CAPITAL UNIVERSITY OF SCIENCE AND TECHNOLOGY, ISLAMABAD

A Novel Hybrid Deep Learning Model for Human Activity Recognition Based on Transitional Activities

by

Saad Irfan Khan

A thesis submitted in partial fulfillment for the degree of Master of Science

in the

Faculty of Computing
Department of Computer Science

2021

I dedicate this work to my parents for their endless love and support. For believing in me and for providing me with everything they could have offered. I would also dedicate this thesis to my supervisor for his continuous guidance and support throughout the year.

CERTIFICATE OF APPROVAL

A Novel Hybrid Deep Learning Model for Human Activity Recognition Based on Transitional Activities

by

Saad Irfan Khan

(MCS191026)

THESIS EXAMINING COMMITTEE

S. No.	Examiner	Name	Organization
(a)	External Examiner	Dr. Muhammad Zubair	RIPHAH, Rawalpindi
(b)	Internal Examiner	Dr. Imtiaz Ahmad Taj	CUST, Islamabad
(c)	Supervisor	Dr. Nadeem Anjum	CUST, Islamabad

Dr. Nadeem Anjum
Thesis Supervisor
December, 2021

Dr. Nayyer Masood
Head
Dept. of Computer Science
December, 2021

Dr. M. Abdul Qadir
Dean
Faculty of Computing
December, 2021

Author's Declaration

I, **Saad Irfan Khan** hereby state that my MS thesis titled "**A Novel Hybrid Deep Learning Model for Human Activity Recognition Based on Transitional Activities**" is my own work and has not been submitted previously by me for taking any degree from Capital University of Science and Technology, Islamabad or anywhere else in the country/abroad.

At any time if my statement is found to be incorrect even after my graduation, the University has the right to withdraw my MS Degree.

(Saad Irfan Khan)
Registration No: MCS191026

Plagiarism Undertaking

I solemnly declare that research work presented in this thesis titled "**A Novel Hybrid Deep Learning Model for Human Activity Recognition Based on Transitional Activities**" is solely my research work with no significant contribution from any other person. Small contribution/help wherever taken has been duly acknowledged and that complete thesis has been written by me.

I understand the zero tolerance policy of the HEC and Capital University of Science and Technology towards plagiarism. Therefore, I as an author of the above titled thesis declare that no portion of my thesis has been plagiarized and any material used as reference is properly referred/cited.

I undertake that if I am found guilty of any formal plagiarism in the above titled thesis even after award of MS Degree, the University reserves the right to withdraw/revoke my MS degree and that HEC and the University have the right to publish my name on the HEC/University website on which names of students are placed who submitted plagiarized work.

(Saad Irfan Khan)
Registration No: MCS191026

List of Publications

It is certified that following publication has been made out of the research work that has been carried out for this thesis:-

1. **Irfan, S.**; Anjum, N.; Masood, N.; Khattak, A.S.; Ramzan, N. A Novel Hybrid Deep Learning Model for Human Activity Recognition Based on Transitional Activities. Sensors 2021, 21, 8227.

(Saad Irfan Khan)
Registration No: MCS191026

Acknowledgement

"In the name of Allah, the Most Merciful and the Most Beneficent."

First and foremost, I am thankful to Almighty Allah for granting me success in every good aspect of life and for providing me with the courage to complete my MS thesis work.

Then I would like to thank my parents, siblings, dearest friends, and all those who encouraged me to achieve this goal.

Particularly, I am very grateful to my mentor and supervisor **Dr. Nadeem Anjum**; whose guidance and encouragement provided an immense contribution towards the completion of my MS thesis.

(Saad Irfan Khan)

Abstract

In recent years, a plethora of algorithms has been devised for efficient human activity recognition. Most of these algorithms consider basic human activities and neglect the postural transitions - because of their subsidiary occurrence and short duration. However, postural transitions assume a significant part in the enforcement of an activity recognition framework that cannot be neglected. This work proposes an ensemble multi-model activity recognition approach that employs basic and transition activities by utilizing multiple deep learning models simultaneously. For final classification, a dynamic decision fusion module is introduced that generates predictions based on the weighted average method. Experiments were performed on the publicly available datasets and the proposed approach achieved the classification accuracy of 96.11% and 98.38% for the transition and basic activities, respectively. The outcomes show that the proposed method is superior to the state-of-the-art methods in terms of accuracy and precision.

Contents

List of Figures

List of Tables

Abbreviations

BLSTM	Bidirectional Long Short Term Memory
CNN	Convolutional Neural Networks
D.L	Deep Learning
DBN	Deep Belief Networks
FC	Fully Connected
FFT	Fast Fourier Transformation
GBDT	Gradient Boosted Decision Trees
HAR	Human Activity Recognition
HAPT	Human Activities and Postural Transitions
LSTM	Long Short Term Memory
M.L	Machine Learning
ReLu	Rectified Linear Unit
SVM	Support Vector Machine
TA	Transition Aware
TED	Transition Event Detection

Symbols

y	Feature Vector
I	Feature matrix
η	Number of Neurons
μ	Learning rate - CNN
κ	Dropout rate
ϱ	Activation Function
γ	Pool size
s	Stride
α	Kernel 1
β	Kernel 2
ζ	Learn rate - LSTM/BLSTM
μ_B	Batch Mean
σ_B^2	Batch Variance

Chapter 1

Introduction

1.1 Human Activity Recognition

Activity Recognition (AR) inculpates the recognition, interpretation, and assessment of daily-life human activities hence known as Human Activity Recognition (HAR). Wearable sensors such as accelerometer, gyroscope, depth, etc. can be attached to assorted body locations to record the movement patterns and actions. In recent years, HAR research has drawn in critical consideration on account of its boundless applications such as fashion [1], surveillance systems [2, 3], smart homes [4] and healthcare [5].

The main ideas of the activity recognition framework are to track and assess human actions and understand the behavior of humans in their respective surroundings. Using multi-type motion patterns, Activity Recognition systems recapture and extract statistical features such as, - temporal, spatial, etc. to predict and understand human behavior. There exist numerous such application tools and platforms that encapsulate HAR generalities and the advanced systems are designed and developed there. Human Activities are generally categorized as basic activities, complex activities, and the postural transitions between or within these activities. Postural transition is a finite movement between two activities that varies between humans in terms of time and actions. Most of the works do not take into account the postural transitions because of their short duration. However, they play an important role in the efficient recognition of activities when multiple

tasks are performed in a shorter duration of time [6]. Generally, HAR systems are based on two categories which are vision-based devices and Sensor-based devices. Vision-based HAR does have a mature base so do they have limitations compared to sensor-based devices.

Smart sensors are embedded in our smartphones nowadays which are compact, and do not cause any unnatural disturbance, and can be worn in quotidian exertion. Such sensors can record any change in human action and can provide real-time observation with a minimum time lag. The most common sensors that come embedded in smartphones are an accelerometer, gyroscope, and compass which can be used to generate tri-axial movement data [7]. These sensors can also be used in conjunction with and fed to machine learning models to recognize activities better. The main difference between an accelerometer and a gyroscope is that one can sense rotation while the other can't. A gyroscope is typically used to generate orientation whereas an accelerometer is used to calculate linear acceleration.

1.1.1 Applications of HAR

Progress in the areas of human activity recognition has significantly bettered our daily lives that can unquestionably be considered ostensible relating to the ascent of daily life contemplation. Several HAR systems have been designed to automate the fore mentioned applications; however, assembling a completely automated HAR framework can be an extremely daunting undertaking since it requires a colossal pool of movement data and methodical classification algorithms. In addition, it is a troublesome undertaking in light of the fact that a solitary movement can be acted in more than one way [8]. HAR applications have also been embedded with the IoT environment. Multiple IoT devices can also be attached to a serialized network to gather data HAR data with sensor and vision-based devices. A generalized IoT-based HAR framework is shown in Figure 1.1. The figure portrays the flow of an Iot device connected to different real world applications.

[10] harnessed multiple detectors such as, - heart rate detector, temperature detector, pressure detector, etc. The system worked on the same principle and hardware as the fore-mentioned. The only key difference was that equal voltage

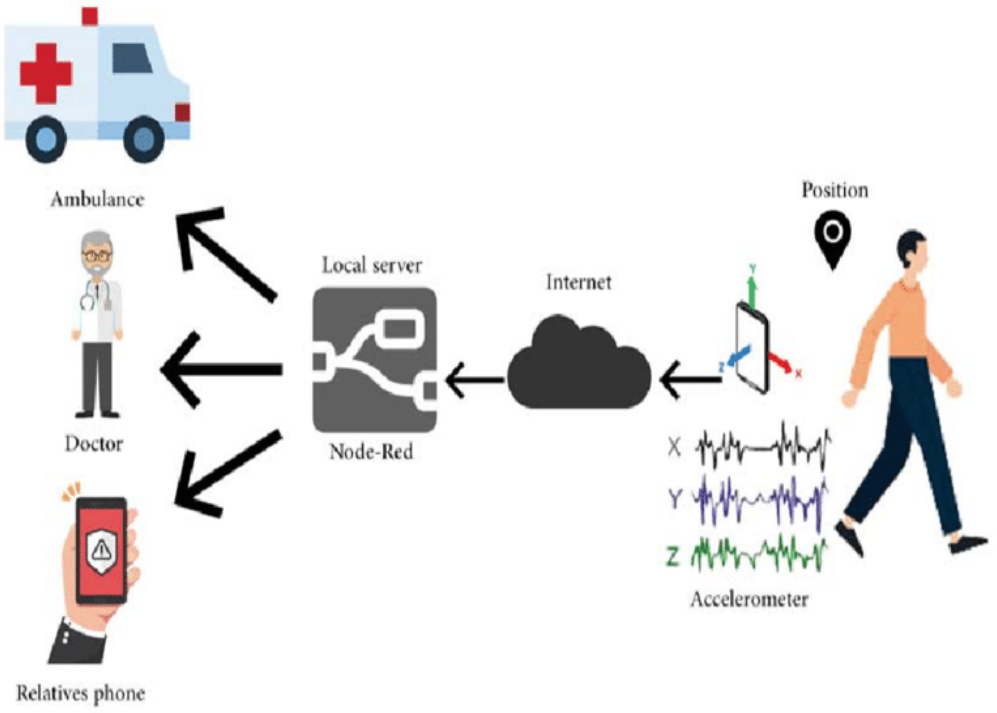

FIGURE 1.1: IoT Based HAR Architecture from [9]

was provided to each sensor along with the inclusion of a step-down transformer. The purpose of the transformer was to manage the load on each sensor such that only the required voltage is provided to the respective sensor. Unlike the previous work, the hardware device was compact with a user-friendly system. Unlike previous works, this approach had no such limitations or drawbacks and was easy to use. In another approach, [11], the magnitude of an automated health guidance system with detectors exercising a robust framework with more focus on security and bareness was introduced. The conjunction of multiple sensors can generate a substantial amount of data and meaningful data which can be used to extract quality features by using a machine learning algorithm.

1.2 Machine Learning (M.L)

Traditional M.L algorithms are utilized as a classification tool and are contrary to achieving satisfactory results. Activity recognition using standard Machine Learning approaches such as Support Vector Machine (SVM) [12, 13, 14, 15], K Nearest Neighborhood KNN [16, 17, 18], Discrete Cosine Transform (DCT) [19, 20,

21], Decision Tree (DT) [22, 23] and Random Forest (RF) [24, 25] etc. have been reported to produce good results under controlled environment [26]. The accuracy of these models heavily depends on the process of feature selection/extraction.

A volume of research has been conducted on activity recognition based on features obtained from a variety of datasets collected using various sources such as accelerometers and gyroscopes. However, it is of great importance that a feature selection preprocess module is applied to select a subset that prunes away any noise and redundancy, which would otherwise only degrade the performance of the recognition system. This computation is also called dimensionality reduction, i.e. selection of features that would complement each other. A variety of feature selection methods have been used to improve performance of activity recognition systems, such as correlation-based [27], energy-based [28], cluster analysis [29], AdaBoost [30], Relief-F [31], Single Feature Classification (SFC), Sequential Forward Selection (SFS) [32], and Sequential Floating Forward Search (SFFS) [33]. The aim of feature selection methods is to drop out features that carry the least information in discriminating an activity, consequently increasing efficiency without compromising robustness. For more on feature selection methods, readers are referred to a survey in [34]. Mi Zhang and Alexander A.Sawchuk have proposed an SVM-based framework for analyzing the effects of feature selection methods on the performance of activity recognition systems [35]. This research has shown that the SFS method has performed better than Relief-F and SFC methods. Essentially, this points out the fact that the topmost relevant features encode more information than the left out features, which considerably increases the performance of the activity recognition system. However, this can not be considered as a fulfilled requirement in building an efficient HAR system as a robust HAR system has more to it then higher average accuracy in recognition of activities.

Likewise, Ahmed et al. [36] demonstrated a feature selection model based on a hybrid SFFS feature selection method that selects/extracts the best features in view of a set of specific rules. Moreover, sets of the best features are formed and contrasted with the next set of features. The final optimal features were input to an SVM classifier for activity classification. However, machine learning techniques

to date have used shallow-structured learning architectures that only use one or two nonlinear layers of feature transformation. And usually shallow architectures refer to statistical features such as mean, frequency, variance, amplitude, etc. that could only be used for low-level activities like running, walking, standing that are well-constrained activity and could not model complex postural situations [37]. Moreover, lack of good quality data due to the costly process of labeling that requires human expertise/domain knowledge is also a bottleneck as well as manual selection of features are vulnerable to a margin of human error that would not generalize well on unseen data, because activity recognition tasks in real-life applications are much complicated and need close collaboration with the feature selection module [38].

1.3 Deep Learning

As the volume of the dataset has increased to an unprecedented level, in recent years, deep learning (D.L) has accomplished noteworthy results in the space of HAR. One of the affecting aspects of deep learning is the automatic feature identification and classification with high accuracy, which consequently produced an appeal in the space of HAR [39]. A substantial amount of uni-model and hybrid approaches have been introduced to gain benefit from deep learning techniques that cater to the shortcomings of the machine learning domain and utilize the multiple levels of features found in different levels of hierarchies. Deep Learning models involve a hierarchy of layers to accommodate low-level and high-level features as well as linear and nonlinear feature transformations at these levels, which helps in learning and refining features. To this end, models such as CNN [40] capable of performing convolutions on dense layers of data, RNN [41] capable of preserving information in the past and LSTM [42] capable of preserving past information with longer sequences of data, etc. are used to overcome the limitations of traditional machine learning algorithms that were dependent on manual features and erroneous selection/classification of features could prompt inconvenient impacts for the applications at hand. A conventional CNN structure is represented in Figure 1.2. Therefore, deep learning networks found a natural application in recognition tasks and have been popularly used for feature extraction in activity

recognition research [43]. One downside to the deep learning paradigm especially using the hybrid architectures is their increased cost of processing the available huge amount of datasets. However, it is worth the cost because a HAR system requires accurate classification results of the deep learning models. However, modern day hardware resources have significantly increased and are capable of dealing with huge amount of data with limited computational cost. Moreover, state-of-the-art tools have also been used introduced which are capable of automatizing several bulky tasks which required extensive time and effort.

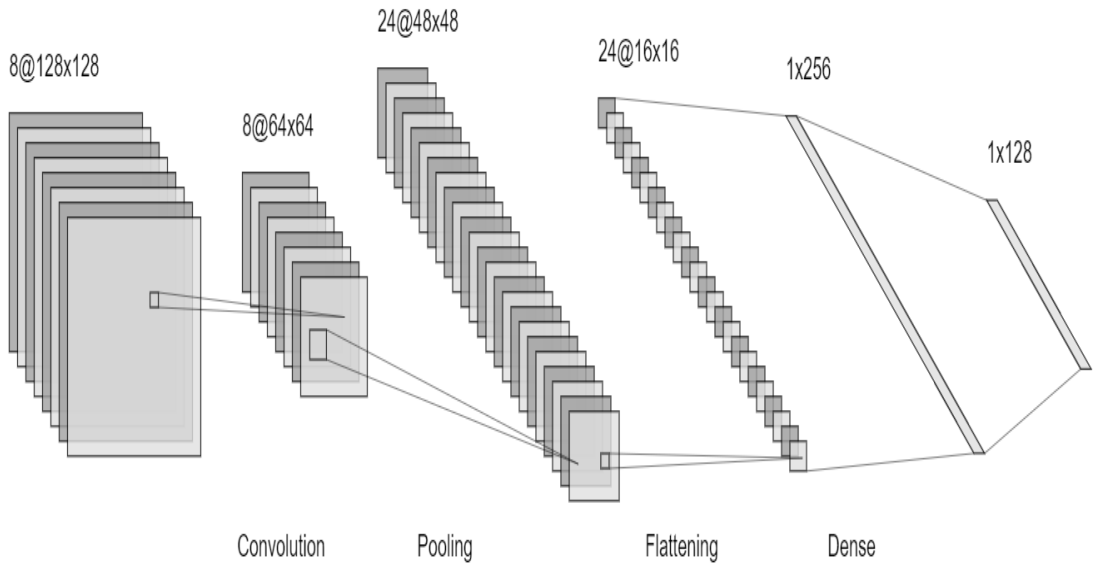

FIGURE 1.2: Overall structure of the Conventional CNN.

1.3.1 Problem Statement and Research Questions

From the detailed discussion in the previous sections, it can be concluded that a plethora of algorithms is available for Human Activity Recognition, however, most of the works ignore the activity transition factor and do not employ transitional activities. However, postural transitions play an important role in improving the activity recognition rate of a HAR system and accurately identifying activities in real-time activity recognition scenarios as well. Moreover, with the availability of transition data, analysis can be done to better recognize, classify, cluster, and predict what human activities are carried out for further decision-making. Based on these limitations and drawbacks, we have devised the following Research

Questions: 1 - Can we accurately identify postural transitions with high accuracy when the number of instances is very small compared to basic activities? 2 - Can we achieve accuracy gain with the simultaneous training of neural networks instead of in a pipeline flow? 3 - Can we increase the classification accuracy of the state-of-the-art by modifying their architecture?

The rest of the thesis is organized as follows: Chapter 2 shows a detailed discussion on the literature related to HAR. Chapter 3 discusses the proposed Methodology and its modules. Chapter 4 provides an in-depth discussion on the experimental environment and results. Chapter 5 concludes the thesis and provides the future prospects of the proposed approach.

Chapter 2

Literature Review

2.1 Introduction

To the current date, HAR is being carried out via two types of devices i.e., Vision-based devices and Sensor-based devices [44].

Depth data is acquired grounded on triangular and time-series techniques. Triangular motion is recorded by utilizing the vision-based devices that are capable of retrieving the depth data which is acquired by observing and recording a similar scene from multiple angles. Vision-based device encapsulates a generic rule which translates the depth as a difference between two similar angles. This may necessitate information regarding the confirmation of vision-based devices which may lead to a newly initiated analysis or change in the whole configuration of the system. To that end, a reasonable approach is based on the light patterns which differentiate between the objects so that the respective object depth can be extracted without a hassle. Moreover, scanners such as LiDAR can record the time that light takes to bounce off of an object (from the sensor back to the sensor) [45].

The milieu between different activities plays an important role to distinguish one activity from another. Tracking a person's movement duration can be very challenging in certain contexts. Either, the movement area of the person may be uneven in the recording and the environmental conditions can cast a shadow or

change the recorded results compared to reality. Observing one movement from multiple points of view may also cause different image obedience. Considering the restrictions of conventional vision-based devices which include the positional issues [46].

Multiple positioned cameras can be useful to cover simultaneous angles and overcome the standpoint issues, more likely in cases where obedience covering various angles can form a conformable slide. Avoiding the background dynamic lighting can be very helpful as it increases the complexity for differentiating backgrounds from the foregrounds. Rotating a camera can increase the complexions so it is better to use multiple cameras simultaneously. In vision-predicated behavior detection, the problem underlying the usage has to be handled explicitly [47].

Generally, movements are recorded in steady time slices. Feeding time-sliced data reduces the load on the HAR system however it needs an external data conversion and time-slicing module anteriorly. The drawback of this method is that it may not always be implementable. Several approaches have been discussed and taken into account this factor as well. Moreover, the accuracy of different activities may vary based on the training and the quality of data collected. Activity recognition rate importantly affects the fleeting degree of activity, particularly when more components are utilized. A robust human behavior detection system ought to be consistent to various paces of accomplishment, precision and naming the weakly labeled data. This gives a sound organization to correlation yet the sets hourly require a portion of the anteriorly referenced dataset. Several sensible datasets have been presented which incorporate labeled data accumulated from films or tapes. As the number of features is available in abundance, they're restricted in the quantity of training and test instances. Likewise, labeling these instances is troublesome and very costly in terms of human resources since there exists no such automated module for efficient labeling of instance [48].

An accelerometer encapsulates an electro-mechanical detector that is designed to measure the static and dynamic acceleration. Statics acceleration is the constant force acting on a body, like soberness or division. These forces are predictable and undeviating to a large extend. For the prototype, the acceleration due to

soberness is constant at 9.8 m/ s, and the soberness force is fair the same at every point on earth. Dynamic acceleration forces are non-uniform, and the semi-formal prototype is a vibration or shock. A motor crash is an excellent prototype of dynamic acceleration. Presently, the acceleration change is unforeseen when compared to its precedent state. The supposition behind accelerometers is that they can discover acceleration and convert it into measurable quanta like electrical signals [49].

Normally, activity takes place in a limited amount of time which is broken into smaller units to differentiate the movement patterns. Taking into consideration the timestamp and performance of the following units, an activity can be broken into segments as; compound exercise, succeeding activity, concurrent activity, and interleaved activity. Predicated behavior, conduct, and activity are filtered based on the ranking of patterns which can then be combined with milieu statistics. To this very day, even modern surveillance and security networks are manually monitored by humans. In actuality, these networks should be handled and maintained automatically without any human involvement. A hyping amount of networked security feeds and cameras dwindle the yield of an observer. To that end, technologically advanced organizations have been seeking a way to automate this process while avoiding the drawbacks of vision-based devices. Examples of such technology can be given as motion sensors embedded with cameras or infrared-based alarm generation such that a person may prompt towards the feeds that have detected unnatural observations [50]. Head-ways in the areas of physics have also extraordinarily consummated our satisfaction that is most certainly presumed in the ascent of quality life expectation. With the increase of cost for daily life needs which include medical and household needs, these technological advancements have provided us with immense time and money-saving measures which have made our surroundings productive and easy-to-deal-with overall. For some time, activity recognition has surfaced as an impactful solution for the disabled and infant care by providing intensive and continuous monitoring. Simple natural exertion is implemented across many hospital and healthcare organizations with positive results. Most of the living probabilistic strategies for activity detection may perform well and apply shallow machine learning algorithms. Notwithstanding, these methods

are way complicated than other conventional strategies because the computational cost increase as the data increase, moreover it is very difficult to fine-tune their hyper-parameters. Due to manual feature selection and fine-tuning, the relevance of the same actions must be considered however they fail to achieve so. Hidden Markov Models considers the features independent to better co-relate with other features, however, this method does not always work similar motions in ambiguous activities. However, the sequence of motion cannot be ignored because the data can be normalized and tagged but this causes the bias problem. To summarize, Hidden Markov Model fails to achieve better results where complex activities are concerned because it fails to distinguish between patterns, hence utilized in case of basic and static activities more [51].

The bias problem can however get overcome by multiple means. It can either be solved by selecting the most relevant dataset, by supervising the learning process, or by increasing the number of features. Another most common solution corresponds to the usage of smaller datasets with more prominent features. This will avoid overfitting as well. Conditional Random Field (CRF) performance is way ahead in natural language processing compared to Hidden Markov Model, however, when it comes to real-time implementation, they fail to achieve better results due to the inclusion of various parameters. Fine-tuning these parameters on sparse data can be a very challenging task and it also needs to be re-tuned if the data sale varies. To that end, it can be concluded that another goal of an activity recognition system is to verbatim dredge common natural activities in a real-time environment. For that specific purpose, numerous information-mining tactics are being used to vaticinate activities with perfection [52]. To that end, the existing literature is classified into two groups: approaches based on (a) basic activities and (b) transition activities. Both of these categories are further incorporate approaches based on conventional machine learning, deep learning, and hybrid approaches. Hybrid works incorporate both machine learning + deep learning and deep learning + deep learning approaches. We have not included the machine learning + machine learning approaches in our research work as we wanted to target deep learning approaches for HAR. The details are provided in the subsequent subsections.

2.1.1 Basic Activities

Wan et al. [53] demonstrated a CNN framework which showed that the fine-tuned conventional CNN still outperforms SVM, Multilayer Perceptrons (MLP), LSTM, and Bidirectional LSTM (BiLSTM) networks. Moreover, the results have shown a significant increase in classification accuracy compared to machine learning classification models such as DT, RF, etc. However, these approaches have limitations as they can only extract simple features. Zhou et al. [54] demonstrated a deep learning framework for weakly labeled data which was able to extract features well. The framework mainly accommodated an auto-labeling technique that was implemented on top of a HAR system and employed a distance-based reward rule strategy to label the data. The newly labeled data was fused with the strongly labeled data and passed through an LSTM module for feature extraction. This approach was focused on the labeling of unlabeled and weakly labeled data rather than the classification accuracy. To that end, it required a large dataset of unlabeled data for accurate labeling which resulted in an increased computational cost. Computational cost incubates the overall time a model takes to train and generates a successful set of predictions.

Activity recognition is considered to be a burdensome bailiwick corresponding to the experimenter owing to the difficulties faced in the activity detection and handling of several denizens generated. The first disquisition on HAR introduced it as a conventional movement detection issue. Traditional algorithms like ANN's, HMM's, SVM's have been considerably implemented concerning the pattern recognition systems, notwithstanding, the most recent works in activity recognition have changed course due to the introduction of deep neural networks. The conventional shallow learning approaches require efficient extraction of features which is dependent on the knowledge of the respective domain and feature engineering. This obstructs the development which also limits the transfer of one domain knowledge to another. Moreover, it's convenient for feting small degree conditioning akin to the conditioning of day-to-day livings, notwithstanding, it's nearly undoable for the conventional shallow learning algorithms to detect complex activities and detect the small movement pattern changes from it. This is due to the fact that

machine learning models strictly follow supervised learning where labeled data has to be fed to the models. Notwithstanding, deep neural networks learn the shallow and deep statistical features little by little which excluded the limitations of M.L algorithms [55].

Likewise, Convolution neural networks have been introduced which show superior performance in automatic feature detection and extraction as they can perform deep convolution on multiple layers of data. Activity recognition presents difficulties related to the analysis of actual movement scenarios, acknowledgment of molding for the availability of numerous dopeheads, diversity for perceived bias, and the detection of unrequired variables and parameters while deploying the same model for multiple data types. Normally huge pools of information are fed to deep learning models during the training because they perform the best when the number of observations is in abundance. Notwithstanding, it is not always compulsory or enough to only label the data as there can be a huge variance in the quality of reading as well. However, this issue can be resolved by transfer learning which is capable of co-relating various features and can extract the dependencies between them. Though it is a very useful method, however, it can cause an obvious increase in the cost of the system as they take a long time in the training of network when the amount of information is too vast. To that end, different approaches regarding transfer learning are being introduced to sift the domain information within multiple networks [56]. Recently, the use of edge-cutting technologies in executing behavior detection frameworks is in its juvenility environment. The literature emphasizes the requirement for re-processing of sensor-generated data in an efficient manner. Therefore, it explains in what manner calls are grounded and analyzed for several colored perspectives, alike perspectives of employment management systems, healthcare, human surveillance, education, and synthetic sectors. Substantial achievements have been grasped in the field of mobile edge computation environment by employing conventional machine learning algorithms, offering smart results in both real-time and predefined IoT applications. New uses for deep scholarship tactics have wax inchmeal probabilities based on several points, similar better than average reckoning faculty providing a refined collection of information. To achieve the goals, deep neural networks with dense layers are

utilized. The more the dense layers, the better the model will be able to extract quality features. However, it will have an impact on the overall computational cost of the approach. They can break down the information into several sub-classes and divide them into dense layers for a better understanding of features and generate required output [57].

To summarize this discussion, deep neural networks can achieve groundbreaking results in the areas of cloud and fog computing. Specifically in edge computing due to the division of layers such that it can be unloaded either with minimal mediator information targeted towards a waitperson. Another beneficial aspect of this methodology is the security of information during the communication between multiple networks. This method is very fast and efficient when dealing with long sequences of time-series information. Notwithstanding, longer sequences of data do cause some drawbacks such as the extraction of useful information becoming limited. This set of data after crossing a specific threshold, will not show any gain in the learning which will eventually drop the accuracy of the network. To handle longer sequences of data, the evolved networks LSTM are used. LSTMs are the updated and advanced versions of conventional RNN's. They encapsulate multiple gated units which control the flow of information while employing a memory cell. This memory cell is capable of storing long-term dependencies and hence LSTM's can better process and deal with longer sequences of time series data. With the sudden advance in knowledge and technology, the newer models have capabilities beyond the scope of conventional M.L networks, however, with this sudden advancement, there are issues sure to come such as limitations of current internet services, connectivity, and the major security portfolios which require utmost attention. We can utilize traditional machine learning models in a traditional setup where minor predictions are required. All in all, even in the private, public, and basic care environments such as homes, parks, offices, healthcare institutes, etc., IoT-based frameworks are a need that cannot be neglected [58]. Chen et al. [59] demonstrated an Attention Based BLSTM (ABLSTM) framework by introducing the concept of attention that assigns weights to features based on their importance for the current recognition scenario. The results showed superior classification accuracy compared to many recent approaches in both categories; shallow as well

as deep learning. Since it was a conventional BLSTM model with an Attention module, therefore it was compared with conventional machine learning works as well. All the experimental evaluations were based on the publicly available pre-processed data and no real-time data collection was performed which is of utmost importance during the testing of a signal-based system. Zhu et al. [60] demonstrated a Deep LSTM (DLSTM) architecture for feature recognition and filtration. Smartphone sensors were employed to train the model on labeled and unlabeled data for human activity recognition.

DLSTM encapsulated multiple LSTM layers between the I/O gates. The raw data was processed through the augmentation module to build the measure of information and the removal of Gaussian noise was initiated to filter irregularities in the final input. The DLSTM separated the low-level features which were dropped out and the high-level features were extracted. The unsupervised data loss embroiling the unlabeled data was calculated and labeled on the basis of predefined rules. Experimental evaluations on the proposed approach based on publicly available datasets showed superior results compared to several recent semi-supervised frameworks within a user-controlled environment. Xu et al. [61] fused a conventional RNN with Inception Neural Network (INN) model targeted at HAR based on wearable sensors to create InnoHAR. The INN architecture is composed of various deep layers consisting of multiple convolution layers that are parallel to pooling layers which is what forms the inception layer. The INN architecture is tested on multiple publicly available datasets and it portrayed superior performance compared to modern Deep-Convolutional-LSTM models. The drawback of this framework was the poor initialization of INN that required a lot of computation to be dealt with and minor changes could require the costly retraining phase to be repeated.

2.1.2 Transition Activities

In both machine and deep learning, tools such as MATLAB are used, which can break down the structures of M.L and D.L networks. They are capable of extracting and evaluating different applications and trends by the training of networks.

Where they learn based on different datasets. M.L networks are used for classification purposes where they train on the basics of real-world data and create classes that are used to categorize the data. Usually, the dataset is composed of raw sensor data which is converted into a sort of feature array before passing it to the network for training. The network, whichever it may be, learns of the basis of some mathematical evaluations and generated classes to actualize the data [62]. Machine Learning encapsulates deep learning, and the basic point of differentiation is the automated feature detection and extraction in deep learning networks. Just like machine learning, input has to be fed manually into the deep neural networks, however, once the input is fed and the training starts, deep neural networks automatically start the feature detection process and they can extract shallow as well as deep features from multiple layers of the input data. This data can either be statistical data, language data, or images. Machine learning is normally integrated concerning the programs incorporating prognosticating production or diving into mainstream applications where low-level pattern detection is required. However, the limits of these applications are convenient because fact that they are based on smaller data pools. To that end, machine learning models can make correct predictions but in a limited context and applications. Several machine learning algorithms have been designed to date which involves K-mean, Apriori, Markov model, KNN, Logistic regression, etc. [63]. In areas like, image recognition, image classification, signal processing, and natural language processing, D.L models are more widely used due to their adaptive nature and automated feature detection. Spatial and temporal features are usually extracted in these areas which require in-depth analysis of the dataset and a capable data classification model as these fields require feature matching and precise selection. Most often used deep learning networks involve CNN, MLPs, Self-Organizing Maps, Deep Belief Networks, LSTM's, Autoencoders, and Deep Q networks. However, machine learning models are way more convenient when the requirement is based on urgency rather than quality and performance as machine learning models can perform faster but smaller tasks and generate the output in a limited amount of time while spending a nominal computation cost. Because in neural networks, the amount and number of features play a major role in the overall computational time of the model. To that end,

for smaller tasks where higher accuracy with in-depth learning is not required, machine learning algorithms are most commonly used as they can reduce the cost, and computational time 10 folds compared to deep neural networks [64]. Deep neural networks require a significant amount of time for training and validation. In the context of real-time scenarios, pre-training deep learning models can be a very good approach. This can be achieved by integrating transfer learning within deep neural networks. To sum it up, deep learning models are not the be-all and end-all for classification problems as they can take a second as well as a month to train. This is strongly dependent on the amount and quality of data. Moreover, the nature of the model utilized plays an important role as well. Researchers who opt to experiment using the deep learning model should be very well aware of the fact that training a robust or deep model with a huge amount of data can take them a very long period to achieve results. Moreover, a neural network needs to be retrained on multiple parameters for it to perform well. Then again, all of this depends on the quality of data [65].

All of the fore mentioned works have a significance of their own, however, all of them are based on basic human actions. None of these works have discussed or employed the transition activities. Though Postural transition may not have to emphasize effect on the system due to their short duration and lower incidence, the validity of this statement is dependent on the application prospects. Shi et al. [66] proposed a standard deviation trend-analysis (STD-TA) based architecture to recognize transition activities. For the reduction of dimensions in data, "statistical features" were extracted and a conventional SVM was utilized for classifier training. The self-collected dataset was based on 8 basic daily life activities and 10 transitions.

Liu et al. [67] demonstrated a novel Transition-aware housekeeping task monitoring approach to outline the importance of activity transition events in housekeeping tasks related to elderly people. The self-generated dataset was divided into three parts which contained the basic housekeeping activities, inter-transition, and intra-transition activities. The approach was based on an SVM model with an embedded transition event detection module and it was able to achieve the

classification accuracy of 81.62%. Ahmad et al. [68] implemented a DBN-based approach that extracted features such as mean, median, auto-regressive coefficients, etc., from the raw data obtained from sensors. To make the features more robust, the features were further processed through a Kernal Principal Component Analysis (KPCA) and Linear Discriminant Analysis (LDA) unit. The proposed network was compared with SVM and ANN networks and has been shown to achieve an accuracy of 95.8%. Gusain et al. [69] proposed a transition-aware Gradient Boosted Decision Tree approach. They implemented incremental learning by utilizing ensembles of SVM. Batches of data were trained on frequent iterations but after the initial cycle of training, all the other cycles were trained on the incorrectly classified data. In the end, the weighted sum of all the machines is calculated and the accuracy of the whole system is computed and the accuracy of the whole system is calculated to be 94.9%. Yulita et al. [70] presented a hybrid model based on a classic KNN and SVM model where the SVM kernel was polynomial. Moreover, they imbued their approach with Radial Basis Function (RBF) and the Sigmoid function. After cross-validation, they managed to achieve an accuracy of 86%. These results were achieved due to the RBF kernel as it is a useful function to solve classification problems by finding non-linear classifiers. If not for the RBF kernel, the following method would not have been possible to show even average results. On the other hand, even with the RBF kernel, the approach does not show emphasizing results. Atrsaei et al. [71] designed a location-independent; postural-transition detection algorithm. Postural transitions were detected by the sensor following the vertical acceleration calculation and kinematic features were extracted to characterize the postural transitions. The approach was focused on the algorithm rather than the accuracy of the system. The proposed approach was independent of the placement of the sensor on the body and produced satisfactory results. Dan Setterquist [72] evaluated multiple networked LSTM's on a collected dataset of basic activities and postural transitions and managed to achieve 89% accuracy in a user-controlled environment. However, utilization of multiple LSTM units in a pipelined flow abruptly slows down the whole model and increases the complexity of the system.

In a more recent study, Wang et al. [73] presented a hybrid D.L method for activity

recognition in which multi-sensor data was passed through a CNN and the output was classified by the LSTM. The fundamental accomplishment of this approach is the activity transition identification alongside basic activities while emphasizing the fact that most of the research works do not employ the postural transitions however in human behavior recognition this is non-negligible, hence an important task to consider. The accuracy of real-time movement recognition is strongly dependent on the detection of postural transitions. The triaxial multi-sensor data was fused and fed to the multilayered CNN. The resultant feature matrix from the CNN was flattened and input to the LSTM module. LSTM was separately trained on the sensor data and a feature fusion was performed before the final classification. The benchmark showed superior results compared to state-of-the-art CNN, LSTM, CNN-BiLSTM, and CNN-GRU models on a publicly available dataset.

Taking into consideration the state-of-the-art technologies and innovations, the M.L trend is shifting from the traditional high-power consuming hardware devices to low-power mobile devices. This shift is referred to as TinyML [74]. This pretty much breaks the high-power consumption barrier in the areas of M.L. By bringing the inference to low-power devices, the responsiveness of the whole system can be increased while reducing the power consumption-based cost of the system. Banbury et al. [75] employed differential neural architecture search (DNAS) to bring forward a MicroNet model deployed on MCU which showed superior results on TinyML benchmark tasks which included audio-based keyword spotting, visual wake words, and anomaly detection. DNAS models were utilized due to their characteristics of requiring low MCU memory and energy. The lower consumption of energy itself is an efficient constraint. To that end, bringing forward a model with low energy consumption supported with a lower memory usage is a groundbreaking achievement that can revolutionized micro devices. The current limitations of TinyML are restricted to shrinking the size of the machine learning model, however, with the passage of time and advancement in technology lightweight neural networks are being designed which can take up to a few hundred KB's of space on the TinyML devices and produce substantial results. In prospect, TinyML can also play an important part in the applications of Augmented Reality (AR)

headsets that need to be kept powered on due to shared constraints. The discussed literature is summarized in Table 2.1. Therefore, the literature signifies the implementation of deep learning networks over conventional machine learning algorithms where the performance and accuracy of the system are concerned. For the introduction of an efficient and scalable HAR system, this paper introduces a novel hybrid model which takes both, basic activities and postural transitions (transition activities), into account. Accordingly, we integrated multiple deep learning models for feature extraction and proposed a decision fusion module for activity recognition. We then benchmarked the performance of the proposed approach on multiple datasets and compared them with the state-of-the-art works.

TABLE 2.1: Literature Summary.

Ref.	Model Type	Network	Accuracy (%)	Transition Activities	Weaknesses
[36]	Machine Learning	SVM + SFFS	96.80	No	Higher accuracy on smaller datasets—increase in data causes decrease in accuracy.
[66]	Machine Learning	STD-TA	80.00	Yes	A conventional SVM with an average accuracy that extracts statistical features to differentiate between transitional and basic activities.
[67]	Machine Learning	SVM-TED	81.62	Yes	A traditional SVM with a transition event detection module to detect postural transitions but lacks accuracy for efficient identification of an action.
[53]	Deep Learning	CNN	91.00	No	Requires strongly labeled data as well as increased features in data.

TABLE 2.2: Literature Summary - Continued from Previous Page.

Ref.	Model Type	Network	Accuracy (%)	Transition Activities	Weaknesses
[59]	Deep Learning	BiLSTM	87.50	Yes	Single BiLSTM unit cannot extract quality features from the input, no past information to correlate the data with. Works better on time series data.
[72]	Deep Learning	Multi-LSTM	89.00	Yes	Multiple pipelined LSTM units used in this approach, causing the network to train slowly and increasing the complexity of the whole model. Any fault or irregularity in a single LSTM unit affects the overall pipeline of LSTM units.
[68]	Deep Learning	DBN	95.80	Yes	DBN makes the network architecture more complex to train, and it has been replaced with ReLu, which better handles the vanishing gradient problem.
[61]	Hybrid	INN + RNN	94.00	No	INN has poor initialization, which makes it hard to debug, thus increasing the cost of the system. Moreover, a fine-tuned CNN can achieve the same or better performance than INN, which is no longer used in state-of-the-art systems.
[69]	Hybrid	GBDT	94.90	Yes	Gives best results on smaller datasets whereas accuracy decreases as the data increase.
[73]	Hybrid	CNN + LSTM	95.80	Yes	The model itself is complex and the CNN used is a conventional CNN with a basic three-layered structure that is not optimized at all. Complex activities and their transitions were not considered.

Chapter 3

Proposed Approach

3.1 Introduction

The architecture for the proposed approach consists of three deep neural networks: LSTM, BiLSTM, and CNN as depicted in Figure 3.1. Three D.L networks are utilized due to the imposition of the decision fusion module in the proposed approach. The proposed approach requires at least three machine learning algorithms to distinguish between the individual model results and efficiently implement the decision fusion module.

3.1.1 Model Selection

Since we targeted the deep learning approach, therefore we utilized deep neural networks. Specifically, the reason for the utilization of CNN was because CNN is computationally efficient and can generate better results with lower computational costs.

LSTM was utilized because they have been proven to perform the best on sequence data. As the data generated from the accelerometer and gyroscope is sequence data, to that end, LSTM can efficiently extract patterns and longer sequences from the sequential data with higher accuracy. Similarly, BLSTM works the same as LSTM but with the dual training of data in BLSTM, it trains faster and shows better classification results. Each deep learning model has its unique

characteristics, however, we can utilize any conventional machine or deep learning model in this approach in a plug-and-play manner.

3.1.2 Feature Conversion

We utilized 2 publicly available datasets to benchmark the proposed approach. One dataset includes basic and postural transitions whereas the second dataset is only based on basic activities. Each of these datasets incorporates a set of features against each activity instance. The raw tri-axial data from the accelerometer and gyroscope sensors has been converted into statistical features by employing FFT. FFT is a time-space domain algorithm that computes the discrete Fourier Transform of a function.

The main functionality of the Fourier Transform function is that it converts the domain of a signal from tie to frequency. This conversion is very useful in convolution operations and can speed up the training of CNN's. On the other hand, from a feature perspective, by converting the signal type from the time t frequency variable, we can further generate several features against a similar type of value.

The raw sensor data is converted into a feature matrix and fed to these models separately. Batch normalization is employed in all three networks to normalize the output of each layer [76]. After the classification results are retrieved from each model, a decision fusion module is initiated for the final classification. The details of the proposed approach are provided in the following sections:

3.1.3 Long Short Term Memory (LSTM)

The LSTM used in this approach is a standard unit contrived of a memory cell, input gate, output gate and a forget gate. The input is converted into a 1D vector of y elements and fed to the model for training and the number of Neurons is configured to be η. "Adam" is configured to be the adaptive optimizer as it performs the best with sparse data. Moreover, the learning rate of μ is adapted to achieve the best results while avoiding the loss of training input. A dropout

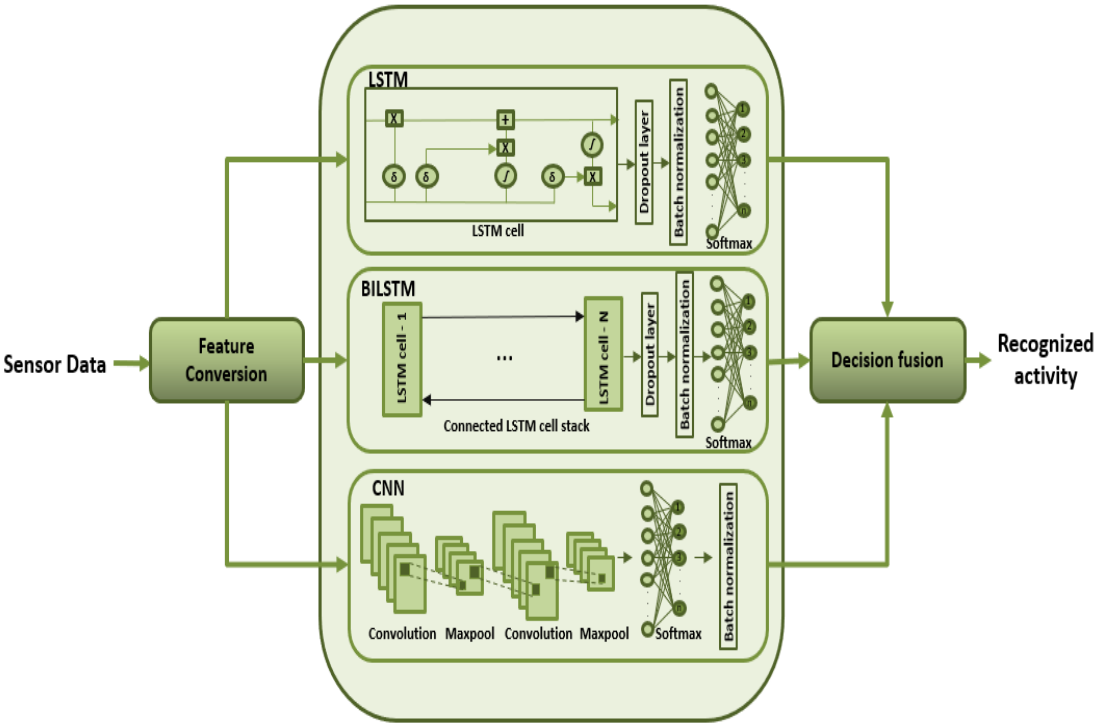

FIGURE 3.1: Architecture of the Proposed System.

rate of κ is used to avoid over-fitting while maintaining the integrity of input and output of neurons. Batch normalization is used after the fully connected layer to normalize the input of every layer in the model and the Softmax layer classifies the results. Table 3.1 shows all the hyper-parameters in LSTM and their respective values for the two datasets involved. The flowchart for LSTM is shown in Figure 3.3. The final output is flattened and normalized which gets classified by the Softmax in all networks.

Figure 3.2 shows the structure of the LSTM cell used. f_t represents the forget gate that tackles the amount of information to be kept and dropped. It is consumed by the *sigmoid* function which scales the values between 0 and 1 thus dropping the values (¡ 0.5). C_t represents the Input gate that quantifies the importance of the next input (X_t) and updates the cell state. The new input (X_t) gets standardized between -1 and 1 by the *tanh* function and the output gets point-wise multiplied by C_t. C_{t-1} represents the state of the cell at the previous timestamp which gets updated after each time step. The information required to update the cell's state is gathered at this point and a bit-wise multiplication is carried out between the previous cell state (C_T) and the forget vector. This gets followed by the bit-wise

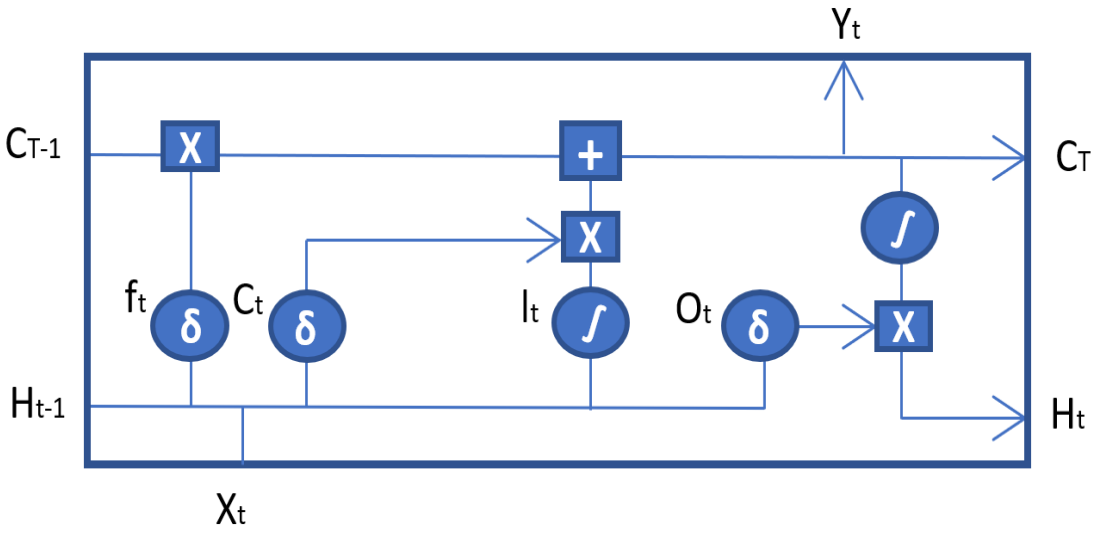

FIGURE 3.2: LSTM Unit

addition with the output of the Input gate and the cell state is updated. Finally, the output gate (O_t) figures out what the next hidden state should be. The hidden state encapsulates the information regarding previous inputs (H_{t-1}). The flow of information through the following gates is mathematically shown in equation (3.2).

$$\begin{aligned} f_t &= sigmoid(X_t * U_f + H_{t-1} * W_f) \\ C_t &= tanh(X_t * U_C + H_{t-1} * W_C) \\ O_t &= sigmoid(X_t * U_O + H_{t-1} * W_O) \end{aligned} \tag{3.1}$$

Where W and U represent the weights corresponding to their respective gates. Initially, the input is fed through X_t and for the first set of instructions to the LSTM, go straight to the Input gate (C_t). The input is standardized between 0 and 1 without any change in the size of the original input and forwarded to the cell state (C_{t-1}). It is to be noted here that one the first run, w do not have any information from the past on either the cell state (C_{T-1}) or the hidden state (X_{t-1}). Hence there are no point-wise operations performed for the first iteration. This is the reason that in the first epoch, the accuracy of every neural network is always very low. The same input is then transferred through the cell state and the hidden state. On the second batch of input, we have the information from the previous state available and after being fed to the LSTM cell, it is initially

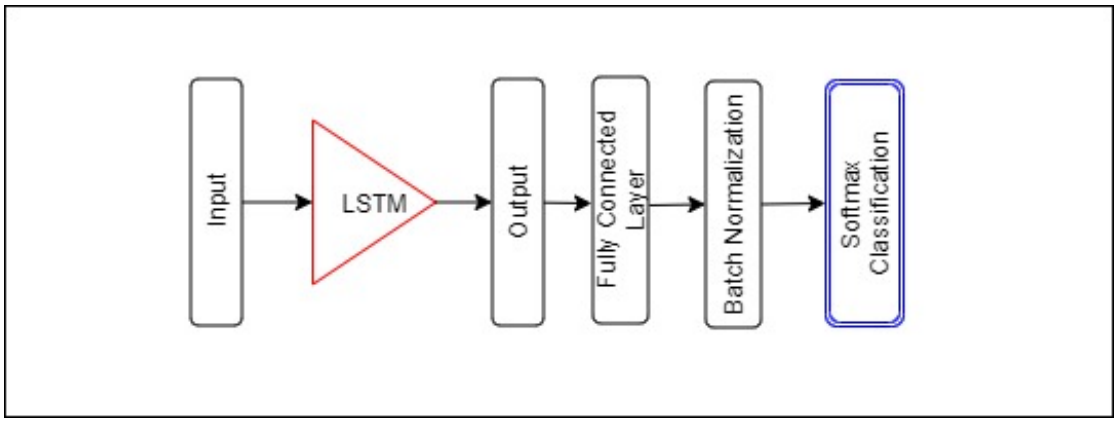

FIGURE 3.3: LSTM Flowchart

pointwise added with the information on H_{t-1} which is also input to the forget gate f_t to update the cell state. This time, the newer input and the previous hidden state also go through the update gate (l_t) where they are normalized between -1 and 1 instead of 0 and 1. This reduces the load on the LSTM unit and avoids over-fitting of the model.

The forget gate also dopes the values closer to 0 and only keeps the highest values i.e., values higher than 0.5. The whole process is repeated for every batch of input until we have a final output vector sequence from Y_t which is classified into labels by softmax. Before the final classification by softmax, batch normalization is also applied to further regularize the LSTM model and a dropout layer is used to drop random connections from each cell.

TABLE 3.1: LSTM Parameters

Parameter	**Value - Dataset A**	**Value - Dataset B**
y	561	60
ζ	0.002	0.002
η	100	50
Optimizer	Adam	Adam
κ	0.5	0.5
Epochs	400	100

3.1.4 Bidirectional Long Short Term Memory (BLSTM)

The BLSTM model utilized is based on dual recurrent layers. The BLSTM model utilized is based on dual recurrent (LSTM) layers 3.4. The topmost layer is referred

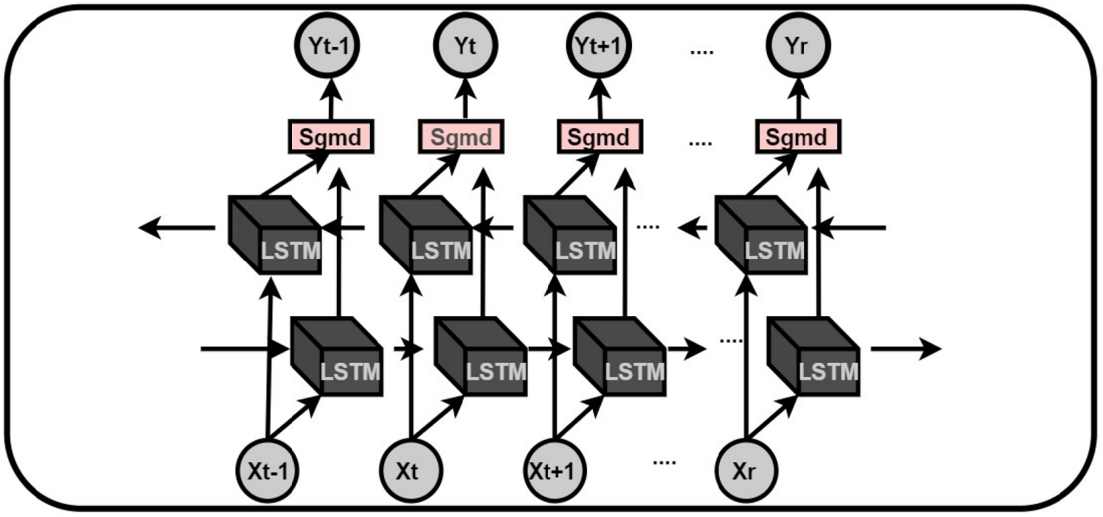

FIGURE 3.4: BLSTM Unit

to as the embedding layer that predicts the output for different time steps. The second layer is the forward LSTM (first recurrent) layer that takes the input in the forward direction. The third layer is the backward LSTM (second recurrent) layer that moves the input in the backward direction. The First recurrent layer runs the input from past to future while the second recurrent layer runs the input from future to past. The second recurrent layer is provided with the reverse sequence of input that preserves the future information. The additional training of data in the BLSTM model shows better results compared to the LSTM's.

The BLSTM hyper-parameters were kept the same as LSTM's to avoid any inconsistencies in the network and to track the changes in performance on multiple datasets. The following BLSTM had 2 LSTM layers incorporated with *ReLu* activation between them. The output from the second LSTM layer was flattened by the FC layer and a random 50% connections were dropped by the dropout layer to avoid overfitting in the network.

The dual training of data provides much better feature extraction and efficiently distinguishes the relation between features. BLSTM parameters are shown in table 3.2. It can be observed from the table that the only different parameter in both datasets is the input size. This was due to lesser number of features in dataset B. A more detailed discussion on datasets is provided in the subsequent Chapters. BLSTM's input and output flow is shown in Figure 3.5.

TABLE 3.2: BLSTM Parameters

Parameter	Value - Dataset A	Value - Dataset B
y	561	60
ζ	0.002	0.002
η	100	50
Optimizer	Adam	Adam
κ	0.5	0.5
Epochs	400	100

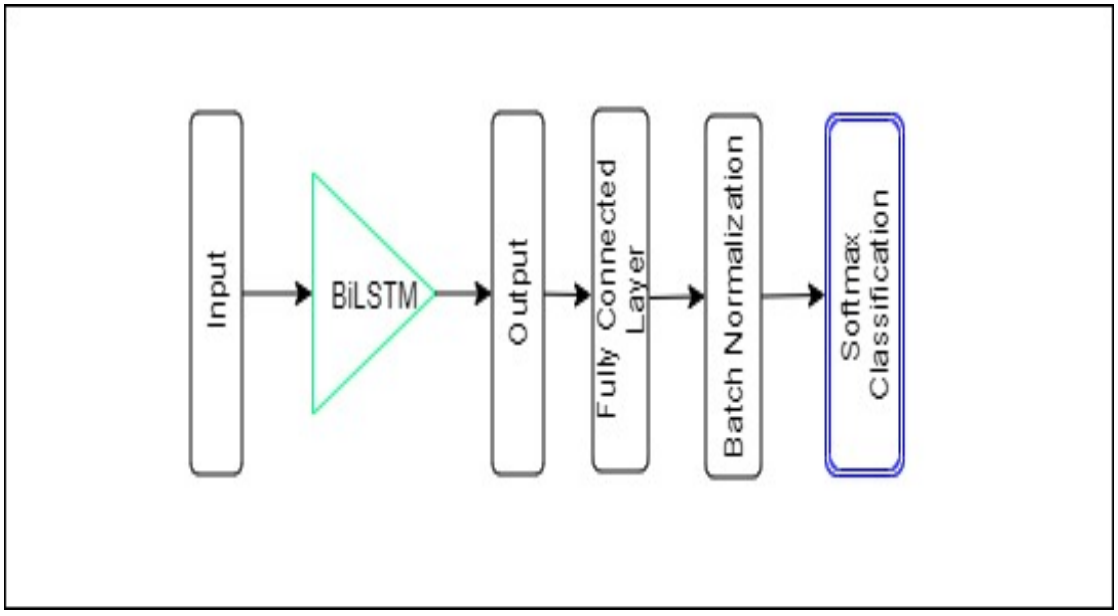

FIGURE 3.5: BLSTM Flowchart

3.1.5 Convolutional Neural Networks (CNN)

In this research work, a 2D-CNN is designed which takes input as a feature matrix "I". The CNN is comprised of an input layer, two hidden layers, a fully connected layer, batch normalization layer, and a softmax layer for classification as shown in Figure 3.6. Each hidden layer is a stack of "Convolution-ReLu-Maxpool" layers. The convolution layer outputs a feature map which is passed through the "Rectified Linear Unit (ReLu)" activation function (ϱ). ReLu is mathematically shown in equation 3.2. The output of ReLu becomes the input to the pooling layer. Amongst various types of pooling strategies, max-pooling selects the maximum element from each block in the feature map.

Max-pool is mathematically shown in equation 3.3 where F represents the max pool filter size, S represents the stride length and I represents the input. The

block to be covered depends on the pool size that is kept as γ. Padding is set to "SAME" and the stride is set as "s", such that the whole input block gets covered by the filter. The step size for both, convolution and pooling layers, is kept the same in all the hidden layers.

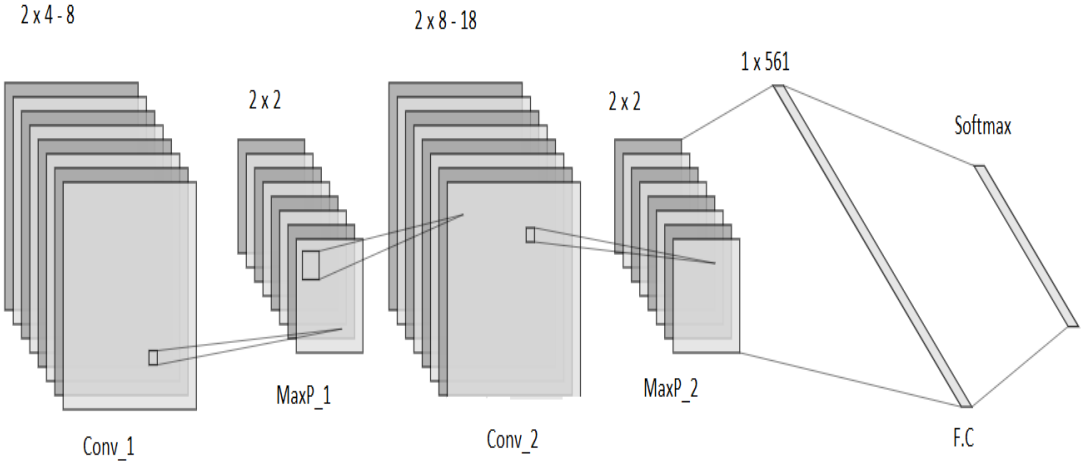

FIGURE 3.6: CNN Unit

$$f(x) = max(0, x) \tag{3.2}$$

$$P_{start}, P_{end} = \frac{[S[\frac{I}{S}] - I + F - S]}{2} \tag{3.3}$$

The number of Kernels in the two convolution layers are α and β respectively. The Kernel sizes are τ and ν respectively. Zero padding is added to fill the edges of the input matrix and a learning rate of ζ is adopted. The input to the convolution layer is of size $h \times w \times d$ where h represents the height of the input, w represents the width of the input and d refers to the dimension of the input. In this approach, the dimension of input is 0 as we are dealing with sparse sensor data. The convolution layer applies a filter of size $f_h \times f_w \times d$, where f_h denotes the filter height and f_w represents the filter width. The convolution layer outputs a volume dimension or feature matrix (f_m) as shown in equation (3.4).

$$f_m = (h - f_h + 1) \times (w - f_w + 1) \times 1 \tag{3.4}$$

The batch normalization layer is utilized after the fully connected layer to normalize the data in all the previous layers and the output is sent to the softmax layer for classification. The mean and variance calculation in batch normalization is shown in equations (3.5),(3.6), where x denotes the batch sample, μ_B represents the batch mean, and σ_B^2 represent the mini-batch variance. Table 3.3 shows all the parameters involved in CNN and their respective values. Figure 3.7 portrays the input flow in CNN.

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{3.5}$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2 \tag{3.6}$$

TABLE 3.3: CNN Parameters

Parameter	**Value - Dataset A**	**Value - Dataset B**
I	24x24	8x8
s	1	1
α	8	8
β	18	18
τ	2x4	2x4
ν	2x8	2x8
γ	2	2
ϱ	ReLu	ReLu
ζ	0.002	0.002
Epochs	50	50

3.2 Model Implementation

Input is fed to each model separately and activities are classified based on their respective labels. LSTM is utilized for its ability to achieve superior results in sequence-to-sequence classification. The LSTM is comprised of an input layer, hidden layers, and output layer. The hidden layers encapsulate memory cells and

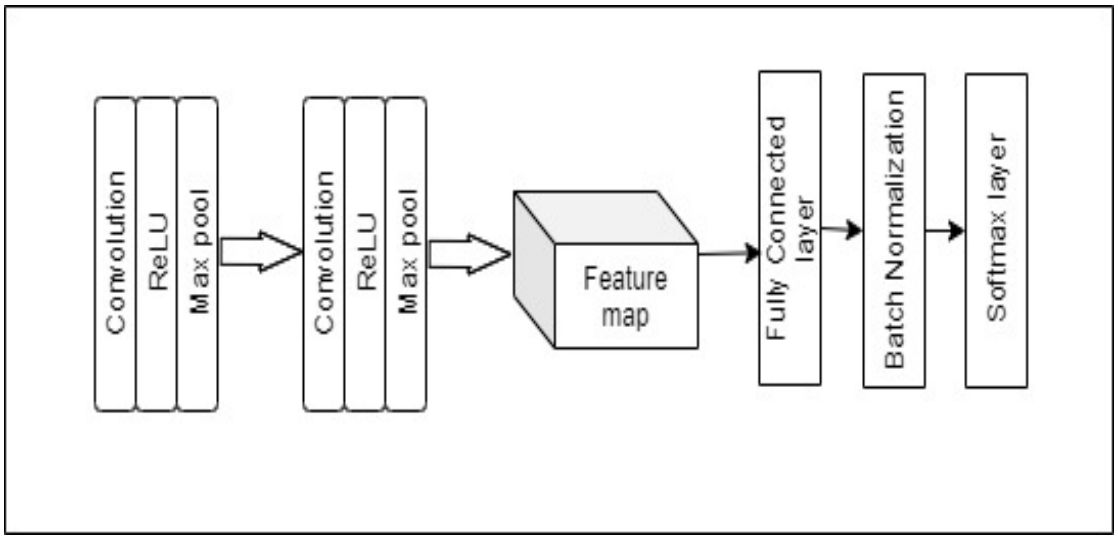

FIGURE 3.7: CNN Flowchart

gated units. The gated units are composed of input gate, forget gate, and output gate. A feature vector is fed to the input gate which contains an update gate that decides which values to update and a *tanh* layer that creates a vector of new values. The forget gate sorts out how much information can be aggregated from the previous gate into the memory cell. *Sigmoid* activation is used to scale the output from the gates between 0 and 1, to speed up the training and reduce the load on the network. The results of the output gate are generated based on the cell's state and flattened by the fully connected layer. All the parameters and input to layers are scaled and standardized by the batch normalization layer and the final output is classified by softmax.

The feature vector is passed to a BLSTM network. BLSTM has the same parameters and works on the same principles as an LSTM. The only point of difference is that in BLSTM, the input is fed to the model twice for training, once from beginning to the end and once from the end to the beginning. Therefore, by utilizing BLSTM, we can preserve information at any time at a point in the future and past which generates a refine feature map. Furthermore, BLSTM speeds up the training process and this dual training of data better classifies the activities compared to LSTM. The final feature map is flattened and classified by softmax. For the precise conversion of data into a matrix for CNN, zero padding is added to the input. Convolutions are performed on the matrix and weights are distributed amongst the filters. A bias is set to update the value of weights after a complete

iteration. The output from the convolution layer is passed through the *ReLu* to convert all the negative values in the resultant feature matrix to zero. Aside from ReLu, leaky ReLu is also used which is more balanced compared to the simple ReLu, however, the training does not stop even when the weight becomes 0 thus the training takes longer. The output of ReLu is input to the max-pool layer to reduce the size of the feature matrix and the overall parameters of the hidden layer. Amongst several pooling techniques; max-pool is the most effective while dealing with sensor data hence utilized in our approach. The feature map from the max pool layer is input to the second hidden layer, and the whole process is repeated twice. The final feature matrix is flattened by a fully connected (FC) layer to form a sequence vector. After the FC layer, the batch normalization layer is used to normalize the output of all layers of the CNN. The Softmax layer predicts polynomial probability distributions and generates categories based on these predictions. The sum of these probabilities is distributed within all the classes such that the sum of all the probabilities becomes 1.

Softmax is utilized in all models for its ability to generate statistical probabilities alongside classes. These probabilities are exerted in the decision fusion module for final classification. After all three networks are trained, the observations (instances) corresponding to each activity are input to the models and each network returns the predicted classes along with the class probabilities.

The predictions are then inter-compared and summed in a decision fusion module and final classification results are generated. The decision fusion module provides equal importance to each network and its predictions. The performance of the approach is then calculated on different benchmark parameters.

We have utilized a 2D CNN in our approach as we wanted to calculate the execution time of the proposed and baseline approach. To that end, we wanted to make our approach as fast as possible in terms of training time. Initially, we utilized a 1D CNN and calculated the accuracy and execution time for it. The same process was repeated for a 2D CNN as well. On a detailed comparison based on 10 iterations for both 1D and 2D CNN, we found almost the same accuracy values. However, in the case of execution time, we found the 2D CNN to be much faster

compared to the 1D CNN. To that end, we utilized the 2D CNN in the proposed approach. Moreover, since CNN performs the best on image type data which is usually 2D or 3D, hence applied in the proposed approach. One may think that image data has nothing to do with the proposed approach, however it should be observed that the one dimensional sensor data is converted to an image like matrix where each row from the datasets have been converted and then input to the model for training and testing.

3.2.1 Decision Fusion

The decision fusion module prioritizes the selection based on the class probabilities. Each returned class accommodates a probability value between 0 and 1 within all three networks. The resultant probabilities of each returned class from all networks are summed respectively and the highest value-based class is designated to be the final recognized class such that:

Let P_i^1 represent the probability of 1^{st} activity class in the i^{th} deep learning model, then the probabilities of the recognized activities $(P^1, P^2, ..., P^M)$ in $(P_1, P_2...., P_n)$ networks can be defined as $P_{total}^1, P_{total}^2, ..., P_{total}^M$ respectively, where n represents the total number of deep learning networks in the model. Then the sum of all the probabilities against each instance can be defined as 3.7:

$$P_{total}^j = \sum_{i=1}^{n} P_i^k, \tag{3.7}$$

where j= 1, 2, . . . , M. The cumulative probability of same resultant classes against each instance is calculated and compared with the cumulative probability of other resultant classes.

For the function $f : g \rightarrow j$, where g is a subset of j and contains the sum of same predicted classes from each model; j represents the set of all generated probabilities from all networks. k takes the argument max of all the values in g and returns the classified activity as shown in equation 3.8. Finally, the class associated with

the highest probability value k is returned as the recognized class. The algorithm for decision fusion is shown in Algorithm 1.

$$g = [P^1_{total}, P^2_{total}, P^3_{total}, ..., P^M_{total}]$$
$$k = \underset{g}{\operatorname{argmax}}\, f(g) \tag{3.8}$$

Algorithm 1 initially shows three trained networks as $CNet$, $LNet$, and $BNet$, where the first one represents the CNN trained network, the second one represents the LSTM trained network and the third one represents the BLSTM trained network. After all the networks are trained, the Testing data is loaded in a variable X, and initially the first instance $X[1]$ is fed to each network.

Each network returns the classification results in the form of labels and their statistical probabilities. The highest probability-based class from each network along with its probability is stored in the $ConfidenceArray$ and $CategoryArray$ respectively. Then the LSTM predicted class's probability is stored in the variable $A1$, the BLSTM's probability against the LSTM class is stored in $A2$ and the CNN's probability against the LSTM class is stored in variable $A3$.

The same process is repeated for BLSTM's and CNN's predicted class and all the probabilities are stored in $B1, B2, B3, C1, C2, C3$. The sum of all the probabilities against LSTM's label is stored in A, the sum of all the probabilities against BLSTM's label is stored in B and the sum of all the probabilities against LSTM's label is stored in C.

The sum of probabilities is stored in a newer matrix K and the argument max is calculated for this matrix. The maximum value along with its original label is stored in the array as $[M, I]$ where M represents the maximum probability and I represents its respective label. This resultant label/class is considered to be the recognized activity and the whole process is repeated for the second instance in X.

Algorithm 1 Algorithm for the Ensemble Decision Fusion

Ensure: train.CNN_Network = *CNet*
Ensure: train.LSTM_Network = *LNet*
Ensure: train.BLSTM_Network = *BNet*
Load TestingFeatures = X
Input X to each network separately
for $i \leftarrow 1$ to X **do**
[LSTMclass,LSTMprob]=classify(LNet,LFeatures(i))
[BLSTMclass,BLSTMprob]=classify(BNet,BFeatures(i))
[CNNclass,CNNprob]=classify(CNet,CFeatures(i))
ConfidenceArray = [max(LSTMprob) max(BLSTMprob) max(CNNprob)]
CatogoryArray = [LSTMclass BLSTMclass CNNclass]
end for
Probability Calculation:
$A1 \leftarrow LSTMprob(LSTMclass)$
$A2 \leftarrow BLSTMprob(LSTMclass)$
$A3 \leftarrow CNNprob(LSTMclass)$
$B1 \leftarrow LSTMprob(BLSTMclass)$
$B2 \leftarrow BLSTMprob(BLSTMclass)$
$B3 \leftarrow CNNprob(BLSTMclass)$
$C1 \leftarrow LSTMprob(CNNclass)$
$C2 \leftarrow BLSTMprob(CNNclass)$
$C3 \leftarrow CNNprob(CNNclass)$
return $A \leftarrow A1 + A2 + A3$
return $B \leftarrow B1 + B2 + B3$
return $C \leftarrow C1 + C2 + C3$
$\{A, B, C$ returns the summed probability of resultant class from each network$\}$
Argument Max:
$k \leftarrow [A\ B\ C]$
$\{k$ returns the argument max of A,B and C$\}$
$\{I$ represents the respective deep learning model$\}$
$[M, I] \leftarrow$max(k)
if $I \leftarrow 1$ **then**
$ClassifiedActivity(i,:) \leftarrow LSTMclass$
end if
if $I \leftarrow 2$ **then**
$ClassifiedActivity(i,:) \leftarrow BLSTMclass$
end if
if $I \leftarrow 3$ **then**
$ClassifiedActivity(i,:) \leftarrow CNNclass$
end if
$\{ClassifiedActivity$ populates a list of Activities $\}$
return *Classified Activities*

A dry run of the decision fusion module is shown in the subsequent section. The benefit of the proposed approach and decision fusion module is that it will generate good results on any type of data. Whether it be image data, textual data, or sparse data, the proposed approach incubates multiple deep learning models which can be used in a plug-and-play manner. To that end, even if one algorithm performs best on a certain type of data while the other only shows average results, e.g., if we want to process natural language processing cases, then the literature signifies the importance and superior performance of LSTMs for this particular domain. Even if CNN does not manage to return superior results, LSTM and BLSTM would cover that drawback and will generate results with higher accuracy which will impact the overall accuracy of the proposed approach. LSTM's show superior performance where time-series data is concerned which itself is a daunting task because there can exist several time lags between time series data.

The main motivation for the usage of LSTM is the handling of vanishing gradients in traditional RNNs. In the proposed approach we have utilized the most basic LSTM structure, however, there exist multiple LSTM variants such as deep convolutional LSTMs, peephole LSTMs, peephole convolutional LSTMs, etc. The usage of each of these variants depends on the type and quantity of data that we are dealing with. Aside from conventional LSTMs, convolutional LSTMs are also widely used due to their exceptional performance in variety of tasks such as deep feature extraction and unsupervised learning.

Seemingly, LSTM/BLSTM shows superior performance while dealing with time-series data, robot control systems, natural language processing, human action recognition, and music composition, etc. On the other hand, CNN outclasses LSTM in image classification, video classification, facial recognition, search engines, and healthcare data science, etc.

3.2.2 Results from Decision Fusion

Before diving into the detailed analysis of results from the datasets, let's assume Figure 3.8 as an illustration of a test instance on the proposed approach as shown

below.

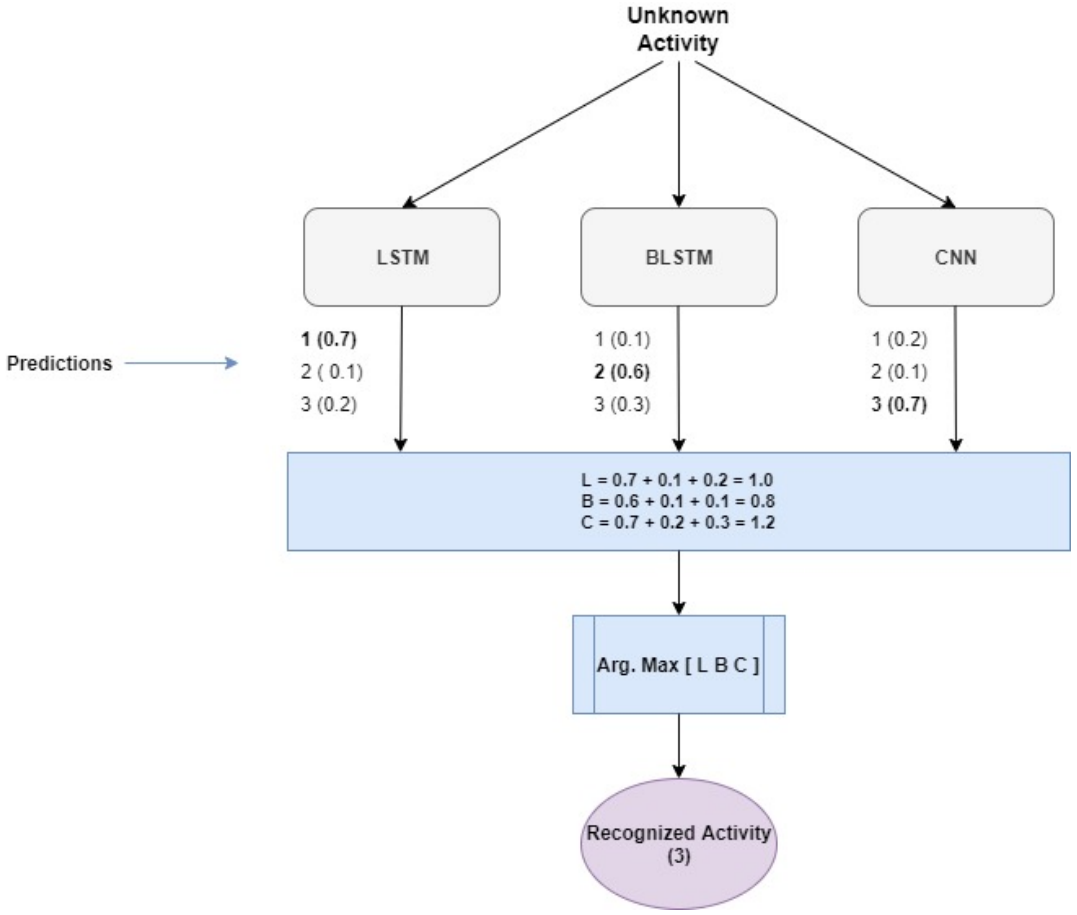

FIGURE 3.8: Illustration of the Decision Fusion Module

Suppose we input an unknown activity x to our model. We already have 3 networks (LSTM, BLSTM, and CNN) trained on the training data. Each network will generate a set of probabilities (with a weighted sum of 1.0 for all the class probabilities) for each class where the class associated with the highest probability value will be the predicted class from each network respectively. From the figure above, it can be observed that the LSTM network predicted x to be Activity 1, BLSTM predicted it to be Activity 2 whereas CNN predicted the activity to be Activity 3.

We have considered a worst-case scenario where each model generates a different prediction. In such a case, the Decision Fusion module fetches the predicted activity label from the first network, i.e., LSTM, and creates a variable L representing LSTM prediction. The module then fetches the probabilities of all the other networks in the model against the same label (in this case, label 1). The probabilities

of different networks (including LSTM's) against the LSTM predicted label are summed and stored in L.

Similarly, the decision fusion module fetches the BLSTM predicted label (in this case 2) and its probabilities from LSTM, BLSTM, and CNN and saves the sum of probabilities in a new variable B representing the BLSTM prediction. After this process is finished, the decision fusion module stores the weighted sums (L, B, and C) in a 1D matrix (in this case Arg. Max) and calculates the argument max of this matrix. Argument max returns the maximum value in the matrix along with its original label. Based on the output of $Arg.Max$, the unknown activity x is recognized in this approach.

Chapter 4

Experimental Results

4.1 Datasets

The choice of dataset plays an important role in the overall implementation and performance of a machine or deep learning network, hence selection of a dataset and generation of the dataset is a challenge on its own.

In the context where a limited amount of data is available and the data is not too dense, M.L is typically used. It will not be far stretched to make a statement that M.L algorithms are originally modeled to deal with supervised learning i.e., dealing with completely labeled data. In state-of-the-art M.L models, it is also possible to input unlabeled data with a semi-supervised learning-enabled environment, however to that end, the raw or informal data has to be converted or either transformed. This transformation can either be a dimensional transformation or standardization. Whereas the conversion of data refers to the generation of features such as mean, median, maximum, average, standard deviation, etc. This conversion can be carried out by using Fast Fourier Transformation (FFT) and input to the machine learning models by using the sliding window method. Conventional D.L models are based on deep neural networks; their performance is strictly dependent on the quality of data such that they require a huge amount of data for precise training and this may sometimes cause overfitting as well. e.g. CNN's can extract features from sensor data, however, in the case of sparse data,

they require a huge pool of training data for accurate classification. Moreover, if the training data is too similar, that causes overfitting in the neural networks. They can efficiently extract short-term dependencies but fail to extract longer dependencies from longer data sequences. To that end, LSTMs are used which are way ahead of CNNs in terms of extracting long-term dependencies and have superior performance in sequential applications like in natural language processing.

Two publicly available datasets are utilized for the experimental analysis of the proposed model. The following two datasets are selected based on the inclusion of transitional activities in the case of the former one and to validate the performance of the proposed approach with lesser input features for the latter one, respectively. Raw sensor data is transformed into a feature matrix for both datasets and then fed to the model. Next, we describe the two datasets that are as follows:

4.1.1 Dataset A: Human Activities and Postural Transition Dataset (HAPT)

HAPT [77] is an extended version of the UCI HAR dataset [78] and accommodates 6 additional postural transitions alongside 6 basic activities. Moreover, the dataset contains the unprocessed raw data composed of triaxial signals generated from a Samsung Galaxy II cellphone's embedded accelerometer and gyroscope sensors. The dataset also includes the fully processed data based on a 561-feature vector which is divided into a training set and test set. The gathered data is based on an experiment that involved 30 users performing 12 activities (6 basic and 6 transitions). The basic activities recorded are "walking, walking upstairs, walking downstairs, sitting, standing and lying". And the transition activities are "stand to sit, sit to stand, sit to lie, lie to sit, stand to lie, and lie to stand". The number of instances for transition activities are relatively lesser compared to those for basic activities because postural transitions are hard to record as they are physically tiring. An overview of the instances in dataset A is shown in Table 4.1.

TABLE 4.1: Human Activities and Postural Transitions Dataset (Dataset A) - Overview

Activity	**Training Instances**	**Test Instances**
Walking	1226	496
Walking Upstairs	1073	471
Walking Downstairs	987	420
Sitting	1293	508
Standing	1423	556
Laying	1413	545
Stand to Sit	47	23
Sit to Stand	23	10
Sit to Lie	75	32
Lie to Sit	60	25
Stand to Lie	90	49
Lie to Stand	57	27

The dataset was divided into a 70-30 proportion, where 70% was allocated for the training of networks and 30% for the testing of networks. It can be observed that the total number of instances for the activity 'Walking' sum up to 1722, however on the division between training and test set, the total instances are divided into 1226 and 496 for training and testing respectively. Similarly, other activities and their instances have been divided based on the same proportion. However, it can be observed that the number of instances for first 6 activities and last 6 activities are very unbalanced.

The choice of the HAPT data-set was due to the inclusion of 6 postural transitions alongside basic activities. Moreover, the dataset contains fully processed data where triaxial signals have been transformed into statistical features by using Fast Fourier Transform (FFT) which includes mean, standard deviation, max, min, etc.

It is to be noted that FFT only converts the variables from time to frequency domain which can lead to further feature extraction. A sample of the original data from the HAPT dataset is shown in table 4.2. The table shows the sample starting and ending points for each activity against a certain participant marked as Exp. ID. Each activity is represented by its respective label. The starting and ending point values are the duration in several seconds. A sample of the actual features in dataset A is shown in Figure 4.1. Each value in the block represents a statistical feature extracted from the time and frequency domain variables. The

features were converted from a *.txt* file into a *.csv* file for separation and padding before feeding them to neural networks. Figure 4.2 shows the actual training labels from dataset A.

TABLE 4.2: Activity Labels & Original Data

Exp. ID	User ID	Activity Label	Starting Point	Ending Point
2	1	2	15920	16598
2	1	3	16847	17471
2	1	2	17725	18425
3	2	5	298	16598
3	2	7	1399	1555
3	2	4	1686	2627
3	2	8	2628	2769
3	2	5	2770	3904
3	2	11	3905	4322

	A	B	C	D	E	F	G	H	I	J	K
1	0.04358	-0.00597	-0.03505	-0.99538	-0.98837	-0.93738	-0.99501	-0.98882	-0.95333	-0.7948	-0.74489
2	0.03948	-0.00213	-0.02907	-0.99835	-0.98294	-0.97127	-0.9987	-0.98331	-0.974	-0.80254	-0.73634
3	0.039978	-0.00515	-0.02265	-0.99548	-0.97731	-0.98476	-0.99641	-0.97583	-0.98597	-0.79848	-0.73634
4	0.039785	-0.01181	-0.02892	-0.99619	-0.98857	-0.99326	-0.99699	-0.98853	-0.99314	-0.79848	-0.75278
5	0.038758	-0.00229	-0.02386	-0.99824	-0.98677	-0.99312	-0.99822	-0.98648	-0.99382	-0.80198	-0.74651
6	0.038988	0.004109	-0.01734	-0.99744	-0.99349	-0.99669	-0.99752	-0.99349	-0.99692	-0.80198	-0.74337
7	0.039897	-0.00532	-0.02046	-0.99702	-0.97731	-0.98778	-0.9969	-0.97745	-0.98939	-0.80061	-0.74337
8	0.039082	-0.01605	-0.03024	-0.99666	-0.977	-0.98667	-0.99638	-0.97759	-0.98931	-0.80061	-0.74962
9	0.039026	-0.00741	-0.0273	-0.99743	-0.97319	-0.98818	-0.99749	-0.97156	-0.99016	-0.80025	-0.74202
10	0.040354	0.004245	-0.01793	-0.99491	-0.98118	-0.99005	-0.9953	-0.98248	-0.99092	-0.79972	-0.73341
11	0.03886	0.001515	-0.01626	-0.99492	-0.9814	-0.9894	-0.9954	-0.98267	-0.99064	-0.79972	-0.73341
12	0.038597	-0.0071	-0.0193	-0.99835	-0.99121	-0.99474	-0.99815	-0.99069	-0.99548	-0.80331	-0.74852
13	0.039495	-0.00609	-0.02219	-0.99924	-0.98946	-0.99731	-0.99897	-0.98865	-0.99744	-0.80331	-0.74698
14	0.038978	-0.00045	-0.01837	-0.99929	-0.99351	-0.9952	-0.99911	-0.99381	-0.99487	-0.80277	-0.74667
15	0.027045	-0.07104	0.00371	-0.75245	-0.59577	-0.50797	-0.80269	-0.62568	-0.57812	-0.4941	-0.46647

FIGURE 4.1: Sample of Actual Features from Training Data (Dataset A)

4.1.2 Dataset B: HumanActivity Dataset

This dataset [79], [80] contains 24,075 observations against 5 human activities (Sitting, Standing, Walking, Running, and Dancing). It can be observed that three activities in this dataset are similar as the ones in dataset A, however, 2 activities are different. Each observation corresponds to 60 features extracted from

	A	B	C	D	E	F	G
1	5						
2	5						
3	5						
4	5						
5	5						
6	5						
7	5						
8	5						
9	5						
10	5						
11	5						
12	5						
13	5						
14	5						
15	7						

FIGURE 4.2: Sample of Actual Labels from Training Data (Dataset A)

raw triaxial data generated through a smartphone's accelerometer and gyroscope sensor. The variables involved are as follows; *'actid'*- is a vector composed of activity Id's in the form of integers ranging from 1 to 5. *'actnames'* - is a vector composed of the activity names corresponding to their respective activity Id's. *'feat'* - is a feature vector composed of 60 features against every observation. *'featlabels'* - is a list of names corresponding to every feature. It is to be noted that the original tri-axial data has been converted into time-frequency domain variables in this dataset. The choice of this dataset was based on the availability of a vast pool of observations against five basic activities. This lead to a better validation of the proposed approach on basic activities. The dataset was divided into a 90-10 proportion for training and testing respectively. It should be noted that unlike dataset A, which was divided into a 70-30 proportion, this dataset is divided into a 90-10 proportion. The reason is that the number of instances for each activity in this dataset are way greater compared to those in dataset A. This dataset is very refined and contains more instances but lesser features. Therefore to keep a somewhat balanced number of instances for testing, such proportion was considered. An overview of the instances in dataset B is shown in Table 4.3.

TABLE 4.3: HumanActivity Dataset (Dataset B) - Overview

Activity	Training Instances	Test Instances
Sitting	5265	585
Standing	5598	622
Walking	4856	540
Running	3561	395
Dancing	2388	267

From the table above, it can be observed that the total number of instances for the 'Sitting' activity was 5850, however on the division between training and testing sets, the number of instances got divided into 5265 and 585 for training and testing respectively. The same division had been carried out for all the activities and their instances in the dataset.

A sample of actual features in dataset B is shown in Figure 4.3. Each value in the block represents a statistical feature. The features were converted from a *.txt* file into a *.csv* file same as dataset A for separation and padding before feeding them to neural networks. Figure 4.4 shows the actual training labels from dataset B.

	A	B	C	D	E	F	G	H	I	J
1	0.764034	0.585142	-0.25694	0.109121	0.083456	0.036783	0.381036	-1	0.067645	0.222877
2	0.764064	0.585018	-0.25694	0.109209	0.083319	0.036554	0.381653	-1	0.066487	0.222145
3	0.76401	0.585013	-0.25708	0.108962	0.083465	0.036516	0.379924	-1	0.066038	0.222926
4	0.764093	0.584778	-0.25729	0.109039	0.083582	0.036911	0.380466	-1	0.066519	0.22355
5	0.764137	0.584614	-0.25732	0.109041	0.083301	0.03666	0.380481	-1	0.066725	0.222048
6	0.764109	0.584732	-0.25736	0.108991	0.08349	0.036442	0.380126	-1	0.066385	0.22306
7	0.764071	0.584624	-0.25744	0.10893	0.083629	0.036744	0.379702	-1	0.066502	0.2238
8	0.764054	0.584379	-0.25727	0.108988	0.083582	0.0367	0.380109	-1	0.066985	0.22355
9	0.76406	0.58434	-0.2572	0.109007	0.083567	0.03673	0.380239	-1	0.065682	0.223471
10	0.764035	0.584394	-0.25718	0.109088	0.083431	0.036489	0.380806	-1	0.067179	0.222745
11	0.764079	0.584344	-0.25728	0.109164	0.083399	0.036609	0.381335	-1	0.067122	0.22257
12	0.763993	0.584298	-0.25721	0.108981	0.083558	0.036821	0.380059	-1	0.066006	0.223423
13	0.763974	0.58426	-0.25706	0.10893	0.083182	0.036834	0.379703	-1	0.066621	0.221413
14	0.7641	0.584369	-0.25687	0.109156	0.083352	0.036809	0.381281	-1	0.066789	0.222321
15	0.764114	0.584303	-0.25673	0.108946	0.083359	0.036686	0.379817	-1	0.066623	0.222361
16	0.764174	0.584394	-0.25667	0.108978	0.083141	0.036466	0.380039	-1	0.067285	0.221197
17	0.764232	0.584481	-0.25676	0.109025	0.083351	0.03667	0.380365	-1	0.065847	0.222317
18	0.764217	0.584435	-0.25677	0.109046	0.083709	0.036779	0.380515	-1	0.066867	0.224231
19	0.764293	0.584383	-0.25685	0.109204	0.083409	0.036809	0.381617	-2.2	0.023058	0.222625
20	0.764245	0.584369	-0.25675	0.108949	0.083239	0.036589	0.379837	-1	0.066748	0.22172

FIGURE 4.3: Sample of Actual Features from Training Data (Dataset B)

	A	B	C	D	E	F
1	1					
2	1					
3	1					
4	1					
5	1					
6	1					
7	1					
8	1					
9	1					
10	1					
11	1					
12	1					
13	1					
14	1					
15	1					
16	1					
17	1					
18	1					
19	1					
20	1					
21	1					

FIGURE 4.4: Sample of Actual Labels from Training Data (Dataset B)

4.2 State-Of-The-Art Approaches

Table 4.4 shows the average recognition rate (accuracy) of various transition-aware approaches. It can be observed that the standard deviation-based trend analysis fused with an SVM can achieve satisfactory results (80%) and achieve almost the same accuracy (81.62%) as the SVM infused with a transition event detection module. Both approaches are based on conventional machine learning models and fail to achieve increased performance on relatively bigger datasets. Comparatively, gradient boosted decision trees outperformed (94.90%) both approaches by calculating the sum of weights from dual training of correctly and incorrectly classified data. The approach is based on a fine-tuned SVM, however, the performance and the accuracy of the proposed approach degrades as the data is increased. This is called the curse of dimensionality when too much data tunes the model to memorize data and causes over-fitting. Consequently, all three SVM exhibits a common limitation corresponding to decreased performance with this increase in data. To fill the gaps, the deep learning approach utilizing multiple LSTM units,

TABLE 4.4: Comparison of the Proposed Approach with State-Of-The-Art Approaches in Terms of Average Accuracy.(Dataset A)

Approach	**Average Accuracy(%)**
STD-TA [66]	80.00
SVM-TED [67]	81.62
LSTM [72]	89.00
GBDT [69]	94.90
DBN [68]	95.80
CNN-LSTM [73]	95.80
Proposed	96.11

to classify transitional activities, has achieved an accuracy of 89% as compared to conventional SVM approaches. However, the deep structure of LSTM with multiple networked cells slowed down the overall model, causing a vanishing gradient and have increased the computational cost. To this end, another approach utilized DBN to handle the vanishing gradient problem and extracted features from raw sensor data. The extracted features were refined by a component analysis Kernel and analyzed by the LDA unit. The LDA unit made substantial contribution in the final accuracy of this approach however it only works best when the amount of data is relatively lesser. Experimental evaluations have shown the DBNs to be much superior with a 95.80% accuracy. DBNs show superior performance in deep feature extraction. However, DBNs have become obsolete due to their complex structure that has been replaced by a much simplified ReLu unit, which has been introduced to handle the vanishing gradient problems in neural networks.

To overcome the limitations of the fore-mentioned approaches, the CNN-LSTM approach demonstrated a pipelined model by feeding the CNN extracted features to LSTM for refinement and feature fusion. The predicted results showed superior and equal results (95.80%) compared to the fore-mentioned state-of-the-art research works. Every approach has a significance of its own, however lacking either in scalability, i.e., having a complex structure, or any irregularity in the pipeline architecture can halt or slow down the system. Suppose if the features extracted from the first model in the pipeline architecture are not quality features and fail to show better results, then the second algorithm in the pipeline flow will fail to extract quality features and final feature matrix will be meaningless. To that end, the proposed approach brings forward a non-pipeline flow of algorithms

which works without any dependency on each other and shows results based on a biased view.

The above table shows the individual activity accuracy, precision, recall, and f-measure calculated on dataset A that incorporates transitional activities alongside basic activities 9 (a total of 12 activities, 6 basics, and 6 transitions respectively). The table below shows the same for dataset B which incorporates basic activities only. The difference in the performance parameters values can be observed in both of the tables. From the dataset samples, it can be observed that for dataset B we have abundant instances per activity in dataset B so the networks had more data to train on. This leads to a robust training of networks on a higher number of instances. Moreover, deep neural networks show the best performance when the number of observations or datasets, in general, is relatively large. To that end, training on a smaller number of features and a larger dataset proved to show much better results compared to that of being trained on a smaller dataset with a higher number of features.

4.3 Quantitative Analysis

4.3.1 Individual Model Results

Before incorporating all employed neural networks into the architecture, we implemented and tested each model individually. To calculate the overall accuracy of each model individually on dataset A and to show the difference in results compared to the decision fusion module, we initially fed the training and testing data to the models with different hyper-parameters. The results based on average accuracy for each model are explained in the subsequent sections.

4.3.1.1 LSTM Accuracy

The LSTM model takes as input a one-dimensional feature vector and processes the input through a single LSTM cell. To keep the complexity of the overall architecture less, only a single LSTM cell was used. Moreover, since we had an

abundant number of features against each instance in the datasets, one LSTM cell managed to achieve better results on test data in a shorter amount of time. The average execution time for ten complete iterations was calculated to be 8 units where a single unit of time represents one minute. The results on testing data are shown in Table 4.5.

It can be observed that the number of epochs was initially configured to be 100 because of the longer sequences of input data. An increment of 100 epochs was initially made which showed a drastic increase in the accuracy up to 300 epochs. After the 300 thresholds, the increment was configured to be 50 up to 450 epochs. It can very well be observed that the accuracy started decreasing on the mentioned number of epochs, hence it was lowered in units of 10 and we managed to achieve the highest accuracy with the number of epochs configured to 400. For the BLSTM, a similar experimental trial was conducted and the number of epochs were configured and will not be discussed in the subsequent section.

The table shows the average accuracy on one dataset as the proposed approach was mainly to be benchmarked on dataset A due to the inclusion of transition activities.

TABLE 4.5: LSTM Average Accuracy on Different Epochs - Dataset A

Epochs	**Accuracy(%)**
100	91.12
200	92.64
300	92.81
350	94.01
400	94.77
450	94.12
420	94.18

4.3.1.2 BLSTM Accuracy

In our BLSTM unit, we utilized two LSTM layers. The initial input to the first LSTM layer was also a one dimensional feature vector which was reprocessed in the second LSTM cell. The results on testing data showed slightly better results compared to LSTM in terms of average accuracy as shown in Table 4.6.

TABLE 4.6: BLSTM Average Accuracy on Different Epochs - Dataset A

Epochs	**Accuracy(%)**
100	93.66
200	93.91
300	93.89
350	94.64
400	94.98
450	94.78
420	94.72

It can be observed from the table above classification accuracy of BLSTM is relatively higher compared to LSTM's. This is because of the dual training of data in bi-directional LSTM. Moreover, bidirectional BLSTM takes lesser time to train compared to LSTM, making it computationally less expensive in terms of time compared to LSTM.

4.3.1.3 CNN Accuracy

In the CNN model, we initially implemented a 1D CNN. The classification results showed almost the same accuracy as achieved in the 2D CNN. However, the execution time taken by the 2D CNN was lesser on the same number of epochs compared to a 1D CNN. The only difference in a 1D and 2D CNN is the movement of convolution kernel on the input array and matrix. This was achieved due to the fact that a 2-dimensional kernel covered more blocks on each input matrix and performed faster convolutions compared to a one-dimensional kernel. This could have been overcome in a one-dimensional CNN by increasing the size of the kernel, however, this reduced the overall classification accuracy of the CNN model. Due to the availability of longer sequences in the datasets against each instance, the number of kernels and their sizes had to be configured in a balanced proportion for the model to be able to extract quality features from the input. As one of the main contributions of the proposed approach was to be computationally efficient in terms of time, this led to the utilization of the 2D CNN. Moreover, the classification results on testing data showed superior results compared to LSTM and BLSTM in terms of average accuracy and execution time, as shown in Table 4.7. There also exists a 3D CNN, however it is strictly used for video-based datasets or cases where real time video data is being utilized.

TABLE 4.7: CNN Average Accuracy on Different Epochs - Dataset A

Epochs	**Accuracy(%)**
100	95.33
150	94.78
90	95.37
80	95.46
70	95.19
60	95.51
50	95.55
40	95.11

It can be observed from the table above that CNN showed better results in a lesser number of epochs compare to LSTM and BLSTM. Initially, the number of epochs was configured to be 100, the same as LSTM and BLSTM. However, on an increment of 50 epochs, the accuracy started decreasing. To calculate the highest accuracy value, we decremented the epochs by 10 till the number of epochs reached 40. After 50 epochs, the accuracy again started to decrease. To that end, the final number of epochs were selected to be 50 which showed the highest accuracy (95.55%) on test data. The overall number of epochs play an important role in the training of a network. If the number of epochs is too high, the model may suffer from overfitting, in a similar manner, if the number of epochs are too low, the model suffers from underfitting and the training accuracy degrades. Hence it is best to utilize deep learning models when the number of instances in training data are in abundance. If very limited amount of data is available, then it is better to use some conventional machine learning algorithm which keeps the overall architecture lightweight and computationally efficient.

LSTM and BLSTM were utilized in this approach due to their exceptional performance in sequence-to-sequence classification alongside sequence to label classification. As accelerometer and gyroscope generate sequence data, hence we have utilized LSTM and BLSTM in our approach. This sums up the reason for the utilization of these three networks in the proposed approach. It is to be noted that for a different type of data, i.e., image, video, textual, etc., we can incorporate multiple machine learning algorithms in the proposed approach in a plug and play manner. We can also utilize a three dimensional CNN for extracting video sequences from real time video data.

TABLE 4.8: Accuracy, Precision, Recall and F-measure of Various Activities - Dataset A

Activity ID	**Accuracy(%)**	**Precision(%)**	**Recall(%)**	**F-measure(%)**
A1	99.34	97.00	99.00	98.00
A2	99.11	98.00	96.00	97.00
A3	99.56	99.00	98.00	98.00
A4	98.13	96.00	92.00	94.00
A5	98.36	94.00	97.00	95.00
A6	100.00	100.00	100.00	100.00
A7	99.49	62.00	78.00	69.00
A8	99.97	91.00	100.00	95.00
A9	99.75	84.00	90.00	87.00
A10	99.59	73.00	76.00	75.00
A11	99.46	80.00	82.00	81.00
A12	99.46	70.00	56.00	62.00

TABLE 4.9: Accuracy, Precision, Recall and F-measure of Various Activities - Dataset B

Activity ID	**Accuracy(%)**	**Precision(%)**	**Recall(%)**	**F-measure(%)**
B1	99.92	100.00	100.00	100.00
B2	99.88	100.00	100.00	100.00
B3	99.58	99.00	99.00	99.00
B4	98.71	96.00	96.00	96.00
B5	98.67	93.00	95.00	94.00

4.3.2 Final Classification Results

Quantitative analysis of the proposed framework has been carried out against state-of-the-art approaches using the metrics: Accuracy, Precision, Recall, and F-measure. The resultant metrics of the recognized activities from 'dataset A' are shown in Table 4.8. The activity id labels "$A1, A2, A3, A4, A5, A6$" represent the basic activities "Walking", "Walking Upstairs", "Walking Downstairs", "Sitting", "Standing", "Lying" respectively; and the labels "$A7, A8, A9, A10, A11, A12$" represent the postural transitions "Stand to Sit", "Sit to Stand", "Sit to Lying", "Lying to Sit", "Stand to Lying", "Lying to Stand" respectively. It can be observed that the basic activities $(A1, ..., A6)$ achieve average precision of 97.33%, average recall of 97%, and an average F1 score of 97%. However, transitional activities $(A7, ..., A12)$ have shown reduced average precision of 76.66%, average recall of 80.33%, and an average F1 score of 78.16%. These results are not consistent with the results obtained for the basic activities.

The reason is the unavailability of abundant observations in the dataset for transition activities which cause over-fitting. Overfitting is the phenomenon of memorizing the seen data (in the case of the small dataset) and consequently, the model would be unable to generalize on unseen data. Compared to basic activities, the total number of observations recorded for transition activities is significantly lesser which caused our proposed model to over-fit. Deep neural networks perform better when the volume of data is larger, however, in this case, the volume of transitional activities varied a lot compared to basic activities which lead to less-than-stellar results. However, the accurate final classification can be observed by the average accuracy of 96.11% which outperformed all the referenced state-of-the-art methods and portrays the overall performance of the proposed approach. Compared to the baseline work, the accuracy gain may not seem much significant, however, the main idea behind the proposed framework was that equal and also higher accuracy can be achieved by not following the traditional pipeline architecture and implementing multiple machine learning models simultaneously. We also showed that we can integrate multiple machine and deep learning models as per the requirement if the hardware resources are in abundance.

The resultant metrics of the recognized activities from 'dataset B' are shown in Table 4.9. The activity id labels "$B1, B2, B3, B4, B5, B6$" represent the basic activities "Sitting", "Standing", "Walking", "Running", "Dancing" respectively. It can be observed that the classified activities show an average precision of 97%, average recall of 98%, and an average F1 score of 97.80%. The proposed approach showed an average accuracy of 98.38% with a higher recall, precision, and F1 score compared to dataset A. The comparative analysis of basic activities in both datasets shows superior classification results on dataset B. The reason is the higher number of observations for individual activity in dataset B compared to dataset A. This leads to robust training of networks in the latter case. Therefore, even though the number of features per observation was significantly more in dataset A, results based on dataset B were superior. Table 4.10 shows the average accuracy of the proposed approach evaluated on the two publicly available datasets. These datasets were selected based on the inclusion of transitional activities and due to the availability of benchmark datasets.

Table 4.11 shows the confusion matrix corresponding to the final classification of activities from dataset A. The diagonal bold entries represent the correctly identified instances of the activities $A1, A2, ..., A12$ respectively. It can be observed that 490 out of 496 instances of $A1$ were correctly identified in the final classification, meanwhile, 3 instances were predicted to belong to class $A2$ and 3 from class $A3$. Similarly, 12 instances from $A2$ and 3 instances from $A3$ were incorrectly predicted to belong to $A1$. To this end, the column entries (excluding the Bold ones) represent the incorrectly classified instances of those particular activities (Bold entries) and the row entries represent the incorrectly classified instances of the bold entries. Moreover, it can also be observed that the number of observations for transition activities is considerably lesser compared to basic activities. Similarly, Table 4.12 shows the confusion matrix relating to the final classification of activities from dataset B, where the diagonal entries represent the correctly classified instances of the activities $B1, B2, ..., B5$.

It can be observed let the classification accuracy of activities on dataset B is greater than that of the activities on dataset A. The reason is that the total number of instances in dataset B is much greater than the total number of instances in dataset A. The total number of activities in dataset A was 12 and the number of instances for those activities, both training, and testing summed up to approximately 11000. Moreover, the total number of instances for transition activities was way lesser compare two basic activities. Meanwhile, the number of activities in data Set B was 6, however, the total number of instances for those activities summed up to approximately 24000. This led to a robust training of networks on dataset B. For a time-based comparison, the CNN-LSTM approach was reproduced according to the base parameters defined in the approach. The raw data were converted into a feature matrix and fed to CNN for feature extraction. The CNN employed was a multilayered network with three hidden layers. Each hidden layer was a stack of convolution and ReLu layers. The output feature matrix from the third pooling layer was passed to LSTM for further feature filtration. LSTM trained separately on input as well and the LSTM extracted features were fused with CNN features to create a robust trained model. Finally, the test set was input to the model for validation and benchmark. The experimental results achieved similar accuracy as

shown in the referenced approach. The baseline accuracy hand computational time was recorded on the same hardware requirements as mentioned in the referenced paper. The computational environment may be different however we tried our best to reproduce the baseline work as it is.

Furthermore, a comparison of the average execution time of the proposed model was carried out with the CNN-LSTM approach on dataset A. Heading forward with explanation, we have represented 5 minutes (300 seconds) as 5 units of time. Figure 4.5 exhibits the difference in execution time of both approaches on 10 iterations(X-axis) where a single iteration refers to one complete execution (predictions) of each approach. Moreover, each label (0, 5, 10, ..., 60) on the Y-axis represents a difference of five units. It can be observed for the first iteration that the CNN-LSTM approach took 52 units of time for one complete execution whereas the proposed approach took only 18 units of time while employing three deep learning models. Similarly, after 10 iterations, the average execution time for the CNN-LSTM approach is calculated to be 51.50 whereas the proposed approach demonstrates an average execution time of 17.20 units. The number of iterations were kept as 10 because after 10 iterations there was no such change in the execution time. This shows that the proposed approach is computationally efficient in terms of average accuracy and execution time compared to baseline work.

The execution time was only calculated on dataset A as the baseline approach had utilized the same dataset in their architecture. Moreover, a detailed comparison had to be made with the baseline approach only and the parameter 'accuracy' was not enough to emphasize the results of the proposed approach on the baseline work. To that end, we reproduced the baseline approach according to the mentioned parameters. A similar hardware environment was also set up and utilized to precisely reproduce the results without any deviation. The software tools may be different as the one used by the baseline approach was not mentioned in the text, however, the language and the machine learning libraries utilized were the same.

A TensorFlow environment in Python was utilized with 16GB of ram and a dual core processor in the baseline and the proposed approach. The software tool

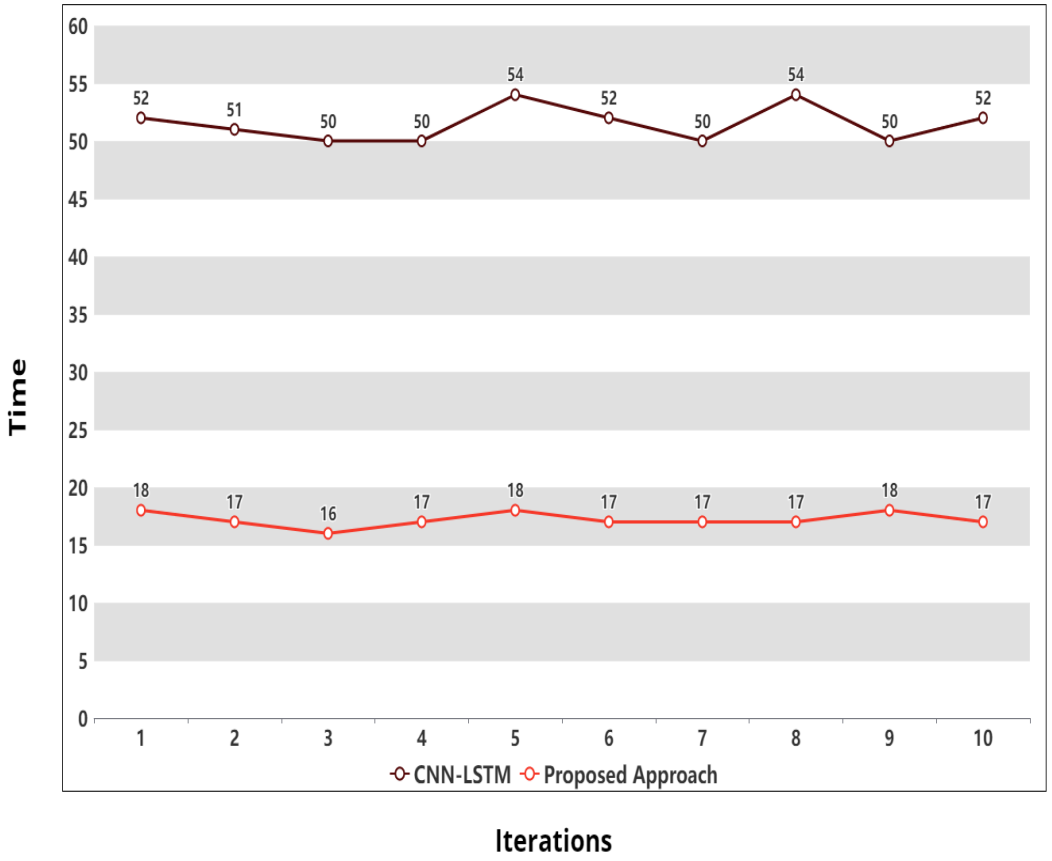

FIGURE 4.5: Execution Time of The CNN-LSTM and Proposed Approach (Dataset A)

TABLE 4.10: Average Accuracy of The Proposed Approach on Two Datasets

	Average Accuracy(%)	
Proposed Approach	Dataset A	Dataset B
	96.11%	98.38%

utilized was the 'Pycharm - community edition' in the proposed approach. We also visualized the baseline approach and the proposed approach in MATLAB software.

The overall results on both tools showed a slight variance in results. To that end, we strictly followed the baseline environment for the overall experimentation and achieved the same results.

Moreover, the selection of the baseline paper was based on the most recent works in human activity recognition related to deep learning environment. This fact led to the filtration of a few state-of-the-art works and the baseline was selected as it provided in-depth detail of implementation as well.

TABLE 4.11: Confusion Matrix of Activities—Dataset A.

		Predicted											
		A1	**A2**	**A3**	**A4**	**A5**	**A6**	**A7**	**A8**	**A9**	**A10**	**A11**	**A12**
Actual	A1	490	12	3	0	0	0	0	0	0	0	0	0
	A2	3	454	7	0	0	0	1	0	0	0	0	0
	A3	3	1	410	0	0	0	0	0	0	0	0	0
	A4	0	0	0	467	15	0	2	0	0	0	1	0
	A5	0	0	0	35	540	0	1	0	0	0	0	0
	A6	0	0	0	0	0	545	0	0	0	0	0	0
	A7	0	4	0	6	1	0	18	0	0	0	0	0
	A8	0	0	0	0	0	0	1	10	0	0	0	0
	A9	0	0	0	0	0	0	0	0	27	0	5	0
	A10	0	0	0	0	0	0	0	0	0	19	2	5
	A11	0	0	0	0	0	0	0	0	3	0	36	6
	A12	0	0	0	0	0	0	0	0	0	6	0	14

TABLE 4.12: Confusion Matrix of Activities—Dataset B.

		Predicted				
		A1	**A2**	**A3**	**A4**	**A5**
Actual	A1	584	1	0	0	0
	A2	0	619	7	0	0
	A3	1	2	536	3	0
	A4	0	0	0	378	14
	A5	0	0	4	14	251

4.3.2.1 Discussion on Confusion Matrices

From table 4.11 and 4.12, it can be observed that, after the decision fusion module, the number of correctly identified instances has drastically increased compared to individual model results. This shows that due to the inclusion of the decision fusion module, the performance of the proposed approach increased and the overall model managed to return accurate predictions against test cases. This can very well be true for real time data as well if used in the proposed approach.

While this may very well be true for the basic activities, however, it can be observed that the number of incorrectly predicted instances for the transition activities also are in abundance. One can assume that the reason for this is the lesser number of observations for transition activities in the dataset for the training of the models. This fact is supported by the literature that deep neural networks perform the best when the number of training data is available in abundance. However, in this specific case, the number of observations for transition activities, in total, for both training and testing was way lesser compared to basic activities. There were also other datasets available with more depth features to benchmark the proposed approach however none of them incorporated transition activities.

Due to these limitations, the models were unable to train efficiently for the transitional activities however due to the inclusion of the decision fusion module, the weighted sum of probabilities was calculated which provided equal importance to each network's predictions and managed to return high precision and accuracy for the overall transitional activities as well. On the other hand, the confusion matrix for activities on dataset B shows a different story. As dataset B incorporated 5 basic activities, different from dataset A, the results showed a better activity recognition rate along with the correctly predicted activities. The reason, as stated before, is the availability of an abundant number of observations against each activity. For dataset A with 12 activities, the average number of observations for each activity (training and testing both) can be roughly considered as 1500 for basic activities and almost 100 for transition activities. Whereas for dataset B, the average number of observations against each activity can be considered as 4800.

It is to be noted that the total number of features for each instance in dataset A was 561 time-frequency domain variables, whereas, for dataset B, each instance incorporated only 60 variables.

This indicated that the models were trained much better on dataset B as they had a greater number of instances per activity to build a more efficient network. However, the issue of lower accuracy due to a lesser number of instances was covered via the decision fusion module by providing equal importance to each network's predictions.

4.3.3 Strengths & Weaknesses

Based on the analysis in the previous sections, the strengths of the proposed approach can be summarised as following:

1 - The proposed approach is scalable.

2 - The proposed approach generates biased predictions and provides equal importance to individual model results.

3- Multiple machine learning and deep learning models can be incorporated in the framework in a plug and play manner. As stated above, the models can either be conventional deep learning algorithms or deep learning models.

The only drawback of the proposed approach is that if multiple models (3+) are integrated in the framework, the overall system becomes more complex and the costly in terms of computational time. Moreover, depending on the type of models being used, hardware resources may need to be upgraded as well. This will overall increase the cost of the system.

4.3.4 Other Applications in HAR

A common example of a generalized RMS is Microsoft's Remote Desktop client. A user can easily access a Remote Computer from any other self-authorized system while being anywhere globally. One can access his/her workplace system remotely from home as if they were in front of their workplace system. More applications of RMS include (but are not limited to) structure monitoring, power plants, network operation centers, airports, smart grids, etc. The main goal of RMS is to provide a semi or fully automated system which can manage, maintain and monitor a specific set of tasks efficiently over a network with reduced cost. This network can be an IoT system or a local network system with a series of connected devices. RMS reduces the cost associated with Manual Monitoring Systems in multiple ways. They decrease the cost of data-gathering as manual data gathering requires multiple workers to perform this task and chances of errors can be quite abundant. However, RMS can perform this task in an automated way which requires fewer personnel and provide an efficient error observation system to avoid and remove

errors. Moreover, RMS are scalable and provide multiple opportunities to implement change. RMS utilizes certain devices to form a network, these devices can either be vision-based devices such as cameras or sensor-based devices such as Accelerometer, Gyroscope, etc. The selection of devices for the RMS is dependent on the environment and requirements. Alongside commercial and household applications, RMS is also being used in sensor-based technologies with applications such as Radars, Satellites, Airplanes, etc. Radar Systems are based on sensors that emit radiations that are scattered or bounced back when they collide with an object which helps in the interpretation of that object. Another impactful application of RMS with sensors is Remote Health Monitoring (RHM). Real-time Health monitoring of patients by a doctor from a remote location has a great impact on the avoidance of irregularities and providing First Aid within the nick of time. RHM shows great promise especially when it comes to elderly patients and physically disabled patients. Different types of wearable sensors or health-monitoring sensors such as heart rate sensors, pulse sensors, Oxygen sensors, Blood pressure sensors, etc. are used in open or closed environments to observe the patients. Any kind of abnormality in the patient's behavior prompts the caretaker or doctor which enables them to take certain measures as soon as possible.

Vision-based sensors are also used to monitor the Health Conditions of patients. A camera is mounted near the patient's vicinity which keeps track of the patient's movement and if the system detects any abnormal movement by the patient, it prompts an alarm to notify the caretaker. However, vision-based devices do have some shortcomings such as environmental limitations, camera angle, lighting and contrast limitations, etc. Similarly, sensor-based devices also have shortcomings of their own such as magnetic interference, faulty sensors, etc. Major work has been done for RHM and newer efficient systems are still being designed to overcome the shortcomings of current systems.

Though various RMS applications are easy to implement, they also produce a lot of data that must be analyzed and exploited to attain certain results. Many of these applications usually run-on web servers and require continuous sending and retrieving of data which causes a delay in the service, and when it is concerned with

Real-time monitoring of patients; this delay can be impactful to life-threatening scenarios. To avoid this persisting problem, an efficient RMS is analyzed based on a layered structure. The layers usually consist of Edge computing, Fog computing, and Cloud computing Cloud computing provides large-scale data processing and computing over the internet. There is no need to manage and maintain local or online servers for data management and processing. Cloud servers are used to handle and compute the data over the internet which reduces the cost and increases the efficiency of RMS. Cloud-based monitoring enables effective remote monitoring and smart resource scheduling by removing delays and data communication issues. Many cloud-based health monitoring systems have been presented to overcome the limitations of manual server-based data communication. A cloud-based server for data communication and processing was also introduced. The data was gathered by a cellphone embedded sensor and sent to the cloud-based server to be accessed by the healthcare workers. Though there are many advantages in migrating to cloud-based servers, so are some concerns. The vast distance between multiple devices can cause high latency in data communication which can cause problems for IoT apps that require low latency. Security and privacy are also major concerns as the data is globally communicated through different channels along with other users, so it may cause data loss and is not prone to cyberattacks.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

This thesis proposed and demonstrated the working of an ensemble multi-model framework for efficient recognition of basic as well as transitional activities. The proposed framework utilized multiple deep learning models i.e., LSTM, BLSTM and CNN followed by a decision fusion module for the final classification of activities. The proposed approach has been tested on the publicly available datasets for both, basic and transition, activities and compared with other state-of-the-art approaches employing the same datasets.

The results exhibited that the proposed approach outperformed the referenced approaches by achieving the classification accuracy of 96.11% on the HAPT dataset and 98.38% on the HumanActivity dataset with transition and basic activities respectively.

Furthermore, it is to be noted that conventional machine learning algorithms and deep earning models can be embedded into the proposed framework in a plug and play manner such that the required models can be incorporated easily in it. The proposed model was created and experimented on in a Python environment with Tensorflow and Keras.

The hardware utilized was a core i5 10th gen processor with 8GB RAM and an Nvidia 1650ti GPU. The algorithm can be used to easily reproduce the work done

and it can run on a conventional system without explicit use of GPU. GPU was utilized in our case to speed up the training in case of a much higher number of epochs, however, it is not at all explicitly required to reproduce the proposed approach.

5.2 Future Prospects

For forthcoming research, the proposed approach can be transformed into a parallel architecture to further improve the processing speed for real-time implementation while putting some effort into compiling a dataset consisting of complex transition activities. Furthermore, different and more M.L & D.L models can be added to the architecture and data can be divided into clusters to train each model on different types of data.

Bibliography

[1] Kurt Bollacker, Natalia Dıaz-Rodrıguez, and Xian Li. "Extending knowledge graphs with subjective influence networks for personalized fashion". In: *Designing Cognitive Cities*. Springer, 2019, pp. 203–233.

[2] Lulu Chen, Hong Wei, and James Ferryman. "A survey of human motion analysis using depth imagery". In: *Pattern Recognition Letters* 34.15 (2013), pp. 1995–2006.

[3] Oluwatoyin P Popoola and Kejun Wang. "Video-based abnormal human behavior recognition—A review". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42.6 (2012), pp. 865–878.

[4] Shoya Ishimaru et al. "Towards reading trackers in the wild: detecting reading activities by EOG glasses and deep neural networks". In: *Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers*. 2017, pp. 704–711.

[5] Oresti Banos et al. "Daily living activity recognition based on statistical feature quality group selection". In: *Expert Systems with Applications* 39.9 (2012), pp. 8013–8021.

[6] Jun-Huai Li et al. "Segmentation and recognition of basic and transitional activities for continuous physical human activity". In: *IEEE access* 7 (2019), pp. 42565–42576.

[7] Lukas Köping, Kimiaki Shirahama, and Marcin Grzegorzek. "A general framework for sensor-based human activity recognition". In: *Computers in biology and medicine* 95 (2018), pp. 248–260.

[8] Djamila Romaissa Beddiar et al. "Vision-based human activity recognition: a survey". In: *Multimedia Tools and Applications* 79.41 (2020), pp. 30509–30555.

[9] Sarah Makhlouf and Ali Abdulshahed. "Internet of Things for Libya Healthcare System: Challenges and Issues". In: Dec. 2020.

[10] Seena Naik and E Sudarshan. "Smart healthcare monitoring system using raspberry Pi on IoT platform". In: *ARPN Journal of Engineering and Applied Sciences* 14.4 (2019), pp. 872–876.

[11] Kate Hoye and Fred Pries. "'Repeat commercializers,'the 'habitual entrepreneurs' of university–industry technology transfer". In: *Technovation* 29.10 (2009), pp. 682–689.

[12] Adithyan Palaniappan, R Bhargavi, and V Vaidehi. "Abnormal human activity recognition using SVM based approach". In: *2012 international conference on recent trends in information technology*. IEEE. 2012, pp. 97–102.

[13] KG Manosha Chathuramali and Ranga Rodrigo. "Faster human activity recognition with SVM". In: *International Conference on Advances in ICT for Emerging Regions (ICTer2012)*. IEEE. 2012, pp. 197–203.

[14] Zhen-Yu He and Lian-Wen Jin. "Activity recognition from acceleration data using AR model representation and SVM". In: *2008 international conference on machine learning and cybernetics*. Vol. 4. IEEE. 2008, pp. 2245–2250.

[15] Lin Sun et al. "Activity recognition on an accelerometer embedded mobile phone with varying positions and orientations". In: *International conference on ubiquitous intelligence and computing*. Springer. 2010, pp. 548–562.

[16] Sadiq Sani, Nirmalie Wiratunga, and Stewart Massie. "Learning deep features for kNN-based human activity recognition." In: CEUR Workshop Proceedings. 2017.

[17] Sadiq Sani et al. "kNN sampling for personalised human activity recognition". In: *International conference on case-based reasoning*. Springer. 2017, pp. 330–344.

[18] Paulo JS Ferreira, João MP Cardoso, and João Mendes-Moreira. "KNN prototyping schemes for embedded human activity recognition with online learning". In: *Computers* 9.4 (2020), p. 96.

[19] Tasweer Ahmad et al. "Using discrete cosine transform based features for human action recognition". In: *Journal of Image and Graphics* 3.2 (2015), pp. 96–101.

[20] Zhenyu He and Lianwen Jin. "Activity recognition from acceleration data based on discrete consine transform and SVM". In: *2009 IEEE International Conference on Systems, Man and Cybernetics*. IEEE. 2009, pp. 5041–5044.

[21] Aziz Khelalef, Fakhreddine Ababsa, and Nabil Benoudjit. "A simple human activity recognition technique using DCT". In: *International Conference on Advanced Concepts for Intelligent Vision Systems*. Springer. 2016, pp. 37–46.

[22] Lin Fan, Zhongmin Wang, and Hai Wang. "Human activity recognition model based on decision tree". In: *2013 International Conference on Advanced Cloud and Big Data*. IEEE. 2013, pp. 64–68.

[23] Thomas Phan. "Improving activity recognition via automatic decision tree pruning". In: *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*. 2014, pp. 827–832.

[24] Zengtao Feng, Lingfei Mo, and Meng Li. "A Random Forest-based ensemble method for activity recognition". In: *2015 37th annual international conference of the ieee engineering in medicine and biology society (embc)*. IEEE. 2015, pp. 5074–5077.

[25] Nurul Retno Nurwulan and Gjergji Selamaj. "Random forest for human daily activity recognition". In: *Journal of Physics: Conference Series*. Vol. 1655. 1. IOP Publishing. 2020, p. 012087.

[26] Long Cheng et al. “Recognition of human activities using machine learning methods with wearable sensors”. In: *2017 IEEE 7th annual computing and communication workshop and conference (CCWC)*. IEEE. 2017, pp. 1–7.

[27] Uwe Maurer et al. “Activity recognition and monitoring using multiple sensors on different body positions”. In: *International Workshop on Wearable and Implantable Body Sensor Networks (BSN'06)*. IEEE. 2006, 4–pp.

[28] Nishkam Ravi et al. “Activity recognition from accelerometer data”. In: *Aaai*. Vol. 5. 2005. Pittsburgh, PA. 2005, pp. 1541–1546.

[29] Tâm Huynh and Bernt Schiele. “Analyzing features for activity recognition”. In: *Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies*. 2005, pp. 159–163.

[30] Paul Viola and Michael Jones. “Rapid object detection using a boosted cascade of simple features”. In: *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*. Vol. 1. Ieee. 2001, pp. I–I.

[31] Kenji Kira and Larry A Rendell. “A practical approach to feature selection”. In: *Machine learning proceedings 1992*. Elsevier, 1992, pp. 249–256.

[32] Juha Reunanen. “Overfitting in making comparisons between variable selection methods”. In: *Journal of Machine Learning Research* 3.Mar (2003), pp. 1371–1382.

[33] Dimitrios Ververidis and Constantine Kotropoulos. “Fast and accurate sequential floating forward feature selection with the Bayes classifier applied to speech emotion recognition”. In: *Signal processing* 88.12 (2008), pp. 2956–2970.

[34] Girish Chandrashekar and Ferat Sahin. “A survey on feature selection methods”. In: *Computers & Electrical Engineering* 40.1 (2014), pp. 16–28.

[35] Mi Zhang and Alexander A Sawchuk. “A feature selection-based framework for human activity recognition using wearable multimodal sensors.” In: *BodyNets*. 2011, pp. 92–98.

[36] Nadeem Ahmed, Jahir Ibna Rafiq, and Md Rashedul Islam. "Enhanced human activity recognition based on smartphone sensor data using hybrid feature selection model". In: *Sensors* 20.1 (2020), p. 317.

[37] Li Deng and Dong Yu. "Deep learning: methods and applications". In: *Foundations and trends in signal processing* 7.3–4 (2014), pp. 197–387.

[38] Yayin Xu et al. "Machine learning in construction: From shallow to deep learning". In: *Developments in the Built Environment* (2021), p. 100045.

[39] Wei Li et al. "On improving the accuracy with auto-encoder on conjunctivitis". In: *Applied Soft Computing* 81 (2019), p. 105489.

[40] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. "Understanding of a convolutional neural network". In: *2017 International Conference on Engineering and Technology (ICET)*. Ieee. 2017, pp. 1–6.

[41] Jun Wang. "Analysis and design of a recurrent neural network for linear programming". In: *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications* 40.9 (1993), pp. 613–618.

[42] Klaus Greff et al. "LSTM: A search space odyssey". In: *IEEE transactions on neural networks and learning systems* 28.10 (2016), pp. 2222–2232.

[43] Jessica R Michaelis et al. "Describing the user experience of wearable fitness technology through online product reviews". In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. Vol. 60. 1. SAGE Publications Sage CA: Los Angeles, CA. 2016, pp. 1073–1077.

[44] Jake K Aggarwal and Lu Xia. "Human activity recognition from 3d data: A review". In: *Pattern Recognition Letters* 48 (2014), pp. 70–80.

[45] Mohammad Arif Ul Alam et al. "LAMAR: Lidar based Multi-inhabitant Activity Recognition". In: *MobiQuitous 2020-17th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*. 2020, pp. 1–9.

[46] L Minh Dang et al. "Sensor-based and vision-based human activity recognition: A comprehensive survey". In: *Pattern Recognition* 108 (2020), p. 107561.

[47] Allah Bux, Plamen Angelov, and Zulfiqar Habib. "Vision based human activity recognition: a review". In: *Advances in computational intelligence systems* (2017), pp. 341–371.

[48] Emiro De-La-Hoz-Franco et al. "Sensor-based datasets for human activity recognition–a systematic review of literature". In: *IEEE Access* 6 (2018), pp. 59192–59210.

[49] Tomas Brezmes, Juan-Luis Gorricho, and Josep Cotrina. "Activity recognition from accelerometer data on a mobile phone". In: *International Work-Conference on Artificial Neural Networks*. Springer. 2009, pp. 796–799.

[50] Pragati Garg, Naveen Aggarwal, and Sanjeev Sofat. "Vision based hand gesture recognition". In: *World academy of science, engineering and technology* 49.1 (2009), pp. 972–977.

[51] LR Rabiner and BH Juang. "Hidden Markov models for speech recognition—strengths and limitations". In: *Speech recognition and understanding*. Springer, 1992, pp. 3–29.

[52] Girija Chetty, Matthew White, and Farnaz Akther. "Smart phone based data mining for human activity recognition". In: *Procedia Computer Science* 46 (2015), pp. 1181–1187.

[53] Shaohua Wan et al. "Deep learning models for real-time human activity recognition with smartphones". In: *Mobile Networks and Applications* 25.2 (2020), pp. 743–755.

[54] Xiaokang Zhou et al. "Deep-learning-enhanced human activity recognition for Internet of healthcare things". In: *IEEE Internet of Things Journal* 7.7 (2020), pp. 6429–6438.

[55] Lu Zhang et al. "From machine learning to deep learning: progress in machine intelligence for rational drug discovery". In: *Drug discovery today* 22.11 (2017), pp. 1680–1685.

[56] Hao Du, Yuan He, and Tian Jin. "Transfer learning for human activities classification using micro-Doppler spectrograms". In: *2018 IEEE International Conference on Computational Electromagnetics (ICCEM)*. IEEE. 2018, pp. 1–3.

[57] Paulo JG Lisboa. "A review of evidence of health benefit from artificial neural networks in medical intervention". In: *Neural networks* 15.1 (2002), pp. 11–39.

[58] Haoxi Zhang et al. "A novel IoT-perceptive human activity recognition (HAR) approach using multihead convolutional attention". In: *IEEE Internet of Things Journal* 7.2 (2019), pp. 1072–1080.

[59] Zhenghua Chen et al. "WiFi CSI based passive human activity recognition using attention based BLSTM". In: *IEEE Transactions on Mobile Computing* 18.11 (2018), pp. 2714–2724.

[60] Qingchang Zhu, Zhenghua Chen, and Yeng Chai Soh. "A novel semisupervised deep learning method for human activity recognition". In: *IEEE Transactions on Industrial Informatics* 15.7 (2018), pp. 3821–3830.

[61] Cheng Xu et al. "InnoHAR: A deep neural network for complex human activity recognition". In: *Ieee Access* 7 (2019), pp. 9893–9902.

[62] Shaohuai Shi et al. "Benchmarking state-of-the-art deep learning software tools". In: *2016 7th International Conference on Cloud Computing and Big Data (CCBD)*. IEEE. 2016, pp. 99–104.

[63] Raymond E Wright. "Logistic regression." In: (1995).

[64] George A Oguntala et al. "SmartWall: novel RFID-enabled ambient human activity recognition using machine learning for unobtrusive health monitoring". In: *IEEE Access* 7 (2019), pp. 68022–68033.

[65] Ganapati Bhat et al. "w-HAR: An activity recognition dataset and framework using low-power wearable devices". In: *Sensors* 20.18 (2020), p. 5356.

[66] Junhao Shi, Decheng Zuo, and Zhan Zhang. "Transition Activity Recognition System Based on Standard Deviation Trend Analysis". In: *Sensors* 20.11 (2020), p. 3117.

[67] Kai-Chun Liu, Chia-Yeh Hsieh, and Chia-Tai Chan. "Transition-aware housekeeping task monitoring using single wrist-worn sensor". In: *IEEE Sensors Journal* 18.21 (2018), pp. 8950–8962.

[68] Mohammed Mehedi Hassan et al. "A robust human activity recognition system using smartphone sensors and deep learning". In: *Future Generation Computer Systems* 81 (2018), pp. 307–313.

[69] Kunal Gusain, Aditya Gupta, and Bhavya Popli. "Transition-aware human activity recognition using extreme gradient boosted decision trees". In: *Advanced Computing and Communication Technologies*. Springer, 2018, pp. 41–49.

[70] IN Yulita and S Saori. "Human Activities and Postural Transitions Classification using Support Vector Machine and K-Nearest Neighbor Methods". In: *IOP Conference Series: Earth and Environmental Science*. Vol. 248. 1. IOP Publishing. 2019, p. 012025.

[71] Arash Atrsaei et al. "Postural transitions detection and characterization in healthy and patient populations using a single waist sensor". In: *Journal of NeuroEngineering and Rehabilitation* 17 (2020), pp. 1–14.

[72] Dan Setterquist. "Using a Smartphone to Detect the Standing-to-Kneeling and Kneeling-to-Standing Postural Transitions". In: *Dissertation* (2018).

[73] Huaijun Wang et al. "Wearable Sensor-Based Human Activity Recognition Using Hybrid Deep Learning Techniques". In: *Security and Communication Networks* 2020 (2020). DOI: https://doi.org/10.1155/2020/2132138.

[74] Pete Warden and Daniel Situnayake. *TinyML*. O'Reilly Media, Incorporated, 2019.

[75] Colby Banbury et al. "Micronets: Neural network architectures for deploying tinyml applications on commodity microcontrollers". In: *Proceedings of Machine Learning and Systems* 3 (2021).

[76] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International conference on machine learning*. PMLR. 2015, pp. 448–456.

[77] Jorge-L Reyes-Ortiz et al. "Transition-aware human activity recognition using smartphones". In: *Neurocomputing* 171 (2016), pp. 754–767.

[78] Davide Anguita et al. "A public domain dataset for human activity recognition using smartphones." In: *Esann*. Vol. 3. 2013, p. 3.

[79] Amine El Helou. *Sensor HAR recognition App*. https://www.mathworks.com/matlabcentral/fileexchange/54138-sensor-har-recognition-app. [Online; accessed 11-August-2021]. 2015.

[80] Amine El Helou. *Sensor data analytics*. https://www.mathworks.com/matlabcentral/fileexchange/54139-sensor-data-analytics-french-webinar-code. [Online; accessed 11-August-2021]. 2015.