Human Activity Recognition using CNN

A thesis

submitted in partial fulfilment of the requirement for the

Degree of

Master of Computer Science and Engineering

of

Jadavpur University

By

MIRAN HADI

Registration No.: 154140 of 2020-21

Class Roll No.: 002010502016

Exam Roll No.: M4CSE22016B

Under the Guidance of

Prof. Susmita Ghosh

Department of Computer Science and Engineering

Jadavpur University, Kolkata-700032

India

2022

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

Certificate of recommendation

This is to certify that the dissertation entitled " **Human Activity Recognition using CNN**" been carried out by **Miran Hadi** (University Registration No.- **1541410 of 2020-2021**, Class Roll No. **002010502016,** Exam Roll No.: **M4CSE22016B**) under my guidance and supervision and to be accepted in partial fulfilment of the requirement for the Degree of **Master of Computer Science and Engineering** the research results presented in this thesis have not been included in any other papers submitted for the award of any degree in any other Universities or institutes

_____

Prof. Dr. Susmita Ghosh (Thesis Supervisor)
Professor, Dept. of Computer Science & Engineering
Jadavpur University Kolkata 700032

Countersigned

_____

Prof. Nandini Mukherjee
Head, Department of Computer Science and Engineering
Jadavpur University Kolkata 700032

_____

Prof. Bhaskar Gupta
Dean, Faculty of Engineering and Technology
Jadavpur University Kolkata 700032

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

Certificate of Approval*

This is to certify that the thesis entitled "**Human Activity Recognition using CNN**" is a bonafide record of work carried out by **Miran Hadi** in partial fulfilment of the requirements for the award of the degree of **Master of Computer Science and Engineering** in the Department of Computer Science and Engineering, Jadavpur University during the period of July 2020 to November 2022. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed OR conclusion drawn therein but approve the thesis only for the purpose for which it has been submitted.

……

Signature of Examiner

Date:

……

Signature of Supervisor

Date:

*Only in case the thesis is approved

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

Declaration of Originality and Compliance of Academic Ethics

I hereby declare that this thesis entitled "**Human Activity Recognition using CNN**" contains literature survey and original research work by the undersigned candidate, as part of her Degree of **Master of Computer Science and Engineering**. All information has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all materials and results that are not original to this work.

Name: Miran Hadi

Registration No: 154140 of 2020-2021

Class Roll No.: 002010502016

Exam Roll No.: M4CSE22016B

Thesis Title: Human Activity Recognition using CNN

…..……..

Signature with Date

Acknowledgement

I would like to start by thanking my parents, by brother and my teachers for helping me deploy all the right resources and for shaping me into a better human being.

I would like to express my deepest gratitude to my advisor, Prof. Susmita Ghosh, Department of Computer Science and Engineering, Jadavpur University for her admirable guidance, care, patience, mental support and for providing me with an excellent atmosphere for doing research.

My thanks to Mr. Rahul Roy, Research Fellow, Department of Computer Science and Engineering. I am deeply grateful to them for the discussions that helped to enrich the content of this thesis.

I am also thankful to Laboratory for giving me the proper laboratory facilities for carrying out my work. This thesis would not have been completed without the inspiration and support of a number of wonderful -- my thanks and appreciation to all of them for being part of this journey and making this thesis possible.

……..

Miran Hadi

Registration No: 154140 of 2020-21

Class Roll No.: 002010502016

Exam Roll No.: M4CSE22016B

Department of Computer Science & Engineering

Jadavpur University

Summary

In recent years, human movement analysis in real-life settings outside the laboratory has experienced increasing attention in sports and medical applications. At the same time, wearable sensors have been evolved as valuable tools. This has allowed to acquire large-scale human movement data that are typically complex, heterogeneous, and noisy. In this context, modelling approaches are required, as the measured quantities often do not directly reflect meaningful biomechanical variables. More recently, machine learning methods have emerged as promising modelling tools that exploit unstructured data for estimating relevant target variables, such as joint kinematics and dynamics.

Although research in this field is still going on, there is a great potential not only to enlarge the range of numerous applications in sports but also to obtain biomechanical measures used to infer the load on body structures, such as joint forces. This applies in particular to unique sports such as ice hockey skating. Furthermore, little research has been conducted with respect to the direct estimation of biomechanical surrogate measures for knee joint load (i.e., joint dynamics) using wearable technology. This is of paramount importance, as ambulatory joint load assessment can improve health diagnostics and rehabilitation of injuries as well as musculoskeletal diseases. The overall aim of this thesis is to assess how, and to what extent, accelerometer sensor data and machine learning techniques can biomechanically quantify sports performance and the load on body structures during the execution of everyday and sport movements.

This work proposes a conventional neuron network (CNN) for successful human activity recognition using Accelerometer. Various experiments are performed on a real-world wearable sensor dataset to verify the effectiveness of the deep learning algorithm. The results show that the proposed that CNN 1D model performs comparison with other CNN 2D model and achieves satisfactory activity recognition performance. Some open problems and ideas are also presented and should be investigated as future research.

List of Figures

List of Tables

CONTENTS

Table of contents Page No.

CHAPTER 1

INTRODUCTION

1.1 Human Activity Recognition

The study and prediction of human movement can be traced back along a fascinating trajectory, to the ancient Greeks, when the famous philosopher Aristotle left documented records about human walking (Harris & Smith (2007)). Skipping forward to the 19th century, the Weber brothers analysed gait temporal and stride parameters using experimental measurements. In their published work, they studied human gait in both mechanical and physiological aspects (Kisner et al. (2017)). In recent decades, the study of human motion has increased exponentially, thanks to the development of video camera systems Gavrila & Davis (1996); Karaulova et al. (2002); Allard et al. (1998); Zatsiorsky & Zaciorskij (2002)). Such camera systems, with or without force-sensing platforms, make the study of human movement feasible and accurate. They are often combined with electromyographic (EMG) sensors and other hardware, and can provide important information about the causal relationship between muscle activation and resulting human movement, as well as about deviating gait patterns that may arise in persons with sensorimotor disorders. They do, however, generally require specialized equipment and are often installed in a lab, which can limit their applicability in some contexts. On this note, other wearable sensors have become prevalent in the past few decades, and can complement or even replace video camera systems in some settings, making it possible to study human motion in more versatile settings (Takeda et al. (2009); Reenalda et al. (2016); Seel et al. (2014b)). These sensors are generally cost-effective, non-obtrusive, and can be worn or attached to the body, inside or outside of a lab environment. They often include inertial measurement units (IMUs), which contain accelerometers, and gyroscopes and magnetometers, to measure segment acceleration and angular velocities, as well as electromyographic (EMG) sensors to measure muscle excitation and foot switches, which detect contact between the foot and the ground. These wearable sensors can be donned simultaneously and capture data at a high sampling frequency, resulting in a large amount of data. Machine learning (ML) is an effective tool to extract prominent features of large datasets and make statistical inference from these data. ML has a great capability of

solving classification, regression, and decision-making problems, and as such, can enhance motion analyses through data-driven prediction.

Human Activity Recognition (HAR) aims to identify the actions carried out given a set of observations of a person and his/her surrounding environment. Recognition can be accomplished by exploiting the information retrieved from various sources such as environmental or body-worn sensors. Some approaches have adapted dedicated motion sensors to fit different human body parts such as waist, wrist, chest and thighs. They have achieved great classification performance. However, these sensors usually make a common user not that comfortable and do not provide a long-term solution for activity monitoring, due to such issues, as sensor repositioning after dressing.

HAR has become an attractive research field due to its importance as well as many challenges brought to the research community. Researchers use these HAR systems as a medium to get information about people's behaviours. The information is commonly collected from the signals of sensors such as ambient and wearable sensors. The data from the signals are then processed through machine learning algorithms and recognizes the events. Hence, such HAR systems can be applied in plenty of useful and practical applications in smart environments such as smart home health-care systems. For example, a smart HAR system can continuously observe patients for health diagnosis and medication. Also, it can be applied for automated surveillance of public places to predict crimes that may occur in the near future.

Human activity recognition, or HAR, is a challenging time series classification task. It involves predicting the movement of a person based on sensor data and traditionally involves deep domain expertise and methods from signal processing to correctly engineer features from the raw data in order to fit a machine learning model. Below figure have the six different activities.

Fig 1. Example of the activities

1.2 Wearable Sensors

Since the appearance of the first commercial hand-held mobile phones in 1979, it has been observed an accelerated growth in the mobile phone market. Mobile devices have almost become easily accessible to virtually everybody now. Smartphones, which are a new generation of mobile phones, are now offering many other features such as multitasking and the deployment of a variety of sensors, in addition to the basic telephony. Current efforts attempt to incorporate all these features while maintaining similar battery lifespans and device dimensions. The integration of these mobile devices in our daily life is rapidly growing. It is envisioned that such devices can seamlessly keep track of our activities, learn from them, and subsequently help us to make better decisions regarding our future actions.

Smartphones have been bringing up new research opportunities for human centred applications where the user is a rich source of context information and the phone is the first-hand sensing tool. Latest devices come with embedded built-in sensors such as microphones, dual cameras, accelerometers, gyroscopes, etc. The use of smartphones with inertial sensors is an alternative solution for HAR. These mass marketed devices provide a flexible, affordable and self-contained solution to automatically and unobtrusively monitor Activities of Daily Living (ADL) while also providing telephony services. Consequently, in the last few years, some works aiming to understand human behaviour using smartphones have been proposed. For instance, one of the first approaches has been exploited an Android smartphone for HAR employing its embedded triaxial accelerometers. Improvements are still expected in topics such as in multi-sensor fusion for better HAR classification, standardizing performance evaluation metrics, and providing public data for evaluation.

Currently, smartphones, wearable devices, and internet-of-things (IoT) are becoming more affordable and ubiquitous. Many commercial products, such as the Apple Watch, Fitbit, and Microsoft Band, and smartphone apps including Runkeeper and Strava, are already available for continuous collection of physiological data. These products typically contain sensors that enable them to sense the environment, have modest computing resources for data processing and transfer, and can be placed in a pocket or purse, worn on the body, or installed at home [4]. Accurate and meaningful interpretation of the recorded physiological data from these devices can be applied potentially to HAR. However, most current commercial products only provide relatively simple metrics, such as step count or cadence. The emergence of deep learning methodologies that extract different discriminating features from the data, and increased processing capabilities in wearable technologies. The ability of simultaneous activity classification and the decreasing size of computing platforms give rise to the possibility of

performing detailed data analysis in situ and in real time. Today's handheld PCs are often more powerful than desktop computers of the 1990s. Once a rare commodity, computers are now embedded in everything - toys, cars, cell phones, and even bread makers.

In the case of wearable sensors in activity recognition, a smartphone is an alternative to them due to the support of the diversity of sensors in it. Handling sensors such as accelerometers and gyroscopes along with the device with wireless communication capabilities made smartphones a very useful tool for activity monitoring in smart homes. Besides, smartphones are very ubiquitous and require almost no static infrastructure to operate it. This advantage makes it more practically applicable than other ambient multi-modal sensors in smart homes. As recent smart phones consist of inertial sensors (e.g., gyroscopes and accelerometers), they can be appropriate sensing resources to obtain human motion information for HAR.

HAR has been actively explored based on a distinguished kind of ambient and wearable sensors. Some instances of such sensors include motion, proximity, microphone, and video sensors. Most of the ambient sensor-based latest HAR researchers have mainly focused on video cameras as cameras make it easy to retrieve the images of surrounding environment. Video sensors are included with some other prominent sensors in some work related to novel ubiquitous applications. Though video sensors have been very popular for basic activity recognition. They face very many difficulties for ordinary people to accept due to a privacy issue. On the contrary, wearable sensors such as inertial sensors can overcome this kind of privacy issues and hence, deserve more focus for activity recognition in smart homes.

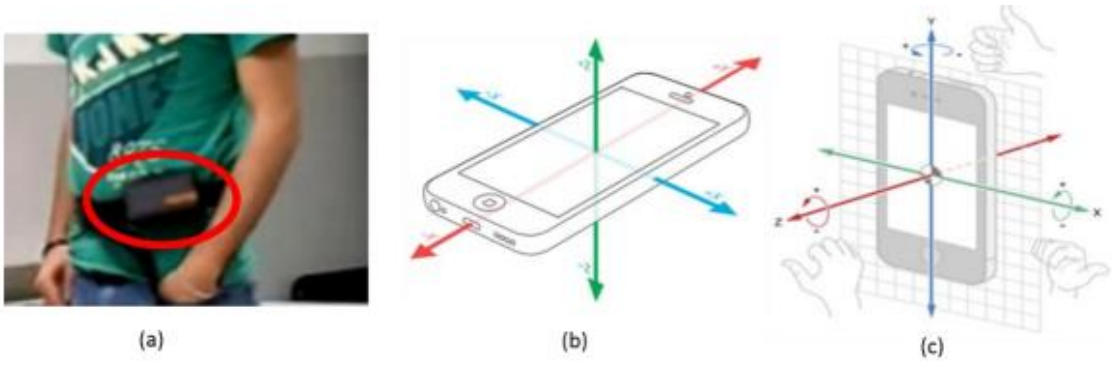


Figure 2. Placement map of smartphone and its sensors

In the past years, many HAR systems used accelerometers to recognize a big range of daily activities such as standing, walking, sitting, running, and lying. For instance, some researchers have already explored the accelerometer data to find out the repeating activities such as grinding, filling, drilling, and sanding. The others, have performed elderly peoples' fall detection and prevention in smart environments. Majority of the afore mentioned systems adopted many

accelerometers fixed in different places of a human body. However, this approach apparently not applicable to daily life to observe long-term activities due to attachment of many sensors in the human body and cable connections. Some studies tried to explore the data of single accelerometers at sternum or waist. These studies have reported substantial recognition results of basic daily activities such as running, walking, and lying.

Public domain UCI data set. Experiments were carried out with a group of 30 volunteers aged 19–48 years. Each person performed six activities (WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING, LAYING_DOWN) wearing a smart phone (Samsung Galaxy S II) on the waist. Using its embedded accelerometer and gyroscope, we were able to make three-axial linear acceleration and three-axial angular velocity available at a constant rate of 50 Hz and trimmed into windows of 128-time steps for a 2.56 seconds windows; this was enough to capture two steps, in the case of walking, for the classification. For x,y and z axis, while user performs six different activities in a controlled environment. These activities include

1. WALKING,
2. WALKING_UPSTAIRS,
3. WALKING_DOWNSTAIRS,
4. SITTING,
5. STANDING,
6. LAYING_DOWN

Below is a video link shows how the above data has be created

Activity Recognition Experiment Using Smartphone Sensors. - YouTube

1.3 Deep Learning

Deep learning is a paradigm of machine learning that uses multiple processing layers to infer and extract information from a large scale of data. Many studies have shown that the use of deep learning can achieve better performances in a range of applications than traditional approaches. Traditional approaches use a set of selected features, also known as “shallow” features, to represent the data for a specific classification task. HAR can be accomplished, for

example, by exploiting the information retrieved from inertial sensors such as accelerometers. In some smartphones these sensors are embedded by default and we benefit from them to classify a set of physical activities (standing, walking, laying, walking, walking upstairs and walking downstairs) by processing inertial body signals through a supervised Machine Learning (ML) algorithm for hardware with limited resources.

1.3.1 Convolution Neural Networks

Popular deep learning approaches include convolutional neural nets (CNN), an deep learning, a **convolutional neural network** (**CNN**, or **ConvNet**) is a class of artificial neural network (**ANN**), most commonly applied to analyse visual imagery's are also known as **Shift Invariant** or **Space Invariant Artificial Neural Networks** (**SIANN**), based on the shared-weight architecture of the convolution kernels or filters that slide along input features and provide translation-equivariant responses known as feature maps. Counter-intuitively, most convolutional neural networks are not invariant to translation, due to the down sampling operation they apply to the input. They have applications in image and video recognition, recommender system, image classification natural language processing financial time series and many other.

Convolutional neural networks are a specialized type of artificial neural networks that use a mathematical operation called convolution in place of general matrix multiplication in at least one of their layers. They are specifically designed to process pixel data and are used in image recognition and processing.

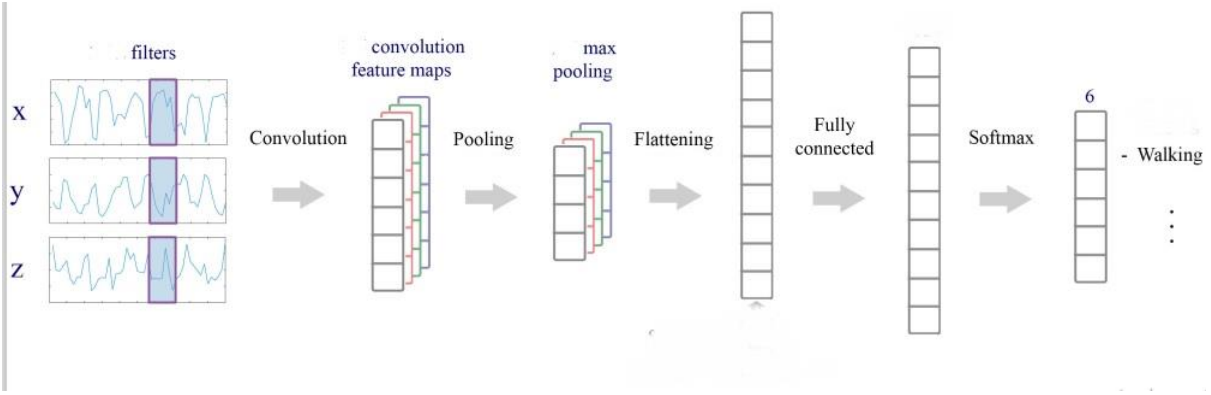


Figure 3: Example of CNN model

1.4 Other Approaches

Recently, smartphones have attracted many activity recognition researchers as they have fast processing capability, and they are easily deployable. For instance, some researchers use wirelessly connected smartphones to collect a user's data from a chest unit composed of the accelerometer and vital sign sensors. The data is later processed and analysed by using different machine learning algorithms.

In [4] of develop an HAR system to recognize five different kinds of transportation activities where data from smartphones inertial sensors are used with a mixture-of expert model for classification. [5] researchers proposed an offline HAR system where a smartphone with built-in triaxial accelerometer sensor is used. A phone is kept in the pocket during experiments. Some scientists also used a smartphone mounted in the waist to collect inertial sensors' data for activity recognition. They used Support Vector Machine (SVM) for activity modelling. In [7], a smartphone is used to recognize six different activities in real-time. Moreover, the researchers have proposed a real-time motion recognition system with the help of a smartphone with accelerometer sensors. [10,11] and [8] use a smartphone with an embedded accelerometer to recognize four different activities in real time.

The development of HAR applications using smartphones has several advantages such as easy device portability without the need for additional fixed equipment, and comfort to a user due to their unobtrusive sensing. This contrasts with other established HAR approaches which use specific-purpose hardware devices such as those in body sensor networks. Although the use of numerous sensors could improve the performance of a recognition algorithm, it is unrealistic to expect that the general public will use them in their daily activities because of the difficulty and the time required to wear them. One drawback of a smartphone-based approach is that energy and services on the mobile phone are shared with other applications and this become critical in devices with limited resources.

In [15], ML methods that are previously employed for pattern recognition include Naive 9 Bayes, and Support Vector Machines (SVMs). In particular, we make use of SVMs for classification as many other studies. Although it is not fully clear which method performs better for HAR, SVMs have confirmed their successful application in several areas including heterogeneous types of recognition such as intrusion detection, fault detection, handwritten character recognition and speech recognition. In ML, fixed-point arithmetic models [8] were

previously studied initially because devices with floating-point units were unavailable or expensive.

1.5. Objective and Aims

The objective of this thesis was to use Machine Learning to perform automatic prediction of human motion, with a focus on data-driven approaches and simulation, beneficial from wearable sensors data after passing it from frequency feature set. Therefore, thesis aims to investigate the feasibility of using wearable sensors and machine learning techniques to obtain meaningful biomechanical measures with respect to different performance of the body Movement.

Aims the six different Human body movement prediction using different CNN 1D models having different filters and kernels and CNN 2D models having different filters and kernels and then finding which CNN Model is best for prediction of human body movement.

.

CHAPTER 2

2.1. Study of Pervious Work

The earlier work investigated the possibilities to use accelerometers as main sensor data for human activity recognition. Many works demonstrated that the usage of dedicated accelerometers can provide good results in the area of activity recognition. The different investigations have one thing in common - multiple accelerometers were placed on different parts of the body, either wired or wireless, and users were required to perform designated movements. [14] The recorded accelerometer values were processed and the resulted features were evaluated using classification algorithms for potential recognition.

The ideas were expanded with the inclusion of additional sensor information. For example, in [8] a heart rate monitor was coupled with data taken from five accelerometers to detect physical activities. The team at Intel Research in Seattle and University of Washington used the multi-modal sensor board (MSB) that had accelerometer, audio, temperature, IR/visible/high-frequency light, humidity, barometric pressure and digital compass. In [6] investigated activity recognition classification of physical activities with multiple MSBs. The group in used a tri-axial accelerometer together with a wearable camera to recognize human activity. Note that these dedicated accelerometers used in the above work are capable of producing sampling rate more than 100Hz and are accurate up to ±10G.
In our tests, we managed to obtain sampling rates around 50Hz using a SAMSUNG smart mobile. Most of the above investigations used accelerometer sampling rates from 30Hz to 50Hz, except for and that sampled the accelerometer data at 76.25Hz and 93Hz respectively. The many research has obtained good results with recognition accuracies between 83% and 96%. These investigations have shown that by analysing the accelerometer data, systems were able to provide good recognition. However, these set ups required multiple accelerometers to be worn and observed. For a typical real user, it can be rather obtrusive and troublesome to wear multiple sensors or sensor boards at specific positions. This factor also partly motivated us to use a smartphone as a non-obtrusive sensor source. The inclusion of other sensors did give a slight improvement on accuracy rate, but two issues remain outstanding. Firstly, not all additions of different sensors improved accuracy

significantly. Secondly, additional sensors also mean more data to be processed and computed. Therefore, an appropriate selection of sensors that provide good recognition will make for the ideal solution. Recently, the three-dimensional accelerometer integrated in smartphones was also investigated as a potential sensor for movement recognition. In, the accelerometer of a Nokia N95 was used as a step counter. The results showed that such smartphones can provide accurate step-counts comparable to some of the commercial, dedicated step counter products, provided the phone is firmly attached to the body. The DiaTrace project uses a mobile phone with accelerometers for physical activity monitoring. The proof-of concept prototype obtained a recognition accuracy of >95% for activity types of resting, walking, running, cycling and car driving. Brezmes et. al. used also the accelerometer data collected with a Nokia N95 with K-nearest neighbour algorithm to detect common movements. These investigations mentioned that the accelerometer sampling rate on smartphones are generally lower as compared to dedicated sensors. The obtained results also showed the potential of smartphones being used for activity recognition. However, there was no comparison performed in identifying suitable algorithms, sampling rates, and features.

The results above have laid down a good foundation for the work here. For activity recognition using a commercial product such as smartphones, one of the many challenges is to find out the relevant criteria such as algorithm and feature extraction methods that will enable successful recognition and utilization of such context information.

CHAPTER 3

3.1. Methodology

This study aims to classify the HAR signals of the UCI HAR dataset employing a CNN model, as shown in Fig. 4. Like all the supervised ML techniques, this algorithm has two stages (i.e., training stage and testing stage). The training stage requires a set of data samples containing various attributes measured from subjects while performing various predefined activities. The supervised learning technique then try to make some “sense” out of the data, find out how the samples that belong to the same class are similar to each other while samples from different classes are diverse, then builds one or more internal models focusing on the crucial attributes that can highlight those contrasting properties to carry out the classification. However, merely feeding the raw data collected from the sensors into the classifier might not be a good idea, because more often than not these time-domain signals contain noise, interference, missing values, and most importantly, time-domain attributes are simply not good enough to make the distinguishable properties perceptible to the classifiers. That is why researchers spend so much time finding and selecting meaningful features from various types of real-life time varying signals, which is also known as feature engineering. Now, although some paper worked with that statistical features and acquired decent results, in this study, we are taking on a slightly different approach. We are extracting frequency features from the raw time-domain accelerometer signals and then feeding these two sets of samples into different CNN model. In the training stage, a preordained portion of the dataset is used to train the machine and build a feasible model, which is then evaluated over the remaining samples.

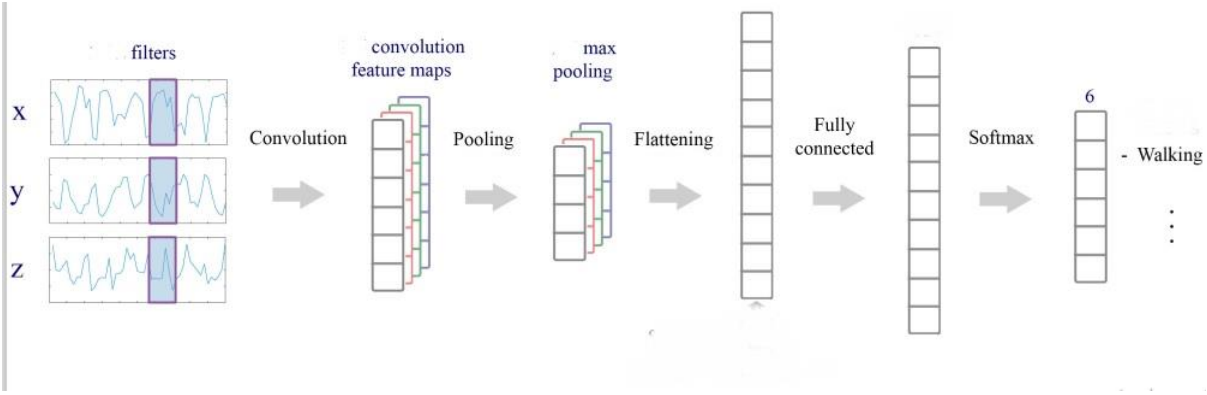
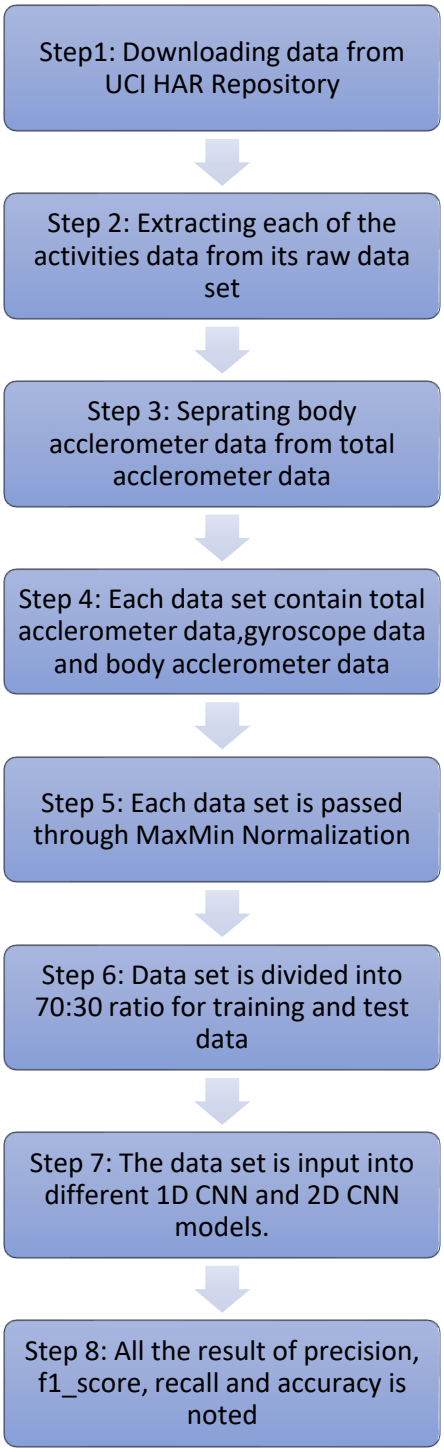


Figure.4 CNN Model

As shown in Fig. 4, we gathered x, y, and z acceleration signals and transformed them into vector magnitude data, and used the vector magnitude data to construct different CNN model for ternary activity classification.

Workflow for Human Activity Recognition using CNN

The workflow for the Human Activity Recognition using CNN is shown below



3.2 Data Set

Public domain UCI data set. Experiments were carried out with a group of 30 volunteers aged 19–48 years. Each person performed six activities (WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING, LAYING_DOWN) wearing a smart phone (Samsung Galaxy S II) of the different genders, heights and weights using a wrist mounted smartphone. Using its embedded accelerometer and gyroscope, action data was recorded using these sensors while each of the subjects was performing six predefined tasks, which, according to the jargon of ML, represent six different classes, we were able to make three-axial linear acceleration and three-axial angular velocity available at a constant rate of 50 Hz and trimmed into windows of 128-time steps for a 2.56 seconds windows; this was enough to capture two steps, in the case of walking, for the classification. The experiments were video-recorded to label the data manually and obtain balanced classes; the data were of high quality.

Below is a video link shows how the above data has be created

[Activity Recognition Experiment Using Smartphone Sensors. - YouTube]

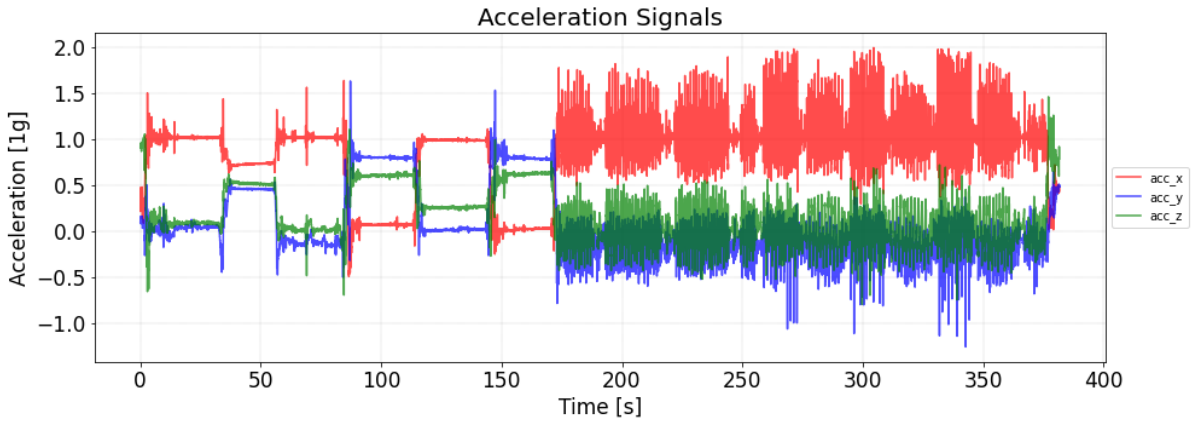


Figure 5: Signal View of Accelerometer Data of one user

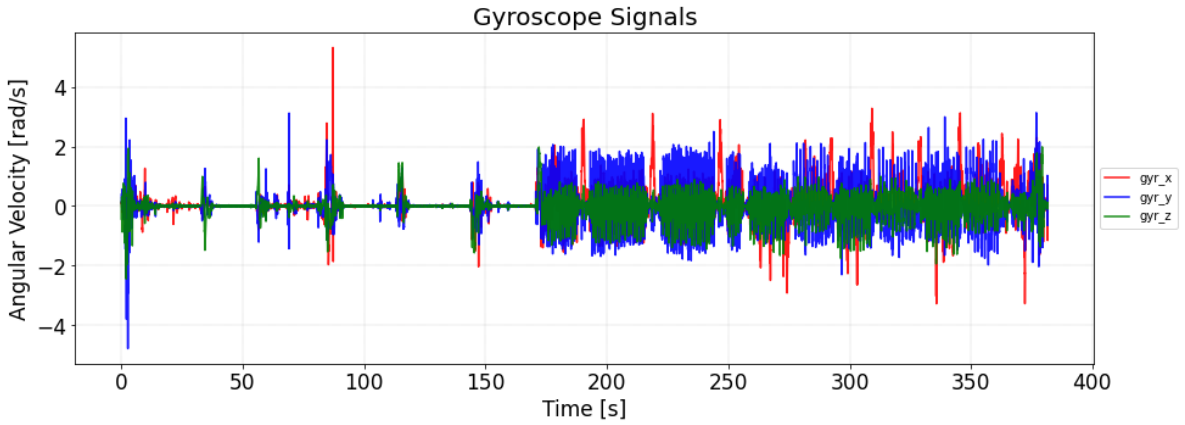


Figure 6:Signal View of gyroscope signal data of one users

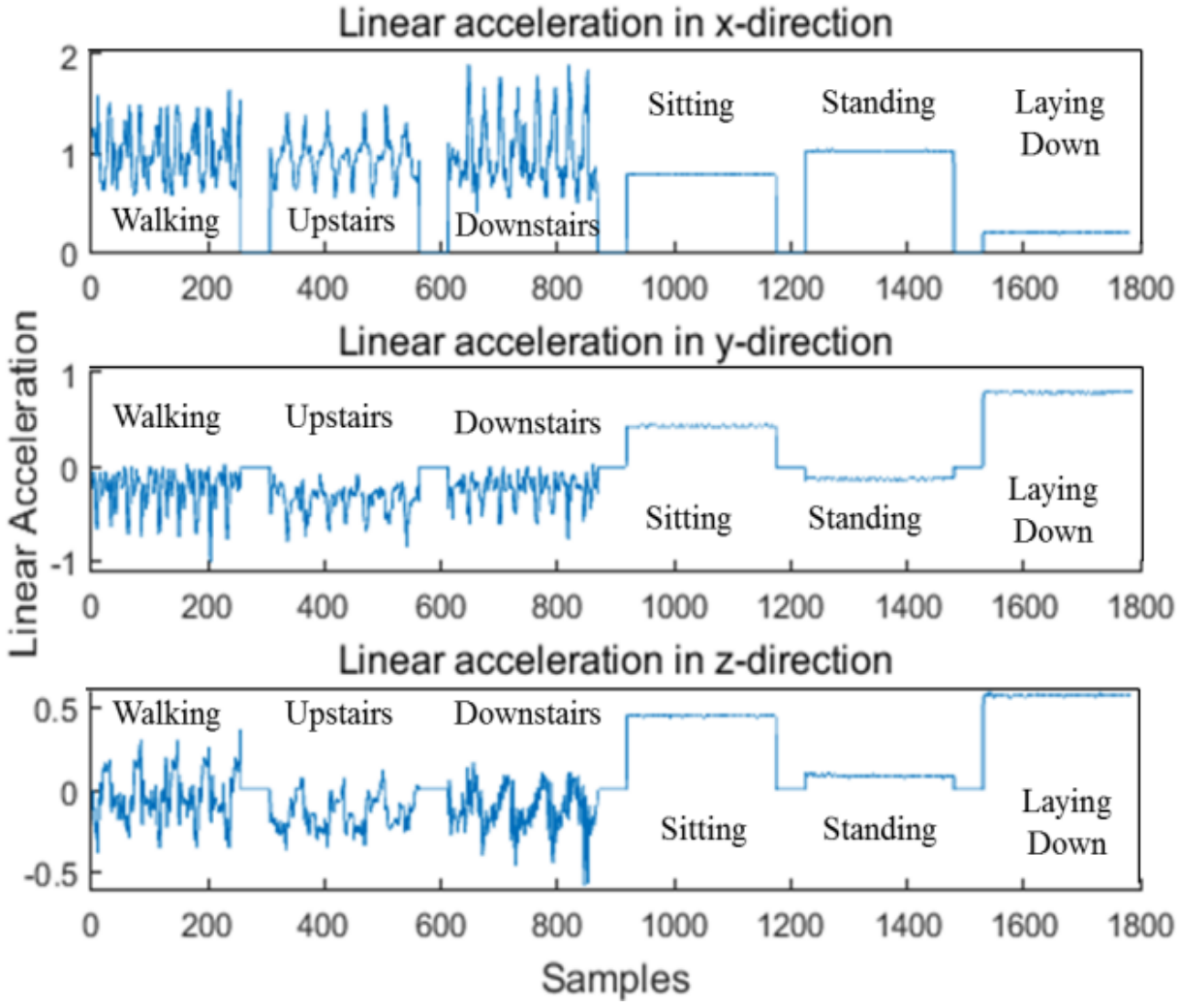


Figure 7: Sample of Signal View of Different Activities

The data set obtained was partitioned randomly into two sets: 70% of the volunteers were selected for generating the training data, and 30% were selected for generating the test data. Each sample had 561 linear (time-independent) hand-made, pre-processed features from

signal analysis (e.g., window's peak frequency), but only nine features were used in our study: triaxial gravity acceleration from the accelerometer and triaxial body acceleration and triaxial angular velocity from the gyroscope. These are raw signals with a time component and do not fall in the frequency domain but rather in the time domain. The sensor data were pre-processed by applying denoising median filters, clipping the approximately 20 Hz mark; they were then sampled in fixed-width sliding windows of 2.56 seconds. Those windows were provided with an overlap of 50% to ease training. Additionally, all features were pre-normalized and bounded within [-1, 1]. The available dataset contains 10,399 samples which are separated into two sets (i.e., a training set and a test set). The former one contains 7,279 samples (70%), whereas the latter one is comprised of the rest 3120 samples (30%). Table I provides more details about the contents of the dataset along with the class identifications and their labels.

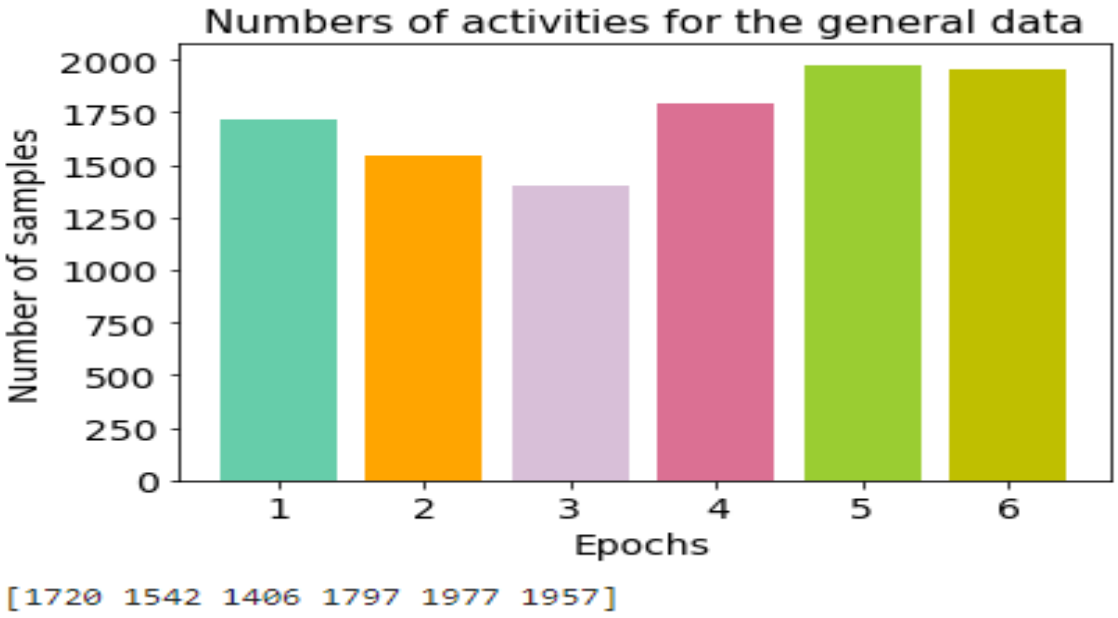


Figure 8: Number of Data for each Activities

3.3 Data Set Processing

Frequency Features From the perspective of the frequency contents, a Human action signal can be viewed as a comprised form of multiple sinusoidal signals of different frequencies. The frequency information of the human action signal simply refers to the values of those frequencies and the amplitudes of those signals in those frequencies. Digital Signal Processing (DSP) offers multiple methods to extract this information to form a HAR signal. In this study, we are going to use the well-known Fast Fourier Transform (FFT). The algorithm of FFT was developed by James Cooley and John Tukey in 1965 as a faster version of the then-popular Discrete Cosine Transform (DFT) to calculate the frequency components of a

time domain signal. If we consider a N-point time-series signal x(N), its N-point DFT is defined by:

$$X(N)_k = \sum_{n=0}^{N-1} x_n \; e^{\frac{-i2\pi kn}{N}}$$

where k=0,1,2,....N-1.FFT is an algorithm for computing the n-point DFT with a computational complexity of O(N logN).Fig.10 presents the graph of the HAR signals after extracting their frequency features using FFT. Then separating signal frequency to gravity signal and body signal. When the signal frequency is below 0.3Hz is gravity signal and the signal frequency above 0.3Hz is body signal It is noticeable that in this figure,

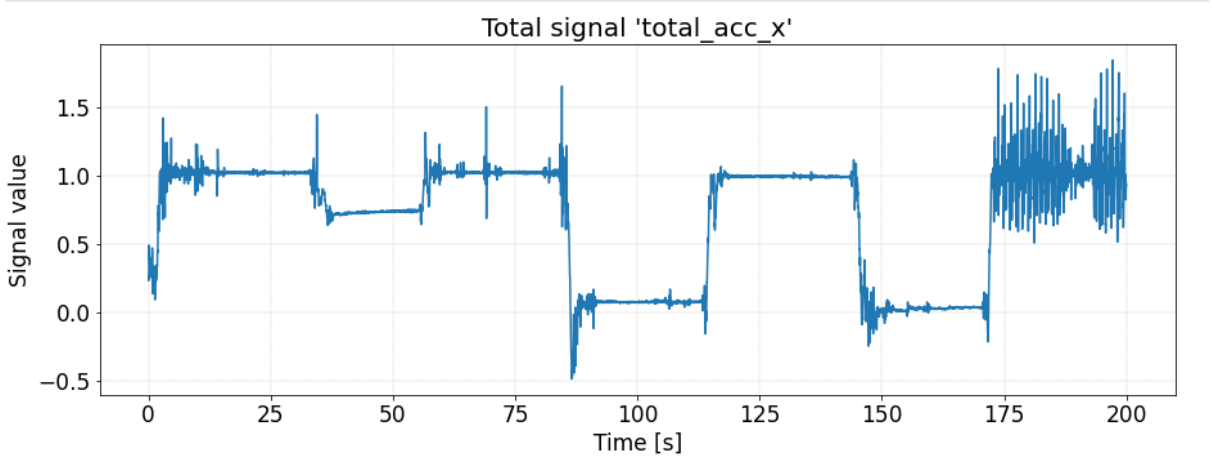


Figure 9: Show the Accelerometer x-axis data of a use

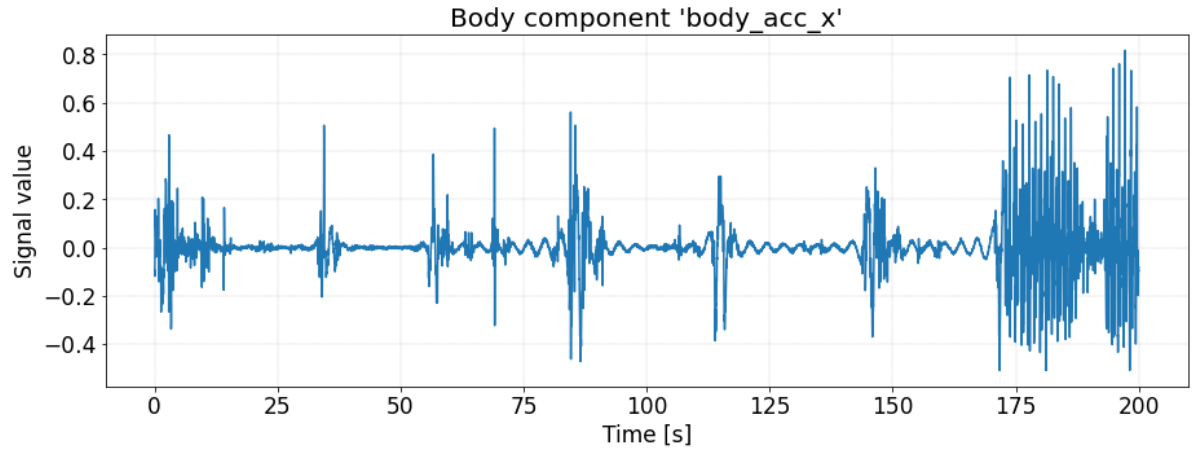


Figure 10: shows the body component after applying frequency feature

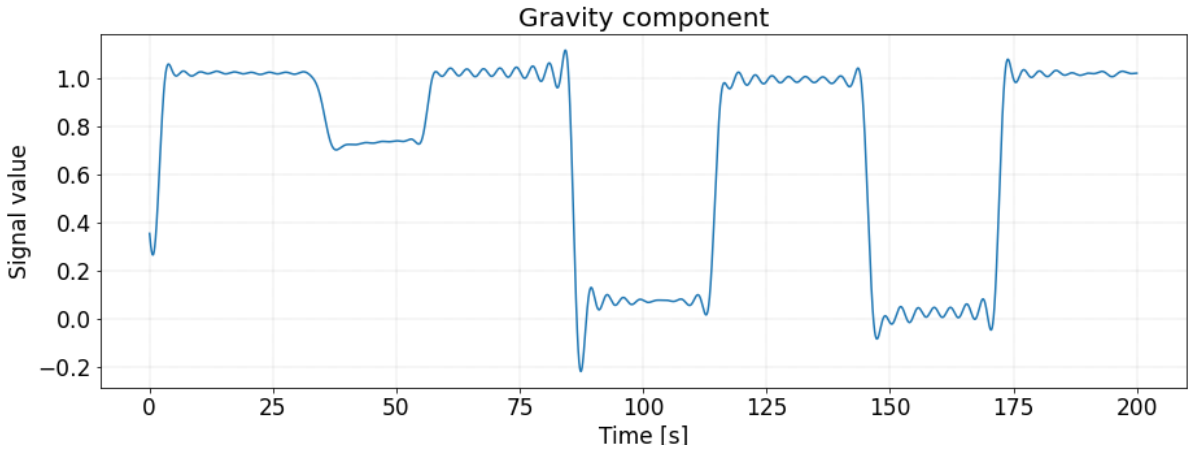


Figure 11: Gravity component after removal of body component

Operational Dataset Preparation

Each of the samples of the UCI HAR dataset contains a subject's body acceleration (Body_acc), Triaxial Angular velocity (Body_gyro) and total acceleration (Total_acc) data in three axes (namely X, Y, and Z) while performing an assigned activity. Fig. 7 shows how each set of data was processed individually to extract frequency and power information from them. After that, the frequency and power features of each signal were concatenated to form a complete feature set which is then done Normalization or Min-Max Scaling is used to transfer feature to similar scale. The new point is calculated as:

$$\text{X\_new} = \frac{x - x(min)}{x(\max) - x(\min)}$$

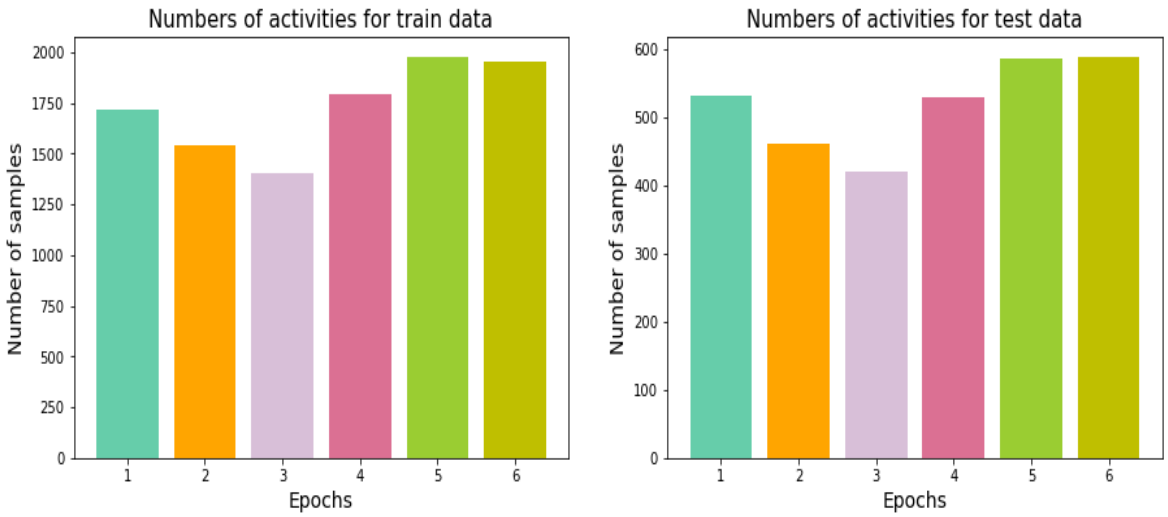
This scales the range to [0, 1] or sometimes [-1, 1]. Geometrically speaking, transformation squishes the n-dimensional data into an n-dimensional unit hypercube. Normalization is useful when there are no outliers as it cannot cope up with them.

Now the normalise feature set which represents the corresponding HAR signal in the classification stage. Finally, the label of the associated class is inserted at the terminal point of the signal.

Input

A fixed time-length accelerometer and gyroscope data in the form of vector magnitude data as explained above was used as input. The input feature set has 128x9 dimensions for one set of classification and in our experiment, there are 7279 set of training sample and 3120 set of test samples.

Below table show the number of different inputs



- Train data and test data are balanced together, so function train_test_split is enough to this project
- Getting 7279 samples of the train data and 3120 samples of test data
- Database is ready to use for

Figure 12: Data after dividing it for test and training data

3.4 CNN Architecture

CNNs are biologically inspired networks for processing data that has a known, grid-like topology. CNN works based on convolution, which is a mathematical operation that slides one function over another and measures the integral of their point-wise multiplication. Convolutional networks are neural networks that use convolution in lieu of general matrix multiplication in at least one of their layers. The idea behind the CNN was obtained from Hubel and Wiesel's interpretation of the operation of the cat's visual cortex, where they observed that specific portions of the visual field excite particular neurons. Although CCN was developed to work mainly with the images that have high dimensionality, they are equally effective on various types of analog and digital signals. CNN has been extensively used in speech recognition, audio processing, machine fault classification, and various types of biomedical

signal recognition such as Electrocardiogram (ECG), electroencephalogram (EEG), Electromyography (EMG). As stated before, we are checking CNNs models (1D an 2 D) in our classification model; all of them will process the frequency features set, as shown in Fig. 10. The basic parameters for all the models are the same.

A convolutional neural network consists of an input layer, hidden layers and an output layer. In any feed-forward neural network, any middle layers are called hidden because their inputs and outputs are masked by the activation function and final convolution. In a convolutional neural network, the hidden layers include layers that perform convolutions. Typically, this includes a layer that performs a dot product of the convolution kernel with the layer's input matrix. This product is usually the Frobenius inner product, and its activation function is commonly ReLU. As the convolution kernel slides along the input matrix for the layer, the convolution operation generates a feature map, which in turn contributes to the input of the next layer. This is followed by other layers such as pooling layers, fully connected layers, and normalization layers.

In a CNN, the input is a tensor with a shape: (number of inputs) × (input height) × (input width) × (input channels). After passing through a convolutional layer, the matrix becomes abstracted to a feature map, also called an activation map, with shape: (number of inputs) × (feature map height) × (feature map width) × (feature map channels).

Convolutional layers convolve the input and pass its result to the next layer. This is similar to the response of a neuron in the visual cortex to a specific stimulus. Each convolutional neuron processes data only for its receptive field. Although fully connected feed forward neural networks can be used to learn features and classify data, this architecture is generally impractical for larger inputs such as high-resolution images. It would require a very high number of neurons, even in a shallow architecture, due to the large input size of matrix, where each cell is a relevant input feature. Using regularized weights over fewer parameters avoids the vanishing gradients and exploding gradients problems seen during backpropagation in traditional neural networks. Furthermore, convolutional neural networks are ideal for data with a grid-like topology (such as images) as spatial relations between separate features are taken into account during convolution and/or pooling.

Pooling layers

Convolutional networks may include local and/or global pooling layers along with traditional convolutional layers. Pooling layers reduce the dimensions of data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. Local pooling combines small clusters, tiling sizes such as 2 × 2 are commonly used. Global pooling acts on all the neurons of the feature map.There are two common types of pooling in popular use: max and average. *Max pooling* uses the maximum value of each local cluster of neurons in the feature map,while *average pooling* takes the average value.

Fully connected layers

Fully connected layers connect every neuron in one layer to every neuron in another layer. It is the same as a traditional multilayer perceptron neural network (MLP). The flattened matrix goes through a fully connected layer to classify the images.

3.4.A. Network Architecture for 1D CNN

The network architecture of our 1D CNN consists of two convolutional layer, one max-pooling layer, then 50% dropout , then flattering the matrix, one deep connected layer with 125 units, and one softmax layer that outputs the probability of each of the six activities. Figure 13 shows the 1D CNN architecture of our proposed method.

1D CNN:

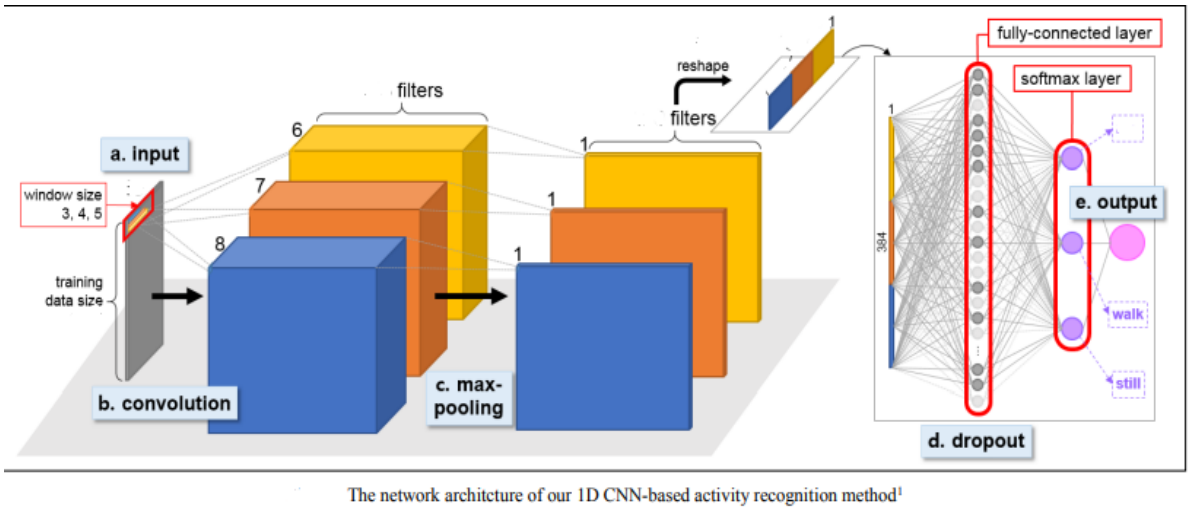
The 1D convolution operations were performed using 6 different Models having different filter and kernel size with two layers of 1D convolution layers having activation function as ReLU

Max-pooling: For each feature set of a given window size and filter type, 1-max-pooling was performed to select the largest feature value. In Figure 12-part c. max-pooling shows the max-pooled result after two convolution layers data as input. The max-pooled of size 2 is used in this architecture.

Dropout: The convolved and max-pooled feature set were then placed as the input to the fully connected neural network with dropout of 50% applied. The dropout was applied to prevent the neural network from overfitting. Figure12 part d. dropout shows the fully connected neural network with dropout. We set the percentage of the dropout to 0.5 in our evaluation experiment to be explained later.

Fully connected layer: After the dropout of 0.5. Then the flattering of Matrix is done for the input in fully connected layer with 125 units neurons and activation function as ReLU

Output: The SoftMax layer was placed as an output layer of the fully-connected layer as shown in Fig. SoftMax layer. Each unit (or node) in the SoftMax layer calculates the probability of each activity (i.e. WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING, LAYING_DOWN). The activity with the highest probability is then determined as the predicted (or recognized) activity and the activity label is outputted to the final node as shown in Fig 12-part e. output



The network architcture of our 1D CNN-based activity recognition method[1]

Figure 13: 1D CNN Network Architecture

1D CNN Model 1

- Layer1
  Filter size= 128 and kernel size= 5 which changes in every model and activation function= ReLU
- Layer 2
  Filter size= 128/2 and kernel size=5 activation function= ReLU
- Then Maxpooling with size 2, then dropout of 0.5 which allows only 50% data to move forward
- then the flattering of the matrix so that it can be input in fully connected layer having one layer of 125 units of deeply connected layers.
- Then the Softmax layers deeply connected having output probability of six different activities.

Below is the description of Model 1

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 124, 128)          5888
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 120, 64)           41024
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 60, 64)            0
_________________________________________________________________
dropout (Dropout)            (None, 60, 64)            0
_________________________________________________________________
flatten (Flatten)            (None, 3840)              0
_________________________________________________________________
dense (Dense)                (None, 125)               480125
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 756
=================================================================
Total params: 527,793
Trainable params: 527,793
Non-trainable params: 0
```

1D CNN Model 2

- Layer1
  Filter size= 128 and kernel size= 3 which changes in every model and activation function= ReLU
- Layer 2
  Filter size= 128/2 and kernel size=3 activation function= ReLU
- Then Maxpooling with size 2, then dropout of 0.5 which allows only 50% data to move forward
- then the flattering of the matrix so that it can be input in fully connected layer having one layer of 125 units of deeply connected layers.
- Then the Softmax layers deeply connected having output probability of six different activities.

Below is the description of Model 2

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_2 (Conv1D)            (None, 126, 128)          3584
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 124, 64)           24640
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 62, 64)            0
_________________________________________________________________
dropout_1 (Dropout)          (None, 62, 64)            0
_________________________________________________________________
flatten_1 (Flatten)          (None, 3968)              0
_________________________________________________________________
dense_2 (Dense)              (None, 125)               496125
_________________________________________________________________
dense_3 (Dense)              (None, 6)                 756
=================================================================
Total params: 525,105
Trainable params: 525,105
Non-trainable params: 0
```

1D CNN Model 3

- Layer1
  Filter size= 128 and kernel size= 7 which changes in every model and activation function= ReLU
- Layer 2
  Filter size= 128/2 and kernel size=7 activation function= ReLU
- Then Maxpooling with size 2, then dropout of 0.5 which allows only 50% data to move forward
- then the flattering of the matrix so that it can be input in fully connected layer having one layer of 125 units of deeply connected layers.
- Then the Softmax layers deeply connected having output probability of six different activities.

Below is the description of Model 3

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_2 (Conv1D)            (None, 122, 128)          8192
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 116, 64)           57408
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 58, 64)            0
_________________________________________________________________
dropout_1 (Dropout)          (None, 58, 64)            0
_________________________________________________________________
flatten_1 (Flatten)          (None, 3712)              0
_________________________________________________________________
dense_2 (Dense)              (None, 125)               464125
_________________________________________________________________
dense_3 (Dense)              (None, 6)                 756
=================================================================
Total params: 530,481
Trainable params: 530,481
Non-trainable params: 0
```

1D CNN Model 4

- Layer1
  Filter size= 64 and kernel size= 11 which changes in every model and activation function= ReLU
- Layer 2
  Filter size= 64/2 and kernel size=11 activation function= ReLU
- Then Maxpooling with size 2, then dropout of 0.5 which allows only 50% data to move forward
- then the flattering of the matrix so that it can be input in fully connected layer having one layer of 125 units of deeply connected layers.
- Then the Softmax layers deeply connected having output probability of six different activities.

Below is the description of Model 4

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 118, 128)          12800
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 108, 64)           90176
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 54, 64)            0
_________________________________________________________________
dropout (Dropout)            (None, 54, 64)            0
_________________________________________________________________
flatten (Flatten)            (None, 3456)              0
_________________________________________________________________
dense (Dense)                (None, 125)               432125
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 756
=================================================================
Total params: 535,857
Trainable params: 535,857
Non-trainable params: 0
```

1D CNN Model 5

- Layer1
  Filter size= 64 and kernel size= 7 which changes in every model and activation function= ReLU
- Layer 2
  Filter size= 64/2 and kernel size=7 activation function= ReLU
- Then Maxpooling with size 2, then dropout of 0.5 which allows only 50% data to move forward
- then the flattering of the matrix so that it can be input in fully connected layer having one layer of 125 units of deeply connected layers.
- Then the Softmax layers deeply connected having output probability of six different activities.

Below is the description of Model 5

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_8 (Conv1D)            (None, 122, 64)           4096
_________________________________________________________________
conv1d_9 (Conv1D)            (None, 116, 32)           14368
_________________________________________________________________
max_pooling1d_4 (MaxPooling1 (None, 58, 32)            0
_________________________________________________________________
dropout_4 (Dropout)          (None, 58, 32)            0
_________________________________________________________________
flatten_4 (Flatten)          (None, 1856)              0
_________________________________________________________________
dense_8 (Dense)              (None, 125)               232125
_________________________________________________________________
dense_9 (Dense)              (None, 6)                 756
=================================================================
Total params: 251,345
Trainable params: 251,345
Non-trainable params: 0
```

1D CNN Model 6

- Layer1
  Filter size= 32 and kernel size= 7 which changes in every model and activation function= ReLU
- Layer 2
  Filter size= 32/2 and kernel size=7 activation function= ReLU
- Then Maxpooling with size 2 ,then dropout of 0.5 which allows only 50% data to move forward
- then the flattering of the matrix so that it can be input in fully connected layer having one layer of 125 units of deeply connected layers.
- Then the Softmax layers deeply connected having output probability of six different activities.

Below is the description of Model 6

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_10 (Conv1D)           (None, 122, 32)           2048
_________________________________________________________________
conv1d_11 (Conv1D)           (None, 116, 16)           3600
_________________________________________________________________
max_pooling1d_5 (MaxPooling1 (None, 58, 16)            0
_________________________________________________________________
dropout_5 (Dropout)          (None, 58, 16)            0
_________________________________________________________________
flatten_5 (Flatten)          (None, 928)               0
_________________________________________________________________
dense_10 (Dense)             (None, 125)               116125
_________________________________________________________________
dense_11 (Dense)             (None, 6)                 756
=================================================================
Total params: 122,529
Trainable params: 122,529
Non-trainable params: 0
```

3.4.B. Network Architecture for 2D CNN

The network architecture of our 2D CNN consists of two convolutional layer, two dropout layers of 0.2, then flattering the matrix, one deep connected layer with 125 units, and one softmax layer that outputs the probability of each of the six activities. Below figure shows the 2D CNN architecture of our proposed method.

2D CNN:

The 2D convolution operations were performed using 4different Models having different filter and kernel size with two layers of 2D convolution layers having activation function as ReLU

Max-pooling: For each feature set of a given window size and filter type, 1-max-pooling was performed to select the largest feature value. Figure 14 max-pooling shows the max-pooled result after two convolution layers data as input.

Dropout: The convolved and max-pooled feature set were then placed as the input to the fully connected neural network with dropout of 20% applied. The dropout was applied to prevent the neural network from overfitting. Figure 14 dropout shows the fully connected neural network with dropout. We set the percentage of the dropout to 0.5 in our evaluation experiment to be explained later.

Fully connected layer: After the dropout of 0.2. Then the flattering of Matrix is done for the input in fully connected layer with 125 units neurons and activation function as ReLU

Output: The softmax layer was placed as an output layer of the fully-connected layer as shown in Fig.14 softmax layer. Each unit (or node) in the softmax layer calculates the probability of each activity (i.e. WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING, LAYING_DOWN). The activity with the highest probability is then determined as the predicted (or recognized) activity and the activity label is outputted to the final node as shown in Fig. 14 output

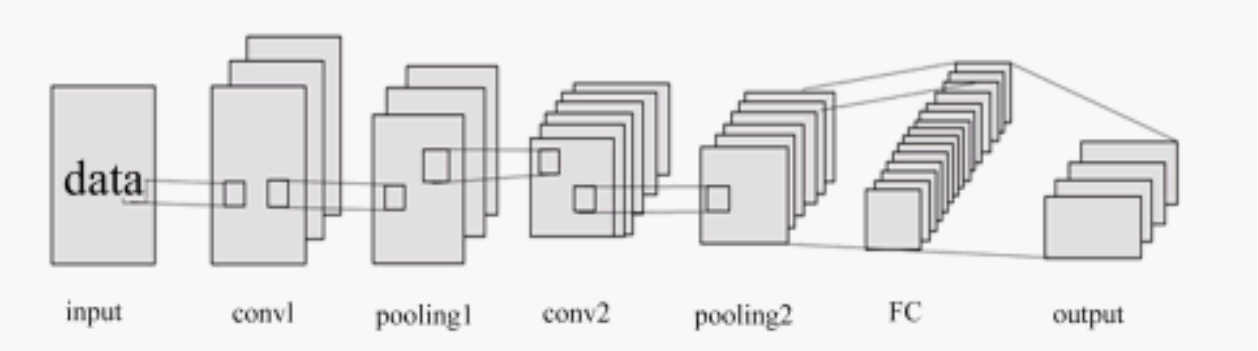


Figure14: 2D Convolutional Layers

2D CNN Model 7

- Layer1
  Filter size= 32 and kernel size= (4,4) which changes in every model and activation function= ReLU
- Layer 2
  Filter size= 64 and kernel size= (4,4) activation function= ReLU
- Then dropout of 0.2 which allows only 20% data to move forward
- then the flattering of the matrix so that it can be input in fully connected layer having one layer of 125 units of deeply connected layers.
- Then the Softmax layers deeply connected having output probability of six different activities.

Below is the description of Model 7

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 125, 6, 32)        544
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 125, 6, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 122, 3, 64)        32832
_________________________________________________________________
dropout_6 (Dropout)          (None, 122, 3, 64)        0
_________________________________________________________________
flatten_6 (Flatten)          (None, 23424)             0
_________________________________________________________________
dense_12 (Dense)             (None, 125)               2928125
_________________________________________________________________
dropout_7 (Dropout)          (None, 125)               0
_________________________________________________________________
dense_13 (Dense)             (None, 6)                 756
=================================================================
Total params: 2,962,257
Trainable params: 2,962,257
Non-trainable params: 0
_________________________________________________________________
```

2D CNN Model 8

- Layer1
  Filter size= 32 and kernel size= (2,2) which changes in every model and activation function= ReLU
- Layer 2
  Filter size= 64 and kernel size= (2,2) activation function= ReLU
- Then dropout of 0.2 which allows only 20% data to move forward
- then the flattering of the matrix so that it can be input in fully connected layer having one layer of 125 units of deeply connected layers.
- Then the Softmax layers deeply connected having output probability of six different activities.

Below is the description of Model 8

```
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 127, 8, 32)        160
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 127, 8, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 126, 7, 64)        8256
_________________________________________________________________
dropout_8 (Dropout)          (None, 126, 7, 64)        0
_________________________________________________________________
flatten_7 (Flatten)          (None, 56448)             0
_________________________________________________________________
dense_14 (Dense)             (None, 125)               7056125
_________________________________________________________________
dropout_9 (Dropout)          (None, 125)               0
_________________________________________________________________
dense_15 (Dense)             (None, 6)                 756
=================================================================
Total params: 7,065,297
Trainable params: 7,065,297
Non-trainable params: 0
_________________________________________________________________
```

2D CNN Model 9

- Layer1
  Filter size= 16 and kernel size= (2,2) which changes in every model and activation function= ReLU
- Layer 2
  Filter size= 32 and kernel size= (2,2) activation function= ReLU
- Then dropout of 0.2 which allows only 20% data to move forward
- then the flattering of the matrix so that it can be input in fully connected layer having one layer of 125 units of deeply connected layers.
- Then the Softmax layers deeply connected having output probability of six different activities.

Below is the description of Model 9

```
Model: "sequential_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 127, 8, 16)        80
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 127, 8, 16)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 126, 7, 32)        2080
_________________________________________________________________
dropout_10 (Dropout)         (None, 126, 7, 32)        0
_________________________________________________________________
flatten_8 (Flatten)          (None, 28224)             0
_________________________________________________________________
dense_16 (Dense)             (None, 125)               3528125
_________________________________________________________________
dropout_11 (Dropout)         (None, 125)               0
_________________________________________________________________
dense_17 (Dense)             (None, 6)                 756
=================================================================
Total params: 3,531,041
Trainable params: 3,531,041
Non-trainable params: 0
```

2D CNN Model 10

- Layer1
  Filter size= 16 and kernel size= (4,4) which changes in every model and activation function= ReLU
- Layer 2
  Filter size= 32 and kernel size= (4,4) activation function= ReLU
- Then dropout of 0.2 which allows only 20% data to move forward
- then the flattering of the matrix so that it can be input in fully connected layer having one layer of 125 units of deeply connected layers.
- Then the Softmax layers deeply connected having output probability of six different activities.

Below is the description of Model 10

```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_6 (Conv2D)            (None, 125, 6, 16)        272
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 125, 6, 16)        0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 122, 3, 32)        8224
_________________________________________________________________
dropout_12 (Dropout)         (None, 122, 3, 32)        0
_________________________________________________________________
flatten_9 (Flatten)          (None, 11712)             0
_________________________________________________________________
dense_18 (Dense)             (None, 125)               1464125
_________________________________________________________________
dropout_13 (Dropout)         (None, 125)               0
_________________________________________________________________
dense_19 (Dense)             (None, 6)                 756
=================================================================
Total params: 1,473,377
Trainable params: 1,473,377
Non-trainable params: 0
```

CHAPTER 4

Results and Analysis

We have discussed the contents of the UCI HAR dataset, our approach to classifying the samples of six different classes contained in it, as well as the techniques and methods that we have employed in the proposed methodology. In this section, we will present the findings of the study. Following the described procedure, we set a classification model where the provided training samples were used to train CNN model, and the rest of the samples were used to test it. The result yields a classification accuracy of 86% on the test samples. Below presents the classification accuracies on both the training and testing samples at each epoch. As seen in the figure, the training accuracy gradually increased with each epoch. The test accuracy, on the other hand, kept fluctuating around the 84% mark and peaked at the 11th and the final epoch.

Apart from the classification accuracy, the confusion matrix of classification is another useful tool to judge the performance of the classification model. The confusion matrix provides more details on the output of the classification process. It tells us how many samples of each class were tested and how many of them were classified correctly based on the model. In the best-case scenario (i.e., when the classification accuracy would be 100%) only the diagonal boxes of the matrix would contain non-zero values or the number of tested samples of the corresponding class, and all the other boxes would contain zeros. Below image provides the confusion matrix of the epoch of our model for HAR classification. It is apparent that the model works very well while distinguishing five of the six classes (Walking, Walking-Upstairs, Walking-Downstairs, and Laying) registering over 91% individual classification accuracies for each class. However, the model faces some difficulties while differentiating the Sitting states from the Standing states, as we can see that 15.8% samples of the former class have been misclassified as the later one. The performance can also improve while separating samples that belong to the three different classes of Walking.

Precision, recall, and F1-score are three commonly used parameters to measure the potency of a classifier. The precision of a class refers to the rate of the correctly classified samples within the positively classified samples of that class. High precision indicates a low false-positive rate and vice versa. Recall of a class, on the other hand, refers to the

fraction of the truly positive instances of the class that the classifier recognizes. A high recall indicates that the classifier made a handful of false-negative predictions and vice versa. Usually, the precision and recall values of all the classes are averaged and expressed as the precision and recall of the model, respectively. F1-score is simply the harmonic mean or the weighted average of the precision and recall values.Data below depicts the precision, recall, and F1-scores at each epoch of the described CNN-based classification model. All three matrices have very close values for both the train and test classifications, which is why they have been represented with only two lines.

To put the outcome of our classification model in context, we have compared our acquired results with that of other four similar studies involving the UCI HAR dataset in Table II. The table shows that in terms of classification accuracy the proposed method outperforms the methods described in [6], [9] and [10] by 5.9%, 0.17%, and 14.01% respectively. Only [7] and [12] have attained better performances than the proposed model. However, [6] and [9] were more successful in classifying the samples of the Sitting class than [7], and our model has good individual accuracy in some classes.

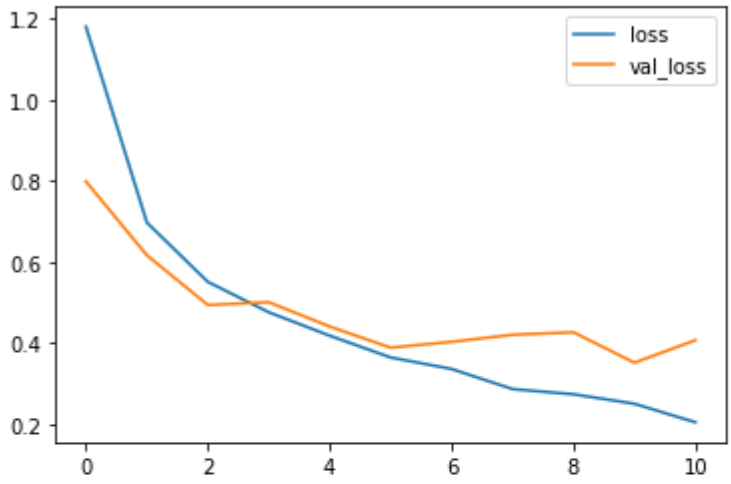
Each Model result has been discussed forward

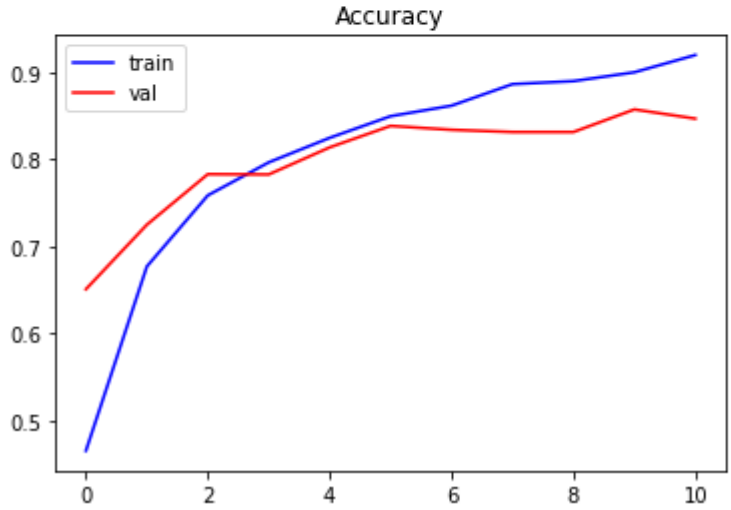
Model 1

1D CNN with filter size=128 kernel size=5

```
---------------------------------Classification report-

              precision    recall  f1-score   support

           0       0.99      1.00      1.00       532
           1       1.00      0.98      0.99       462
           2       0.98      0.99      0.98       421
           3       0.72      0.53      0.61       530
           4       0.63      0.92      0.75       586
           5       0.90      0.71      0.80       589

    accuracy                           0.85      3120
   macro avg       0.87      0.86      0.85      3120
weighted avg       0.86      0.85      0.84      3120


--------------------------------------Confusion matrix-

[[532   0   0   0   0   0]
 [  2 452   6   0   2   0]
 [  3   0 416   0   2   0]
 [  0   0   0 279 216  35]
 [  0   0   2  30 541  13]
 [  0   0   1  76  91 421]]
```

----------Loss Values plot-





----------Accuracy Values plot-

Model 1

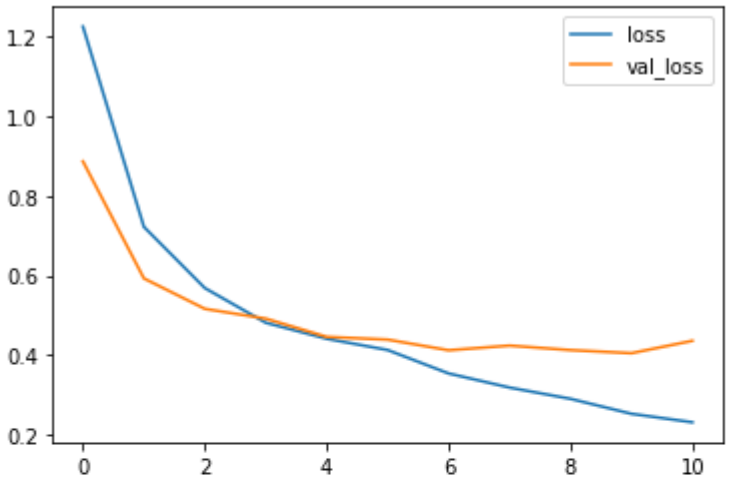
Accuracy = 85%

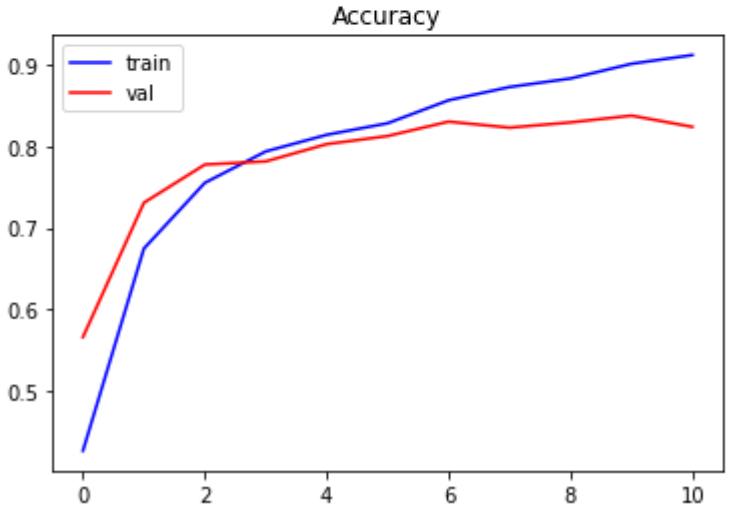
Model 2

1D CNN with filter size=128 kernel size=3

```
--------------------------------Classification report-

              precision    recall  f1-score   support

           0       0.97      1.00      0.99       532
           1       1.00      0.95      0.97       462
           2       0.97      1.00      0.98       421
           3       0.62      0.54      0.58       530
           4       0.64      0.83      0.72       586
           5       0.85      0.69      0.76       589

    accuracy                           0.82      3120
   macro avg       0.84      0.83      0.83      3120
weighted avg       0.83      0.82      0.82      3120


-------------------------------------Confusion matrix-

[[532   0   0   0   0   0]
 [ 13 439   8   1   1   0]
 [  0   1 420   0   0   0]
 [  0   0   0 287 191  52]
 [  2   0   3  73 488  20]
 [  1   0   1 100  83 404]]
```

----------Loss Values plot-





----------Accuracy Values plot-

Model 2

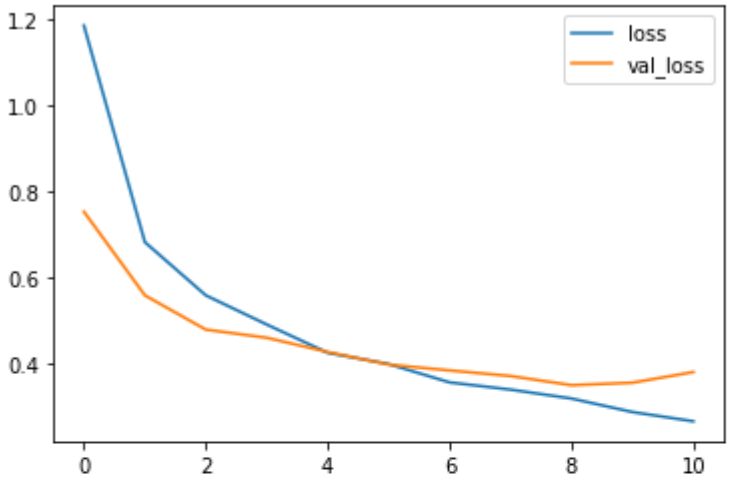
Accuracy = 82%

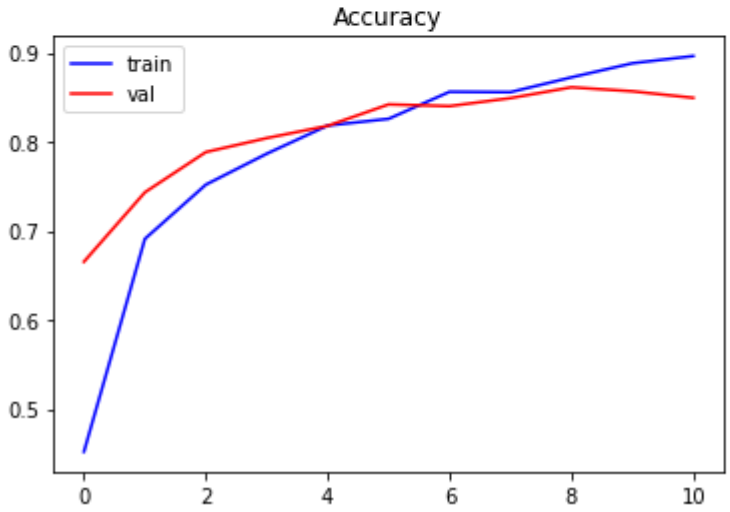
Model 3

1D CNN with filter size=128 kernel size=7

```
---------------------------------Classification report-

              precision    recall  f1-score   support

           0       0.99      1.00      0.99       532
           1       1.00      0.97      0.98       462
           2       0.97      0.99      0.98       421
           3       0.75      0.58      0.65       530
           4       0.65      0.89      0.75       586
           5       0.86      0.73      0.79       589

    accuracy                           0.85      3120
   macro avg       0.87      0.86      0.86      3120
weighted avg       0.86      0.85      0.85      3120


-------------------------------------Confusion matrix-

[[532   0   0   0   0   0]
 [  4 447   8   0   3   0]
 [  2   0 417   0   1   1]
 [  0   0   0 305 174  51]
 [  0   0   3  47 520  16]
 [  2   0   1  55 102 429]]
```

----------Loss Values plot-





----------Accuracy Values plot-

Model 3

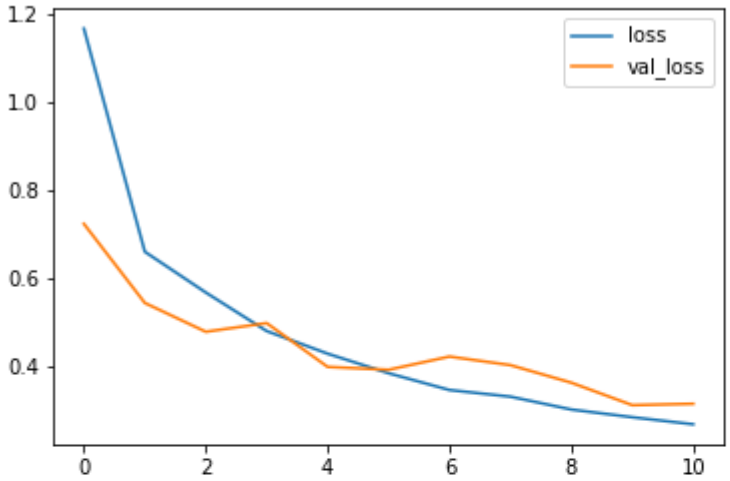
Accuracy = 85%

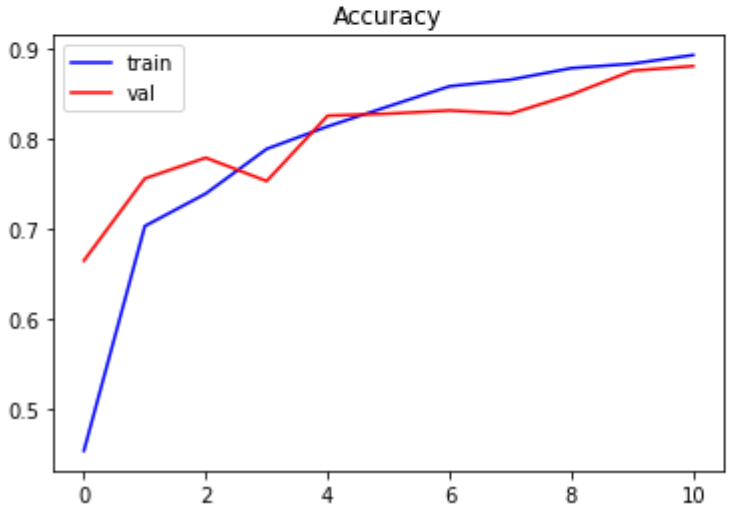
Model 4

1D CNN with filter size= 64 kernel size=11

```
---------------------------------Classification report-

              precision    recall  f1-score   support

           0       0.98      1.00      0.99       532
           1       1.00      0.99      0.99       462
           2       0.99      1.00      0.99       421
           3       0.75      0.71      0.73       530
           4       0.75      0.83      0.79       586
           5       0.88      0.81      0.84       589

    accuracy                           0.88      3120
   macro avg       0.89      0.89      0.89      3120
weighted avg       0.88      0.88      0.88      3120


-------------------------------------Confusion matrix-

[[532   0   0   0   0   0]
 [  2 457   3   0   0   0]
 [  1   0 420   0   0   0]
 [  2   0   0 378 108  42]
 [  4   0   1  74 487  20]
 [  1   0   2  55  56 475]]
```

----------Loss Values plot-





----------Accuracy Values plot-

Model 4

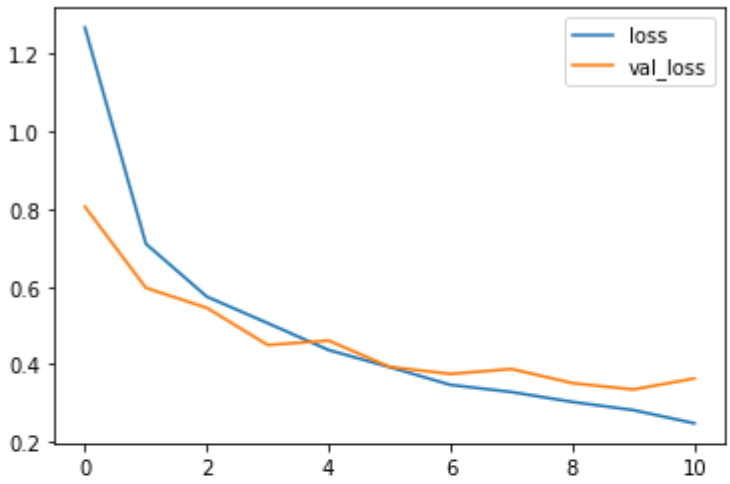
Accuracy = 88% (the best accuracy among all model)

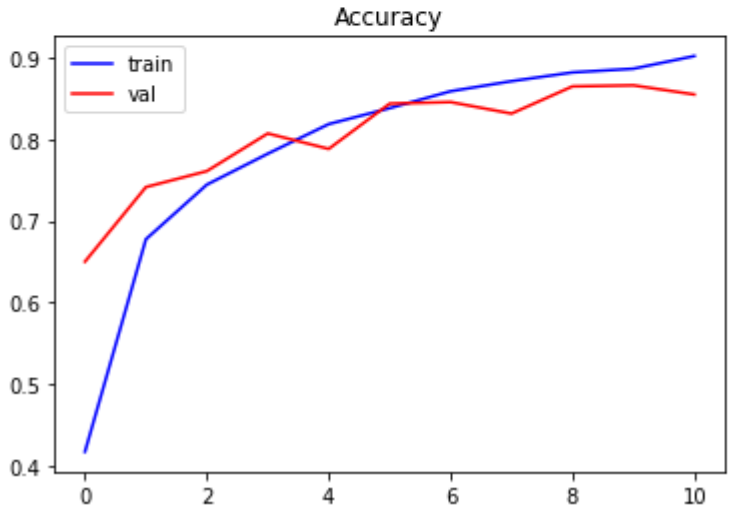
Model 5

1D CNN with filter size=64 kernel size=7

```
---------------------------------Classification report-

              precision    recall  f1-score   support

           0       0.99      0.99      0.99       532
           1       0.98      0.99      0.99       462
           2       0.96      1.00      0.98       421
           3       0.83      0.59      0.69       530
           4       0.63      0.95      0.76       586
           5       0.93      0.67      0.78       589

    accuracy                           0.85      3120
   macro avg       0.89      0.86      0.86      3120
weighted avg       0.88      0.85      0.85      3120


-------------------------------------Confusion matrix-

[[526   2   3   0   1   0]
 [  1 458   1   0   2   0]
 [  1   1 419   0   0   0]
 [  0   1   2 312 191  24]
 [  0   2   3  18 559   4]
 [  2   1   8  47 138 393]]
```

----------Loss Values plot-





----------Accuracy Values plot-

Model 5

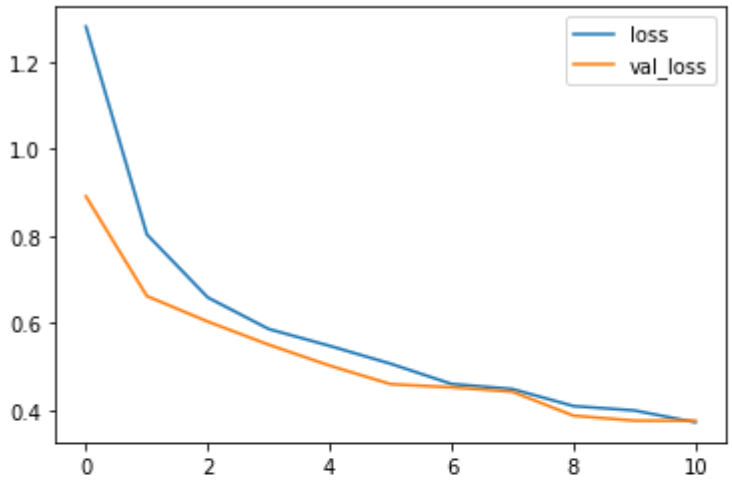
Accuracy = 85%

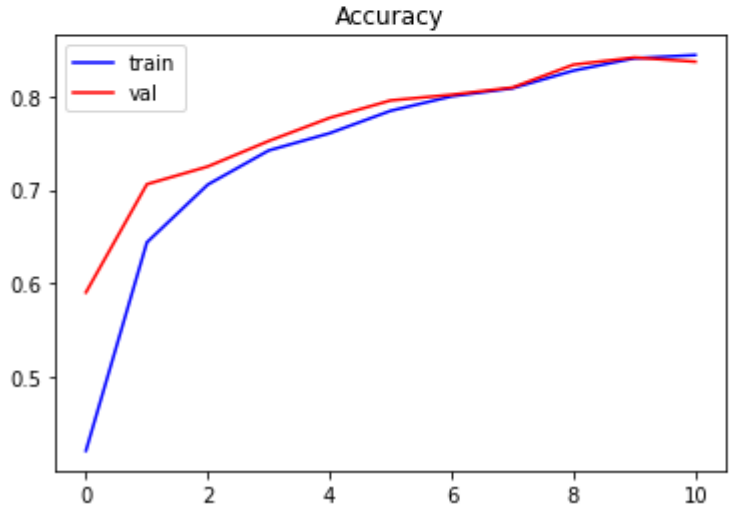
Model 6

1D CNN with filter size=32 kernel size=7

```
--------------------------------Classification report-

              precision    recall  f1-score   support

           0       0.99      1.00      0.99       532
           1       0.98      0.99      0.99       462
           2       1.00      0.99      0.99       421
           3       0.61      0.72      0.66       530
           4       0.78      0.60      0.68       586
           5       0.75      0.80      0.77       589

    accuracy                           0.84      3120
   macro avg       0.85      0.85      0.85      3120
weighted avg       0.84      0.84      0.84      3120


-------------------------------------Confusion matrix-

[[531   1   0   0   0   0]
 [  2 458   0   1   1   0]
 [  0   4 415   0   1   1]
 [  0   3   0 384  66  77]
 [  2   0   0 153 354  77]
 [  1   0   1  87  31 469]]
```

----------Loss Values plot-





----------Accuracy Values plot-

Model 6

Accuracy = 84 %

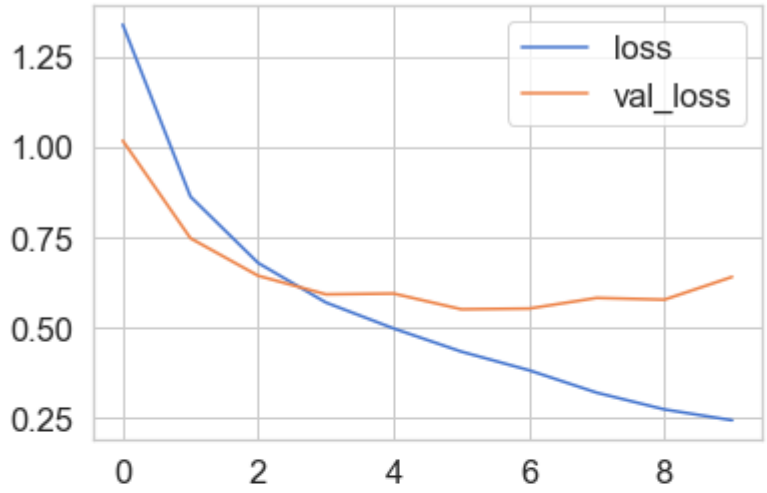
It's the only model where there is less difference between training accuracy and test accuracy.

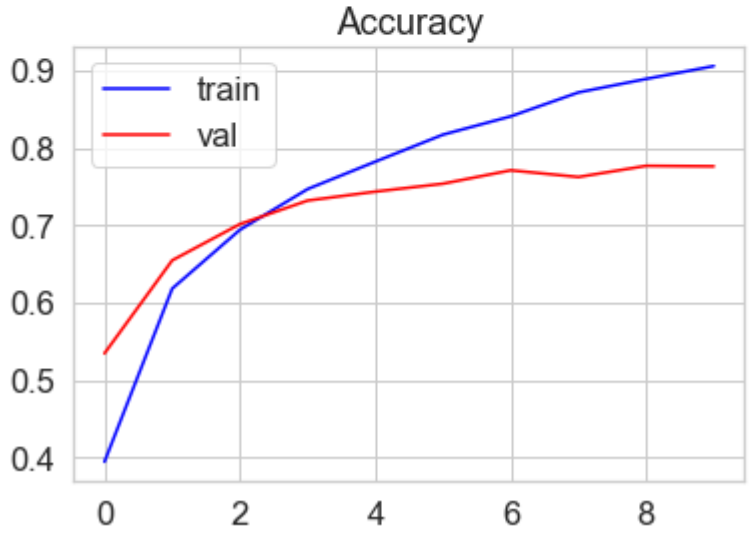
Model 7

2D CNN with filter size = 32 kernel size = (4,4)

```
--------------------------------Classification report-

              precision    recall  f1-score   support

           0       0.96      0.98      0.97       532
           1       0.94      0.97      0.96       462
           2       0.97      0.93      0.95       421
           3       0.61      0.46      0.53       530
           4       0.61      0.76      0.68       586
           5       0.65      0.64      0.65       589

    accuracy                           0.78      3120
   macro avg       0.79      0.79      0.79      3120
weighted avg       0.78      0.78      0.77      3120


-------------------------------------Confusion matrix-

[[520   6   5   0   1   0]
 [ 10 447   2   0   3   0]
 [  7  19 391   0   2   2]
 [  1   0   0 244 163 122]
 [  2   2   3  60 444  75]
 [  1   0   2  94 115 377]]
```

----------Loss Values plot-





----------Accuracy Values plot-

Model 7

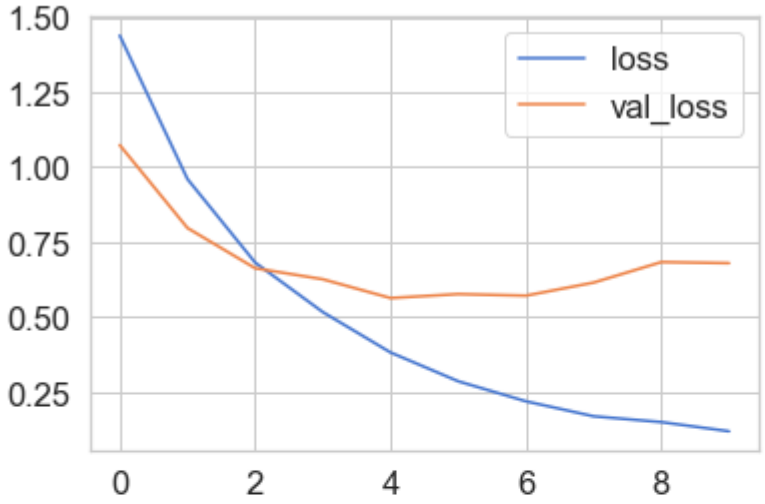
Accuracy = 78%

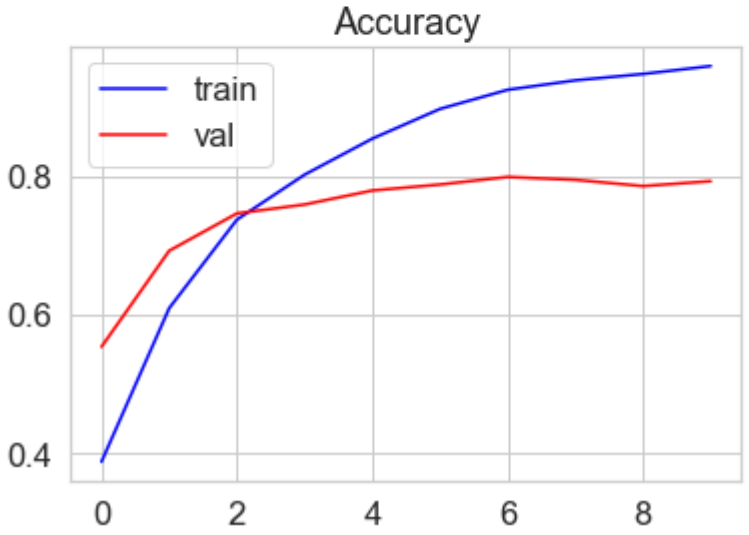
Model 8

2D CNN with filter size = 32 and kernel size = (2,2)

```
--------------------------------Classification report-

              precision    recall  f1-score   support

           0       0.94      0.89      0.92       532
           1       0.85      0.95      0.90       462
           2       0.94      0.93      0.93       421
           3       0.61      0.62      0.61       530
           4       0.69      0.75      0.72       586
           5       0.79      0.68      0.73       589

    accuracy                           0.79      3120
   macro avg       0.80      0.80      0.80      3120
weighted avg       0.80      0.79      0.79      3120


-------------------------------------Confusion matrix-

[[476  39   8   3   5   1]
 [  6 441   3   3   7   2]
 [  7  21 390   0   1   2]
 [  8   7   3 326 117  69]
 [  5   8   4  98 440  31]
 [  4   2   7 106  69 401]]
```

----------Loss Values plot-





----------Accuracy Values plot-

Model 8

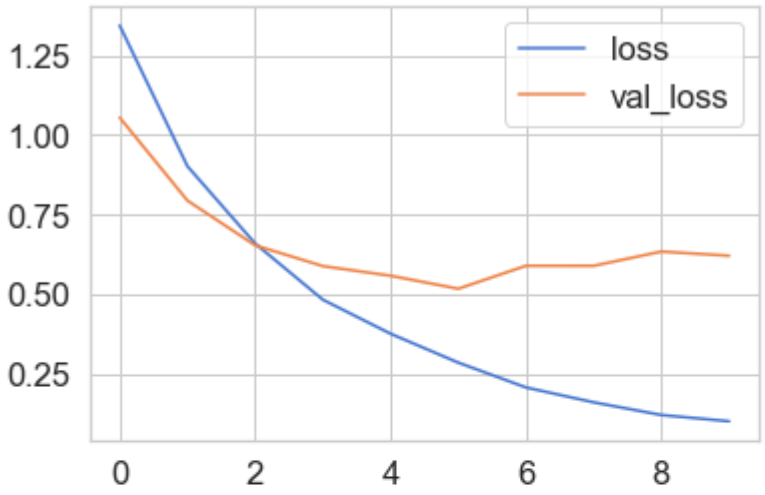
Accuracy = 79%

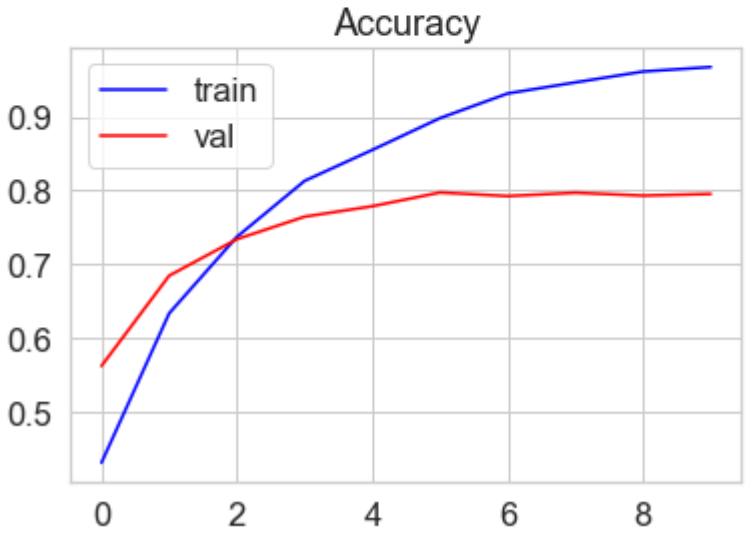
Model 9

2D CNN with filter size = 16 and kernel size = (2,2)

```
--------------------------------Classification report-

              precision    recall  f1-score   support

           0       0.91      0.97      0.94       532
           1       0.95      0.90      0.93       462
           2       0.97      0.94      0.95       421
           3       0.57      0.64      0.60       530
           4       0.70      0.73      0.71       586
           5       0.77      0.66      0.71       589

    accuracy                           0.80      3120
   macro avg       0.81      0.81      0.81      3120
weighted avg       0.80      0.80      0.80      3120


-------------------------------------Confusion matrix-

[[516   5   5   1   5   0]
 [ 26 418   4   5   9   0]
 [ 13   6 394   3   1   4]
 [  4   4   2 338 108  74]
 [  4   4   2 110 429  37]
 [  5   2   1 131  63 387]]
```

----------Loss Values plot-





----------Accuracy Values plot-

Model 9

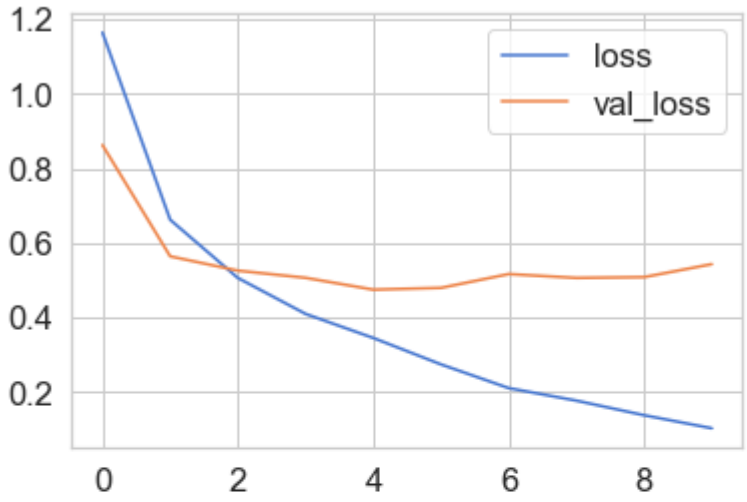
Accuracy = 80%

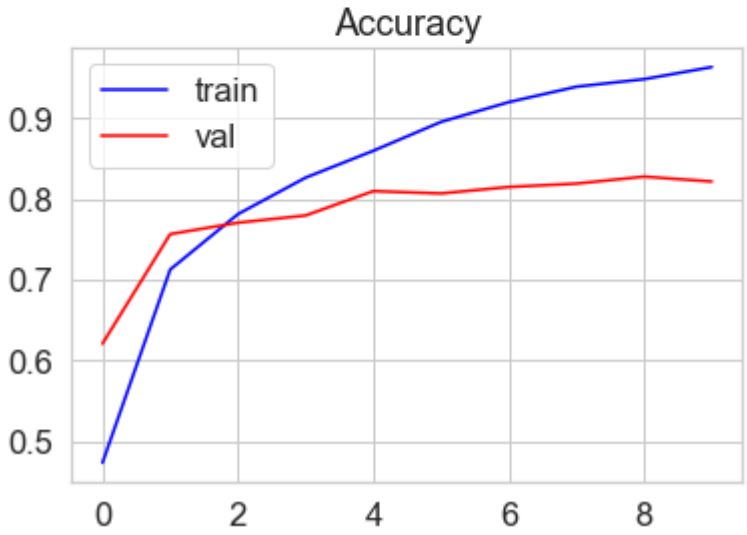
Model 10

2D CNN with filter size = 16 and kernel size = (4,4)

```
---------------------------------Classification report-

              precision    recall  f1-score   support

           0       0.99      0.95      0.97       532
           1       0.92      1.00      0.96       462
           2       0.99      0.97      0.98       421
           3       0.67      0.57      0.62       530
           4       0.70      0.77      0.73       586
           5       0.73      0.75      0.74       589

    accuracy                           0.82      3120
   macro avg       0.83      0.83      0.83      3120
weighted avg       0.82      0.82      0.82      3120


--------------------------------------Confusion matrix-

[[503  24   4   0   1   0]
 [  1 460   0   0   1   0]
 [  1  13 407   0   0   0]
 [  2   0   0 302 118 108]
 [  1   3   0  74 451  57]
 [  0   1   1  75  71 441]]
```

----------Loss Values plot-





----------Accuracy Values plot-

Model 10

Accuracy = 82%

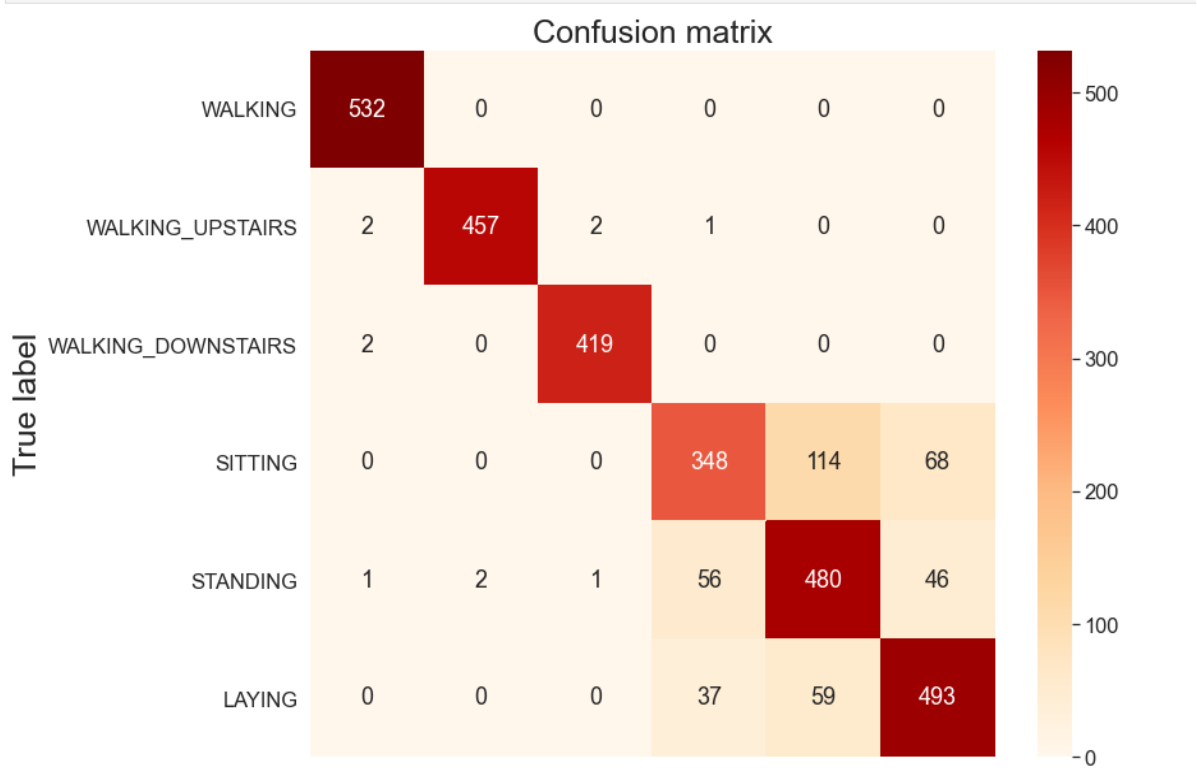
From the above Result we can say that,

The Model which gives the best result among all is model 4 consist of 1D CNN having filter size = 64 and kernel size= 11

The best results achieved in the model 4, loss values plot looks stable during epochs accuracy in each model around 88%, the highest in among all the model

models sometimes have problems to distinguish between activities number 3 (Sitting) and 4 (Standing). Probably the reason is that during the sitting and standing signals from sensors are similar

Below is the Confusion Matrix of Model 4



All the result is obtained by dividing the data into 70:30 ratio where 70% of data is used for training and 30% of data used for testing

Now all the above Models is run 5 time with same ratio (70:30). And the result of precision, F1_score, recall and accuracy is noted.

Then I calculate max, mean and standard deviations of precision, f1_score, recall and accuracy for the 5 times run data

		presision			recall			f1_score		
		max_Pre	mean_pre	std_pre	max_recal	mean_rec	std_recall	max_f1sc	mean_f1s	std_f1scor
	Walking	0.994393	0.986669	0.00614	1	0.998496	0.002192	0.997188	0.992535	0.002987
	Walking_u	0.997792	0.979163	0.020003	1	0.992208	0.007321	0.994595	0.985477	0.008393
	Walking_d	0.997519	0.994202	0.002423	0.985748	0.973872	0.012297	0.990453	0.983883	0.005781
	Sitting	0.736573	0.704108	0.032834	0.641509	0.572453	0.04587	0.655532	0.629364	0.021574
	Standing	0.727273	0.711052	0.021502	0.84471	0.814334	0.031933	0.77708	0.758486	0.013216
Model_1	Laying	0.899061	0.804053	0.058492	0.887946	0.804075	0.080595	0.834335	0.798287	0.026131
	Walking	0.994186	0.990203	0.002675	0.994361	0.984586	0.010986	0.991565	0.987341	0.004477
	Walking_u	0.995261	0.977089	0.013269	0.995671	0.974892	0.032986	0.986006	0.975493	0.013248
	Walking_d	0.98801	0.960671	0.045488	1	0.986223	0.007421	0.983412	0.972534	0.021079
	Sitting	0.724138	0.649148	0.051409	0.684906	0.628302	0.060836	0.664193	0.634258	0.024309
	Standing	0.841727	0.732514	0.06472	0.897611	0.731399	0.097126	0.753582	0.72342	0.017774
Model_2	Laying	0.892544	0.801705	0.073243	0.887946	0.77657	0.089515	0.796043	0.780619	0.013509
	Walking	1	0.994448	0.007389	1	0.995865	0.004352	0.999059	0.995126	0.002655
	Walking_u	1	0.991926	0.010821	0.997835	0.986147	0.011003	0.993492	0.988929	0.004784
	Walking_d	0.995272	0.98557	0.010954	1	0.995724	0.006266	0.99763	0.99057	0.005474
	Sitting	0.781538	0.72003	0.039354	0.769811	0.656226	0.096911	0.733154	0.679762	0.04595
	Standing	0.825147	0.734595	0.070292	0.895904	0.77099	0.086621	0.778481	0.74486	0.023876
Model_3	Laying	0.859551	0.817443	0.032004	0.864177	0.813582	0.03277	0.823328	0.814275	0.006822
	Walking	0.996255	0.987433	0.007434	1	0.999624	0.000752	0.998124	0.993476	0.003658
	Walking_u	0.997826	0.99266	0.003957	1	0.992208	0.009345	0.996757	0.992404	0.004752
	Walking_d	0.997619	0.992475	0.00634	0.997625	0.995724	0.001778	0.996433	0.994081	0.00246
	Sitting	0.752599	0.72688	0.019098	0.80566	0.730566	0.077826	0.769369	0.726212	0.039344
	Standing	0.824497	0.790365	0.048632	0.81058	0.744027	0.059975	0.814053	0.763881	0.035712
Model_4	Laying	0.86372	0.806024	0.04305	0.874363	0.829542	0.029614	0.828622	0.816053	0.010225
	Walking	0.996241	0.986774	0.012458	1	0.993233	0.00729	0.996241	0.989911	0.004984
	Walking_u	0.997807	0.986673	0.00701	0.995671	0.987879	0.004242	0.991285	0.987246	0.002057
	Walking_d	0.997602	0.980592	0.014625	1	0.991924	0.007748	0.99284	0.98611	0.004961
	Sitting	0.733189	0.686851	0.057939	0.696226	0.622642	0.05019	0.682139	0.649375	0.02489
	Standing	0.744479	0.711764	0.023541	0.865188	0.767577	0.06026	0.77377	0.737125	0.027692
Model_5	Laying	0.9117	0.818597	0.071635	0.855688	0.789474	0.056931	0.812968	0.798717	0.013707
	Walking	0.998117	0.983609	0.016279	1	0.995865	0.003834	0.997178	0.989606	0.007192
	Walking_u	0.989201	0.980766	0.01193	0.995671	0.987446	0.006762	0.991379	0.984064	0.008042
	Walking_d	0.997619	0.979034	0.013333	0.995249	0.992399	0.005701	0.996433	0.985648	0.009114
	Sitting	0.651163	0.597557	0.03786	0.715094	0.587925	0.068764	0.619787	0.58829	0.016699
	Standing	0.789593	0.709078	0.056355	0.790102	0.675768	0.085153	0.710668	0.685184	0.020908
Model_6	Laying	0.86618	0.777082	0.067644	0.928693	0.770458	0.103793	0.796834	0.764479	0.028435
	Walking	0.973485	0.927519	0.048837	0.988722	0.971053	0.017293	0.969811	0.947897	0.025512
	Walking_u	0.9413	0.922792	0.014013	0.974026	0.944156	0.03016	0.956337	0.933261	0.021425
	Walking_d	0.98731	0.973099	0.010789	0.966746	0.939192	0.020521	0.965599	0.955679	0.010804
	Sitting	0.650235	0.599241	0.035794	0.65283	0.588679	0.045721	0.628083	0.592159	0.025582
	Standing	0.727768	0.702835	0.019877	0.759386	0.709215	0.040261	0.722176	0.70495	0.015966
Model_7	Laying	0.769088	0.725083	0.030995	0.789474	0.695756	0.05483	0.744596	0.708353	0.027236

	Walking	0.964286	0.943994	0.021813	0.964286	0.92782	0.025208	0.948669	0.935331	0.009391
	Walking_u	0.909283	0.895777	0.014983	0.965368	0.945022	0.013593	0.9282	0.919569	0.007124
	Walking_d	0.982143	0.966511	0.01397	0.945368	0.936342	0.011772	0.956731	0.951035	0.004654
	Sitting	0.713528	0.655582	0.031573	0.681132	0.597358	0.063556	0.649281	0.62153	0.023175
	Standing	0.719643	0.691777	0.027447	0.832765	0.76314	0.047872	0.736237	0.723894	0.011598
Model_8	Laying	0.806452	0.769215	0.035776	0.813243	0.735144	0.043815	0.757655	0.749705	0.006851
	Walking	0.981481	0.937591	0.023374	0.960526	0.933459	0.024148	0.945996	0.935015	0.009414
	Walking_u	0.957346	0.908665	0.038303	0.974026	0.934199	0.033549	0.937301	0.919991	0.012736
	Walking_d	0.97284	0.949732	0.013937	0.942993	0.938242	0.003359	0.953995	0.943894	0.006719
	Sitting	0.667647	0.61758	0.029518	0.635849	0.557358	0.071416	0.616715	0.581515	0.033271
	Standing	0.72549	0.695875	0.032314	0.783276	0.718771	0.034034	0.710069	0.705625	0.005336
Model_9	Laying	0.781132	0.727682	0.031522	0.797963	0.748048	0.035347	0.747218	0.736345	0.009377
	Walking	0.984645	0.949172	0.028059	0.994361	0.981579	0.010526	0.97493	0.964732	0.010528
	Walking_u	0.986364	0.956376	0.026065	0.978355	0.951948	0.01787	0.962306	0.953704	0.00868
	Walking_d	0.975248	0.96654	0.007339	0.976247	0.955819	0.017036	0.969267	0.961007	0.006085
	Sitting	0.633152	0.593194	0.033538	0.60566	0.544528	0.058965	0.611429	0.5646	0.029327
	Standing	0.694136	0.673883	0.022296	0.784983	0.725597	0.042111	0.719803	0.697507	0.013607
Model_10	Laying	0.76556	0.715522	0.025456	0.721562	0.690662	0.037141	0.714286	0.701626	0.013277

Below table shows maximum, mean, Standard deviations of the train accuracy and test accuracy for the 5 time run data.

	train_Accuracy			test_Accuracy		
	max	mean	std	max	mean	std
Model_1	92	91	0.72	86	85	0.87
Model_2	94	93	0.7	84	84	0.75
Model_3	92	91	0.69	87	86	1.09
Model_4	91	89	1.27	89	87	1.36
Model_5	91	88	1.86	86	85	1.21
Model_6	85	84	0.84	83	82	0.69
Model_7	97	96	1.5	80	80	0.55
Model_8	98	97	0.89	81	81	0.44
Model_9	96	91	1.94	81	80	0.74
Model_10	94	93	1.01	80	80	0.55

CHAPTER 5

5.1 CONCLUSIONS

The paper describes a CNN-based HAR classification model and tests it on the UCI HAR dataset extracting the frequency of the samples. The obtained results yield a 88% classification accuracy. However, the model can be further modified by tuning specific parameters of CNN and adding more nodes and layers in the CNN architecture.

In this paper, the significance of HAR research is analysed, and an overview of emerging methods in the field is provided. Convolution neural networks have been used in many innovations in natural language processing, image recognition, and other predictions. This technology was adapted to the HAR task. We proposed the different framework of the CNN network. This deep network can enhance learning ability for faster learning in early training. We also found that window size was a key parameter. Too small a window did not guarantee continuity of information, and too large a window caused classification errors.

5.2 Future Scope

A new set of features can also be extracted and fed in an additional channel of CNN to improve the model's performance, which is subjected to future studies. The issue with the low classification accuracy of the Sitting class must be addressed as well. We are also interested in evaluating our model using other HAR datasets, including an updated version of the UCI HAR dataset that contains Postural Transitions.

Future work should explore a more efficient way to tune parameters. Although the grid search was workable, the searching range must be changed manually, and the values are always fixed. It will be important to find an adaptive way to automatically adjust the searching process and also make the neural network's architecture evolve, such by as automatically reshaping, adding, and removing various layers. Finally, applying the CNN network to other fields could be revealing. A good model should have outstanding generalization. Indeed, focusing on time series prediction problems has value.

REFERENCES

[1] D. Spicer and M. Weber, “Computers Timeline of Computer History Computer History Museum.” [Online]. Available: https://www.computerhistory.org/timeline/computers/. [Accessed: 10-May-2019].

[2] M. A. Labrador and O. D. Lara Yejas, “Human activity recognition: using wearable sensors and smartphones.” CRC Press, 2013.

[3] Y. Fu, “Human activity recognition and prediction.” Springer, 2016.

[4] J. Wang, Z. Liu, and Y. Wu, “Human Action Recognition with Depth Cameras. Cham: Springer International Publishing,” 2014.

[5] J. Reyes-Ortiz, D. Anguita, A. Ghio, L. Oneto, and X. Parra, “UCI Machine Learning Repository: Human Activity Recognition Using Smartphones Data Set,” 2012. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones. [Accessed: 20-Apr-2019].

[6] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, “Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine,” Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 7657 LNCS, pp. 216–223, 2012.

[7] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, “A Public Domain Dataset for Human Activity Recognition Using Smartphones.” 2013.

[8] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, “Energy efficient smartphone-based activity recognition using fixedpoint arithmetic,” J. Univers. Comput. Sci., vol. 19, no. 9, pp. 1295–1314, 2013.

[9] S. Bhattacharjee, S. Kishore, and A. Swetapadma, “A Comparative Study of Supervised Learning Techniques for Human Activity Monitoring Using Smart Sensors,” in 2018 Second International Conference on Advances in Electronics, Computers and Communications (ICAECC), 2018, pp. 1–4.

[10] P. Bota, J. Silva, D. Folgado, and H. Gamboa, “A Semi-Automatic Annotation Approach for Human Activity Recognition,” Sensors (Basel)., vol. 19, no. 3, pp. 1–23, 2019.

[11] W. Jiang and Z. Yin, “Human activity recognition using wearable sensors by deep convolutional neural networks,” MM 2015 - Proc. 2015 ACM Multimedia. Conf., pp. 1307–1310, 2015.

[12] B. Almaslukh, A. Jalal, and A. Abdelmonim, “An Effective Deep Autoencoder Approach for Online Smartphone-Based Human Activity Recognition,” Int. J. Comput. Sci. Netw. Secur., vol. 17, no. 4, pp. 160–165, 2017.

[13] A. Zheng and A. Casari, “Feature Engineering for Machine Learning and Data Analytics - Principles and Techniques for Data Scientists”. 2018.

[14] N. Sikder, K. Bhakta, A. Al Nahid, and M. M. M. Islam, “Fault diagnosis of motor bearing using ensemble learning algorithm with FFT-based pre-processing,” in 2019 International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST), 2019, pp. 564–569.

[15] P. Stoica and R. Moses, “Spectral analysis of signals,” 2005.

[16] M. Sewak, R. Karim, and P. Pujari, “Practical Convolutional Neural”. Packt Publishing, 2018.

[17] I. Goodfellow, Y. Bengio, and A. Courville, “Deep learning”.

[18] C. C. Aggarwal,” Neural networks and deep learning” : a textbook. 2018.

[19] P. Golik, Z. Tüske, R. Schlüter, and H. Ney, “Convolutional neural networks for acoustic modelling of raw time signal in LVCSR,” INTERSPEECH 2015, 2015.

[20] K. Bhakta, N. Sikder, A. Al Nahid, and M. M. M. Islam, “Fault Diagnosis of Induction Motor Bearing Using Cepstrum-based Preprocessing and Ensemble Learning Algorithm,” in 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE), 2019, pp. 1–6.

[21] G. Hackeling,” Mastering Machine Learning” with scikit-learn. Packt Publishing, 2014.