Evolving Systems
https://doi.org/10.1007/s12530-021-09373-6

ORIGINAL PAPER

Evaluation of deep learning model for human activity recognition

Owais Bhat[1] · **Dawood A Khan**[1]

Received: 20 February 2020 / Accepted: 2 March 2021

Abstract
Recognizing a person's physical activity with certainty makes an important aspect of intelligent computing. Modern smart devices are equipped with powerful sensors that are suitable for sensor-based human activity recognition (AR) task. Traditional approaches to human activity recognition has made significant progress but most of those methods rely upon manual feature extraction. The design and selection of relevant features is the most challenging task in sensor-based human AR problem. Using manually extracted features for this task hinders the generalization of performance and these handcrafted features are also incapable of handling similar and complex activities with certainty. In this paper, we propose a deep learning based method for human activity recognition problem. The method uses convolutional neural networks to automatically extract features from raw sensor data and classify six basic human activities. Furthermore, transfer learning is used to reduce the computational cost involved in training the model from scratch for a new user. The model uses the labelled information from supervised learning, to mutually enhance the feature extraction and classification. Experiments carried on benchmark dataset verified the strong advantage of proposed method over the traditional human AR algorithms such as Random Forest (RF) and multiclass Support Vector Machine (SVM).

Keywords Deep learning · Sensors · Activity recognition · Classification · Convolutional neural network

1 Introduction

Recent years have witnessed a rapid increase in the use of smart portable devices. These devices are becoming more and more sophisticated. Current generation smart phones, fitness trackers, and music players now integrates many varied and powerful sensors (Yuan et al. 2018). These sensors include gyroscope, GPS sensors, accelerometer, direction sensors, temperature sensors etc. The availability of different sensors in these communication devices creates new opportunities for data-mining applications (Kwapisz et al. 2011; Brezmes et al. 2009; Wang and Chen 2017; Angelov et al. 2010). The continuous data extracted from these sensors opens up new areas of intelligent computing (fitness trackers, security, health monitors etc.) (Chennuru et al. 2010; Wu et al. 2011, 2013). One such area is human activity recognition (AR). AR is a classification problem where continuous stream of data from sensors is generated and from this continuous stream only a portion of the data that varies with different activities is of significance. To detect and classify the patterns from this (multi-variate) stream of data is a challenging task (Rokni and Ghasemzadeh 2017; Angelov and Zhou 2008). Mostly used technique in AR is to divide this continuous data stream into overlapping segments or windows. A smaller segment size may not capture all the features of motion states, and a larger segment can incorporate noise due to involvement of multiple states (Reddy et al. 2010). Typically, a segment of one second is often utilized for activity classification, which has been validated by prior work (Peterek et al. 2014). Each window is processed separately and is transformed into feature vectors. Performance of the methods used in recognizing a person's activities depends upon the extracted features and most of the approaches rely upon hand-crafted statistical features like mean, variance, entropy or correlation coefficients (Figo et al. 2010; Bao and Intille 2004). Feature extraction from other domains; time-domain features (e.g., Zero-crossings of acceleration), frequency-domain features (e.g., FFT DC component of acceleration) (Krause et al. 2003) and discrete-domain features (e.g., Dynamic Time

✉ Owais Bhat
owais031@gmail.com

[1] Department of Computer Sciences, North Campus, University of Kashmir, Srinagar, Jammu and Kashmir 190006, India

Published online: 11 March 2021

Warping of acceleration) are also used in some methods (Huynh and Schiele 2005; Krause et al. 2003). The problem of designing hand-crafted features is well studied in other fields like image recognition Lowe (1999) and requires a strong domain-specific knowledge. Recent years have witnessed a tremendous increase in the computing power as a result new techniques like deep learning is attracting the researchers around the globe. Deep learning has shown good performance in speech and image recognition (Tang et al. 2012; Bagct and Bai 2007; Choudhary and Hazra 2019), where CNN is used to extract the features from the raw signals automatically (Krizhevsky et al. 2017). CNNs has the great potential to be used in human AR due to its ability to store features in its complex layered architecture. In this work, an automatic feature extraction method for human AR is proposed. The proposed technique uses convolutional neural networks to automate the feature extraction process and classify between different activities with high degree of certainty. The rest of this paper is organized as follows. A survey of the relevant literature is presented in the next section highlighting shortcomings of the existing methods. In methodology section the proposed model architecture is briefly presented followed by the results and discussion section which shows the experimental results along with comparison of proposed method with different classification algorithms. Finally, the conclusion of this work along with some remarks is presented.

2 Related work

Activity recognition (AR) using smart phone sensors (accelerometer) can be considered as a classification problem which is well addressed in the past. In human AR problem, extracting features from time-series sensor data is the most challenging task and there is a very small portion of data that is relevant to recognizing human activities and also these basic activities can last up to only few seconds. So handcrafted features often fails to capture all the relevant details in time series data resulting in poor performance of an algorithm in classifying different activities. Features such as mean, entropy, correlation coefficients and standard deviation are mostly used by human AR classifiers (Figo et al. 2010). Work presented in (Kwapisz et al. 2011) has shown that multilayer perceptron model trained on various handcrafted features produced most accurate results. Extracting these hand-crafted features becomes challenging when the data is generated from continuously changing and uncontrolled environments. (Sharma et al. 2008; Khan 2013) have used neural networks and decision trees as classifiers respectively, but showed poor performance in predicting the similar activities. Recently, approaches like principal component analysis (PCA) (Plotz et al. 2011) and Boltzman machine (Vollmer et al. 2013; Bhattacharya et al. 2014) were used to automatically extract features from raw signals, but they do not have the capability to model the complex non-linear dependencies from the linear combination of extracted features (Bengio 2009). To avoid handcrafted feature extraction, a new approach for this task is based on deep learning. These deep learning techniques have resulted in good performance in image and speech recognition and several works although a few in number have been conducted to apply it to activity recognition problem. Authors in (Yang et al. 2015; Ordonez and Roggen 2016) used the data from multiple sensors and generated a single sensor image which is passed to the CNN for classification. There is also difference in CNN architectures, in (Yang et al. 2015) one convolution and two fully connected layers were used, in (Ronaoo and Cho 2015; Ronao and Cho 2016) three and four convolutional layers were used. Increasing the number of layers is expected to learn more abstract features but it often leads to data over fitting, so a proper balance needs to be maintained here. Authors in Thomas Platz and Olivier (2012) used deep belief network in classifying different human activities but these unsupervised models did not make use of available labelled information and also lack the effective feature extraction capability provided by convolutional, pooling and rectifier units.

In this work a method for human AR is proposed where a CNN is used to (1) extract and learn features automatically from the time series sensor data (2) classify different activities based on the features extracted from inner layers (3) preserve the knowledge represented in the lower layers for a new user (Transfer Learning), so that computational cost involved in training the model from scratch is reduced.

3 Methodology

In this section, we describe the structure of CNN and present the system architecture proposed in this paper.

3.1 CNN for feature extraction and classification

CNN is a neural network structure which is inspired by the structure of the brain where information is stored in series of patterns, in sequential order. Similar to a child who learns to recognize different objects over time, a neural network is taught to generalize the input and make appropriate predictions. In a regular neural network there are series of fully connected hidden layers, input from these lower layers is transformed and presented to the outer layer that represents the predictions. In CNN, neurons present in inner layers are organized along different dimensions like width, height, depth and are not fully connected. Neurons in lower layer connect to a set of neurons in the higher layer. Finally the

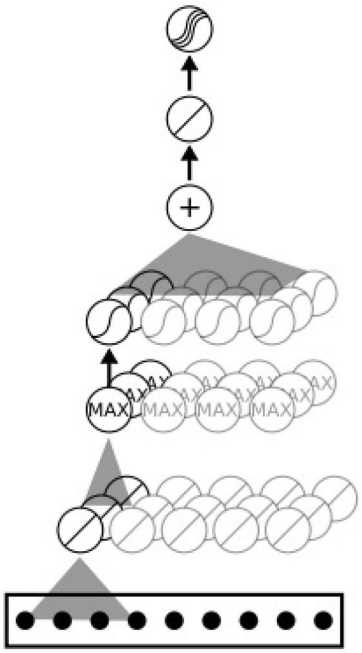

Fig. 1 CNN layers

probability scores of output are produced as a single vector which is organized along the depth. The overall structure of CNN is described in Fig. 1.

Feature extraction: In lower layers of the network, several Convolution and Pooling layers are added from where features are extracted. The convolution operation involves the combination of two functions to produce a third function. In convolutional layers, the input data is used with a filter or a kernel to produce a feature map. These filters slide over the input data by performing the matrix multiplication at every location and store the result into a feature map. The sliding of filters over the input data is called a Stride. It is the size of the step taken by the convolutional filter while moving over the input. We can obtain different feature maps by performing the several convolutions over the input data. Padding is added to avoid the shrinking of Feature maps and allowsthe kernel and stride size to fit into the input. A nonlinear output is produced from the convolutional layer by using an activation function. Three commonly used activation functions are sigmoidal, hyperbolic tangent and ReLU (Rectified Linear Unit). For the jth feature map in the ith layer of CNN, the value at xth row is denoted as u_{ij}^{x}. Formally, the value u_{ij}^{x} is given by

$$u_{ij}^{x} = tanh\left(b_{ij} + \Sigma_m \Sigma_{p=0}^{pi-1} w_{ijm}^{p} u_{(i-1)m}^{x+p}\right), \tag{1}$$

where tanh(.) is the tangent function, m is the index over the set of feature maps in the $(i-1)$th layer connected to the current feature map,b_{ij} is the bias for this feature map, w_{ijm}^{p} is the value at the position p of the convolutional kernel, and P_i is the length of convolutional kernel. After the convolution layer, it is common to add a pooling layer between CNN layers to reduce and summarize the obtained representation. This is usually done in two ways: max pooling, where maximum value from the sub region is selected

$$u_{ij}^{x} = max_{1\le q\le Qi}\left(u_{(i-1)j}^{x+q}\right). \tag{2}$$

Or average pooling, where average value of data from the region of lower layer is selected.

$$u_{ij}^{x} = \frac{1}{Qi}\Sigma_{1\le q\le Qi}\left(u_{(i-1)j}^{x+q}\right), \tag{3}$$

where Q_i is the length of the pooling region.

Pooling is introduced in CNN to address the problem of over fitting, this also helps in reducing the number of parameters and computational effort needed in training the network.

Classification: This part of our network is same as a regular neural network where the features are mapped into the output classes. CNN at this level is fully connected and all the activations from multidimensional data from lower layers is flattened to produce the single dimensional output. The output at this layer is governed by the softmax function.

$$u_{ij} = \frac{exp(u_{(i-1)j})}{\Sigma_{j=1}^{c} exp(u_{(i-1)j})}, \tag{4}$$

where C is the number of output classes.

Finally the Softmax layer computes the probability distribution of the predicted class. All the layers mentioned are trained as a whole using a back propagation algorithm and weights are optimized with gradient descent.

3.2 Architecture

The convolutional neural network architecture is shown in Fig. 2.

The model consists of a max pooling layer separating the two convolution layers. The convolution layer is followed by a fully connected layer which is further, connected to a Softmax layer. Both, max pool and convolution layers are temporal or 1 Dimensional. The first convolutional layer has a filter size and depth of 60 (number of channels, as output from convolutional layer), after performing the 1D convolution over the input channel data, output is passed through ReLU activation function. Training hidden layers

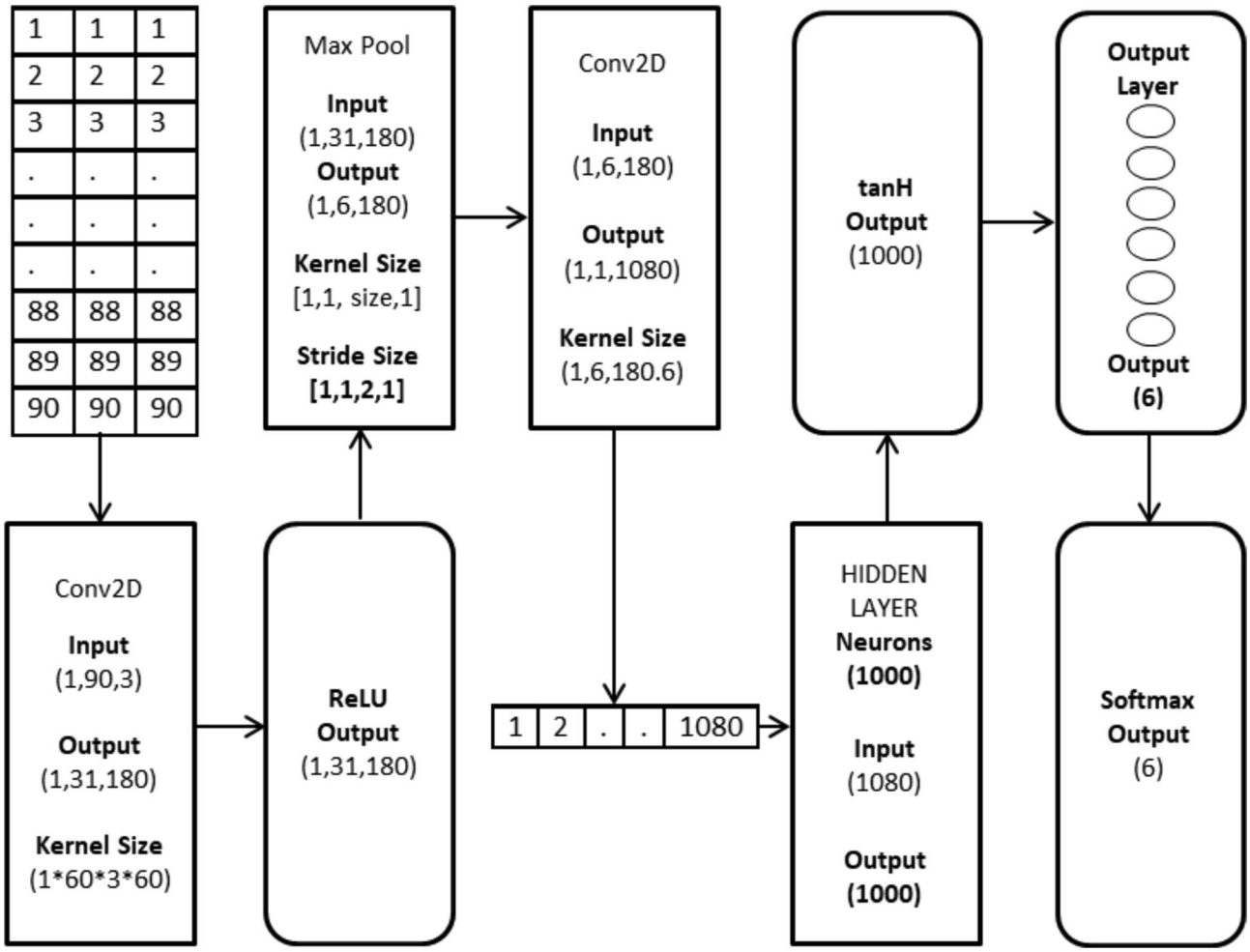

Fig. 2 CNN architecture

with ReLU has proven to be useful in avoiding the vanishing gradient problem. The pooling layer has filter size of 20 and with a stride of 2. The convolution layer at the top takes an input from max-pooling layer, applying the filter size of 6 and have tenth of depth as of max pooling layer. The output is flattened for the fully connected layer which consists of 1000 neurons. It uses tanh(.) function as defined by Eq. 1 as non-linearity. Finally, the Softmax layer is defined in Eq. 4 to output the probabilities of class labels.

Transfer Learning: Deep learning models require large amount of labeled data and sometimes collecting the bulk of data becomes difficult. In these situations Transfer Learning (TL) becomes useful and computationally efficient approach in learning features across similar and different domains. TL is used where knowledge learned from tasks for which a lot of labeled data is available is used in settings where only little labeled data is available. This help in simplifying the generalization process by starting from patterns that have been learned for different or similar task. In our CNN based model we employed transfer learning by using the same architecture while freezing inner layers except the classification layer to retain weights of these layers and retrained outer layers with few numbers of instances in the target domain. Tables 1 and 2 presents the confusion matrix and performance metrics after using transfer learning, respectively. CNN model that used TL shows better performance in terms of accuracy and other performance metrics in contrast to CNN model that does not use transfer learning (Tables 3, 4). Table 5 shows the experimental setup of the CNN model which was trained on a Core i5 processor with 6 GB of RAM and 2 GB NVIDIA graphics card.

Hyperparmeters: The proposed architecture is considered after assessing the effect of possible combination of CNN hyperparameters (number of layers, filter size, depth of feature map) on network performance. Optimal architecture is selected after making adjustments in number of layers. As shown in Fig. 3, adding the third layer to the network did not result in significant improvement in accuracy. Similarly,

Table 1 Confusion matrix for proposed CNN using TL

	Downstairs	Jogging	Sitting	Standing	Upstairs	Walking
Downstairs	**618**	19	12	0	22	30
Jogging	36	**1825**	0	0	103	36
Sitting	3	0	**357**	21	1	11
Standing	1	0	2	**275**	6	4
Upstairs	38	21	0	0	**648**	81
Walking	17	62	0	1	133	**2488**

Table 2 Metric evaluation for proposed CNN using TL

	Downstairs	Jogging	Sitting	Standing	Upstairs	Walking
TP	618	1825	357	275	648	2488
TN	6075	4779	6464	6561	5818	4008
FN	95	102	14	22	265	162
FP	83	175	36	13	140	213
Precision	0.8815	0.9125	0.9083	0.9548	0.8223	0.9211
Recall	0.8667	0.9470	0.9622	0.9259	0.7097	0.9388
F-score	0.8741	0.9294	0.9345	0.9401	0.7619	0.9299

Table 3 Confusion matrix for proposed CNN without TL

	Downstairs	Jogging	Sitting	Standing	Upstairs	Walking
Downstairs	**601**	24	0	9	19	48
Jogging	41	**1815**	0	2	100	42
Sitting	2	0	**346**	28	5	12
Standing	2	0	4	**261**	14	7
Upstairs	46	31	0	9	**608**	94
Walking	16	69	2	9	175	**2430**

Table 4 Metric evaluation for proposed CNN without TL

	Downstairs	Jogging	Sitting	Standing	Upstairs	Walking
TP	601	1815	346	261	608	2430
TN	6063	4747	6472	6526	5770	3967
FN	107	124	6	57	313	203
FP	100	185	47	27	180	271
Precision	0.8573	0.9075	0.8804	0.9062	0.7715	0.8996
Recall	0.8488	0.9360	0.9829	0.8207	0.6601	0.9229
F-score	0.8530	0.9215	0.9288	0.8613	0.7115	0.9111

Table 5 Experimental setup

Parameter	Value
Input height	1
Input vector size	90
Number of channels	3
Learning rate	0.0001–0.001
Batch size	10–15
Hidden neurons	1000
Kernal size	60
Depth	60

optimal values for filter size and feature maps are selected by making adjustments using greedy approach. Latter part of our network is a regular ANN with softmax layer providing the probability distribution of class labels.

4 Results and discussion

In this section, we present different metrics that are used in evaluating the performance of machine learning models used for classification, followed by the discussion on experiment

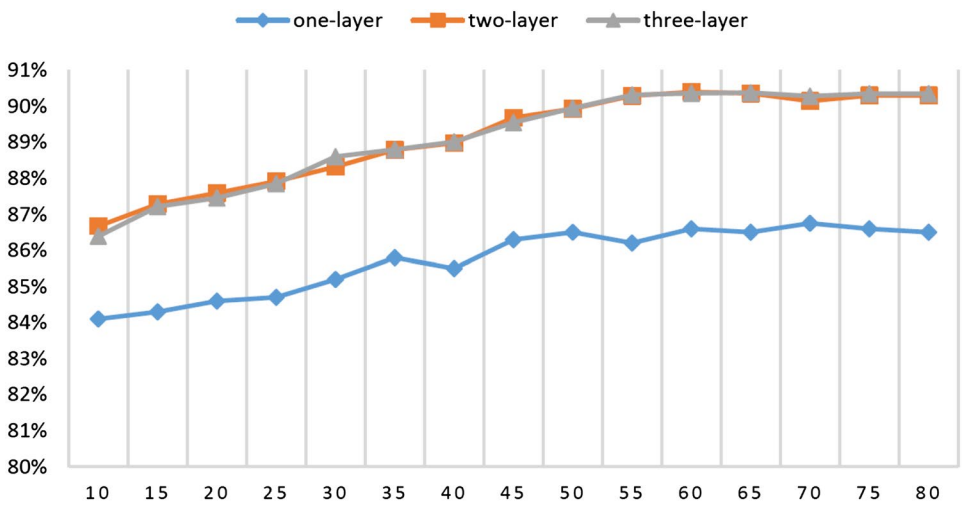

Fig. 3 Accuracies of three layer structure on test data with feature maps

results and finally comparison of the proposed method are presented.

Evaluation metrics In problems involving classification, we use training data as labeled (x, y) pairs to estimate a function $= g(x)$ in order to make predictions. In case of binary classification $y^{(i)} \epsilon \{0, 1\} \forall\, i$ and in multiclass classification y take any one of the discrete values. We often choose g(x) as:

$$g(x) = argmaxP'(Y = y|X). \tag{5}$$

In most of the problems involving machine learning we are given N training examples labeled as: $(x^1, y^1), (x^2, y^2), \ldots, (x^N, y^N)$. Where, $x^{(i)}$ is a feature vector of m discrete features for the ith training example. $y^{(i)}$ is a corresponding label for ith training example.

For any machine learning algorithm including classification tasks, the metrics we select for evaluation are very important. Performance of these algorithms while doing evaluation and comparisons is established on metrics we choose. They are the scales of measuring different characteristics and the weights each of these characteristics carry. They ultimately influence our choice of selecting a particular machine learning algorithm. Classification problems are most widely addressed by machine learning algorithms and as such there are number of metrics to evaluate the prediction capability of these algorithms. Every classification algorithm predicts four outcomes (Fig. 4) and these four outcomes are used in calculation of different metrics.

True positive (TP): correct positive prediction. False positive (FP): incorrect positive prediction. True negative (TN): correct negative prediction. False negative (FN): incorrect negative prediction. Metrics that are widely used in classification problems are as follows:

Classification accuracy It is the most frequently used metric in classification tasks. It is the ratio of correct predictions made to the total number of predictions. Accuracy as an evaluation metric is suitable only when number of training examples is almost equally distributed among different classes.

Logarithmic Loss is a performance metric where a probability of a given training example is calculated based on a degree of membership towards a particular class. The values of probability which ranges between 0 and 1, gives the measure of performance of a classification algorithm. Correct and incorrect predictions contribute proportionally towards the performance of an algorithm. Smaller the logarithmic loss better are the results.

Area under ROC curve An receiving operative curve (ROC) curve is used as a performance metric in case of binary classification (Fig. 5). Perfect model predictions are given by models with an area of 1.0; while as 0.5 area represents average model score. ROC is actually given by specificity and sensitivity Table 6.

Confusion matrix Confusion matrix is another important metric for measuring the accuracy of a model with two or more than two classes. In a confusion matrix model predictions are provided in the form of a table. Cells inside the

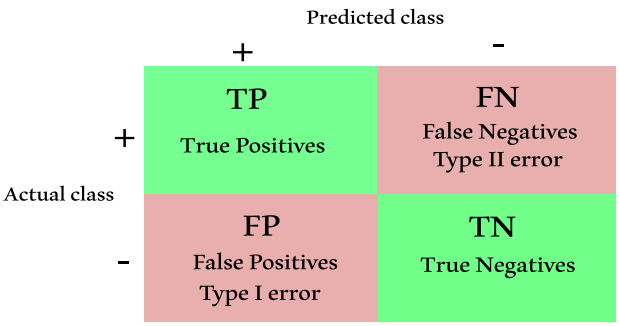

Fig. 4 Outcomes of a classifier

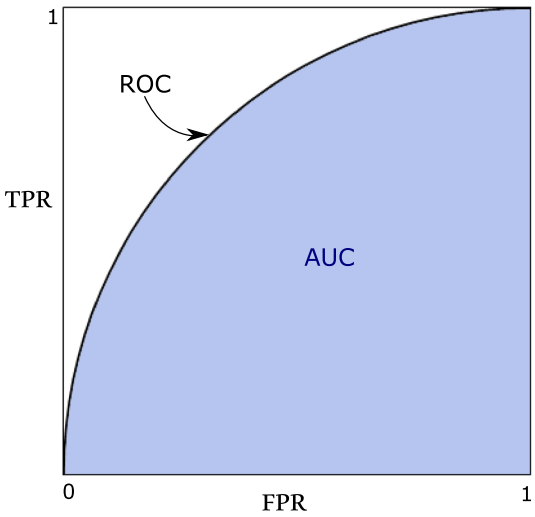

Fig. 5 Receiving operative curve

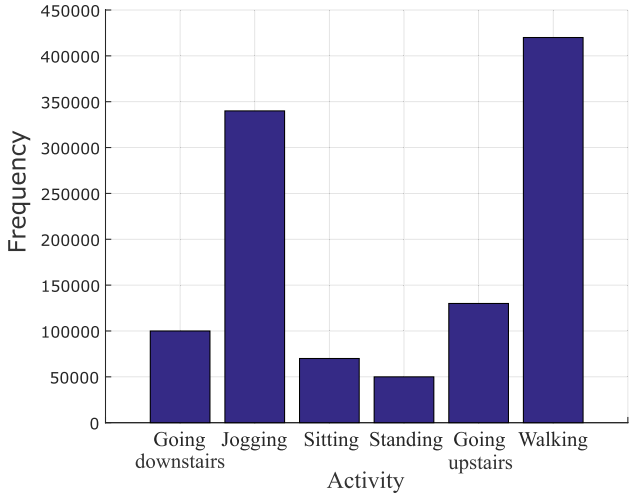

Fig. 6 Data distribution

Table 6 ROC metrics

Metric	Formula	Equivalent
False Positive Rate FPR	$FP/(TN+FP)$	Specificity
True Positive Rate TPR	$TP/(TP+FN)$	Recall, sensitivity

table provide number of predictions made by the classifier and diagonal cells represent the correct predictions.

We used publically available dataset[1] provided by Wireless Sensor Data Mining (WISDM) lab for our evaluation. The data was collected in a controlled environment where users performed different activities. The activities included jogging, walking, moving up-stairs, moving down-stairs, sitting and standing. 36 different users carried their smart phones in their right trouser pockets and the sensor (accelerometer) data was sampled at 20 values per second. Figure 6 shows the data distribution based on the available activity dataset.

The time series data is segmented using a sliding window technique with a window size of 90 data points. The segmented window size equals 4.5 s of continuous sensor data, as we are moving each time by 45 point the step size is 2.25 seconds i.e. 50% overlap. The acceleration values are normalized to have zero mean and unit standard variance for CNN-based approach. To compute the gradients for our deep neural network, parameters were optimized with stochastic gradient descent using back propagation algorithm. The Hyper-parameter values used in the experiment are shown in the Table 7.

[1] AIM '94 (1994). Diabetes Data Set [online]. Website https://archive.ics.uci.edu/ml/datasets/Diabetes [accessed 22 November 2018].

The test data is separated by leaving a one-subject out from the rest of the training data. From the test data, transfer instances were built by randomly taking the labeled instances for each activity and finally the classification layer were trained. To establish some baseline results we trained two other classifiers; Random Forest (RF) and Support Vector Machines (SVM) on the same dataset and their confusion matrix along with their evaluation metric tables are presented in Tables 8, 9, 10, and 11, respectively. Optimal parameters Table 12 for SVM, RF classifiers were selected by using cross validation. Tables 1 and 2 shows the confusion matrix and corresponding metrics of evaluation for our CNN model. Figure 7 shows the average performance of all the classifiers over all possible scenarios and Table 13 shows comparison of the classifiers with respect to different metrics. As presented in Fig. 7 proposed CNN architecture demonstrates the strong advantage over the baseline methods used for this task of activity recognition.

5 Conclusion

In this paper, we proposed a solution for AR problem using CNN architecture with 1D or Temporal Convolution and max pooling layers. CNN learns the features automatically from the accelerometer time series data and predicts the activity with an accuracy of around 90%. To reduce the computational training costs for a new user the features stored by the lower layers are preserved by freezing the lower layers and training the upper classification layer with few instances. Due to the robust architecture it has a small running time but further experiments needs to be carried out with larger datasets and on different devices. Future studies will include data from other sensors to learn the more complex features.

Table 7 Hyperparameter values used in experimentation

No. of training epochs	Batch size	Learning rate	Training loss	Training accuracy	Testing accuracy
8	10	0.0001	0.513	0.863	0.854
8	10	0.001	7.461	0.870	0.854
8	20	0.0001	20.653	0.868	0.860
8	20	0.001	4.512	0.785	0.782
15	10	0.0001	0.475	0.890	0.879
15	10	0.001	0.131	0.792	0.776
15	20	0.0001	1.487	0.896	0.884
15	20	0.001	0.391	0.867	0.853

Table 8 Confusion matrix for SVM

	Downstairs	Jogging	Sitting	Standing	Upstairs	Walking
Downstairs	**592**	37	0	3	21	48
Jogging	48	**1802**	0	2	106	42
Sitting	2	0	**335**	36	5	15
Standing	2	0	7	**260**	14	5
Upstairs	48	31	0	9	**602**	98
Walking	23	75	3	9	174	**2418**

Table 9 Metric evaluation for SVM

	Downstairs	Jogging	Sitting	Standing	Upstairs	Walking
TP	592	1802	335	260	602	2418
TN	6047	4728	6468	6572	5763	3962
FN	123	143	10	58	320	208
FP	109	198	58	28	186	283
Precision	0.8445	0.9012	0.8524	0.9027	0.7639	0.8952
Recall	0.8279	0.9264	0.9710	0.8176	0.6529	0.9207
F-score	0.8361	0.9135	0.9078	0.8580	0.7040	0.9078

Table 10 Confusion matrix for Random Forest

	Downstairs	Jogging	Sitting	Standing	Upstairs	Walking
Downstairs	**597**	41	0	2	12	49
Jogging	57	**1789**	0	0	125	29
Sitting	0	0	**318**	43	12	20
Standing	9	0	0	**266**	11	2
Upstairs	68	14	0	18	**590**	98
Walking	22	101	25	0	196	**2357**

Table 11 Metric evaluation for Random Forest algorithm

	Downstairs	Jogging	Sitting	Standing	Upstairs	Walking
TP	597	1789	318	266	590	2357
TN	6014	4715	6453	6520	5727	3972
FN	156	156	25	63	356	198
FP	104	211	75	22	198	344
Precision	0.8516	0.8945	0.8091	0.9236	0.7487	0.8726
Recall	0.7928	0.9197	0.9271	0.8085	0.6236	0.9225
F-score	0.8211	0.9069	0.8641	0.8622	0.6805	0.8968

Table 12 Parameters for SVM, RF classifiers

SVM	rbf kernel	$\gamma = 0.001$	C = 1000
RF	Estimators=1000	Depth = 100	Maxfeatures=Log2

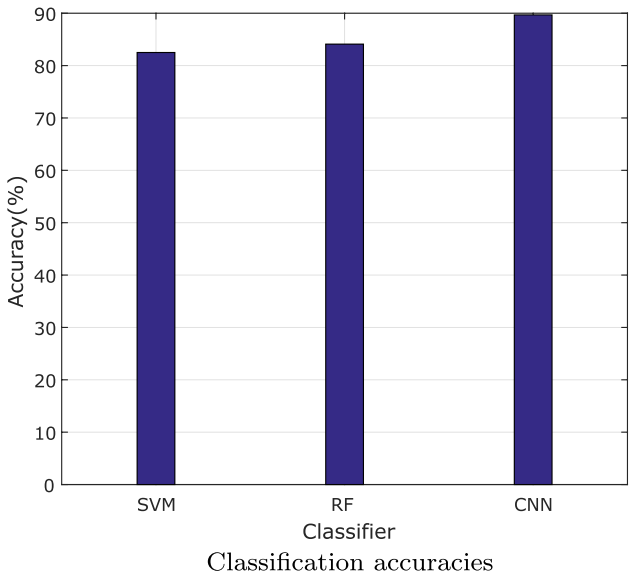

Fig. 7 Classification accuracies

Table 13 Comparative analysis

	Accuracy	Error	Precision	Recall	F1-score	Cohen's Kappa
CNN-TL	0.904	0.096	0.900	0.891	0.895	0.869
CNN	0.882	0.117	0.870	0.861	0.864	0.839
SVM	0.874	0.125	0.859	0.852	0.854	0.829
RF	0.861	0.138	0.850	0.832	0.838	0.812

Funding This research did not receive any specific grant from funding agencies in the public,commercial, or not-for-profit sectors.

Declarations

Conflict of interest The Authors declare that there is no conflict of interest.

References

AM (2013) Recognizing physical activities using wii remote. Int J Inf Educ Technol 3(1):60

Angelov PP, Zhou X (2008) Evolving fuzzy-rule-based classifiers from data streams. IEEE Trans Fuzzy Syst 16(6):1462–1475

Angelov P, Filev DP, Kasabov N (2010) Evolving intelligent systems: methodology and applications. Wiley, Hoboken

Bagct U, Bai L (2007)A comparison of daubechies and gabor wavelets for classification of MR images. In: 2007 IEEE International Conference on Signal Processing and Communications. IEEE, 2007

Bao L, Intille SS (2004) Activity recognition from user-annotated acceleration data. In: International Conference on Pervasive Computing. Springer, pp 1–17

Bengio Y (2009) Learning deep architectures for ai. Found Trends® Mach Learn 2(1):1–127

Bhattacharya S, Nurmi P, Hammerla N, Plötz T (2014) Using unlabeled data in a sparse-coding framework for human activity recognition. Pervasive Mob Comput 15:242–262

Brezmes T, Gorricho J-L, Cotrina J (2009) Activity recognition from accelerometer data on a mobile phone. In: International Work-Conference on Artificial Neural Networks. Springer, pp 796–799

Chennuru S, Chen P-W, Zhu J, Zhang JY (2010) Mobile lifelogger–recording, indexing, and understanding a mobile user's life. In: International Conference on Mobile Computing, Applications, and Services. Springer, pp 263–281

Choudhary P, Hazra A (2019) Chest disease radiography in twofold: using convolutional neural networks and transfer learning. Evol Syst 1–13

Figo D, Diniz PC, Ferreira DR, Cardoso JM (2010) Preprocessing techniques for context recognition from accelerometer data. Pers Ubiquit Comput 14(7):645–662

Huynh T, Schiele B (2005) Analyzing features for activity recognition. In: Proceedings of the 2005 joint conference on Smart objects and ambient intelligence innovative context-aware services: usages and technologies-sOc-EUSAI 05. ACM Press

Krause A, Siewiorek D, Smailagic A, Farringdon J (2003) Unsupervised, dynamic identification of physiological and activity context in wearable computing. In: Seventh IEEE International Symposium on Wearable Computers, 2003. Proceedings. IEEE

Krizhevsky A, Sutskever I, Hinton GE (2017) ImageNet classification with deep convolutional neural networks. Commun ACM 60(6):84–90

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

Lowe D (1999) Object recognition from local scale-invariant features. In: Proceedings of the Seventh IEEE International Conference on Computer Vision. IEEE, 1999

Ordonez FJ, Roggen D (2016) Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. Sensors 16(1):115

Peterek T, Penhaker M, Gajdoš P, Dohnálek P (2014) Comparison of classification algorithms for physical activity recognition. In: Innovations in bio-inspired computing and applications. Springer, pp 123–131

Plotz T, Hammerla NY, Olivier P (2011) Feature learning for activity recognition in ubiquitous computing. In: IJCAI Proceedings-International Joint Conference on Artificial Intelligence, vol. 22, no. 1, 2011, p 1729

Reddy S, Mun M, Burke J, Estrin D, Hansen M, Srivastava M (2010) Using mobile phones to determine transportation modes. ACM Trans Sens Netw (TOSN) 6(2):13

Rokni SA, Ghasemzadeh H (2017) Synchronous dynamic view learning: a framework for autonomous training of activity recognition models using wearable sensors. In: Proceedings of the 16th ACM-IEEE International Conference on Information Processing in Sensor Networks - IPSN. ACM Press

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

Ronaoo CA, Cho S-B (2015) Evaluation of deep convolutional neural network architectures for human activity recognition with smartphone sensors, pp 858–860

Sharma A, Lee Y-D, Chung W-Y (2008) High accuracy human activity monitoring using neural network. In: Convergence and Hybrid Information Technology, 2008. ICCIT'08. Third International Conference on, vol. 1. IEEE, 2008, pp 430–435

Tang Y, Salakhutdinov R, Hinton G (2012) Robust boltzmann machines for recognition and denoising. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, jun 2012

Thomas Platz NYH, Olivier P (2012) Feature learning for activity recognition in ubiquitous computing. IJCAI

Vollmer C, Gross H-M, Eggert JP (2013) Learning features for activity recognition with shift-invariant sparse coding. In: International conference on artificial neural networks. Springer, pp 367–374

Wang Z, Chen Y (2017) Recognizing human concurrent activities using wearable sensors: a statistical modeling approach based on parallel hmm. Sens Rev 37(3):330–337

Wu P, Zhu J, Zhang JY (2013) Mobisens: a versatile mobile sensing platform for real-world applications. Mob Netw Appl 18(1):60–80

Wu P, Peng H-K, Zhu J, Zhang Y (2011) Senscare: Semi-automatic activity summarization system for elderly care. In: International Conference on Mobile Computing, Applications, and Services. Springer, pp 1–19

Yang J, Nguyen MN, San PP, Li X, Krishnaswamy S (2015) Deep convolutional neural networks on multichannel time series for human activity recognition. Ijcai 15(2015):3995–4001

Yuan G, Wang Z, Meng F, Yan Q, Xia S (2018) An overview of human activity recognition based on smartphone. Sens Rev