expensive to implement and may not be accessible at all times and places. On the contrary, human movements could be recorded using wearable sensors and mobile phones. These attached components, through their accelerometers and gyroscopes, record all the user activities, which make them more practical and affordable. The emergence of modern artificial intelligence techniques and the vast amount of human activity data leveraged the scale of applications for human activity recognition.

Healthcare is one of the main fields that benefit from HAR (Bidargaddi et al. 2007), where healthcare providers can monitor and analyze the wellbeing and behavior of their patients (Woznowski et al. 2016). For example, Powell et al. (2007) utilized HAR for clinical assessments of tremor. They employed a combination of triaxial microelectromechanical System (MEMS) inertial sensors, a range of embedded components, and signal processing algorithms to quantitatively capture and analyze tremor. HAR contributes significantly to many other fields such as business management, transport

✉ Luay Alawneh
lmalawneh@just.edu.jo

Tamam Alsarhan
thalsarhan14@cit.just.edu.jo; tamamhazza@sjtu.edu.cn

Mohammad Al-Zinati
mhzinati@just.edu.jo

Mahmoud Al-Ayyoub
maalshbool@just.edu.jo

Yaser Jararweh
yijararweh@just.edu.jo

Hongtao Lu
htlu@sjtu.edu.cn

[1] Faculty of Computer and Information Technology, Jordan University of Science and Technology, Irbid, Jordan

[2] Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China

Published online: 07 January 2021

and traffic management, surveillance and security, eldercare, and sports (Wang et al. 2019). For example, advertising companies utilize HAR in the understanding of their customers' needs and behaviors to send personalized messages by following the principle of Know Your Customer (KNC) (Woodruff and Gardial 1996).

Data acquisition is the initial step in the classification process. Various sensor types have been used with human activity recognition tasks. Video sensors are being utilized in smart homes to detect human activities (Mozer 1998; Jalal et al. 2014), where a series of images are captured to detect human activities to distinguish between normal behaviors and anomalies (Behera et al. 2012; Gutchess et al. 2007). Although this approach has several advantages, privacy and cost constraints hinder its applicability to a broader range of domains.

To overcome these shortcomings, practitioners and researchers have investigated the use of inertial sensors in medical field applications. Such sensors include gyroscopes and accelerometers, which are used to detect fall states (Wu and Xue 2008) and monitor human movements (Sabatini et al. 2005). Gyroscopes measure angular rotation while accelerometers measure acceleration in uniaxial, biaxial or triaxial dimensions. Nowadays, smartphones, smartwatches, smart glasses, and smart helmets commonly include one or more of these sensors. In this paper, we focus on the use of accelerometer data with triaxial dimensions.

Several studies investigated the application of classical machine learning approaches for the classification of human activities (Chen et al. 2017; Sefen et al. 2016; Sukor et al. 2018). However, most of these approaches rely on feature extraction techniques that are constrained by the quality of the selected domain parameters. Hence, affecting their generalization to a broader range of activities. Moreover, the accuracy of these methods may degrade when applied to people who were not included in the training phase (Kolekar and Dash 2016; Paul and George 2015).

Another HAR challenging issue is the resemblance among several human activities, due to the large intra-class scatter and small inter-class separation, which makes it difficult to distinguish among them. Several studies targeted this problem by utilizing margin mechanisms to acquire more knowledge about discriminative feature representations to further enhance intra-class compactness and inter-class diversity (Lv et al. 2020; Deng et al. 2019).

Deep learning approaches such as convolutional neural networks (CNN) (Zeng et al. 2014; Jordao et al. 2018) and artificial neural networks (ANN) (Oniga, and Suto 2014, Li et al. 2018) emerged to overcome the limitations imposed by classical machine learning approaches. Applying deep learning to time series data proved to be useful in the handling of interdependencies found in data (Cho et al. 2014). Recent studies proposed to use recurrent neural networks (RNN) for time series classification as their dynamic structure enables them to benefit from the temporal information of the input sequence (Uddin et al. 2020). Moreover, incorporating a prediction task with the training process significantly improves the learning process and the model generality (Husken and Stagge 2003).

In this paper, we apply data augmentation on time series data to further improve the accuracy of deep learning models in the context of HAR. We use the traditional Vanilla RNN (Husken and Stagge 2003), Long-Short Term Memory (LSTM) (Hochreiter and Schmidhuber 1997), and the Gated Recurrent Units (GRU) (Chung et al. 2014) models. As opposed to the traditional Vanilla RNN model, LSTM efficiently handles serial information with long dependencies using its gating scheme for data representation. The GRU resembles LSTM as it has a forget gate but with fewer parameters. Studies showed that GRU is able to incorporate context information when applied to sequence learning (Chung et al. 2014). Furthermore, GRU handles the correlations between close data points in time series sequences. Unlike LSTM, the GRU provides a balance between the previous and new memory contents by exposing its memory content at each time step.

Bidirectional RNN is a special version of the unidirectional RNN. It divides RNN neuron states into Forward and Backward states and runs the input in two opposite directions from past to future (just like unidirectional RNN), and then from future to the past. It exploits the bidirectional temporal dependencies from data. Studies dealing with time series data showed that the bidirectional approach's accuracy prevails when compared to other artificial neural network structures (Schuster and Paliwal 1997). In this paper, we study the performance of the bidirectional approach on the target RNN models for HAR. In our previous short study (Alsarhan et al. 2019), we only investigated the effectiveness of the bidirectional GRU model on the raw UniMiB SHAR (Micucci et al. 2017) dataset. In this paper, we compare the performance of the Vanilla RNN, LSTM and GRU models on the UniMiB SHAR, the WISDM (Kwapisz et al. 2011), and the UCI-HAR (Anguita et al. 2013) open-source datasets. More importantly, we investigate the impact of two time series data augmentation techniques on the performance of the three RNN models.

The rest of the paper is organized as follows. In Sect. 2, we present several related studies and compare them to our work. In Sect. 3, we discuss the target RNN models, followed by our approach in Sect. 4. Section 5 presents the evaluation and analysis of the results. Finally, we conclude our paper in Sect. 6 with a highlight on the future direction.

2 Related work

Several research studies used deep learning approaches on time series data for HAR (Nweke et al. 2018). Foerster et al. (1999) initiated the efforts of assessing the applicability of using accelerometer data for HAR. Recently, with the wide usage of wearable devices, this topic has gained more attention (Uddin et al. 2017; Hinton et al. 2006). Ramasamy and Roy (2018) presented a comprehensive survey summarizing the challenges that face HAR and the recent machine learning methods in this field. Hammerla et al. (2016) conducted a comparative study and showed that RNN approaches outperformed other machine learning techniques for HAR. Torres-Huitzil and Alvarez-Landero (2015) studied HAR on people who suffer from metabolic syndrome. To balance their movements, they built a hierarchical HAR system using neural networks. As a result, they were able to recognize five types of activities, which are static, walking, running, and walking up and down stairs. The experiments showed that their recognition system achieved a high classification rate of up to 92%.

Deep learning techniques provided accurate real-time classifications in the study proposed by Ravi et al. (2016), where they defined a set of features found in the sensor data to enhance the training and recognition accuracy. Husken and Stagge (2003) proposed to use recurrent neural networks for the categorization of time series input sequences. The results showed that using time series data improves the generalization ability of the model. Vepakomma et al. (2015) proposed a model for extracting handcrafted features from data captured by motion sensors. The model automatically feeds the data into the deep neural network (DNN) model to extract discriminative features for activity recognition (AR). Similarly, Plötz et al. (2011) proposed a partial weight sharing technique and applied it to accelerometer signals, aiming to obtain further improvements to the AR accuracy. For this sake, they performed principal component analysis and then fed the data to a DNN model. Moreover, they used the Restricted Boltzmann Machine (RBM) for feature extraction. Although they designed their model to advocate fully connected neural networks, it failed to capture the local dependencies among signals.

Uddin et al. (2020) used kernel principal component analysis (KPCA) to enhance the extracted features in data collected from multiple sensors such as electrocardiography, accelerometer and magnetometer. Ullah et al. (2019) proposed to use a stacked LSTM network for HAR that consists of a five LSTM cells. The sensor data is first preprocessed using a single layer network. The network uses the L_2 Regularizer in the cost function for model generalization. The proposed approach slightly improved the accuracy with an increase of 0.93%.

Hassan et al. (2018) investigated the use of a Deep Belief Network (DBN) for the classification of human activities using data gathered by smartphones equipped with inertial sensors. Their approach provided better results when compared to traditional machine learning approaches such as SVM. Jiang and Yin (2015) used Deep Convolutional Neural Networks (DCCN) for HAR. Their approach assembles the accelerometer and gyroscope data into an activity image. Then, the model automatically learns and extracts the optimal features from the activity image in order to use them in the recognition task.

Simonyan and Zisserman (2014) studied two DCCN architectures for the classification of videos for activity recognition. They used the Softmax activation function to decrease the overfitting in the models. Zeng et al. (2014) proposed to use CNN to extract discriminative features for activity recognition from motion sensors. More clearly, they introduced a partial weight sharing technique and applied it to accelerometer signals, aiming to obtain additional improvements on the recognition accuracy. Zahin and Hu (2019) proposed a semi-supervised learning approach to overcome the problem with scarce labeled data that combines CNN with vibrational auto-encoder (VAE) for HAR. The VAE is responsible for extracting the salient characteristics of human activity data and providing useful criteria for the reconstruction of compressed sensing. VAE generates new data points with slight difference from the original data by learning the true data distribution of a training set. CNN is used to extract discriminative features and to produce low-dimension latent codes. The results showed that using VAE improved the accuracy.

LSTM networks handle the vanishing gradient problem and the long dependencies in serial data (Gers et al. 2002). Thus, several studies utilized them for time series information. Mehdiyev et al. (2017) proposed a multi-stage deep learning approach for multivariate time series classification problems. The first stage uses unsupervised feature extraction using stacked LSTM Autoencoders. The next stage uses the deep feedforward neural network for making predictions. The advantage of this approach is the extraction of useful features without domain expert intervention. The authors validated their approach on a real-world dataset from the steel industry.

Murad and Pyun (2017) proposed using deep recurrent neural networks to overcome the long-range dependencies problem in variable-length input sequences. They compared the effectiveness of LSTM networks on several datasets. The results showed that using LSTM outperformed the traditional machine learning models such as SVM and KNN and other deep learning approaches such as deep believe networks (DBNs) and CNNs. Liciotti et al. (2020) proposed using LSTM for modeling spatio-temporal sequences captured by smart home sensors. They tested their approach

on the Center for Advanced Studies in Adaptive Systems dataset. The results showed that LSTM outperformed other machine learning approaches.

Graves et al. (2008) utilized the concept of incremental learning and the long-range contextual processing using the bidirectional long short-term memory (BLSTM) approach on two large unconstrained handwriting databases where they achieved 79.9% recognition accuracy. Veeriah et al. (2015) proposed dRNN by using a differential gating scheme for the LSTM network. They assessed their approach on the KTH 2D and the MSR Action3D datasets. The proposed dRNN approach achieved better recognition results when compared to the original LSTM model.

Chung et al. (2014) proposed to use traditional RNN, LSTM and GRU models in the classification of lexical utterances. The authors compared their approach to the ngram-based and the feedforward neural network language models, and the boosting classifiers. The results showed that the proposed approach outperformed these models. More clearly, the experiments showed that the traditional RNN model is a good fit for short utterances while LSTM and GRU provide better results for longer ones. Vu et al. (2017) used Self-Gated RNN for HAR on sensor data gathered from wearable devices. Their approach was superior to the traditional RNN approach. However, it provided similar results to the LSTM and GRU approaches.

Shen et al. (2018) used GRU for the prediction of financial time series data. The results showed that the GRU model using the Softmax activation function improved the prediction accuracy. Moreover, they extended their experiments by replacing the output layer in the GRU by an SVM model. The results showed that the prediction accuracy of the proposed GRU-SVM model slightly improved for the tested datasets.

Xia et al. (2020) proposed a deep learning approach for HAR that uses a two-layer LSTM architecture followed by convolutional layers. They tested their approach on publicly available datasets and achieved an accuracy of up to 95.8%. The study showed that using the LSTM-CNN approach provides a better generalization ability. Moreover, using the LSTM-CNN uses few model parameters and avoids the usage of complex feature extraction while maintaining high accuracy.

Che et al. (2018) proposed using a GRU-based model for handling missing values in multivariate time series data to overcome the shortcomings in the traditional RNN approach, which provides similar results to traditional machine learning approaches such as SVM. The GRU-based approach handles the missing information problem efficiently by including masking and time interval directly inside the GRU architecture. The proposed GRU-D model provided similar time and space complexity compared to the traditional RNN models when tested on synthetic and real-world datasets.

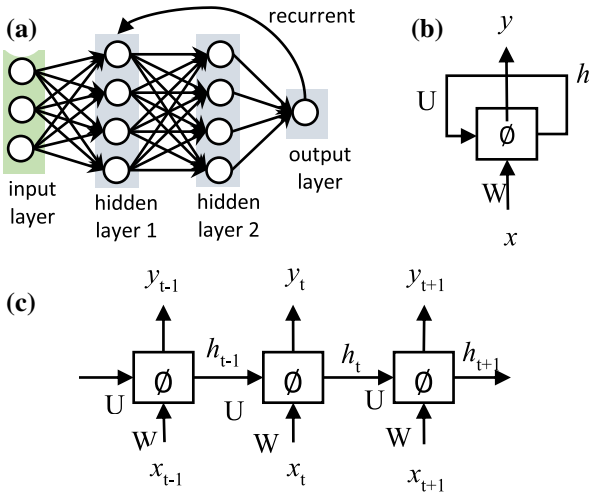

Fig. 1 Recurrent neural network architecture

Our approach investigates the application of data augmentation on different RNN models using three open-source datasets. In Section V, we compare the results of our approach with the state-of-the-art solutions. Our experimental results, supported by confidence intervals, showed that applying data augmentation on the target datasets significantly improves the accuracy of HAR models.

In the following, we present a background of neural networks and the different RNN models used in our study.

3 Recurrent neural networks

RNN is a popular artificial neural network model with a specific architecture where the relation between nodes performs a directed graph along the data sequence. Hence, this makes it applicable for tasks that contain sequences of data such as speech, text, genome, and numerical time series (Ravuri and Stolcke 2015). RNN models are commonly used for the classification of time series and sequential data.

Figure 1a depicts a typical neural network with a recurrent branch from the output node to the first hidden layer in the network. Figure 1b represents the basic architecture of an RNN, where it has an input, output, activation function, and a recurrent loop. The structure of RNN allows it to take a sequence of inputs, as shown in Fig. 1c, which represents an unfolded RNN into a full network. The power of RNN relies on the hidden state, which captures the information from one node to another along the sequence. Thus, it keeps propagating until the end of the structure. Generally, RNN receives the input x_0 from the sequence of inputs, performs some calculations resulting in $h_{0,}$ which, together with $x_{1,}$ compose the input to the next step. Similarly, the output h_1

with the input x_2 will be the input to the next step, and so on. It should be noted that y_t is the same as h_t.

The value of x_t represents the input of the system at time step t, whereas h_t, which is called the hidden state, is the output of the system at step t. Furthermore, W represents the weight matrix of the input to the hidden layer at time t, and U is the weight matrix of the hidden layer at time t-1. Finally, $\varnothing$ represents the activation function. The value of h_t is calculated using Eq. 1. The input x_t is modified by W and h_{t-1} is modified by U. The result from the two is then added. The weights determine the importance of current inputs and hidden states.

$$h_t = \phi\left(Wx_t + Uh_{t-1}\right) \tag{1}$$

RNN learns weights (W, U) through training using the backpropagation technique that is popular in every neural network structure. The network then determines the accuracy of the model by using an error function (loss function) and calculates the derivative of the loss function with respect to the weight. Finally, the network uses the resulting values to update the old weights aiming to acquire better results in the next training steps. Additionally, the network uses an activation function to simplify the mathematical calculations related to the application of the backpropagation.

Sharing the parameters along all steps is a crucial feature for RNN. The same task is performed at each step, but with different inputs. Accordingly, this significantly reduces the total number of parameters needed during learning. Moreover, RNN captures the dependencies between inputs, and hence preservers remembering them. As such, RNN has become widely used to solve many complicated real-life problems such as natural language processing, next-word prediction, time series anomaly detection, and music composition.

In the following, we present the three RNN models that we target in our study. We start by the simple Vanilla RNN model, followed by the Long-Short Term Memory (LSTM) model. Finally, we explain the Gated Recurrent Units model (GRU).

A. Vanilla RNN

Vanilla RNN is a simple circular neural network. As illustrated in Fig. 2, Vanilla RNN mainly involves inputs, weights, and outputs. It then builds a connection among sequences of inputs, which is mandatory in many applications such as text prediction and voice recognition. Vanilla RNN uses the *tanh* or *sigmoid* as its activation function.

Theoretically, Vanilla RNNs are supposed to handle long-term dependencies effectively. Unfortunately, as the input sequence grows, Vanilla RNN becomes unable to connect the information efficiently. The gap between the current input and the relevant data becomes huge whenever the sequence expands, causing the *long-term dependency problem*, as shown in Fig. 3. As a result, the learning performance and the model accuracy will be impacted negatively.

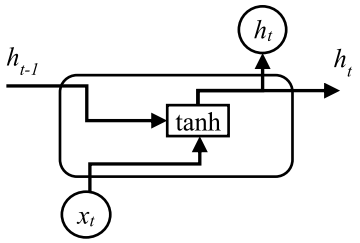

Fig. 2 Vanilla RNN cell architecture

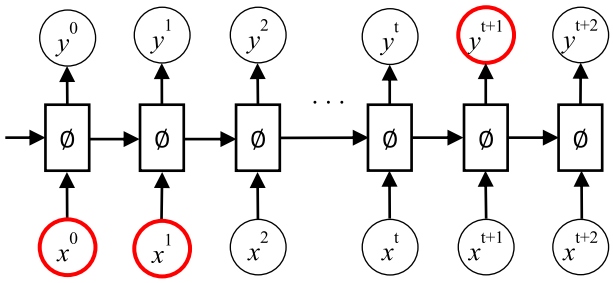

Fig. 3 Long-term dependency problem

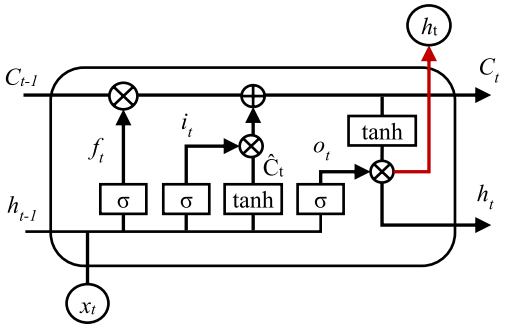

Fig. 4 LSTM cell architecture

B. Long-short term memory (LSTM)

LSTM is a version of RNN with an enhanced function to calculate the hidden state. Hochreiter and Schmidhuber (1997) proposed LSTM to solve the long-term dependency problem by adding the gating mechanism. As illustrated in Fig. 4, the LSTM cell uses the concept of gating to determine what data to keep or erase from memory. Gating is a method to transfer needed information selectively. It is achieved by

constructing a combination of the current inputs, memory, and previous states.

LSTM consists of three gates. The *input* gate, the *forget* gate, and the *output* gate. The *forget* gate is responsible for selectively choosing which data to remember and which to throw away. This decision is made through the *sigmoid* layer as shown in Eq. 2.

$$f_t = \sigma\left(x_t W^f + h_{t-1} U^f\right) \tag{2}$$

The output is 0 or 1, where 0 means *forget* and 1 means *keep*. The second gate is the *input* gate. Another sigmoid layer is used to determine the values to be updated, as shown in Eq. 3.

$$i_t = \sigma\left(x_t W^i + h_{t-1} U^i\right) \tag{3}$$

The *tanh* function creates a vector of candidates that could be added to the state as shown in Eq. 4.

$$\hat{C}_t = \tanh\left(x_t W^g + h_{t-1} U^g\right) \tag{4}$$

Next, LSTM updates the old cell state C_{t-1} to be C_t as shown in Eq. 5.

$$C_t = \sigma\left(f_t \times C_{t-1} + i_t \times \hat{C}_t\right) \tag{5}$$

Finally, in the *output* gate, LSTM uses a *sigmoid* function to determine which part of the cell state will come out as shown in Eq. 6.

$$o_t = \sigma\left(x_t W^o + h_{t-1} U^o\right) \tag{6}$$

In Eq. 7, by multiplying o_t with $\tanh(C_t)$, we implicitly determine which part to take out.

$$h_t = \tanh\left(C_t\right) \times o_t \tag{7}$$

C. Gated recurrent units (GRU)

GRU is an improved version of the standard RNN and a simplified version of LSTM (Gers et al. 2002). Hence, GRU is a variation of the LSTM, and in some cases, they produce equally excellent results. Similar to LSTM, GRU is designed to reset or update its memory adaptively. Hence, GRU has a *reset* gate and an *update* gate, which are identical to the *forget* and the *input* gates in LSTM. However, GRU fully exposes its memory content in each time step and also balances between the previous and the new memory content using leaky integration controlled by the update gate. Figure 5 presents the GRU structure, which is similar to the LSTM structure but with fewer parameters that enable it to capture long-term dependencies more easily.

The *update* gate monitors the amount of memory content that must be forgotten from the previous time step.

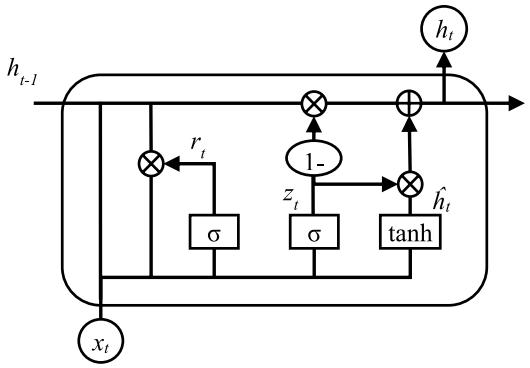

Fig. 5 GRU cell architecture

Moreover, it controls the amount of memory content that must be added from the current time step. Equation 8 captures this behavior.

$$z_t = \sigma\left(W_z \cdot \left[h_{t-1}, x_t\right]\right) \tag{8}$$

The model uses the *reset* gate to decide the amount of the past information to forget as given in Eq. 9.

$$r_t = \sigma\left(W_r \cdot \left[h_{t-1}, x_t\right]\right) \tag{9}$$

New memory content is introduced by using the *reset* gate as calculated in Eq. 9 and relevant past information is stored as shown in Eq. 10.

$$\hat{h}_t = tanh\left(W \cdot \left[r_t \times h_{t-1}, x_t\right]\right) \tag{10}$$

Finally, the network calculates the hidden state h_t, which is a vector that carries information for the current unit and passes it down to the network. Thus, the update gate is essential since it decides what is needed from the current memory content $\hat{h}_t$ and the previous step h_{t-1}. Equation 11 calculates the value of h_t.

$$h_t = \left(1 - z_t\right) \times h_{t-1} + z_t \times \hat{h}_t \tag{11}$$

Consequently**,** GRUs can store and filter the information using their *update* and *reset* gates, making them a preferred choice if trained appropriately. In the next section, we present our approach which utilizes the three RNN models discussed in this section.

4 Approach

In this paper, we study the effect of data augmentation of time series data, collected from smartphone accelerometer sensors, on the performance of several RNN models in the context of HAR. The accelerometer data is a three-dimensional vector where each data point contains a number of

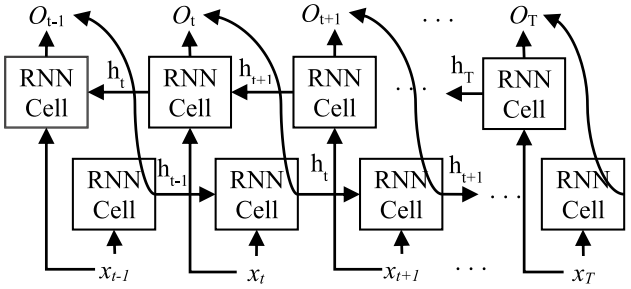

Fig. 6 Bidirectional RNN structure

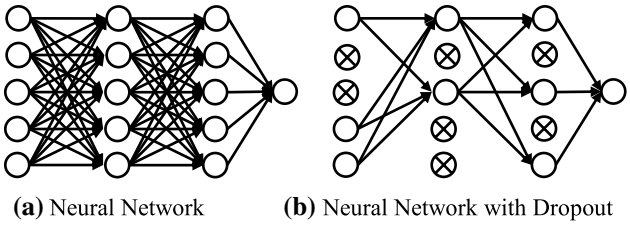

Fig. 7 Dropout in neural networks

recorded samples along each Cartesian direction during a time window. For example, if the sampling rate is 20 Hz (20 samples per second), then a data point encompassing three seconds will contain 180 values (60 for x, 60 for y, and 60 for z dimension). It is preferred to have a three-second window for each sample. This is motivated by the cadence of an average person which is within 90–130 steps per minute and that at least a full cycle (two steps) is preferred on each window sample.

Our approach iteratively trains the Vanilla RNN, LSTM, and GRU models, by tuning their hyper-parameters, using raw data until they reach maximum accuracy. Then, we augment the raw data and retrain the model using the raw and augmented data. Data augmentation increases the size of training data which helps in improving the generalization performance of the model.

In this work, we apply the bidirectional RNN approach. A bidirectional RNN model consists of two separate layers that divide the state neurons of a regular RNN into *forward* (positive time direction) and *backward* (negative time direction) states. The outputs of the *forward* and *backward* layers are concatenated which make it possible to obtain the forward and backward information at each time step in the sequence. Figure 6 depicts the bidirectional RNN structure where the two independent layers share the same input sequence (x_{t-1}–x_T) while the outputs from the two layers are concatenated and represented in the sequence (O_{t-1}–O_T). This approach enhances the learning process due to the dependency found between the neighboring data pairs.

The designs of LSTM and GRU enable the utilization of the information from earlier data points in time series data. Thus, combining the information from the future data, which is useful for prediction, with the past data will improve the learning experience as shown in (Shen et al. 2018).

In our approach, we use the cross-entropy loss (*log* loss) function as a cost function for the model accuracy. The loss function is required to determine the model's accuracy in the classification problem. A smaller loss value means that the deviation from the actual value is less. Thus, the cost function measures the model's ability to find the correlation between the test and labeled data.

Updating the weights and biases in the network is the responsibility of the optimization function. In this work, we use the Adam optimization function, an adaptive version of the stochastic gradient descent, for model training (Kingma and Ba 2014). The Adam optimizer is a reliable optimizer which ensures fast and accurate results when updating the network parameters.

To prevent the overfitting in the model, we apply the widely used stochastic regularization method known as the *dropout* technique (Srivastava et al. 2014). A model that over-fits the data is merely useless. This problem arises when the loss function is only accurate for the training data. This means that the loss function is very small for training data while it is very large for testing data. The main objective of the dropout technique is to prevent the neurons in the network from excessive co-adapting which result in a lack of model generalization. This method works by randomly hiding some of the neurons along with their connections from the network during each training batch. Then, the model will be trained and optimized based on the current structure. The dropout will be iteratively applied to ensure that all the neurons are part of the training process.

The *dropout* provides a smart way for regularization by reducing the network tendency to become over-dependent on some neurons as they may not be available all the time. Figure 7a shows an example of a neural network. On the other hand, Fig. 7b depicts the corresponding network after applying the *dropout* function.

The training process is performed by dividing the dataset into training and testing sets. We allocate the first 80% of the data for training and the remaining 20% for testing (Dobbin and Simon 2011). The training set is further split into two subsets (one for training and one for validation). The validation subset is used for hyper-parameter tuning. The process of selecting the training and testing data is preceded by a random shuffling step to ensure that the training and testing sets contain all the possible classes and cases.

We train the model by tuning a set of well-known hyper-parameters until it reaches high prediction accuracy. The hyper-parameters that we configure for the model are the

Table 1 Moving average smoothing example

	0	1	2	3	4	5	6	7	8	9	10
S	7.61	7.32	7.28	7.75	8.04	8.22	8.39	8.58	8.66	8.67	8.74
S_A	7.68	7.40	7.45	7.69	8.00	8.22	8.40	8.54	8.64	8.69	8.51

number of epochs, batch size, hidden units, learning rate and the dropout percentage. An epoch is defined as feeding the entire dataset to the network only once. It is comprised of a number of iterations to go through the whole dataset. The batch size is the number of training examples in each iteration. The learning rate is used to control how fast the network parameters are being changed to reach the optimal value by minimizing the loss function.

We use the same hyper-parameter values in the training of the three RNN models. We select the parameters values based on the model that provides the best accuracy with the lowest training time. A lower time means that a lower cost is incurred by the training and the HAR process. The model starts by receiving the dataset samples. Then, it randomly initializes the learning parameters. After that, the model computes the cell states based on the input data and the training parameters. Furthermore, the prediction of the model is then computed. The model proceeds to determine the cross-entropy loss. Finally, the model uses the Adam optimizer to adjust the parameters based on the calculated entropy loss. The training process will repeat until the model obtains the maximum accuracy. We measure the accuracy of the model using Eq. 12.

$$\text{accuracy} = \frac{\text{Correct Classifications}}{\text{Total Classifications}} \tag{12}$$

We calculate the training and the testing accuracies when we apply the model to the training and testing data, respectively. Both accuracies are essential to identify the overfitting of the model.

The main contribution of this paper is to study the effect of data augmentation on the accuracy of the HAR process. This is driven by the relatively small size of the available datasets. In the following, we present the two augmentation techniques we use in our study.

A. Data augmentation

Adding more data to the training set is an effective way for enhancing the performance of a deep learning model. Aside from gathering more instances from the raw data that are representative of the classification task, we can generate more data from the one we already have. Data augmentation is used to adjust the existing samples in a dataset by applying a small transformation to the input and use these transformed inputs as additional training data. In this paper, we applied the moving average and the exponential smoothing augmentation techniques.

1. Moving average smoothing

As we are dealing with time series data, we can use the *moving average* method to generate more data with the same class label (Yager 2008). Calculating the moving average for a specific point in time series data requires calculating the average of the observed values that surround that point. Hence, we can produce a new series of values that comprise the average of raw observations in the original time series. This technique is called moving average since the window of a specific width slides along the time series to calculate the average values in the new series. For example, at time t, a moving average (MA) of width 3 (window size is 3) with equal weights would be the average of the observed values at times ($t-1$, t, $t+1$) as shown in Eq. 13.

$$MA = \frac{(v(\mathrm{t}-1) + v(\mathrm{t}) + v(\mathrm{t}+1))}{3} \tag{13}$$

Table 1 shows an example of a sample data series S and the corresponding augmented series S_A. S_A at position 0 is calculated as the average of S at -1, S at 0, and S at 1. S_A at positions 0 and 10 are calculated by padding the original list by the average of its values (padded values are added at positions -1 and 11).

2. Exponential smoothing

Exponential smoothing is a very popular technique for smoothing of time series data where it assigns exponentially decreasing weights for older observations. Exponential smoothing proved to be efficient in forecasting methods (Hyndman 2008). The basic idea is that recent observations provide more relevant information compared to older ones. It revises the forecast based on newly available information. We use the simple exponential smoothing technique modeled using Eq. 14. The current smoothed value S_t is computed based on the previous observation X_{t-1} and the previous smoothed value S_{t-1}. The value of α determines the level of smoothing where a value of 1 means that there is no smoothing and a value of 0 means that the smoothed value depends completely on the previous smoothed observation.

$$S_t = \alpha \times X_{t-1} + (1-\alpha) \times S_{t-1} \tag{14}$$

Table 2 Exponential smoothing example

	0	1	2	3	4	5	6	7	8	9	10
S	7.61	7.32	7.28	7.75	8.04	8.22	8.39	8.58	8.66	8.67	8.74
S_A	8.11	7.66	7.35	7.29	7.70	8.01	8.20	8.37	8.56	8.65	8.67

Table 3 Hyper-parameter tuning–range of values

Parameter	Range of Values
Epochs	1–75
Dropout	0.1, 0.25, 0.35, 0.5, 0.6, 0.7
Batch size	16, 32, 64, 128, 256, 320
Learning rate	0.0001, 0.001, 0.005, 0.01, 0.1
Hidden units	35, 50, 75, 100, 125, 150, 175, 200, 250

Table 2 shows an example of exponential smoothing with 0.9 α value. The value of $S_A{}^0$ is the average of the values in the original sequence S.

By using data augmentation, we can increase the number of records, which should improve the training of the model. In this work, each original sample will be smoothed once. This means that we will have double the number of labeled samples for training when compared to the original dataset.

5 Evaluation

We assessed our approach on the UniMiB SHAR (Micucci et al. 2017), the WISDM (Kwapisz et al. 2011), and the UCI-HAR (Anguita et al. 2013) datasets using Tensorflow (Abadi et al. 2016). We conducted the experiments on an Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60 GHz using 24 vCPUs with 256 GB of RAM.

We performed the hyper-parameter tuning by adjusting the value of one parameter at a time and measuring its effect on accuracy until the maximum accuracy is reached. The tuning experiments were conducted on the three datasets. We started the tuning using the UniMiB SHAR dataset. This process was repeated for each parameter until the accuracy has stopped improving. Table 3 presents the range of values used in the tuning of the models. The experiments showed that our tuning process achieved the best accuracy for the three datasets when setting the values of the *dropout*, *batch size*, *learning rate*, and *hidden units* at 0.5, 64, 0.001, and 200, respectively.

In the following, we present the results of our time series data augmentation approach on each dataset using the three RNN models supported by confidence intervals. Furthermore, we compare the accuracy of our data augmentation approach with the state-of-the-art solutions.

A. UniMiB SHAR dataset

The UniMiB SHAR (Micucci et al. 2017) dataset was collected using Samsung Galaxy Nexus I9250 phones running Android OS version 5.1.1. The phone is equipped with the Bosh BMA220, a triaxial low-g accelerometer, which measures the acceleration in three dimensions.

The dataset consists of 11,771 records of daily life activities and fall states collected by 30 persons. Each record spans a 3-s non-overlapping window where each Cartesian dimension consists of 151 values (i.e. a record vector size is 453 with a sampling rate of 50 Hz). This acceleration vector is the main feature used for the model training. The UniMiB SHAR dataset contains 17 distinct classes of different daily living activities (ADL) and fall states. The ADL classes are Standing Up from Sitting, Standing Up from Lying, Walking, Running, Going Upstairs, Jumping, Going Downstairs, Lying Down from Standing and Sitting Down. Fall states include Falling Forward, Falling Right, Falling Back, Hitting Obstacle, Falling with Protection Strategies, Falling Backward-Sitting-Chair, Syncope and Falling Left.

The ADL category consists of 7,579 records of nine different classes while the fall states category consists of 4,192 records of eight fall classes. In our experiments, we implemented four different classification tasks. The first task, AF-2, is a simple task to measure the model ability to distinguish between daily activities and fall states. Thus, all ADL types will be considered as one class and all fall types will be considered as one class as well. The number of test records used for this task is 2,354 which were not included in the model training. The second task, F-8, targets the classification of the different types of fall states. Thus, the model will be trained on the data samples that correspond to the fall states only. The model was tested using 838 records. The third classification task is A-9. The A-9 task is concerned with the model ability to distinguish between the different types of daily activities. We used 1516 test records for this task. Similarly, the model will be trained using the 7579 ADL states only. The fourth task, AF-17, is the most important task since it enables the model to distinguish the exact type of activity or fall state. This task consists of all the 17 classes in the dataset and uses 2354 records for testing.

Table 4 presents the accuracy and the execution time for each task obtained from the first training attempt for the three RNN models using the assigned hyper-parameter values. The GRU and LSTM models achieved high accuracy values for the AF-2 task. The accuracy for the Vanilla version reached only 69.8%. However, GRU reached only 70.1% accuracy for F-8. On the other hand, Vanilla reached

Table 4 UniMiB SHAR–initial results

Parameter	AF-2	F-8	A-9	AF-17
Epochs	50	30	32	30
Iterations	45	100	94	128
Batch size	200	32	64	74
Hidden units	35	100	100	50
Model	Accuracy (%)			
Vanilla	69.8	20.3	25.2	12.9
LSTM	95.6	39.7	79.3	40.3
GRU	98.8	70.1	85.6	50.9
	Execution time (minutes)			
Vanilla	30	15	20	31
LSTM	37	24	29	41
GRU	36	22	28	38

Learning rate: 0.001, dropout: 0.5

Table 5 UniMiB SHAR–enhanced results

Parameter	AF-2	F-8	A-9	AF-17
Epochs	50	50	50	50
Iterations	148	52	94	148
Model	Accuracy ± Confidence interval (%)			
Vanilla	79.8 ± 1.6	23.3 ± 2.9	26.6 ± 2.2	18.6 ± 1.6
LSTM	98.1 ± 0.6	47.0 ± 3.4	85.1 ± 1.8	80.4 ± 1.6
GRU	98.9 ± 0.4	75.9 ± 2.9	96.0 ± 1.0	93.9 ± 1.0
	Execution time (minutes)			
Vanilla	40	20	30	45
LSTM	170	44	70	179
GRU	169	41	68	172

Batch size: 64, learning rate: 0.001, hidden units: 200, dropout: 0.5

only 20.3%. The high similarity between the different types of fall states leads to this confusion, thus resulting in misclassification issues. For example, the *Syncope* fall state resembles *Falling leftward*, *Generic falling backward*, and *Falling backward-sitting-chair*. The accuracy of the GRU and LSTM models for the A-9 task reached 85.6% and 79.3% respectively. The Vanilla model achieved only 25.2% accuracy for A-9 which means that it is inferior for this kind of problems. Nevertheless, the accuracy for the AF-17 task reached only 50.9% in case of GRU but still provided higher accuracy than the other models. Clearly, the GRU model is still prevalent in terms of accuracy. The accuracy of the LSTM is relatively good, and its training cost was relatively similar to that of GRU.

We continued the tuning of the hyper-parameters to improve the recognition accuracy. Table 5 presents the accuracy of the three models with enhanced parameters along with the confidence intervals for statistical significance validation. We calculate the binomial confidence interval at 95% confidence level. The results show that the tuned parameters improve the accuracy of the three models on all tasks. However, the vanilla RNN model still performs poorly. In the case of GRU, the accuracy for the AF-17 task improved significantly and reached 93.9% using the enhanced parameters. However, the accuracy for F-8 task is still low compared to the other tasks. This is due to the high similarity among the different fall classes. The average prediction time for each data sample (record) is 0.38 s. This shows that the generated model has a feasible response time. The testing of the model spent 41 s using the 2,354 records for the AF-17 task.

Table 6 UniMiB SHAR–moving average smoothing

Model	AF-2	F-8	A-9	AF-17
Epochs	50	50	50	50
Iterations	294	104	189	294
	Accuracy ± Confidence interval (%)			
Vanilla	81.0 ± 1.6	20.4 ± 2.7	29.0 ± 2.3	17.2 ± 1.5
LSTM	98.9 ± 0.4	68.0 ± 3.2	94.2 ± 1.2	90.1 ± 1.2
GRU	99.0 ± 0.4	89.0 ± 2.1	99.0 ± 0.5	97.9 ± 0.6
	Execution time (minutes)			
Vanilla	87	47	61	100
LSTM	270	98	140	281
GRU	268	98	139	280

Batch size: 64, learning rate: 0.001, hidden units: 200, dropout: 0.5

Using data augmentation techniques to increase the size of the training data showed significant improvements in the accuracy of the three RNN models especially for the F-8 task. Table 6 shows the training accuracies and times for the moving average smoothing technique along with the confidence intervals in order to show the significance of the data augmentation techniques. Nevertheless, the number of iterations increased due to the larger dataset generated by augmentation.

Table 7 shows the results for the exponential smoothing approach. It is clear that using this approach had a significant improvement on the accuracy of F-8 as it reached 95.8% for GRU. The results show that the GRU provides the best performance in terms of accuracy and training time. Thus, we were able to improve the accuracy of the models using data augmentation for the UniMiB SHAR dataset. Moreover, the confidence intervals show that the exponential smoothing technique significantly improves the model quality when compared to the accuracies in Table 5.

We applied the Z-test (Johnson et al. 2000) to further validate the significance of the proposed augmentation approach. Table 8 presents the *Z*-Score and the *p*-value measures on the AF-17 task for the LSTM and GRU models (raw data model vs. augmented data model). The *p*-value is very small which means that the two data augmentation

Table 7 UniMiB SHAR–exponential smoothing

Model	AF-2	F-8	A-9	AF-17
Epochs	50	50	50	50
Iterations	294	104	189	294
	Accuracy ± Confidence interval (%)			
Vanilla	82.3 ± 1.5	19.3 ± 2.7	26.5 ± 2.2	20.4 ± 1.6
LSTM	98.8 ± 0.4	76.0 ± 2.9	96.2 ± 1.0	93.8 ± 1.0
GRU	99.3 ± 0.3	95.8 ± 1.4	97.6 ± 0.8	98.1 ± 0.6
	Execution time (minutes)			
Vanilla	99	70	80	110
LSTM	276	130	184	284
GRU	281	140	197	290

Batch size: 64, learning rate: 0.001, hidden units: 200, dropout: 0.5

Table 8 P-value (raw vs. augmented data models)

Model	Z-Score		P-VALUE	
	MAS	ES	MAS	ES
LSTM	− 9.387	− 13.718	< 0.00001	< 0.00001
GRU	− 6.921	− 7.340	< 0.00001	< 0.00001

Table 9 UniMiB SHAR–related studies

Study	Method	Accuracy (%)
Li et al. (2018)	Standard CNN	74.97
Gao et al. (2020)	Att-based Residual Network	79.03
Teng et al. (2020)	Layer-wise CNN with local loss	78.07
Micucci et al. (2017)	KNN	82.86
Ferrari et al. (2020)	AdaBoost-HC	85.44
Mukherjee et al. (2020)	CNN-LSTM	91.6
	CNN-Net	92.1
	EnsemConvNet	92.6

approaches (MAS and ES) significantly improved the accuracy of the LSTM and GRU models.

Table 9 presents the related studies that targeted the UniMiB SHAR dataset for the AF-17 task along with the technique used and the corresponding recognition accuracy. Using the exponential smoothing data augmentation technique significantly improves the accuracy of the GRU model and achieves better accuracy than the state-of-the-art approaches.

B. WISDM dataset

The activity prediction dataset v1.1 (Kwapisz et al. 2011) was collected from 36 persons under controlled laboratory conditions in 2010 by the Wireless Sensor Data Mining (WISDM) Lab. The dataset consists of 1,098,207 labeled samples of six classes which are *walking* (38.6%), *jogging* (31.2%), *upstairs* (11.2%), *downstairs* (9.1%), *sitting* (5.5%), and *standing* (4.4%). The activities were recorded at 20 Hz sampling rate (one sample every 50 ms). Each record contains 60 non-overlapping samples in the 3D format from the same subject. A total of 1,092,600 samples have been used in our experiments from the original dataset resulting in a total of 18,210 records. We used 2196 records for the testing of the models.

Table 10 WISDM–initial results

Parameter	Original data	Moving average smoothing	Exponential smoothing
Epochs	33	33	33
Iterations	137	274	274
Model	Accuracy (%)		
Vanilla	43	43.2	43.3
LSTM	59.58	59.98	66.4
GRU	77.0	73	83.4
	Execution time (minutes)		
Vanilla	32	59	60
LSTM	72	156	171
GRU	80	153	149

Batch size: 64, learning rate: 0.001, hidden units: 50, dropout: 0.5

Table 10 presents the initial results of the three models on the WISDM dataset and the augmented versions using the moving average smoothing and the exponential smoothing techniques. The Vanilla RNN did not show any improvement after applying augmentation on the dataset. The LSTM accuracy improved only using the exponential smoothing technique. Furthermore, the GRU performed better than the LSTM and the Vanilla RNN on the original dataset. The GRU accuracy improved in case of the exponential smoothing technique. However, its accuracy degraded when using the moving average smoothing technique.

Table 11 shows the training results of the three models using the WISDM dataset with enhanced parameters along with the confidence intervals. Similar to the UniMiB SHAR dataset, the GRU model achieved the best results based on the specified hyper-parameters when applied on the original non-augmented data. Additionally, using exponential smoothing had a significant improvement on the accuracy of the three models especially when applying the exponential smoothing technique where the accuracy reached 97.13%. It should be noted that the accuracy of the Vanilla RNN and GRU models degraded when using the moving average smoothing approach even after using the enhanced parameters. The average prediction time for one data sample is 0.41 s. The total testing of the model took 40.5 s to complete.

Table 11 WISDM–enhanced results

Parameter	Original data	Moving average smoothing	Exponential Smoothing
Epochs	35	35	35
Iterations	137	274	274
Model	Accuracy ± Confidence Interval (%)		
Vanilla	44.3 ± 2.1	39.4 ± 2.0	46.75 ± 2.1
LSTM	72.9 ± 1.9	73.2 ± 1.9	88.75 ± 1.3
GRU	87.8 ± 1.4	83.22 ± 1.6	97.13 ± 0.7
	Execution time (minutes)		
Vanilla	40	81	90
LSTM	85	171	190
GRU	90	170	157

Batch size: 64, learning rate: 0.001, hidden units: 200, dropout: 0.5

Table 12 P-value (raw vs. augmented data models)

Model	Z-score		P-value	
	MAS	ES	MAS	ES
LSTM	− 0.224	− 13.340	0.411	< 0.00001
GRU	4.313	− 11.711	< 0.00001	< 0.00001

Table 13 WISDM–related studies

Study	Method	Accuracy (%)
Kwapisz et al. (2011)	Logical Regression	78.1
Kwapisz et al. (2011)	J48	85.1
Chen et al. (2016)	LSTM	92.1
Li and Trocan (2019)	CNN	95.75
Sukor et al. (2018)	PCA	92.1
Zainudin et al. (2015)	J48 + MLP + LR	93.0

Table 12 presents the *Z*-Score and the *p*-value measures for the LSTM and GRU models. The *p*-value for the MAS in case of LSTM was greater than 0.05 significance level which means that the MAS approach did not significantly improve the accuracy of the model. For the other tests, the *p*-value is very small which means that the data augmentation approach significantly improved the accuracy of the LSTM and GRU models.

Table 13 presents the related studies that targeted the WISDM dataset along with the techniques used and the corresponding recognition accuracies. Our approach outperforms the state-of-the-art studies and reaches 97.13% accuracy when using the exponential smoothing technique.

Table 14 UCI-HAR–initial results

Parameter	original data	Moving average smoothing	Exponential smoothing
Epochs	36	36	36
Iterations	132	264	264
Model	Accuracy (%)		
Vanilla	41.0	40.6	42
LSTM	69.3	69.5	72
GRU	76.6	76	80
	Execution time (minutes)		
Vanilla	42	76	80
LSTM	88	190	193
GRU	82	167	169

Batch size: 128, learning rate: 0.001, hidden units: 50, dropout: 0.5

C. UCI-HAR dataset

The UCI-HAR dataset (Anguita et al. 2013) consists of 10,299 daily living activity instances recorded from 30 persons (19–48 years old) carrying waist-mounted smartphones with embedded inertial sensors. The sampling rate of the recordings is 50 Hz. Each data sample or instance consists of a 2.56 s window with a 50% overlap (128 readings). The dataset is separated where 70% of the data (7352 instances) is used for training and 30% of the data (2947 instances) is used for testing. The dataset includes the Walking, Walking Upstairs, Walking Downstairs, Sitting, Standing and Laying ADL activities.

Table 14 presents the initial results of the three RNN models on the UCI-HAR dataset. The GRU model provides

Table 15 UCI-HAR–enhanced results

Parameter	Original data	Moving average smoothing	Exponential Smoothing
Epochs	50	50	50
Iterations	132	264	264
Model	Accuracy ± Confidence interval (%)		
Vanilla	42 ± 1.8	42.3 ± 1.8	42.6 ± 1.8
LSTM	85.3 ± 1.3	86 ± 1.3	86.9 ± 1.2
GRU	92.4 ± 1.0	93.5 ± 0.9	97.9 ± 0.5
	Execution time (minutes)		
Vanilla	45	70	66
LSTM	90	160	156
GRU	88	207	210

Batch size: 64, learning rate: 0.001, hidden units: 200, dropout: 0.5

Table 16 P-value (raw vs. Augmented data models)

Model	Z-score		P-value	
	MAS	ES	MAS	ES
LSTM	− 0.766	− 1.776	0.222	0.0379
GRU	− 1.649	− 9.826	0.0496	< 0.00001

Table 17 UCI-HAR- Related Studies

Study	Method	Accuracy (%)
Anguita et al. (2013)	SVM	96.37
Ronao and Cho (2016)	FFT + Convnet	95.75
Jiang and Yin (2015)	DCNN + SVM	97.59
Xia et al. (2020)	LSTM + CNN	95.78
Cho and Yoon (2018)	CNN + Data Sharpening	97.62

better accuracy when compared to the other two models. Moreover, the augmentation techniques provided a slight improvement on the accuracy of the three RNN models. The GRU showed the best improvement using the exponential smoothing technique where the accuracy reached 80%.

Table 15 shows the results of the three models using the UCI-HAR dataset with the enhanced parameters along with the confidence intervals. In case of moving average smoothing, the LSTM and GRU models showed slight but insignificant improvements. On the other hand, using the exponential smoothing techniques, the accuracy of the LSTM and GRU models significantly improved and reached 86.9% and 97.9% respectively. The Vanilla RNN model did not show any considerable improvement even after applying data augmentation. The average prediction time for one sample was 0.39 s. The time for testing the model was 52.3 s.

Table 16 presents the *Z*-Score and the *p*-value measures for the accuracies of LSTM and GRU models, with and without augmentation. The *p*-value for the MAS in case of LSTM was greater than 0.05 significance level which means that the MAS did not improve the accuracy of the model significantly. Moreover, for GRU, the *p*-value in case of MAS is very close to 0.05. This means that for stricter confidence levels such as 0.04, the alternative hypothesis, which states that the augmentation approach provides significant improvement to the accuracy, will be rejected. Thus, the *null* hypothesis will be selected, which means that the two models provide similar accuracy levels. For the other tests, the *p*-value is smaller than 0.05 which means that the ES augmentation approach significantly improved the accuracy of the LSTM and GRU models.

Table 17 shows a list of related studies for the UCI-HAR datasets. Our exponential smoothing approach with GRU provided similar results compared to the state-of-the-art approaches. For example, Cho and Yoon (2018) used a divide-and-conquer approach for the classification of human activities where they first use a binary classifier and then apply multi-class classifiers on the initial two classes. Furthermore, they apply sharpening on the testing data to further enhance the prediction accuracy. The sharpening is performed on the test data using a Gaussian filter to remove minor features by the attenuation of high frequency signals. In our approach, we applied the augmentation technique during the training process to the raw data to increase the size of training data.

6 Conclusion

In this paper, we investigated the effectiveness of time series data augmentation on the accuracy of the Vanilla RNN, LSTM, and GRU models for human activity recognition. Using data augmentation minimizes the overfitting problem by increasing the size of training data. We used the UniMiB

SHAR, the WISDM, and the UCI-HAR open-source datasets for training and evaluation. The results showed that the LSTM and the GRU models benefited from applying data augmentation techniques while the Vanilla RNN model did not show any considerable enhancements. The results showed that using exponential smoothing data augmentation is promising and improves the accuracy significantly.

As part of our future directions, we intend to apply the selected RNN models on more datasets to leverage the generalization of the approach. We also aim to explore and compare the effectiveness of other data augmentation techniques such as window warping and window slicing. Moreover, we should look for a reliable automatic parameter tuning technique which could further enhance the accuracy of the models. We intend to investigate the efficiency of the Grid Search, the Random Search, the Bayesian Optimization, Tree-structured Parzen Estimators (TPE), and the meta-learning approach. Another direction could be achieved by considering data from other types of sensors such as gyroscopes.

Finally, we intend to apply the models on an Edge-Cloud environment and study different machine learning approaches such as federated learning, semi-supervised learning, and semi-supervised incremental learning.

References

Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Kudlur M (2016) Tensorflow: A system for large-scale machine learning. In: 12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16), pp 265–283

Alsarhan T, Alawneh L, Al-Zinati M, Al-Ayyoub M (2019) Bidirectional gated recurrent units for human activity recognition using accelerometer data. In: 2019 IEEE SENSORS, IEEE, pp 1–4

Anguita D, Ghio A, Oneto L, Parra X, Reyes-Ortiz JL (2013) A public domain dataset for human activity recognition using smartphones. In: Esann, p 3

Behera A, Hogg DC, Cohn AG (2012) Egocentric activity monitoring and recovery. Asian conference on computer vision. Springer, Berlin, pp 519–532

Bidargaddi N, Sarela A, Klingbeil L, Karunanithi M (2007) Detecting walking activity in cardiac rehabilitation by using accelerometer. In: 2007 3rd International Conference on Intelligent Sensors, Sensor Networks and Information. IEEE, pp 555–560

Che Z, Purushotham S, Cho K, Sontag D, Liu Y (2018) Recurrent neural networks for multivariate time series with missing values. Sci Rep 8(1):1–12

Chen Y, Zhong K, Zhang J, Sun Q, Zhao X (2016) Lstm networks for mobile human activity recognition. 2016 International conference on artificial intelligence: technologies and applications. Atlantis Press, Paris

Chen Z, Zhu Q, Soh YC, Zhang L (2017) Robust human activity recognition using smartphone sensors via CT-PCA and online SVM. IEEE Trans Industr Inf 13(6):3070–3080

Cho H, Yoon SM (2018) Divide and conquer-based 1D CNN human activity recognition using test data sharpening. Sensors 18(4):1055

Cho K, Van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y (2014) Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: Conference on empirical methods in natural language processing (EMNLP 2014), Doha, Qatar, pp 1724–1734

Chung J, Gulcehre C, Cho K, Bengio Y (2014) Empirical evaluation of gated recurrent neural networks on sequence modeling. In: NIPS 2014 workshop on deep learning

Deng J, Guo J, Xue N, Zafeiriou S (2019) Arcface: additive angular margin loss for deep face recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 4690–4699

Dobbin KK, Simon RM (2011) Optimally splitting cases for training and testing high dimensional classifiers. BMC Med Genom 4(1):31

Ferrari A, Micucci D, Mobilio M, Napoletano P (2020) On the personalization of classification models for human activity recognition. IEEE Access 8:32066–32079

Foerster F, Smeja M, Fahrenberg J (1999) Detection of posture and motion by accelerometry: a validation study in ambulatory monitoring. Comput Hum Behav 15(5):571–583

Gao W, Zhang L, Teng Q, Wu H, Min F, He J (2020) DanHAR: dual attention network for multimodal human activity recognition using wearable sensors.

Gers FA, Schraudolph NN, Schmidhuber J (2002) Learning precise timing with LSTM recurrent networks. J Mach Learn Res 3:115–143

Graves A, Liwicki M, Fernández S, Bertolami R, Bunke H, Schmidhuber J (2008) A novel connectionist system for unconstrained handwriting recognition. IEEE Trans Pattern Anal Mach Intell 31(5):855–868

Gutchess D, Checka N, Snorrason MS (2007) Learning patterns of human activity for anomaly detection. Intelligent computing: theory and applications. International Society for Optics and Photonics, Washington, p 65600Y

Hammerla NY, Halloran S, Plötz T (2016) Deep, convolutional, and recurrent models for human activity recognition using wearables. In: Proceedings of the twenty-fifth international joint conference on artificial intelligence, pp 1533–1540

Hassan MM, Uddin MZ, Mohamed A, Almogren A (2018) A robust human activity recognition system using smartphone sensors and deep learning. Future Gener Comput Syst 81:307–313

Hinton GE, Osindero S, Teh YW (2006) A fast learning algorithm for deep belief nets. Neural Comput 18(7):1527–1554

Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780

Husken M, Stagge P (2003) Recurrent neural networks for time series classification. Neurocomputing 50:223–235

Hyndman R, Koehler AB, Ord JK, Snyder RD (2008) Forecasting with exponential smoothing: the state space approach. Springer Science and Business Media, Cham

Jalal A, Kamal S, Kim D (2014) A depth video sensor-based life-logging human activity recognition system for elderly care in smart indoor environments. Sensors 14(7):11735–11759

Jiang W, Yin Z (2015) Human activity recognition using wearable sensors by deep convolutional neural networks. In: Proceedings of the 23rd ACM international conference on Multimedia, pp 1307–1310

Johnson RA, Miller I, Freund JE (2000) Probability and statistics for engineers. Pearson Education, London, p 642

Jordao A, Kloss R, Schwartz WR (2018) Latent HyperNet: exploring the layers of convolutional neural networks. In: 2018 International Joint Conference on Neural Networks (IJCNN), IEEE, pp 1–7

Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv:1412.6980

Kolekar MH, Dash DP (2016) Hidden markov model based human activity recognition using shape and optical flow based features. In: 2016 IEEE Region 10 Conference (TENCON), IEEE, pp 393–397

Kwapisz JR, Weiss GM, Moore SA (2011) Activity recognition using cell phone accelerometers. ACM SIGKDD Explor Newsl 12(2):74–82

Li H, Trocan M (2019) Deep learning of smartphone sensor data for personal health assistance. Microelectron J 88:164–172

Li F, Shirahama K, Nisar MA, Köping L, Grzegorzek M (2018) Comparison of feature learning methods for human activity recognition using wearable sensors. Sensors 18(2):679

Liciotti D, Bernardini M, Romeo L, Frontoni E (2020) A sequential deep learning application for recognising human activities in smart homes. Neurocomputing 396:501–513

Lv T, Wang X, Jin L, Xiao Y, Song M (2020) Margin-based deep learning networks for human activity recognition. Sensors 20(7):1871

Mehdiyev N, Lahann J, Emrich A, Enke D, Fettke P, Loos P (2017) Time series classification using deep learning for process planning: a case from the process industry. Proced Comput Sci 114:242–249

Micucci D, Mobilio M, Napoletano P (2017) Unimib shar: A dataset for human activity recognition using acceleration data from smartphones. Appl Sci 7(10):1101

Mozer MC (1998) The neural network house: an environment hat adapts to its inhabitants. In: Proceedings of AAAI Spring Symposium of Intelligent Environments.

Mukherjee D, Mondal R, Singh PK, Sarkar R, Bhattacharjee D (2020) EnsemConvNet: a deep learning approach for human activity recognition using smartphone sensors for healthcare applications. Multimed Tools Appl 79(41):31663–31690

Murad A, Pyun JY (2017) Deep recurrent neural networks for human activity recognition. Sensors 17(11):2556

Nweke HF, Teh YW, Al-Garadi MA, Alo UR (2018) Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: state of the art and research challenges. Expert Syst Appl 105:233–261

Oniga S, Sütő J (2014) Human activity recognition using neural networks. In: Proceedings of the 2014 15th International Carpathian Control Conference (ICCC), IEEE, pp 403–406

Osmani V, Balasubramaniam S, Botvich D (2008) Human activity recognition in pervasive health-care: supporting efficient remote collaboration. J Netw Comput Appl 31(4):628–655

Paul P, George T (2015) An effective approach for human activity recognition on smartphone. In: 2015 IEEE International Conference on Engineering and Technology (ICETECH), IEEE, pp 1–3

Plötz T, Hammerla NY, Olivier P (2011) Feature learning for activity recognition in ubiquitous computing. In: 22nd international joint conference on artificial intelligence, IJCAI 2011, pp 1729–1734

Powell HC, Hanson MA, Lach J (2007) A wearable inertial sensing technology for clinical assessment of tremor. In: 2007 IEEE Biomedical Circuits and Systems Conference, IEEE, pp 9–12

Ramasamy SR, Roy N (2018) Recent trends in machine learning for human activity recognition—a survey. Wiley Interdiscip Rev 8(4):e1254

Ravi D, Wong C, Lo B, Yang GZ (2016) A deep learning approach to on-node sensor data analytics for mobile or wearable devices. IEEE J Biomed Health Inform 21(1):56–64

Ravuri S, Stolcke A (2015) Recurrent neural network and LSTM models for lexical utterance classification. In: Sixteenth Annual Conference of the International Speech Communication Association.

Ronao CA, Cho SB (2016) Human activity recognition with smartphone sensors using deep learning neural networks. Expert Syst Appl 59:235–244

Sabatini AM, Martelloni C, Scapellato S, Cavallo F (2005) Assessment of walking features from foot inertial sensing. IEEE Trans Biomed Eng 52(3):486–494

Schuster M, Paliwal KK (1997) Bidirectional recurrent neural networks. IEEE Trans Signal Process 45(11):2673–2681

Sefen B, Baumbach S, Dengel A, Abdennadher S (2016) Human activity recognition. In: Proceedings of the 8th International Conference on Agents and Artificial Intelligence. SCITEPRESS-Science and Technology Publications, Lda, pp 488–493

Shen G, Tan Q, Zhang H, Zeng P, Xu J (2018) Deep learning with gated recurrent unit networks for financial sequence predictions. Proced Comput Sci 131:895–903

Simonyan K, Zisserman A (2014) Two-stream convolutional networks for action recognition in videos. Advances in neural information processing systems. Springer, Cham, pp 568–576

Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958

Sukor AA, Zakaria A, Rahim NA (2018) Activity recognition using accelerometer sensor and machine learning classifiers. In: 2018 IEEE 14th International Colloquium on Signal Processing and its Applications (CSPA), IEEE, pp 233–238

Teng Q, Wang K, Zhang L, He J (2020) The layer-wise training convolutional neural networks using local loss for sensor-based human activity recognition. IEEE Sens J 20(13):7265–7274

Torres-Huitzil C, Alvarez-Landero A (2015) Accelerometer-based human activity recognition in smartphones for healthcare services. Mobile health. Springer, Cham, pp 147–169

Uddin MZ, Hassan MM, Almogren A, Zuair M, Fortino G, Torresen J (2017) A facial expression recognition system using robust face features from depth videos and deep learning. Comput Electr Eng 63:114–125

Uddin MZ, Hassan MM, Alsanad A, Savaglio C (2020) A body sensor data fusion and deep recurrent neural network-based behavior recognition approach for robust healthcare. Inform Fus 55:105–115

Ullah M, Ullah H, Khan SD, Cheikh FA (2019) Stacked Lstm network for human activity recognition using smartphone data. In: 2019 8th European Workshop on Visual Information Processing (EUVIP), IEEE, pp 175–180

Veeriah V, Zhuang N, Qi GJ (2015) Differential recurrent neural networks for action recognition. In: Proceedings of the IEEE international conference on computer vision, pp 4041–4049

Vepakomma P, De D, Das SK, Bhansali S (2015) A-Wristocracy: Deep learning on wrist-worn sensing for recognition of user complex activities. In: 2015 IEEE 12th International conference on wearable and implantable body sensor networks (BSN), IEEE, pp 1–6

Vu TH, Dang A, Dung L, Wang JC (2017) Self-gated recurrent neural networks for human activity recognition on wearable devices. In: Proceedings of the on Thematic Workshops of ACM Multimedia 2017, pp 179–185

Wang J, Chen Y, Hao S, Peng X, Hu L (2019) Deep learning for sensor-based activity recognition: a survey. Pattern Recogn Lett 119:3–11

Woodruff RB, Gardial SF (1996) Know your customer: new approaches to customer value and satisfaction blackwell. Cambridge University, Cambridge

Woznowski P, King R, Harwin W, Craddock I (2016) A human activity recognition framework for healthcare applications: ontology, labelling strategies, and best practice. In: IoTBD, pp 369–377

Wu GE, Xue S (2008) Portable preimpact fall detector with inertial sensors. IEEE Trans Neural Syst Rehabil Eng 16(2):178–183

Xia K, Huang J, Wang H (2020) LSTM-CNN architecture for human activity recognition. IEEE Access 8:56855–56866

Yager RR (2008) Time series smoothing and OWA aggregation. IEEE Trans Fuzzy Syst 16(4):994–1007

Zahin A, Hu RQ (2019) Sensor-based human activity recognition for smart healthcare: a semi-supervised machine learning. International conference on artificial intelligence for communications and networks. Springer, Cham, pp 450–472

Zainudin MS, Sulaiman MN, Mustapha N, Perumal T (2015) Activity recognition based on accelerometer sensor using combinational classifiers. In: 2015 IEEE Conference on Open Systems (Icos), IEEE, pp 68–73

Zeng M, Nguyen LT, Yu B, Mengshoel OJ, Zhu J, Wu P, Zhang J (2014) Convolutional neural networks for human activity recognition using mobile sensors. In: 6th International Conference on Mobile Computing, Applications and Services, IEEE, pp 197–205