A Comparison of Unidirectional and Bidirectional LSTM Networks for Human Activity Recognition

Luay Alawneh, Belal Mohsen, Mohammad Al-Zinati, Ahmed Shatnawi, Mahmoud Al-Ayyoub
Jordan University of Science & Technology
{lmalawneh, mhzinati, ahmedshatnawi , maalshbool}@just.edu.jo, blmohsen14@cit.just.edu.jo

***Abstract*—Human activity recognition targets identifying different classes of human movements using data gathered from various types of sensors. Deep learning approaches, such as Recurrent Neural Networks, are gaining interest in the classification of human activities using time series data. Long-Short Term Memory is a recurrent neural network approach that is well suited for the classification of time series data where it handles the vanishing gradient and the long-term dependency problems efficiently. In this paper, we compare the human activity recognition accuracy of the unidirectional and bidirectional Long-Short Term Memory models on two different datasets that represent accelerometer data. The results show that the bidirectional approach slightly enhances the recognition quality over the unidirectional approach. However, the bidirectional approach spends more time during the training, which may hinder its applicability on large datasets.**

Keywords—Accelerometers, Deep Learning, Recurrent Neural Networks (RNN), Classification

I. Introduction

Human activity recognition (HAR) aims to identify the types of human actions and movements. The accuracy of HAR is a major concern, and if applied properly, will have a direct impact on people's welfare [1][2][3]. HAR uses modern artificial intelligence techniques on data collected from different types of sensors. The wide use of smartphones and their support for different types of sensors, such as accelerometers, make them suitable and practical for the collection of human activity data.

HAR is a classification problem where it targets the recognition of specific movements and actions such as walking, running, and falling. Recently, HAR has been widely studied and gained more attention for its applicability in different domains such as security, transport, traffic management, and healthcare [4]. Classical classification approaches have been used for human activity recognition [5]. The problem with these approaches is their reliance on heuristic handcrafted feature extraction techniques that affect their generalization.

Deep learning is a branch of machine learning that is based on artificial neural networks [6]. It can work on supervised, semi-supervised, or unsupervised problems. Several deep learning approaches, such as recurrent neural networks (RNN), have been applied to different fields such as computer vision, speech recognition, natural language processing, bioinformatics, and medical image analysis.

Deep learning approaches proved to provide similar, and in some cases, superior results to existing AI approaches. They overcome the constraints of classical classification approaches as they succeed in dealing with time-sequential data that embody correlations between close data points in a sequence [7]. However, using RNN for time series classification can be challenging as performance is highly dependent on the availability of trained data.

Recent studies proposed to use Long-Short Term Memory (LSTM) [8][9], an RNN approach, which is useful for the classification of time series data. LSTM is a modified version of RNN, which improves the handling of long-term dependency problems, as it is easier to remember past data in memory. Moreover, LSTM resolves the vanishing gradient problem found in the original RNN approach [10][11]. It is capable of handling complex serial information with long dependencies since it utilizes a gating scheme for data representation.

The purpose of this work is to study the accuracy and performance of the unidirectional and the bidirectional LSTM models for HAR on time series data gathered from mobile accelerometer sensors. We run these two different LSTM models for the classification of several daily life activities and fall states on two different datasets with different classes of movements. The first dataset is the UniMiB SHAR dataset [12] that contains 17 different classes of activities and fall states. The second dataset is provided by the Wireless Sensor Data Mining (WISDM) Lab [13], which contains six classes of human motions collected under controlled laboratory conditions.

The paper is organized as follows. In Section II, we discuss some recent studies that used different machine learning approaches for human activity recognition and time series analysis. In Section III, we explain the LSTM architecture. Section IV presents our approach followed by an evaluation in Section V. Finally, we conclude our paper in Section VI with a highlight on future directions.

II. Related Work

HAR utilizes feature extraction methods using supervised classification techniques [14][15]. An early study proposed by Foerster et al. [16] demonstrated the usefulness of accelerometer data for human activity recognition. Ravi et al. [17] presented a deep learning approach that combines extracted features learned from sensor data with complementary information from a range of shallow features, which led to accurate and real-time activity classifications. Husken et al. [18] used dynamic RNN for time series. The results showed that the inclusion of a prediction task during learning strongly supports the learning process. Moreover, the generalization ability was significantly improved.

Hassan et al. [19] proposed to use Deep belief networks for HAR on data gathered from inertial sensors in smartphones. Their approach outperformed traditional machine learning approaches such as SVM and ANN. Jiang et al. [20] used Deep Convolutional Neural Networks for HAR, where they combined sequences of accelerometers and gyroscopes into a novel activity image. Their model automatically extracts and learns the optimal features from the activity image for the recognition task.

Mehdiyev et al. [21] proposed a multi-stage deep learning approach for multivariate time series classification. The authors used the stack LSTM Autoencoder Network model that extracts features in an unsupervised or self-supervised manner. Then, the forward neural network performs the classification on the time series data from the stack LSTM Autoencoder network. The neural network consists of three hidden layers, where each layer has 200 neurons. The training of the network took almost 100 epochs, which included changing some parameters such as activation functions, number of layers and neurons, and the type of optimization function to improve the accuracy. They used the average accuracy and recall as classification metrics. Their model achieved an average accuracy of 87.49%.

Graves et al. [22] proposed the bidirectional long short-term memory (BLSTM). BLSTM applies the concept of incremental learning and handles long-range contextual processing. The authors tested BLSTM on two large unconstrained handwriting databases where its recognition accuracy reached 79.7%.

Veeriah et al. [23] proposed a differential gating scheme for the LSTM neural network (dRNN). They tested dRNN on the KTH 2D action recognition and the MSR Action3D datasets. The results showed that the dRNN model outperformed the conventional LSTM approach.

Chung et al. [8] proposed a combination of RNN and LSTM models for Lexical Utterance Classification. They compared their approach to the ngram-based language models (LMs), feed forward neural network LMs, and boosting classifiers. The proposed approach outperformed the approaches mentioned above, and the results showed that the RNN worked well for short utterance series while LSTM worked efficiently for long series.

Studies showed that LSTM models performed well with long sequence applications [24][9]. Vu et al. [25] proposed to use Self-Gated RNN for human activity recognition on data gathered from wearable sensors. The results showed that their approach outperformed standard RNN and maintained comparable results to those of LSTM.

Finally, Karim et al. [26] proposed an approach to transform univariate time series classification models into a multivariate time series classification model. They augment the fully convolutional block with a squeeze-and-excitation block in order to enhance the accuracy. They applied their approach to activity recognition and action recognition, where they achieved good accuracy. The proposed model uses minimum memory requirements, which makes it efficient to work on constrained environments. Even though their proposed model outperformed existing models, it requires an additional preprocessing step for the raw data.

III. Long-Short Term Memory Architecture

In this section, we explain the Long-Short Term Memory (LSTM) architecture and the unidirectional and bidirectional structures.

Hochreiter [27] introduced LSTM to avoid the long-term dependency problem by adding the gating mechanism. LSTM is an RNN with an enhanced function to calculate the hidden state. Figure 1 depicts the building block (cell) of LSTM. This cell determines which data to keep in memory and which data to ignore using the concept of gating, where the latter selectively transfers needed information.

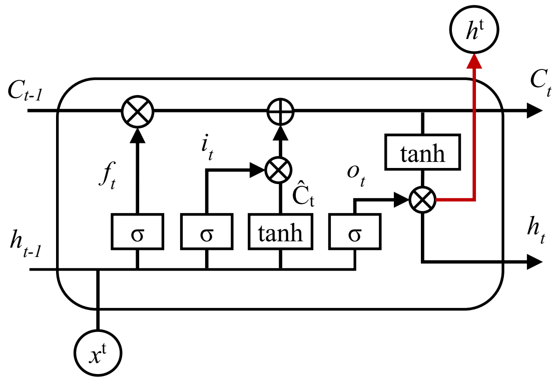


Figure 1. LSTM Cell Architecture

The LSTM cell consists of the *input* gate, the *forget* gate, and the *output* gate. The *forget* gate is responsible for selecting which data to remember and which data to erase. The *sigmoid* layer is responsible for this decision as shown in Equation 2.

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

The output is 0 or 1, where 0 means *forget* and 1 means *keep*. The second gate is the *input* gate. This gate uses another sigmoid layer that determines which values to update as shown in Equation 3.

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

The *tanh* function creates a vector of new candidate values that could be added to the cell state as shown in Equation 4.

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

The cell state is then ready for the update by concatenating both f_t and $\hat{C}_t$. The LSTM updates the old cell state (C_{t-1}) to be (C_t) as shown in Equation 5.

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

Equation 6 calculates the output of the sigmoid gate o_t.

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

When multiplying o_t with *tanh* (C_t), we implicitly determine which part to take out as seen in Equation 7.

$$h_t = \tanh(C_t) \times o_t \qquad (7)$$

The *output* gate, using a *sigmoid* function, determines which part of the cell state will come out, where *tanh* of the cell state is used to make values between -1 and 1.

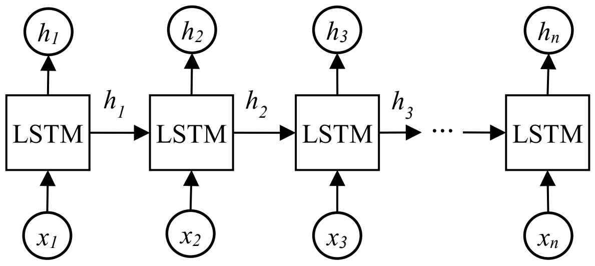


Figure 2. Unidirectional LSTM Layer Structure

This LSTM cell makes the building blocks of the RNN. Figure 2 shows the unidirectional LSTM structure. This architecture resembles the RNN architecture with LSTM cells instead. It shows how each LSTM cell's output is being inputted to the next LSTM cell in the same layer.

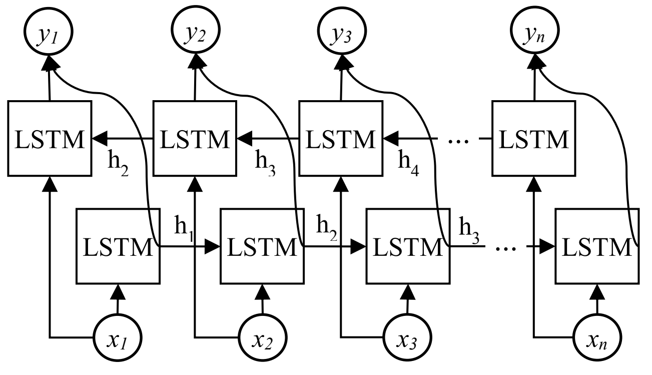


Figure 3. Bidirectional LSTM Layer Structure

Figure 3 shows the bidirectional LSTM structure where the forward layer is responsible for a positive time direction, and the backward layer is responsible for a negative time direction. The outputs from each LSTM cell in the two layers are concatenated in the output y_n.

IV. Approach

We study the usefulness of two LSTM models in recognizing human activities of data gathered from mobile phone sensors.

The human activity recognition task is a multiclass classification problem. Thus, we classify the activity samples into one specific class among different candidate classes. Activities are recorded as time series data using accelerometer sensors. Given a series of triaxial accelerometer data, the LSTM model is supposed to generate hypotheses $\tilde{y}$ of the actual set of labels y. An advantage of LSTM is that it can handle long-term dependencies. Accordingly, it has the ability to remove or add information to the cell state, carefully regulated by the gating scheme. Thus, LSTM cells are able to remember important information about the received input. This enables them to be very precise in the prediction of the next input.

We divide the data into training and testing sets (80% for training and 20% for testing [28]). Our model determines the first 80% of the samples as training data while the remaining 20% as testing data. Furthermore, we randomize the data using shuffling to ensure that the training and the testing sets contain all the possible cases defining the problem. The data samples of n dimensional patterns obtained by concatenating the m acceleration values (x, y, z) recorded along each Cartesian direction. For example, if the sampling rate is 20 Hz and the window size is 3 seconds. Then, the vector will include 180 acceleration values where each dimension contains 60 samples.

Classification problems require using a loss function for describing the model accuracy. The loss function determines the deviation of the results from the true values. The smaller the loss is, the more accurate the results are. We used the cross entropy loss (log loss function) as a cost function, which measures the inaccuracy of the model by determining the model's ability to find a relationship between input data and output labels.

We chose the Adam optimization function, a widely used optimizer, for the training of the model [29]. Adam is an adaptive version of the stochastic gradient descent. The criteria for selecting the optimizer depend on how fast and efficient the optimizer is when updating the network parameters (weights and biases).

A common problem in machine learning is model overfitting. If the model is over-fitted, the resulting model is almost useless. This problem is observed when the loss function of the model is small in the training data while it is large in the testing data. Model overfitting causes a large difference between the training accuracy and the testing accuracy.

Our model tries to restrain the overfitting by using the *Dropout* function, which is a formalization method that is widely used in modern deep learning environments [30]. It randomly selects some of the cells (neurons) in the neural layer and temporarily hides them. Then, it performs the training and the optimization process of the neural network in a certain loop. In the next loop, it will hide some other cells (neurons) until the end of the training. Thus, it randomly skips some neurons at training to force the other neurons to pick up the slack. For the bidirectional model, we applied the *Dropout* in both LSTM layers. *Dropout* is a smart way for regularization. It reduces the network tendency to become over-dependent on some neurons as they may not be available all the time.

Figure 4 presents the training process of the model. The first step after inputting the raw data to the LSTM model is the initialization of the learning parameters. Then, we compute the LSTM cell states, followed by computing the model prediction. After calculating the cross entropy loss, we train the model using Adam optimizer, followed by adjusting the training parameters accordingly. We repeat this process until the model achieves the maximum accuracy level.

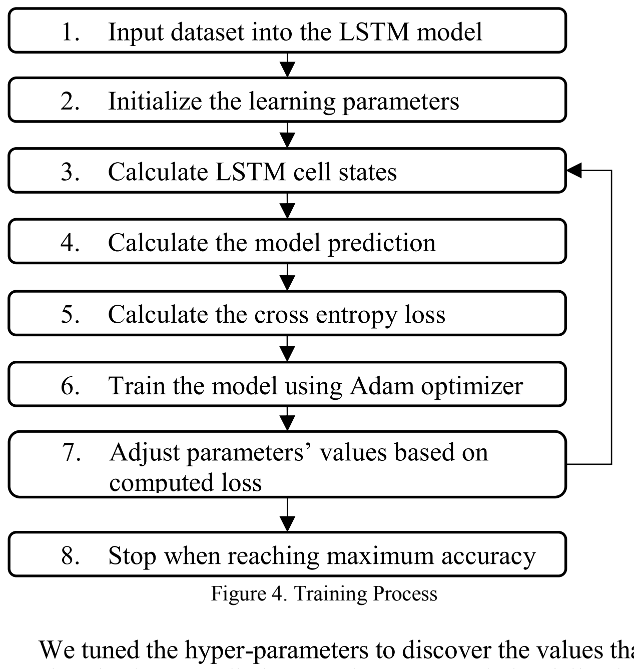

1. Input dataset into the LSTM model
2. Initialize the learning parameters
3. Calculate LSTM cell states
4. Calculate the model prediction
5. Calculate the cross entropy loss
6. Train the model using Adam optimizer
7. Adjust parameters' values based on computed loss
8. Stop when reaching maximum accuracy

Figure 4. Training Process

We tuned the hyper-parameters to discover the values that acquire the best prediction results. We used the following training hyper-parameters for classification.

1. *Number of Epochs*: an epoch represents one full pass on the entire dataset.
2. *Number of iterations*: number of batches required to perform one epoch.

3. *Batch size*: number of training examples in a single batch.
4. *Optimizer function*: minimizes the error function in a specific model.
5. *Hidden units*: hidden neurons in LSTM structure.
6. *Learning rate*: how fast weights are changed.

We used the training accuracy and the testing accuracy as a measure of recognition quality. We measure the accuracy using Equation 8.

$$accuracy = \frac{\text{The amount of correct classifications}}{\text{The total amount of classifications}} \quad (8)$$

We calculate the training accuracy when we apply the model to the training data, while we calculate the testing accuracy when we apply the model to the testing data. Both accuracies are important to identify the overfitting of the model.

V. Evaluation

We conducted the experiments on a Windows 10 workstation where the code is written in Python 3 using Keras v2.2.2 and TensorFlow v1.10.0. The CPU is a 4-core Intel(R) Core(TM) i7-6700HQ, 2.60 GHz, and 8 GB of RAM. We tested the models using two different datasets. The UniMib Shar [12] and the WISDM dataset [13]. In the following, we present the accuracy of the two models on the two datasets as well as the training times.

a. UniMib Shar Dataset

The dataset contains 11771 data samples for daily life activities, and fall states acquired using smartphones. Data collection is performed by 30 persons (24 females and six males) ranging from 18 to 60 years old [12]. The different characteristics of the subjects in the UniMiB SHAR dataset are shown in Table 1.

Table 1. The Characteristics of Subjects

	Female	**Male**	**Total/Range**
Subjects	24	6	30
Age	18-55	20-60	18-60
Height (cm)	160-172	170-190	160-190
Weight (kg)	50-78	55-82	50-82

Samsung Galaxy Nexus I9250 with the Android OS version 5.1.1 was used in the experiments that aimed to build the dataset using its Bosh BMA220 acceleration sensor. This triaxial low-g acceleration sensor allows measurements of acceleration in three perpendicular axes (x, y, and z). For each activity, the accelerometer data vector is made of three vectors of 151 values (total vector size is 151 x 3 = 453), one for each acceleration direction where each activity spans 3 seconds with a sampling rate is 50Hz. Overall, the main feature used from this dataset is the acceleration value in the x, y, z axes.

Samples are divided into 17 fine-grained classes and grouped into two coarse-grained classes. The first class (*A*) contains samples of nine types of daily living activities (*ADL*), and the second class (*F*) includes samples of eight types of falls. The dataset contains 7579 ADL states and 4192 fall states.

We used four different classification tasks for the evaluation (*AF-2*, *F-8*, *A-9*, and *AF-17*).

AF-2 consists of two classes obtained by considering all the ADLs as one class and all the fall states as another class. It allows evaluating the classifier's robustness in distinguishing between ADLs and fall states.

F-8 consists of eight classes obtained by considering all the classes of fall states. It allows evaluating the ability of the classifier to differentiate among different types of fall states.

A-9 consists of nine ADL classes. It determines the ability of the classifier to distinguish among different types of ADLs.

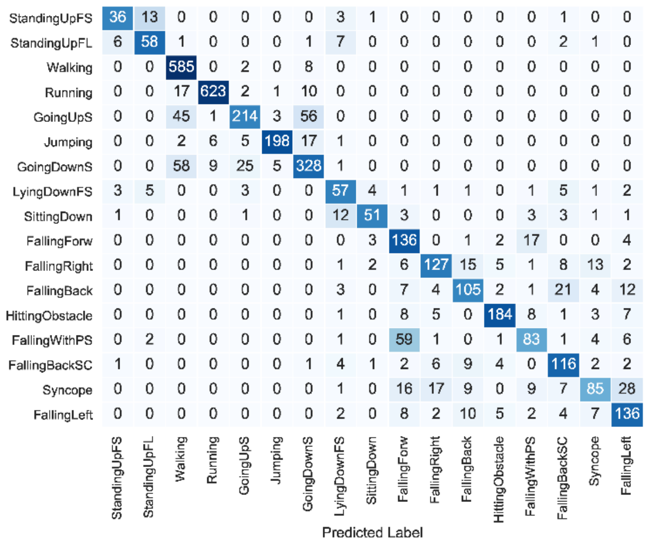
Finally, *AF-17* consists of 17 classes (nine classes of ADLs and eight classes of fall states). It evaluates the ability of the classifier to determine the type of movement regardless of being an ADL or a fall state.

Table 2 shows the results obtained from running the unidirectional and the bidirectional LSTM models on the raw data for the four tasks along with the values of the hyper-parameters. These results were achieved after several experiments and tuning of the parameters.

Table 2. UniMib Shar Training Accuracy and Time

Parameter	**AF-2**	**F-8**	**A-9**	**AF-17**
Epochs	50	100	70	50
Iterations	45	13	12	148
Batch size	200	256	512	64
Learning rate	0.001	0.001	0.001	0.001
Hidden units	200	200	200	200
Dropout	0.5	0.5	0.5	0.5
Accuracy				
Unidirectional	98.87%	73.46%	90.25%	80.59%
Bidirectional	99.25%	74.89%	91.2 %	83.31%
Training Time (minutes)				
Unidirectional	48	10	9	310
Bidirectional	101	17	15	685

Figure 5 shows the confusion matrix for the AF-17 task using the unidirectional approach.



	StandingUpFS	StandingUpFL	Walking	Running	GoingUpS	Jumping	GoingDownS	LyingDownFS	SittingDown	FallingForw	FallingRight	FallingBack	HittingObstacle	FallingWithPS	FallingBackSC	Syncope	FallingLeft
StandingUpFS	36	13	0	0	0	0	0	3	1	0	0	0	0	0	1	0	0
StandingUpFL	6	58	1	0	0	0	1	7	0	0	0	0	0	0	2	1	0
Walking	0	0	585	0	2	0	8	0	0	0	0	0	0	0	0	0	0
Running	0	0	17	623	2	1	10	0	0	0	0	0	0	0	0	0	0
GoingUpS	0	0	45	1	214	3	56	0	0	0	0	0	0	0	0	0	0
Jumping	0	0	2	6	5	198	17	1	0	0	0	0	0	0	0	0	0
GoingDownS	0	0	58	9	25	5	328	1	0	0	0	0	0	0	0	0	0
LyingDownFS	3	5	0	0	3	0	0	57	4	1	1	1	0	1	5	1	2
SittingDown	1	0	0	0	1	0	0	12	51	3	0	0	0	3	3	1	1
FallingForw	0	0	0	0	0	0	0	0	3	136	0	1	2	17	0	0	4
FallingRight	0	0	0	0	0	0	0	1	2	6	127	15	5	1	8	13	2
FallingBack	0	0	0	0	0	0	0	3	0	7	4	105	2	1	21	4	12
HittingObstacle	0	0	0	0	0	0	0	1	0	8	5	0	184	8	1	3	7
FallingWithPS	0	2	0	0	0	0	0	1	0	59	1	0	1	83	1	4	6
FallingBackSC	1	0	0	0	0	0	1	4	1	2	6	9	4	0	116	2	2
Syncope	0	0	0	0	0	0	0	1	0	16	17	9	0	9	7	85	28
FallingLeft	0	0	0	0	0	0	0	2	0	8	2	10	5	2	4	7	136

Figure 5. AF-17 Confusion Matrix for Unidirectional Approach

For the binary classification (AF-2), the accuracy was 98.87% for unidirectional and 99.25% in the case of the bidirectional model. However, we only reached 74.89% accuracy in the case of F-8. The misclassification usually occurs in some fall states due to the similarity between some pairs such as Syncope and Falling leftward, Generic falling backward, and Falling backward-sitting-chair. For A-9, the accuracy was up to 91.2%, while for AF-17, it reached 80.59% in the case of unidirectional and 83.31% for bidirectional. Finally, the training time for the bidirectional approach was considerably higher than that of the unidirectional model with only minor accuracy enhancement.

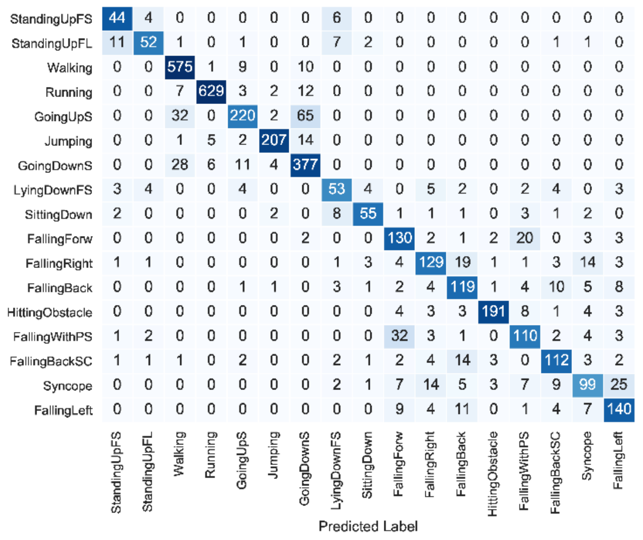


Figure 6. AF-17 Confusion Matrix for Bidirectional Approach

Figure 6 shows the confusion matrix for the AF-17 task using the bidirectional approach.

b. WISDM Dataset

The second dataset in our experiments, the activity prediction dataset v1.1 [13], is from the Wireless Sensor Data Mining (WISDM) Lab. It was collected through controlled laboratory conditions in 2010. The dataset contains 36 users with 1,098,207 labeled samples of six classes. The sampling rate of data collection is 20Hz, which means one sample every 50 milliseconds. The classes are distributed in the dataset as walking (38.6%), jogging (31.2%), Upstairs (11.2%), Downstairs (9.1%), sitting (5.5%), and standing (4.4%).

We processed the data to make a labeled activity every three seconds. Each labeled activity should belong to the same user and the same activity from the raw data. Thus, each data vector is made of three vectors of 60 values. The total number of data points is 18210, where each point contains 60 samples in 3D format (total vector size is 60 x 3 = 180). Since each window contains 60 samples, we used 1092600 samples from the dataset.

TABLE 3. WISDM TRAINING ACCURACY AND TIME

Parameter	**WISDM (A-6)**	
Epochs	50	
Iterations	228	
Batch size	64	
Learning rate	0.001	
Hidden units	200	
Dropout	0.5	
Model	**Accuracy**	**Training Time (minutes)**
Unidirectional	97.68%	375
Bidirectional	98.11%	794

Table 3 shows the parameter values used in the training phase. Also, it shows the accuracy and the training time for both unidirectional and bidirectional LSTM. The difference in accuracy was less than 0.05%. However, the training time for the bidirectional model was more than double the training time of the unidirectional model. This observation shows that using the bidirectional approach for this dataset could be an overhead and not necessary.

Figure 7 and Figure 8 show the confusion matrix for WISDM using the unidirectional approach and the bidirectional approach, respectively.

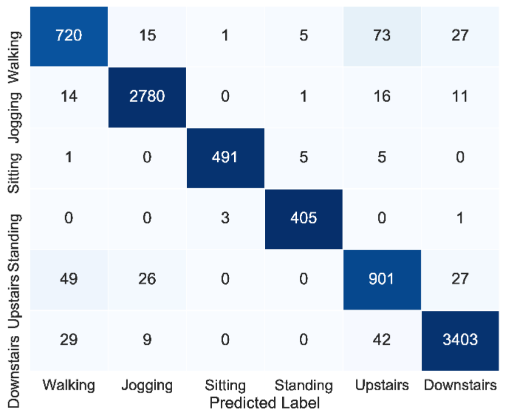


Figure 7. WISDM Confusion Matrix for Unidirectional Approach

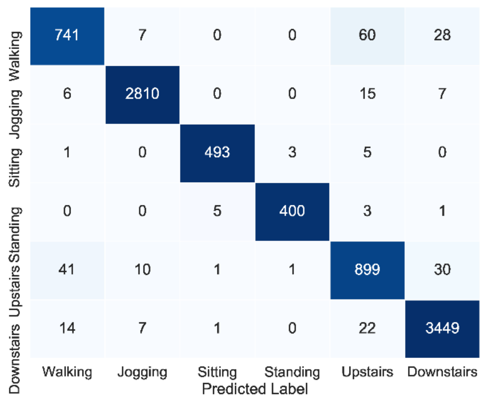


Figure 8. WISDM Confusion Matrix for Bidirectional Approach

VI. CONCLUSION

In this paper, we presented a comparison of the unidirectional and bidirectional LSTM models for human activity recognition. We used two different datasets for the comparison. The results showed that using the bidirectional LSTM slightly improves on the training accuracy for these two datasets. Moreover, we achieved an accuracy of up to 99% for the UniMiB SHAR dataset and 98.11% for the WISDM dataset. In the future, we intend to train the two models on more datasets. Additionally, we intend to use various machine learning and neural network approaches in the classification of human activity recognition. This enhancement will provide more generalization for the behavior of the two LSTM approaches.

REFERENCES

[1] V. Osmani, S. Balasubramaniam, D. Botvich, “Human activity recognition in pervasive health-care: Supporting efficient remote collaboration,” Jo. of Net. and Comp. App., 31(4), 2008, pp. 628–655.

[2] P. Woznowski, R. King, W. Harwin, I. Craddock, “A human activity recognition framework for healthcare applications: ontology, labelling strategies, and best practice,” In Proc. of the International Conference on Internet of Things and Big Data (IoTBD) INSTICC, 2016, pp. 369–377.

[3] N. Bidargaddi, A. Sarela, L. Klingbeil, M. Karunanithi, “Detecting walking activity in cardiac rehabilitation by using accelerometer,” Proc. 3rd Int. Conf. Intell. Sensors Sensor Netw. Inf., pp. 555-560, Dec. 2007.

[4] J. Wang, Y. Chen, S. Hao, X. Peng, L. Hu, “Deep learning for sensor-based activity recognition: A survey,” Pattern Recognition Letters, 2018.

[5] Z. Chen, Q. Zhu, Y. C. Soh and L. Zhang, "Robust Human Activity Recognition Using Smartphone Sensors via CT-PCA and Online SVM," in IEEE Transactions on Industrial Informatics, vol. 13, no. 6, pp. 3070-3080, Dec. 2017.

[6] Y. Bengio, A. Courville, P. Vincent, "Representation learning: a review and new perspectives," IEEE Transactions on Pattern Analysis, Machine Intell. 35, 1798–1828 (2013).

[7] K. Cho, B. Van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Holger, Y. Bengio, “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation,” 10.3115/v1/D14-1179.

[8] S. Ravuri, A. Stolcke, “Recurrent neural network and LSTM models for lexical utterance classification,” In INTERSPEECH-2015, 135-139.

[9] F. Gers, N. Schraudolph, J. Schmidhuber, “Learning Precise Timing with LSTM Recurrent Networks,” Journal of Machine Learning Research, 3:115–143, 2002.

[10] S. Venugopalan, H. Xu, J. Donahue, M. Rohrbach, R. Mooney, K. Saenko, “Translating videos to natural language using deep recurrent neural networks,” arXiv preprint arXiv:1412.4729, 2014.

[11] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, G. Toderici, “Beyond short snippets: Deep networks for video classification,” arXiv preprint arXiv:1503.08909, 2015.

[12] D. Micucci, M. Mobilio, P. Napoletano, “UniMiB SHAR: A new dataset for human activity recognition using acceleration data from smartphones,” IEEE Sens. Lett. 2016, 2, 15–18.

[13] J. R. Kwapisz, G. M. Weiss, S. A. Moore, "Activity Recognition using Cell Phone Accelerometers," Proceedings of the Fourth International Workshop on Knowledge Discovery from Sensor Data KDD-10, 2010.

[14] M. Uddin, M. Hassan, A. Almogren, M. Zuair, G. Fortino, J. Torresen, “A facial expression recognition system using robust face features from depth videos and deep learning,” Comput. Electr. Eng., 2017.

[15] G. Hinton, S. Osindero, Y. Teh, “A fast learning algorithm for deep belief nets,” Neural Comput. 18(7):1527–1554, 2006.

[16] F. Foerster, M. Smeja, and J. Fahrenberg, “Detection of posture and motion by accelerometry: a validation study in ambulatory monitoring,” Computers in Human Behavior, vol. 15,no. 5, pp. 571–583, 1999.

[17] D. Ravi, C. Wong, B. Lo, G. Yang, “A deep learning approach to on-node sensor data analytics for mobile or wearable devices,” IEEE Jornal of Biomedical and Health Informatics. 21(1): 56–64, 2017.

[18] M. Husken, P. Stagge, “Recurrent neural networks for time series classification,” Neurocomputing, 50 (2003), pp. 223-235.

[19] M. Hassan, M. Uddin, A. Mohamed, A. Almogren, “A robust human activity recognition system using smartphone sensors and deep learning,” Future. Gener. Comput. Syst. 81:307–313, 2018.

[20] W. Jiang, Z. Yin, “Human activity recognition using wearable sensors by deep convolutional neural networks,” In Proc. of the 23rd Annual ACM Conference on Multimedia Conference. pp. 1307–1310, 2015.

[21] N. Mehdiyev, J. Lahann, A. Emrich, D. Enke, P. Fettke, P. Loos, “Time Series Classification using Deep Learning for Process Planning: A Case from the Process Industry,” Procedia Comput. Sci. 2017, 114, 242–249.

[22] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber, “A Novel Connectionist System for Unconstrained Handwriting Recognition,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 31, no. 5, pp. 855-868, 2009.

[23] V. Veeriah, N. Zhuang, G.-J. Qi, “Differential recurrent neural networks for action recognition,” In Proceedings of IEEE International Conference on Computer Vision (ICCV), pages 4041–4049, 2015.

[24] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling,” arXiv preprint arXiv:1412.3555, 2014.

[25] T. Vu, A. Dang, L. Dung, J-C. Wang, “Self-Gated Recurrent Neural Networks for Human Activity Recognition on Wearable Devices,” In Proceedings of the on Thematic Workshops of ACM Multimedia 2017 (Thematic Workshops '17). ACM, New York, NY, USA, 179-185.

[26] F. Karim, S. Majumdar, H. Darabi, and S. Harford, “Multivariate lstmfcns for time series classification,” Neural Networks, Volume 116, Pages 237-245, August 2019.

[27] S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural computation, 9(8):1735-1780, 1997.

[28] K. Dobbin and R. Simon, “Optimally splitting cases for training and testing high dimensional classifiers,” BMC Medical Genomics, vol. 4, no. 1, p. 31, 2011.

[29] D. Kingma, J. Ba, “Adam: A method for stochastic optimization,” arXiv preprint arXiv:1412.6980, 2014.

[30] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” J. Machine Learning Res. 15, 1929–1958 (2014).