

July 2022

Deep Learning and Feature Engineering for Human Activity Recognition: Exploiting Novel Rich Learning Representations and Sub-transfer Learning to Boost Practical Performance

Ria Kanjilal
University of South Florida

Follow this and additional works at: <https://digitalcommons.usf.edu/etd>

 Part of the [Electrical and Computer Engineering Commons](#)

Scholar Commons Citation

Kanjilal, Ria, "Deep Learning and Feature Engineering for Human Activity Recognition: Exploiting Novel Rich Learning Representations and Sub-transfer Learning to Boost Practical Performance" (2022). *USF Tampa Graduate Theses and Dissertations*.
<https://digitalcommons.usf.edu/etd/9384>

This Dissertation is brought to you for free and open access by the USF Graduate Theses and Dissertations at Digital Commons @ University of South Florida. It has been accepted for inclusion in USF Tampa Graduate Theses and Dissertations by an authorized administrator of Digital Commons @ University of South Florida. For more information, please contact scholarcommons@usf.edu.

Deep Learning and Feature Engineering for Human Activity Recognition: Exploiting Novel
Rich Learning Representations and Sub-transfer Learning to Boost Practical Performance

by

Ria Kanjilal

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Electrical Engineering
College of Engineering
University of South Florida

Major Professor: Ismail Uysal, Ph.D.
Nasir Ghani, Ph.D.
Mia Naeini, Ph.D.
Robert Karam, Ph.D.
Deniz Dayicioglu, Ph.D.

Date of Approval:
June 23, 2022

Keywords: Data Augmentation, Healthcare, Spectrotemporal Representation,
Subject-specific Learning, Unsupervised Feature Learning

Copyright © 2022, Ria Kanjilal

Acknowledgments

I am deeply indebted to my Ph.D. supervisor Dr. Ismail Uysal for his continuous encouragement and guidance as I worked over many years towards this accomplishment. This endeavor would not have been possible without his cordial support. I would also like to express my deepest gratitude to Dr. Nasir Ghani, Dr. Mia Naeini, Dr. Robert Karam and Dr. Deniz Dayicioglu for their willingness to serve as my supervisory committee members and their valuable time reviewing my dissertation work. I'm extremely grateful to my supervisor for providing me with the necessary research funding as well as USF's Department of Electrical Engineering for the teaching assistantship support such that I could complete my research work. A special thanks to my friend Mainak Kundu. I could not have undertaken this journey without his support and encouragement. Many thanks go to my friends and lab partners, Rania Elashmawy and Dr. Muhammed Kucuk. Most importantly, I wish to thank my parents and my brother for providing me with their unfailing support and continuous encouragement throughout my academic life. Finally, I would like to thank God Almighty for giving me the strength, opportunity and guidance for achieving my goal.

Table of Contents

List of Tables	iv
List of Figures	vi
Abstract	ix
Chapter 1: Introduction	1
1.1 Motivation	1
1.2 Deep Learning versus Feature Engineering	3
1.3 Empowering Deep Feature Learning in HAR	5
1.4 Sub-transfer Learning for Retuning the Outlier User Accuracy	6
Chapter 2: Introduction to Machine Learning and Deep Learning	9
2.1 Introduction	9
2.2 What is Machine Learning?	9
2.3 Relation Between ML, DL and AI	10
2.4 Model Selection Paradigm	10
2.4.1 Data Preprocessing	12
2.4.1.1 Feature Engineering and Feature Processing	12
2.4.1.2 Feature Selection	14
2.4.2 Learning Algorithms	15
2.4.2.1 Supervised Learning	15
2.4.2.2 Unsupervised Learning	15
2.4.2.3 Semi-supervised Learning	16
2.4.2.4 Self-supervised Learning	16
2.4.2.5 Reinforcement Learning	17
2.4.3 Cross Validation	17
2.4.3.1 Holdout Cross Validation	18
2.4.3.2 Leave-one-out Cross Validation	18
2.4.3.3 Leave-P-out Cross Validation	18
2.4.3.4 K-fold Cross Validation	19
2.4.3.5 Stratified K-fold Cross Validation	19
2.4.4 Loss Function	20
2.4.4.1 Mean Squared Error	20
2.4.4.2 Mean Absolute Error	21
2.4.4.3 Mean Squared Logarithmic Error	21
2.4.4.4 Binary Cross-Entropy Loss	21
2.4.4.5 Categorical Cross-Entropy Loss	22

2.4.5	Model Optimization	22
2.4.5.1	Model Hyperparameters	22
2.4.5.2	Stochastic Gradient Descent Optimizer	23
2.4.5.3	Adam Optimizer	23
Chapter 3:	Deep Learning versus Feature Engineering	24
3.1	Introduction	24
3.2	Experimental Setup	25
3.2.1	Dataset	25
3.2.1.1	UniMiB-SHAR Dataset	25
3.2.1.2	ExtraSensory Dataset	27
3.2.2	Classification Algorithm	28
3.2.3	Feature Extraction	32
3.2.3.1	FE Using UniMiB-SHAR Dataset	33
3.2.3.2	FE Using Modified ExtraSensory Dataset	35
3.2.4	Feedforward ANN Setup	35
3.2.5	RNN-LSTM Setup	36
3.2.6	1D-CNN Setup	37
3.3	Results and Discussion	39
3.3.1	Experimental Results	40
3.3.1.1	Results of UniMiB-SHAR Dataset	40
3.3.1.2	Results of Modified ExtraSensory Dataset	45
3.4	Summary and Conclusions	48
Chapter 4:	Empowering Deep Feature Learning in HAR	49
4.1	Introduction	49
4.2	Design and Methodology	50
4.2.1	Dataset	50
4.2.2	Data Preprocessing	51
4.2.3	Classification Algorithms	51
4.2.3.1	DFNN	54
4.2.3.2	AE-DFNN	55
4.2.3.3	1D-CNN	57
4.2.3.4	2D-CNN	59
4.2.3.5	RNN-LSTM	61
4.2.3.6	Data Augmentation: SMOTE	64
4.3	Experimental Setup	65
4.3.1	Leave-One-Subject-Out Cross-validation	65
4.3.2	Model Hyperparameters	66
4.3.3	Feature Engineering	67
4.4	Results and Discussion	67
4.4.1	Statistical Analysis of the Performances of Classifiers	71
4.4.2	Computational Complexity	72
4.4.3	Subject-based Classification Accuracies on Four Datasets	75
4.4.4	Deep Learning vs. Feature Engineering on Reduced Size Data	75

4.5	Summary and Conclusions	79
Chapter 5:	Sub-transfer Learning for Retuning the Outlier User Accuracy	82
5.1	Introduction	82
5.2	Methodology	82
5.2.1	Dataset	84
5.2.2	One-Dimensional Convolutional Neural Network	84
5.2.3	Synthetic Minority Oversampling Technique	88
5.3	Design and Experimental Setup	90
5.3.1	Sub-transfer Learning Model	90
5.3.2	Subject-specific Learning Model	91
5.3.3	Experimental Setup and Model Design	91
5.4	Results and Discussion	93
5.4.1	Statistical Significance Analysis of the Classification Models	98
5.5	Summary and Conclusions	98
Chapter 6:	Conclusions, Contributions and Future Work	100
6.1	Deep Learning versus Feature Engineering	100
6.1.1	Contribution 1	101
6.2	Empowering Deep Feature Learning in HAR	101
6.2.1	Contribution 2	102
6.3	Sub-transfer Learning for Retuning the Outlier User Accuracy	102
6.3.1	Contribution 3	103
6.4	Future Work	103
References	104
Appendix A:	Copyright Permissions	119
About the Author	End Page

List of Tables

Table 3.1	Nine activities of daily living	25
Table 3.2	Mean accuracies of four classifiers on AF-2 UniMiB-SHAR dataset . .	41
Table 3.3	Mean accuracies of four classifiers on A-9 UniMiB-SHAR dataset . .	41
Table 3.4	Computational complexity of classifiers on AF-2 and A-9 UniMiB-SHAR dataset	42
Table 3.5	Mean accuracies of four classifiers on modified ExtraSensory dataset .	46
Table 4.1	Activities of daily living categorized into four classes	52
Table 4.2	Number of observations of four Adult-Youth datasets	52
Table 4.3	The topologies for the different classifiers using feature learning for human activity recognition	65
Table 4.4	Average accuracies for the different classifiers using feature learning versus the subject agnostic feature engineering used in the latest work on this dataset (Mannini <i>et al.</i> [64])	68
Table 4.5	Average classification accuracies with and without data augmentation on the youth-ankle dataset	70
Table 4.6	Statistical significance analysis between the performances of the classifiers on Adult dataset	73
Table 4.7	Statistical significance analysis between the performances of the classifiers on Youth dataset	73
Table 4.8	Computational complexities of five classifiers for a single subject on four datasets	74
Table 4.9	Classification accuracies and computational time of RNN-LSTM classifier based on three random subjects of four datasets	75
Table 4.10	Subject-based classification accuracies for four classifiers on adult-ankle dataset	76

Table 4.11	Subject-based classification accuracies for four classifiers on adult-wrist dataset	77
Table 4.12	Subject-based classification accuracies for four classifiers, with and without SMOTE on youth-ankle dataset	78
Table 4.13	Subject-based classification accuracies for four classifiers on youth-wrist dataset	78
Table 4.14	Subject-based classification accuracies for DFNN and FE-DFNN classifiers on reduced size adult dataset	80
Table 4.15	Subject-based classification accuracies for DFNN and FE-DFNN classifiers on reduced size youth dataset	81
Table 5.1	Experimental results of sub-transfer and subject-specific learning models	96
Table 5.2	Statistical significance analysis between the performances of the learning representations on twelve subjects of four datasets	98

List of Figures

Figure 2.1	Comparison between the high label representation of (a) a traditional programming model and (b) a machine learning model	11
Figure 2.2	Interconnection between deep learning, machine learning and artificial Intelligence	11
Figure 2.3	Hierarchy of ML algorithms	17
Figure 2.4	Holdout cross validation	18
Figure 2.5	Leave-one-out cross validation	19
Figure 2.6	10-fold cross validation	20
Figure 3.1	Time-series representations of single-axis acceleration data for each class of activities of daily living for the UniMiB-SHAR dataset	26
Figure 3.2	Time-series representations of single-axis acceleration data for the two classes of activities of daily living for the modified ExtraSensory dataset	27
Figure 3.3	Optimal hyperplanes of a linear (left) and a radial basis function kernel (right) SVM	28
Figure 3.4	A sample topology of the feedforward neural network used for A-9 with two hidden layers	29
Figure 3.5	Schematic representation of the RNN-LSTM network	30
Figure 3.6	A high level representation of 1D-CNN classifier	32
Figure 3.7	Mean and variance features for the UniMiB-SHAR dataset	34
Figure 3.8	Pitch, roll, yaw, zero crossing rate of the mean and magnitude features for the UniMiB-SHAR dataset	36
Figure 3.9	Mean and variance features for the modified ExtraSensory dataset	37
Figure 3.10	Pitch, roll, yaw, zero crossing rate of the mean and magnitude features for the modified ExtraSensory dataset	38

Figure 3.11	Confusion matrix of AF-2 raw data for 1D-CNN classifier	43
Figure 3.12	Confusion matrix of A-9 raw data for RNN-LSTM classifier	43
Figure 3.13	Mean accuracy plot of raw and feature-based models of UniMiB-SHAR AF-2 dataset based on various topologies	44
Figure 3.14	Mean accuracy plot of raw and feature-based models of UniMiB-SHAR A-9 dataset based on various topologies	45
Figure 3.15	Mean accuracy plot of raw and feature-based models of modified ExtraSensory dataset based on various topologies	47
Figure 4.1	Human activity recognition using unsupervised feature learning . . .	50
Figure 4.2	Example observations from different subjects and sensor placements for both re-sampled (marked by blue) and filtered (marked by green) signals	53
Figure 4.3	Time-series representations of single-axis acceleration data for four classes of activities of daily living for the adult-ankle and adult-wrist datasets	54
Figure 4.4	Architecture of a simple DFNN used in the experiments	56
Figure 4.5	Autoencoder deep feedforward neural network model and training of learned features (encoder output) using the DFNN	57
Figure 4.6	Feature learning using a one-dimensional convolutional neural network	58
Figure 4.7	Transformation of 1D time domain signal to 2D frequency domain representation	62
Figure 4.8	Frequency domain representation of sample observations from the four datasets	63
Figure 4.9	Feature learning using a two-dimensional convolutional neural network	63
Figure 4.10	Schematics of RNN-LSTM network used in this experiment	64
Figure 4.11	Boxplot distributions of classification accuracies across different classifiers on all 4 datasets	68
Figure 4.12	Average classification accuracies of 2D-CNN classifier with no SMOTE and different percentages of SMOTE on the youth-ankle dataset	70
Figure 5.1	Time-series representations of single-axis acceleration data for four classes of activities of daily living for the youth-ankle and youth-wrist datasets	83

Figure 5.2	Framework of 1D-CNN classifier	87
Figure 5.3	Generating synthetic samples using SMOTE	89
Figure 5.4	Block diagram of sub-transfer learning and subject-specific learning models	92
Figure 5.5	Boxplot distribution to compare the classification accuracies of sub-transfer learning and subject-specific learning models on Adult-youth dataset	97

Abstract

A significant gap exists in our knowledge of how domain-specific feature extraction compares to unsupervised feature learning in the latent space of a deep neural network for a range of temporal applications including human activity recognition. This dissertation aims to address this gap specifically for human activity recognition using acceleration data. To ensure reproducibility, we use two publicly available datasets, UniMiB-SHAR and ExtraSensory, with a well-established history in the human activity recognition literature. We methodically analyze the performance of 64 different combinations of i) learning representations (in the form of raw temporal data or extracted features), ii) traditional and modern classifiers with different topologies on iii) both binary (fall detection) and multi-class (daily activities of living) datasets. We report and discuss our findings and conclude that while feature engineering may still be competitive for activity recognition task, trainable front-ends of modern deep learning algorithms can benefit from raw temporal data especially in large quantities. In fact, this study claims state-of-the-art where we significantly outperform the most recent literature on UniMiB-SHAR dataset in both activity recognition (88.41% vs. 98.02%) and fall detection (98.71% vs. 99.82%) using raw temporal input.

We further improve the generalization capability of deep learning networks by introducing a richer way to harness the spectral properties of biological time series in addition to temporal features. A Stanford research group proposed subject agnostic features as state-of-the-art when applied to a large dataset with many participants of different ages. In this dissertation, we demonstrate that implicit feature learning in the latent spaces of deep learning algorithms can be powerful alternatives to using finely tuned domain-specific features for human activity recognition. In fact, when using a spectrotemporal representation of the raw sensor data in the form of spectrograms, a standard convolutional neural network without any prior

conditioning on the features, statistically significantly outperforms the prior state-of-the-art using subject agnostic features in all the different partitions of the dataset with a significant 29.8% reduction in the overall average error rate.

Finally, we look at one of the most important practical challenges for human activity recognition where a commercial algorithm can achieve very low accuracies for some outlier subjects. In this context, we propose a method to exploit a source model to fine tune the parameters for each specific subject to enhance their classification performances. In the literature, most of the research follow the mean classification accuracy as a performance metric. However, some outlier users which provide low accuracies in classification can demote the overall performance of a motion recognition system when compared to the median accuracy reported on any given dataset. To find a solution to the problem, we study several approaches in determining the impact of outlier users on activity recognition task and propose a novel approach, sub-transfer learning which demonstrate that the principles of transfer learning can be applied within the same dataset when coupled with augmentation techniques. Our results show that on the most difficult users with the lowest subject specific accuracies, our performance gains can be as much as 15% when using only a few additional samples for re-tuning. Finally, we demonstrate that the performance improvements of the proposed model are statistically significantly better than the source or subject-specific models across a range of datasets with demographically diverse users and sensor locations.

Chapter 1: Introduction

1.1 Motivation

The most impactful research for our society almost always involves advances in health-care and related disciplines as we strive to learn more about the disease trends and risk factors of some chronic diseases, outcomes and methodologies of treatments, pattern of care by continuous monitoring of patients' health, medical costs etc. A recent projection about the disease trends in the future reports that the ubiquity of diabetes will increase by 54% to include more than 54.9 million Americans by 2030 [82]. Moreover, in [104], the authors predict that nearly one in every four adults will have severe obesity by 2030 and the prevalence will be higher than 25% in twenty-five states. In this regard, researchers argue that continuous monitoring and recognition of daily physical activities can reduce the risk of chronic diseases including diabetes, obesity and cardiovascular problems. Moreover, the population of the United States is aging. The U.S. Department of Health and Human Sciences projects approximately 55 million people in the US aged 65 or older by 2030, which is almost double its value in 1990. A recent study found that one in three adults older than 65 years of age falls each year which is the leading cause of both fatal and non-fatal injuries for this age group. In fact, they predict that there will be 7 deaths due to falls every hour by 2030. The economic costs are staggering with direct medical costs of falls reaching \$30 billion annually. A simple wireless wristband with a fall detection algorithm can prevent a vulnerable person from lying in the emergency room for hours or days due to future complications.

On the other hand, we saw that the COVID-19 pandemic has had an unprecedented impact on the mental health of the front-line hospital staff. A recent report shows that up to 37% clinicians globally and 47% U.S. healthcare workers plan to leave their present role

by 2025 for their psychological wellbeing [50]. In this scenario, a contact-free continuous monitoring system can improve the conditions for the patients and reduce healthcare costs by a significant amount [58].

Human activity recognition (HAR) plays an important role in the recent emergence of personalized healthcare technologies via continuous monitoring and recognition of regular physical activities of humans using accelerometer data. The developments associated with HAR depend on the technological advancement of low-cost sensor-based systems and when integrated in smart devices can help healthcare providers monitor daily symptoms of patients to provide effective care. Researchers have more recently enlisted the help of mobile apps to collect sensory data on human motions at an unprecedented scale using readily available accelerometers and gyroscopes on smartphones, smart watches, etc. The ubiquitous availability of these sensors enables the developers to advance mobile and web-based healthcare applications using various data science and machine learning techniques [73]. There are different types of real life HAR applications such as still image based HAR [91, 96], video surveillance HAR [110], wearable sensor based HAR [83] etc. The environmental and stationary constraints, noisy information captured by the camera due to the presence of non-target people in the scene, and complexity and cost of video processing units are the main drawbacks of vision based HAR [89]. On the contrary, sensor based HAR systems receive more accurate and efficient signals from the multiple sensors placed on the body and reduce the overall cost and complexity of the processes required for meaningful representation of data to correctly distinguish between activities of daily livings such as jogging and running [22]. Most prevalent sensor based HAR systems use wearable sensors including accelerometer, magnetometer, and gyroscope which are embedded into portable smart devices including smartwatches, smart bands, smartphones, glasses etc. [107] for continuous monitoring of daily activities.

1.2 Deep Learning versus Feature Engineering

Over the past several decades, feature engineering has been the driving force behind many contemporary applications of machine learning such as Mel frequency cepstral coefficients for speech recognition [62] or edge-detection filters for object classification [76]. In recent years, deep learning, which attempts to find abstract structures in data without relying on explicit feature extraction methods common to many supervised learning schemes, has gained significant traction [23]. As big data becomes more readily available, it becomes practically more feasible to apply structurally deep and complex algorithms including the convolutional neural networks, deep Boltzmann machines, sparse auto-encoders, generative adversarial networks and transfer learning for a wide range of applications [59, 84, 21, 108]. New training cost optimization techniques such as stochastic gradient descent and parallel processing allow for training deep networks which are orders of magnitude larger than their predecessors. Simply, deep learning mimics the operational and organizational behavior of the human brain, which works through abstraction [7]. For example, objects and sounds are represented as electrical signals traveling through different types of connections with different strengths between neurons in the visual and auditory cortexes respectively. Deep learning takes inspiration from this and relies on higher-level representations of features embedded in the data instead of human engineered characteristics.

In the context of mobile health applications, big data analytics led to significant improvements in the accuracy of HAR or motion classification [93]. For instance, James Bartlett et al. prepared a large dataset, ActionNet, which contains 10.3 million observations over 13 different activity labels [8]. Two different deep learning architectures were studied with the final observation that the performance of deep neural networks outweighs the predictive power of traditional feature-engineering approach. Feature-engineering is a process which extracts key characteristics from raw data through domain-specific expertise to ease the input-requirements of the classifier by generating inputs that are mathematically and computationally more convenient to process. Researchers focused on comparative analysis of

the performance of deep learning vs. expert feature engineering on various fields such as reinforcement learning [51], healthcare and clinical practices [105] and food related applications [27] etc. For example, Yang Jiang et al. represented a comparative study on deep neural network vs. expert features and concluded that feature-engineering models are better when considering a single optimized threshold [47]. Nevertheless, the authors also acknowledged the limitation of their study due to limited sample-size. Other researchers explored a combination of feature engineering and deep learning such as a new comprehensive study on feature extraction vs. deep modeling for medical image classification [61], where the authors propose a deep learning model comprised of a multi-layer perceptron after extracting the expert features of medical images using a convolutional neural network. Another study proposed an unsupervised deep learning architecture named local deep-feature alignment which implements a dimension reduction technique to map the data points into a learned low-dimensional subspace [111]. On the other hand, video based HAR and Human Pose Recovery systems have gained compelling attention in the field of computer vision and pattern recognition within the deep learning framework [90, 40, 39]. However, there is no comprehensive study on model comparisons for HAR using raw data with deep learning versus expert features. Presently, deep learning methods including Recurrent Neural Networks such as long short-term memory (RNN-LSTM) have demonstrated competitive performance on raw data compared with the state-of-the-art models using feature engineering for HAR [70]. In addition, for time-series classification, the one-dimensional convolutional deep neural networks (1D-CNNs) have shown state-of-the-art performance in terms of accuracy and computational complexity [94]. Researchers proved that the performance of 1D-CNN can be improved by the tuning of its hyper parameters such as the kernel size and filters [60, 74]. Apart from HAR [20, 33], 1D-CNN models are used in various fields of machine learning applications such as radiology [106], split learning [5] and prediction of smoking events [4] etc.

1.3 Empowering Deep Feature Learning in HAR

The approach of deep learning versus feature engineering has drawn significant attention in the machine learning community. For most of the past decade, feature engineering has been the dominant approach for better recognition by carefully casting handcrafted features from data using traditional machine learning tools. In [36], J. Heaton shows an empirical representation of what types of domain-specific features are best fit for which machine learning models. In a recent study [80], the feature engineering approach is reviewed for clinical knowledge before applying machine learning algorithms to analyze its impact on computational complexity and performance. The engineered features are further incorporated in various studies including geological data [46], financial fraud detection [44], and reinforcement learning [51]. Generally speaking, if the dataset size is large enough, the feature extraction methods may not be feasible to extract comparably rich information from the data especially for complex and non-linear machine learning algorithms. Alternatively, deep feature learning can more naturally utilize large volumes of raw data through computationally complex and expensive algorithms where much of the feature extraction takes place in an unsupervised manner. In fact, researchers have achieved unprecedented levels of accuracy using unsupervised automatic feature learning from raw data in the hidden latent spaces of deep neural networks in a variety of applications including natural language processing [95], deep clustering on image datasets [55], DNA sequence processing [101], healthcare and biomedicine [69], solid-state materials science [88], human activity recognition [48] etc. Deep learning techniques allow incorporating feature extraction and classification tasks into a complex single process. Nevertheless, there is a significant gap in our knowledge to optimally determine what particular approach among deep learning and feature engineering would perform better in different fields [47], specifically for healthcare applications.

With widely successful experimental results on a range of applications, deep learning has become the most popular framework to change the future of healthcare. A recent report from Report Linker claims that the expected growth of AI-healthcare market will exceed \$35

billion by 2030 [3]. In healthcare, the researchers use the electronic health record data which stores patient information such as medical history and lab results to create a predictive model of the patients' health. Another report regarding home healthcare technologies used in elder care, specifically home tele-health and safety monitoring notes that the global market for elder care technologies should grow from \$5.7 billion in 2017 to \$13.6 billion by 2022 [2] such as continuous monitoring of daily activities and falls using 3-D motion sensors. In [86], G. L. Santos *et al.* have implemented deep learning models using convolutional neural network (CNN) classifiers for fall detection. In [42], the authors have used support vector machine classifiers to detect falls with high sensitivity, specificity, and accuracy. In the context of HAR, G. Ogbuabor *et al.* [73] investigated the role of motion sensors such as gyroscope and accelerometer sensors to develop an effective HAR system using artificial neural networks. In this paper, the authors showed that the accelerometer sensor data performed better than the gyroscope sensor data whereas the highest accuracy was achieved when sensors' data were combined. In the context of activity recognition, various classification algorithms such as RNN-LSTM [75], CNN [33], DeepConvoLSTM [37], and Deep Autoencoder [99] have been explored to develop efficient HAR systems.

1.4 Sub-transfer Learning for Retuning the Outlier User Accuracy

At its core, HAR consists of an input processing unit where the sensory data is transformed into a specific learning representation to be used in a machine learning algorithm for classification. Recently, with the advance and successful deployment of deep learning techniques, new research is directed from the conventional approach of feature engineering to automatic unsupervised deep feature learning from raw data in many different applications including image classification [103], language modelling [81], bioinformatics and biomedicine [11] etc. However, a significant gap exists in our knowledge regarding the optimum selection of a particular approach specifically for healthcare and HAR. Some of the results in medical diagnosis seem promising but when it comes to fall detection and activity recognition the

details are sparse. In literature, the researchers have been doing a comprehensive study to realize the proper learning algorithms for HAR datasets. In [24], the authors have implemented CNNs to recognize the daily activities on the data collected from accelerometer, gyroscope and magnetometer sensors. In another study, R. Mutegeki et al. proposed a hybrid CNN-LSTM (long-short-term-memory) architecture to classify human activities [72]. In recent HAR research, the scientists have been exploring transfer learning [87], where a model developed for a task is repurposed as the starting point for a model on a second task. Transfer learning is a machine learning method where the knowledge from the prior training is transferred to perform a new classification task. In [6], the authors have developed a HAR transfer learning framework which transfers a reusable portion of the offline classifier to new users to do the activity recognition.

In this dissertation [48, 49], we provide in-depth looks into these two fundamentally different approaches to HAR while introducing a novel way to harness the spectral properties of biological time series in addition to temporal features. However, in [72], the performances of different classification algorithms reveal that a healthcare application such as HAR is very much dependent on the specifics of subjects. Similarly, we have observed significant changes in accuracy from subject to subject when the algorithm is tested on never-before-seen data. For some subjects the accuracies are significantly lower where the lowest accuracy average across all classes is 56% as opposed to the maximum average accuracy of 99% for some subjects and the average accuracy of 90% for all subjects. In our search of the literature, we haven't encountered a method that specifically addresses the performance issues for this subset of users where the performance is less than ideal. Hence, in addition to the comprehensive investigation of feature learning versus feature engineering, in this dissertation we also study two different approaches: sub-transfer learning and subject-specific learning to specifically address subject specific accuracy problem and provide a thorough analysis to boost the performances of outlier users. Sub-transfer learning is a novel method which can be used as a powerful alternative to improve the subject-oriented classification

accuracies by using as few a number of samples as possible from the test subject to perform auxiliary training of the classifier in addition to the prior training using the rest of the subjects similar to transfer learning. We then compare subject-by-subject classification accuracies of sub-transfer learning model with subject-specific learning (as a baseline model) where the classifier is trained only on a few samples of the test subject and tested on the remaining samples of the same test data without any prior training to demonstrate how practical methods can be improved for challenging subjects.

Chapter 2: Introduction to Machine Learning and Deep Learning

2.1 Introduction

Humans have unparalleled superiority in building models for any complex phenomenon and critically analyzing them to acquire knowledge and understand their mechanisms. The knowledge we obtain from such models helps us theorize the underlying operations and predict the probable outcomes of the phenomenon in future [56]. As we build more and more accurate models, we gain better insight into the true operational characteristics to ultimately create an artificial replica of the same phenomenon. This probabilistic model-based approach is often scientifically related to human intelligence with the concept of artificial neural networks (ANN) which mimic, in a very broad manner, the functions of the human brain. An ANN consists of a collection of connected nodes called artificial neurons, which are, at a high level, similar to the basic units or neurons in a biological brain. The interconnections in the ANN have associated weights which are adjusted during the model learning phase much like the biological neural network organically forming and changing during the learning process.

2.2 What is Machine Learning?

Machine learning (ML) is an analytical method including various computational algorithms that can learn from the data by identifying its patterns and making decisions with minimal user intervention. In other words, ML is a tool which can create complex non-linear mappings between the input and output of a predictive model used in decision making.

As an example, in Figure 2.1, the traditional programming describes a paradigm where a programmer or a user develops a program including a set of rules based on the input which

is then fed to the computer. The computer determines the output according to the specific rules of the program and the corresponding input. If the output is not satisfactory, the programmer goes back to the program and changes the rules to update the results. In the case of ML approach, this process is automated. In other words, the computer performs a non-linear mapping between the input and output according to the algorithm or program to be learned where the model is trained for a certain number of iterations to accelerate the decision-making process by updating the parameters of the model. After training, when the model is provided with an input for testing, it will predict an output. ML algorithms are applied in a diverse range of fields including finance, image processing, bioinformatics, biomedicine, pattern recognition, computer vision and healthcare. The interconnections between machine learning, deep learning (DL) and artificial intelligence (AI) is discussed in the next section.

2.3 Relation Between ML, DL and AI

Artificial intelligence allows machines to perform specific tasks based on algorithms. The term AI was first introduced by John McCarthy at the Dartmouth Summer Research Project on Artificial Intelligence conference, a summer workshop that was held in 1956 [65]. ML is a form of AI that emphasizes the ability of machines to learn from data rather than through explicit programming. The term ML was coined in 1959 by Arthur Samuel, a computer scientist at IBM and a pioneer in AI and computer gaming [85]. Finally, DL is a subset of ML, where multi-layer ANNs are used to perform the prediction and classification tasks on vast amounts of data to train a similarly large set of model parameters. Figure 2.2 displays the strong interconnection between the fields of DL and ML as parts of the widespread AI.

2.4 Model Selection Paradigm

The most important part of building a model is to understand the mechanism and the underlying nature of the phenomenon which is being modeled. A typical ML pipeline consists

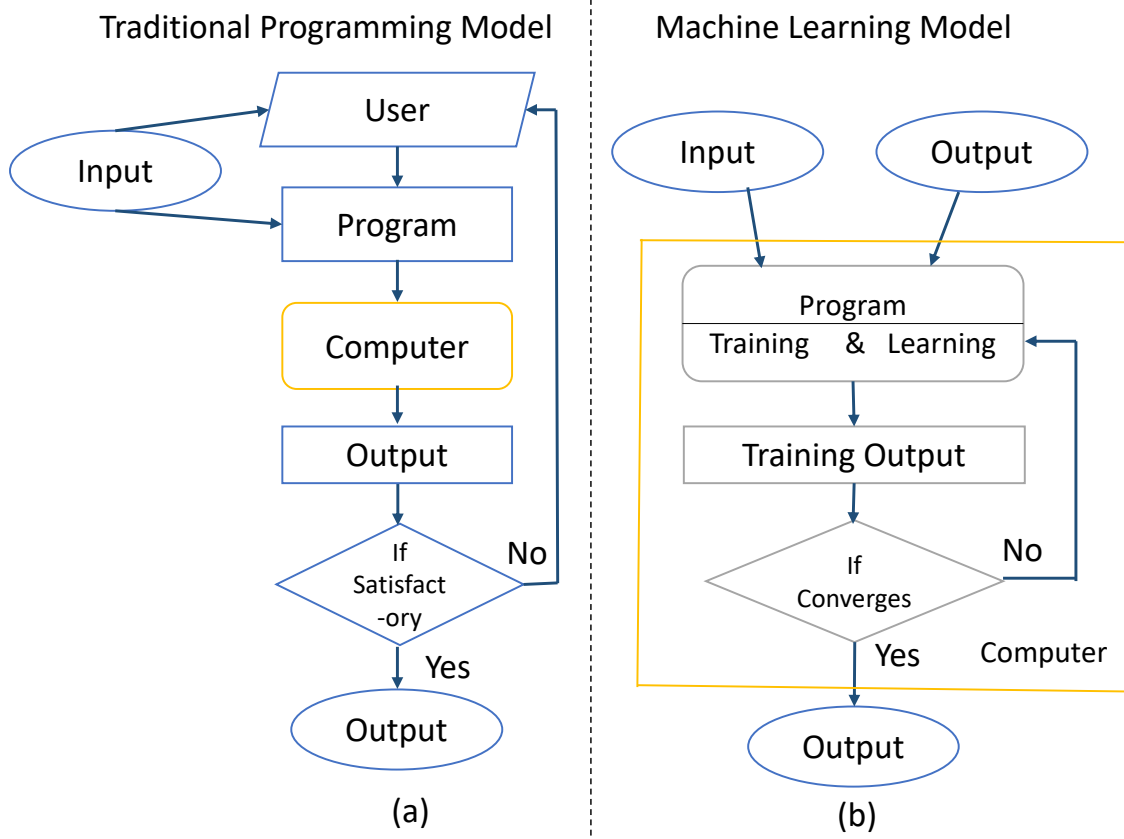


Figure 2.1: Comparison between the high level representation of (a) a traditional programming model and (b) a machine learning model

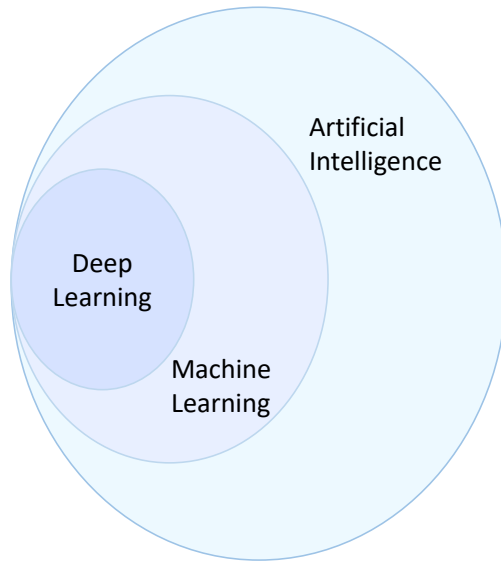


Figure 2.2: Interconnection between deep learning, machine learning and artificial Intelligence

of the following steps: data acquisition, data preprocessing and model training followed by validation on a hold-out subset of the data [71]. In [98], the author addresses the limitations of the model selection process and focuses instead on minimizing the error over the data (to achieve the maximum likelihood of model parameters over the dataset under consideration), while keeping the bias and variance of the model as small as possible. During this process, train-test split, cross-validation and other partitioning techniques are applied to properly determine and boost the model's expected performance.

2.4.1 Data Preprocessing

A dataset includes multiple observations where each observation is called a sample. Each sample is represented by a set of values called features and labeled by different categories of output classes based on the type of the learning problem. For instance, a human activity recognition dataset can be represented as a set of observations in time domain with the domain specific features extracted from the three axes of the accelerometer sensor to create a compressed one-dimensional observation. The dataset is preprocessed and normalized before training to prevent bias. The data preprocessing includes the following steps:

2.4.1.1 Feature Engineering and Feature Processing

Feature engineering (FE) is a machine learning technique where raw data is transformed into more meaningful variables called features to better represent the problem-statement of the model. In FE, domain-specific features are extracted from the data to speed up the learning process while improving the performance of the model. Some of the common features that we extract are mean, variance, maximum in a range, minimum in a range, magnitude etc. For motion related data, some important features are pitch, roll, yaw, zero crossing rate of mean etc. For natural language processing, features can be the number of words, number of capital words, number of punctuations, number of unique words, number of stop words, average sentence length etc.

On the other hand, after visualizing the data, sometimes we need to transform the features further to make them more meaningful. This process is called feature processing. The techniques for feature processing are listed below.

- **Imputation:** Missing values are one of the important issues in the dataset to handle with while feeding the data to the model. These missing values can be included in the dataset due to several reasons such as data flow interruptions, human errors, privacy concerns, and others. Imputation method can fill out these missing values by assigning some numbers based on the data of completed survey or census on a current population. For categorical column variables, missing values are replaced by the highest value in the column.
- **Handling Outliers:** To remove the outliers from a dataset, we use outlier handling method. This technique can be used to generate more accurate representation of a dataset which improves the model performance. Various techniques for handling outliers are removal, replacing values, capping, discretization etc.
- **Log Transform:** In this method, a skewed distribution is turned into a normal or less-skewed distribution by taking the log of the values in a column and utilizing it.
- **One-Hot-Encoding:** This is a one of the popular encoding processes in ML where an element of a finite set is represented by the index in that set. Here, only one element has its index set to “1” and remaining elements are assigned indices within the range $[0, n-1]$. Since it is more challenging to the algorithm to understand the categorical variables, this method transformed the categorical data to numerical format and enables the model to group categorical data without an information loss.
- **Scaling:** This technique changes continuous features to identical in terms of the range. Basically, there are two common ways of scaling:

(i) Normalization (or min-max normalization) where scaling of the data is done within a fixed range between 0 and 1. If X is a feature vector and X_{max} and X_{min} are the maximum and minimum values of the features respectively, then the normalization can be defined as

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (2.1)$$

(ii) Standardization (or z-score normalization) defines that the scaling of the data is done by considering the value of standard deviation. In (2.2), X is a feature vector, the mean of a given distribution X is defined as μ and corresponding standard deviation is defined as σ .

$$z = \frac{X - \mu}{\sigma} \quad (2.2)$$

2.4.1.2 Feature Selection

Feature selection automatically selects the important and more independent features using a scoring method such as correlation to rank the data and reduce the number features by using statistical techniques [109].

Principal component analysis is a popular feature selection method which can reduce the dimensions of the feature space by identifying the important statistical relationships in the features through the covariance matrix. It transforms the existing data based on these relationships, extracts the important features and drops the others. During linear transformation of the covariance matrix, the eigenvectors and eigenvalues are computed. The data is transformed using these eigenvectors to its principal components [114]. Other popular feature selection techniques include wrapper methods, filter methods, and embedded methods etc [18].

2.4.2 Learning Algorithms

ML algorithms are generally classified into four different categories.

2.4.2.1 *Supervised Learning*

In supervised learning the algorithm is trained on labeled data. During training, the model establishes a linear or non-linear relationship between the input & output pairs included in the training set by computing a differentiable loss between the model prediction and the actual output. In ANNs, this process is performed via backpropagation through the network to learn from the error (or cost function) by updating the weight and bias parameters to optimize the predictions. The model is then validated and tested on data not used in the training process to predict the output. Supervised learning can be applied to perform two general tasks: regression analysis and classification. For regression analysis, the labels are continuous variables whereas for classification, the labels are categorical class labels. An example of supervised learning is image classification using convolutional neural network.

2.4.2.2 *Unsupervised Learning*

Unlike supervised learning, unsupervised learning allows the model to learn from the data without using explicit labels. The algorithm self-detects the trends or patterns in the data and groups them by creating associated clusters of data points. Due to the absence of labels unsupervised learning cannot build an explicit relationship between an input-output pair. However, the algorithm still perceives a relationship between data points in an abstract manner by creating a latent structure of the dataset. In the learning process, the algorithm can adapt by dynamically changing the parameters of these hidden structures. Two major subcategories of unsupervised learning are clustering and principal component analysis for dimensionality reduction.

2.4.2.3 Semi-supervised Learning

Semi-supervised learning algorithm is an intermediate between supervised and unsupervised learning where training takes place on both labeled and unlabeled data. Since generation of labeled data is very costly and there is a tremendous volume of unlabeled data in the real world, this algorithm focuses on utilizing mixed datasets consisting of both labeled and unlabeled data points in the training process. The main motivation behind semi-supervised learning is to use the labeled data points of a mixed dataset in supervised learning first to label the unlabeled part of the dataset which can be used in supervised learning later. In this technique, the unlabeled data points are used to improve the generalization performance of the model even if the pseudo-labeling may not be entirely accurate. Graph-based algorithm, SGAN etc. are semi-supervised learning algorithms.

2.4.2.4 Self-supervised Learning

Self-supervised learning aims to leverage many unlabeled data points for supervised learning [79]. The self-supervised learning is called autonomous supervised learning as it uses naturally available information as labels for supervised learning tasks. Like supervised learning, self-supervised learning algorithm maps its inputs to outputs from labeled input-output pairs. In the specific case of self-supervised learning, the model is pretrained on a similar task before it is trained on the general target task.

When a labeled dataset corresponding to the target task is small, or the model could benefit from auxiliary data, self-supervised learning can be used to leverage a more extensive and unlabeled dataset of a similar type as the data for the target task and create the labels for the pretext task. An example of self-supervised learning algorithm is 'wav2vec' developed by Facebook.

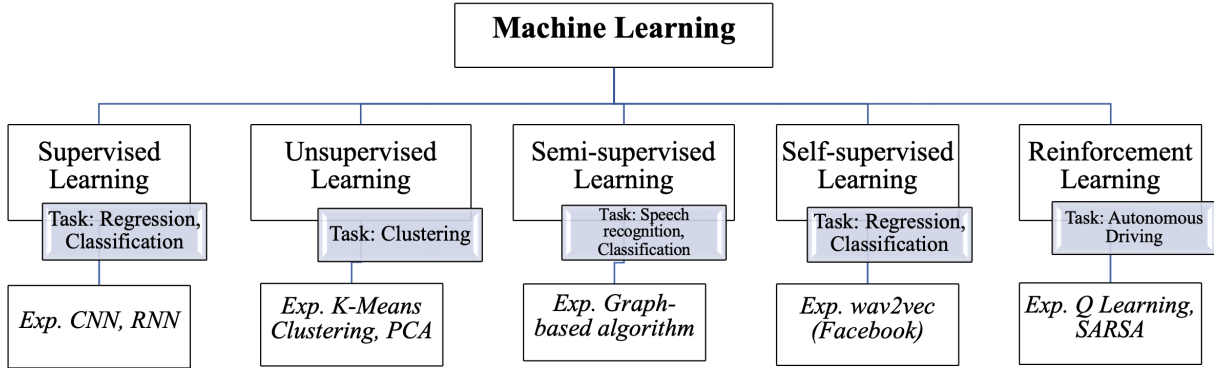


Figure 2.3: Hierarchy of ML algorithms

2.4.2.5 Reinforcement Learning

The motivation behind reinforcement learning is how human beings to learn in real life scenarios. Reinforcement learning algorithms perform a series of actions to maximize the reward for a particular situation which leads to a more satisfactory outcome for a given task [77]. The reinforcement agent makes a decision on what action to perform based on the specifics of the situation for a given task from which the algorithm learns and corrects itself in every step of the model. More specifically, the algorithm uses a variation of noting-and-eliminating-error method to find more optimal steps to take for a better reward. The desired outputs are reinforced, and the non-desired outputs are punished [1]. Reinforcement learning algorithms are Q-learning, SARSA etc. Figure 2.3 shows the hierarchy of machine learning algorithms and the corresponding tasks performed by each algorithm.

2.4.3 Cross Validation

To improve and properly assess the performance of the ML model on a fixed dataset cross validation technique is used to achieve an unbiased training and testing operation. Dataset is partitioned to prevent overfitting and selection bias in the training data for better optimization of the model before the actual deployment in a new environment. Various cross validation techniques are,

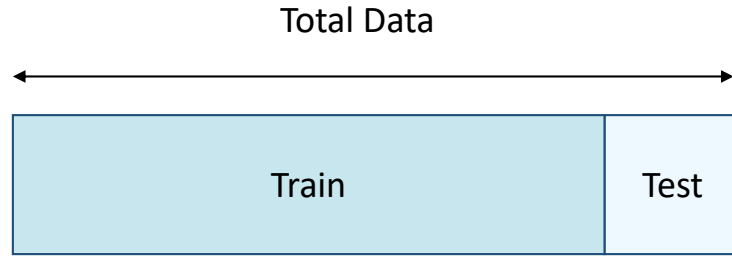


Figure 2.4: Holdout cross validation

2.4.3.1 Holdout Cross Validation

This technique is known as the train-test-split which is shown in Figure 2.4. The dataset is randomly split into training and testing partitions where the test dataset has fewer data points compared to the training data. Since the holdout cross validation performs the training and testing on these unique partitions once, it provides less optimization compared to the other methods. However, this technique is straightforward and less resource intensive especially when the dataset is sufficiently large.

2.4.3.2 Leave-one-out Cross Validation

A more exhaustive method, Leave-one-out cross validation ensures that each observation in the dataset is used as test data by identifying a single sample for each train-test trial and grouping the remaining samples as the training set. Multiple train-test operations are carried out and the model is subsequently evaluated based on the average of single trial performance metrics. Figure 2.5 shows how to partition the train-test splits using leave-one-out cross validation where the dataset has N number of samples.

2.4.3.3 Leave- P -out Cross Validation

As a variation of the previous method, P samples are set aside for testing instead of a single sample to cover all possible combinations of the train and test splits. The number of data points could be a user defined number or for subject based data, P could indicate

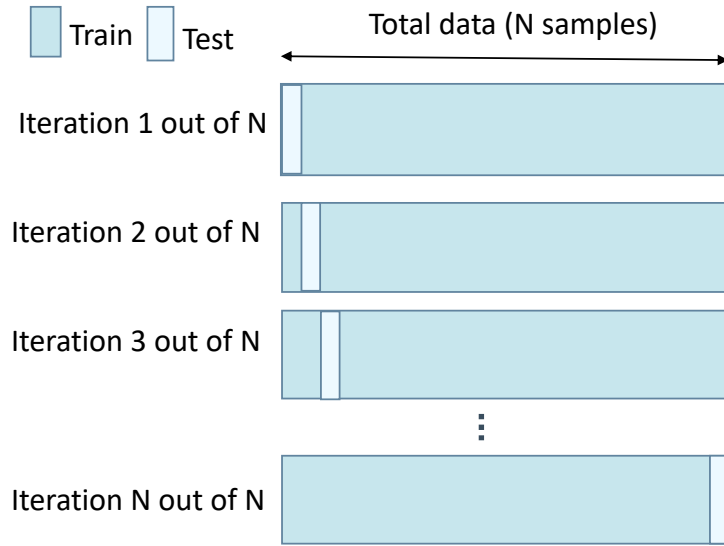


Figure 2.5: Leave-one-out cross validation

the number of samples for each subject to create a leave-one-subject-out variation of this cross-validation technique.

2.4.3.4 *K-fold Cross Validation*

A popular method for scientific datasets, K -fold cross validation creates K equally sized testing partitions within the original dataset where each train-test trial uses one of these partitions for testing and the rest for training to produce a stable and robust average result compared to holdout cross validation. Figure 2.6 displays the train-test partitioning of a dataset using K -fold cross validation method where $K = 10$.

2.4.3.5 *Stratified K-fold Cross Validation*

As often is the case with healthcare related datasets, a large imbalance of the class labels in the dataset could mean the randomized K -fold cross validation generates a train or test set with very few samples belonging to the minority class and downgrades the performance of the model. Stratified K -Fold splits the dataset on K folds while ensuring approximately the same percentage of samples for each target class is represented in each fold.

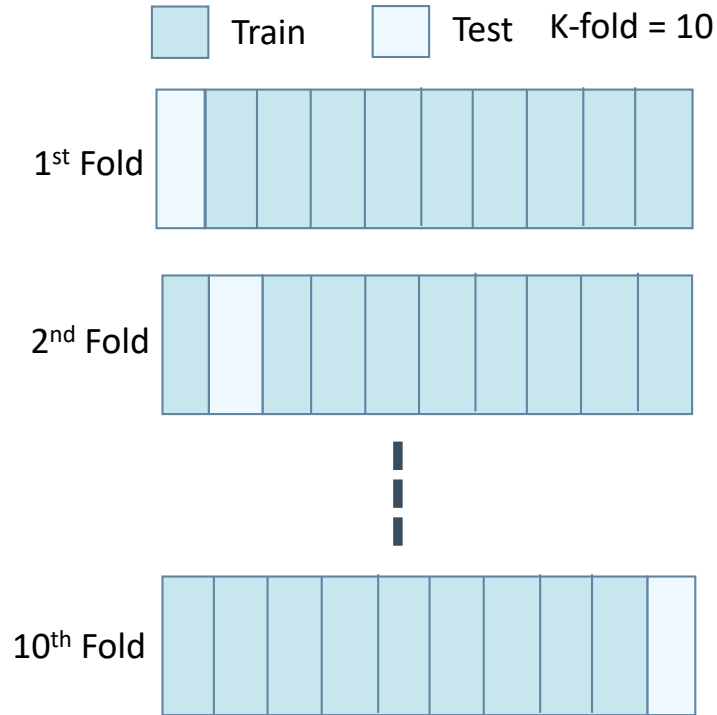


Figure 2.6: 10-fold cross validation

2.4.4 Loss Function

Loss function calculates the error between the predicted and target outputs where the ultimate aim of the model is to minimize the loss function to increase the accuracy and get better predictions. Commonly used loss functions in ML include:

2.4.4.1 Mean Squared Error

Mean square error (MSE) finds the average of the squared difference between the true value and the predicted value across all samples. The lower the value of MSE, the higher the accuracy of the model. MSE is known as the L2 loss function and defined in (2.3) where \hat{y}_i is the predicted output of the i^{th} scalar value in the model, y_i is the corresponding target value, and n is the output size.

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (2.3)$$

2.4.4.2 Mean Absolute Error

Mean Absolute Error (MAE) finds the average of the absolute difference between the true value and the predicted value across all samples. It is known as the L_1 loss function and computes a non-directional error by calculating the average of the magnitudes. The formula for MAE is defined in (2.4), where \hat{y}_i is the predicted output of the i^{th} scalar value in the model, y_i is the corresponding target value, and n is the output size.

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (2.4)$$

2.4.4.3 Mean Squared Logarithmic Error

Mean Squared Logarithmic Error (MSLE) calculates the log difference of the true and predicted values and find the ratio which is shown in (2.5).

$$MSLE = \frac{1}{n} \sum_{i=1}^n (\log(y_i + 1) - \log(\hat{y}_i + 1))^2 \quad (2.5)$$

where, \hat{y}_i is the predicted output of the i^{th} scalar value in the model, y_i is the corresponding target value, n is the output size and $\log(y_i + 1) - \log(\hat{y}_i + 1) = \log \frac{y_i+1}{\hat{y}_i+1}$.

2.4.4.4 Binary Cross-Entropy Loss

Binary Cross-Entropy (BCE) is called log loss and the output of the loss function is calculated as the probability for the predicted class between 0 and 1. This loss function is applied mostly for binary classification problems. The formula for binary cross entropy loss is defined as [38],

$$BCE = -\frac{1}{n} \sum_{i=1}^n y_i (\log(p(y_i))) + (1 - y_i) \log(1 - p(y_i)) \quad (2.6)$$

where, \hat{y}_i is the predicted output of the i^{th} scalar value in the model, y_i is the corresponding target value, n is the output size. In (2.6), $p(y_i) = \hat{y}_i$.

2.4.4.5 Categorical Cross-Entropy Loss

Categorical cross entropy loss (CCE) is used for multiclass classification tasks. CCE is a variation of the entropy loss function, and it calculates the difference between two probability distributions using the formula below:

$$CCE = -\frac{1}{n} \sum_{k=1}^K \sum_{i=1}^n y_i^k \log(p(y_i, k)) \quad (2.7)$$

In (2.7) $p(y_i) = \hat{y}_i$ which is the predicted output of the i^{th} scalar value in the model, y_i is the corresponding target value, n stands for training examples, K is number of classes, and y_i^k is the target label for training example i for class k [38].

2.4.5 Model Optimization

Optimization is an important part of the machine learning model. Optimization is done to update both the internal parameters such as the weights and biases of the network to reduce the loss function as well as the external parameters such as hyperparameters including the number of learning elements, learning rate, and depth of the network for better generalization. The techniques for model optimization are discussed below.

2.4.5.1 Model Hyperparameters

The optimization algorithm estimates the error gradient using the learning rate. According to the gradient descent optimization rule, the algorithm estimates the error gradient for the present state using the samples from the training data and then updates the weights of the model by finding the derivative of the loss function and the direction of its steepest descent. For instance, in neural networks, this is accomplished by using backpropagation where the weight-update is found by scaling the error gradient with the “learning rate” [15]. The learning rate can be between 0.0 and 1.0 and is used to adjust how quickly the model can be adapted to the problem. If a learning rate is too large, the model converges too

quickly to a sub-optimal solution. On the other hand, for a very small learning rate, the process can get stuck without convergence or a very slow convergence.

There is an extra layer to adjusting the speed of convergence using another hyperparameter called the momentum which defines the velocity with which the learning rate itself can be increased to reach the minima.

2.4.5.2 Stochastic Gradient Descent Optimizer

Stochastic gradient descent (SGD) is one of the most popular optimization algorithms which computes a gradient for the whole dataset to update weights in the opposite direction of the gradient to find a local minima on the error curve generated by the loss function. Some optimizers such as Adagrad, Adadelata and RMSprop are adaptive gradient descent algorithms and they provide an alternative to classical SGD for faster convergence.

2.4.5.3 Adam Optimizer

The limitation of using the conventional stochastic gradient descent algorithm is that its hyper parameters depend on the model and the value of the learning rate remains unchanged during training, where a fixed value of the learning rate is applied to update all the parameters. In case of adaptive learning algorithms such as Adam, individual learning rates are considered to update different parameters by combining the concepts of both RMSprop and SGD algorithm with momentum. For scaling of the learning rate, Adam optimizer uses the squared gradients like RMSprop. Moreover, it follows the operation of the SGD with momentum by using the moving average of the gradient instead of the instantaneous gradient itself [16].

Chapter 3: Deep Learning versus Feature Engineering

Parts of this chapter have been published in the journal of Neural Processing Letters in 2021, Springer and copyright permission is attached to Appendix A

3.1 Introduction

With the advance of deep learning, researchers have achieved unprecedented levels of accuracy using unsupervised automatic feature learning from raw data in the hidden latent spaces of deep neural networks in a variety of fields from image recognition to language modeling. However, a significant gap exists in our knowledge to optimally determine what particular approach would perform better, specifically for healthcare applications. Some of the results in medical diagnosis seem promising but when it comes to fall detection and activity recognition the details are sparse. Our research focuses on these predictive models using both raw data and feature engineering for robust comparative analysis and design.

In this chapter, to explore deep learning versus feature engineering, we use two publicly available datasets of moderate size and complexity and study the human activity recognition (HAR) performance of a wide range of modern topologies such as the feedforward deep neural networks, recurrent neural network with long short-term memory (RNN-LSTM) and one-dimensional convolutional neural network (1D-CNN) to compare and contrast expert features and raw data. We extract 13 domain specific features from raw temporal data and compare classification accuracies with subsequent analysis of the models using feature-engineering and raw data with deep learning. In the next section we describe the experimental setup in detail including the dataset and classification algorithms used in the experiments followed by the results with a detailed discussion of the performance comparisons.

Table 3.1: Nine activities of daily living

Dataset	Activities of daily living
A-9	Standing up from sitting on a chair Standing up from lying on the bed Walking Running Going up the stairs Jumping Going down the stairs Lying down on the bed from standing Sitting down

3.2 Experimental Setup

3.2.1 Dataset

3.2.1.1 UniMiB-SHAR Dataset

To ensure reproducibility of this work, a publicly available dataset with a literature history is used. UniMiB-SHAR dataset includes 11771 samples for falls and 7579 samples for other human activities, referred to as activities of daily living (ADL), performed by 30 different agents of ages between 18 and 60 [68]. The data is further categorized with 17 sub-classes of the 9 different ADLs and 8 different types of falls. On this dataset, we perform two different classification tasks: (i) identify fall vs. no-fall observations (this task is labeled as AF-2, following the original paper for comparative purposes), and (ii) classify 9 different activities of daily living (this task is labeled as A-9, following the original paper for comparative purposes). Table 3.1 shows the 9 activities of daily living.

In Figure 5.1, the single-axis acceleration data for these 9 ADLs and fall are shown where each subplot in the figure is a representative sample of raw temporal acceleration data for one of the three axes (x,y,z) which demonstrates the biggest dynamic range of motion as the dominant axis.

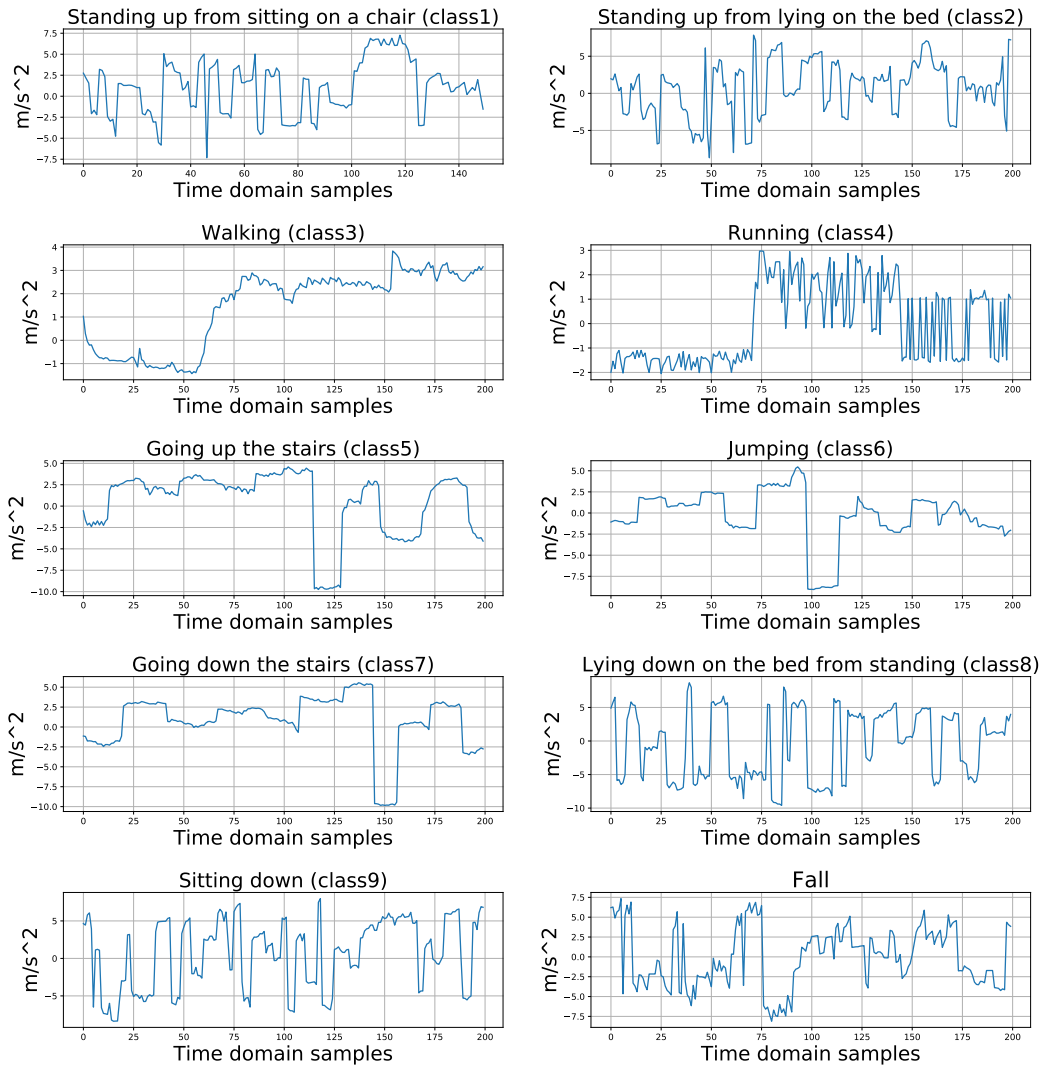


Figure 3.1: Time-series representations of single-axis acceleration data for each class of activities of daily living for the UniMiB-SHAR dataset

3.2.1.2 ExtraSensory Dataset

To compare the performances of the deep learning and feature engineering classifiers, we investigated a secondary publicly available dataset called ExtraSensory [97]. In the original paper, the authors use smartphone and smartwatch sensors for recognizing the activities of the users in their natural behavior. The labeled data which spans over 300,000 minutes is collected using a mobile application. This dataset includes 100 context labels from sensors and the activities are performed by 60 users. For our study, we split the data into a low dimensional version of 5775 observations and a high dimensional version of 55938 observations with two labels: (i) lying down and (ii) sitting. Figure 3.2 shows the time series representation of single axis acceleration data for these two classes of the modified extrasensory dataset.

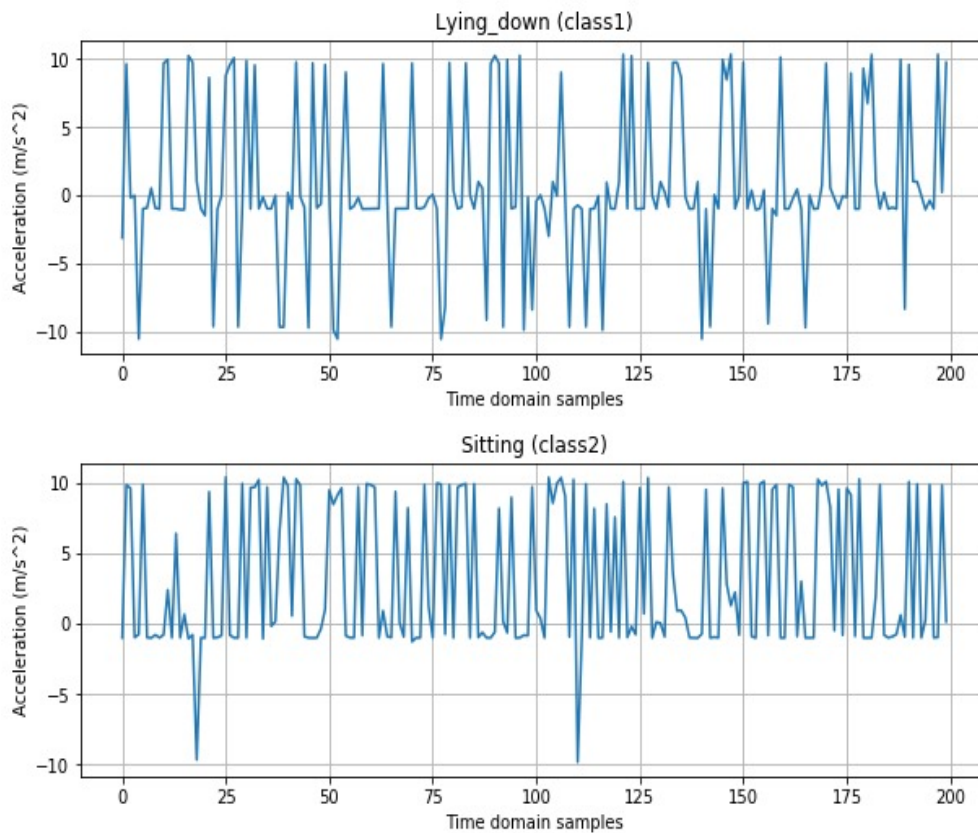


Figure 3.2: Time-series representations of single-axis acceleration data for the two classes of activities of daily living for the modified ExtraSensory dataset

3.2.2 Classification Algorithm

In this chapter, we investigate four different classification algorithms in our quest to explore the differences in feature engineering and raw deep learning when it comes to HAR. First, we use a support vector machine (SVM) classifier which is trained for a direct comparison with the results of the original paper. SVM constructs an optimal hyperplane or a set of hyperplanes in a high or infinite dimensional space. The hyperplane in Figure 3.3 which has the largest distance to the nearest data points of any class would achieve a good separation characterized by the larger margins for a lower generalization error [12].

To maintain a reasonable computational load, SVM uses various kernel functions. In this project, the popular Radial Basis Function (RBF) kernel is used to separate the otherwise not linearly separable data. The right image on Figure 3.3 displays the advantages of using a non-linear kernel to be able to separate two classes which are not linearly separable.

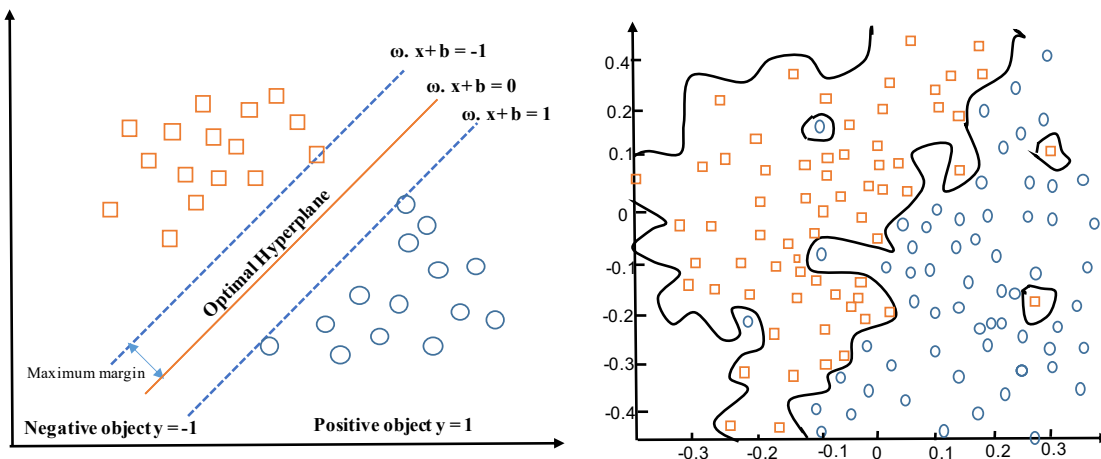


Figure 3.3: Optimal hyperplanes of a linear (left) and a radial basis function kernel (right) SVM

In addition to the SVM classifier, a deep feedforward artificial neural network (ANN) is also trained to improve the classification rate. ANN as a multi-layer perceptron (MLP) performs a nonlinear mapping between the input (x) and the output (y) such that $y = f(x; \theta)$ by learning the parameter set θ for the best possible approximation [31]. The layers of MLP include units to transform the linear sum of inputs to the unit. Each layer is represented as

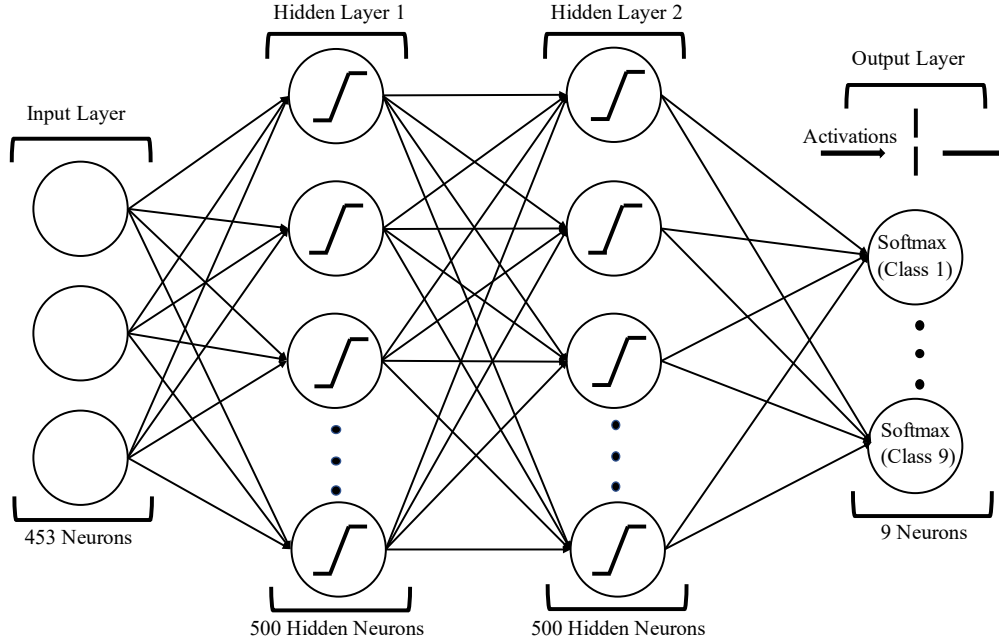


Figure 3.4: A sample topology of the feedforward neural network used for A-9 with two hidden layers

$y = f(\sum_i w_i x + b)$, where f is the non-linear activation function, w_i are the set of parameters or connection weights for neuron i , x is the input vector which is often the output of the previous layer, and b is the bias vector. In a feedforward MLP, the information flows in one direction from the input layer through the hidden layers to the output. Hidden layers increase the non-linearity of the network architecture and cause changes in the representation of the data for better generalization over the mapping function. Figure 3.4 shows a sample feedforward neural network architecture which is used in our study. This sample network consists of two hidden layers with 500 hidden neurons in each layer. The input dimension of this network is 453 (representing the size of the raw time-series vector for each observation) and the output layer consists of 9 neurons (representing each of the 9 activity labels).

To further document the performance of deep architectures using raw data versus feature-based models, a recurrent long-short-term-memory network is also implemented. To explore the latent qualities of the deep neural network, we exploit the temporal dependencies within the time-series dataset via recurrent connections of LSTM cells [115]. In a recurrent network,

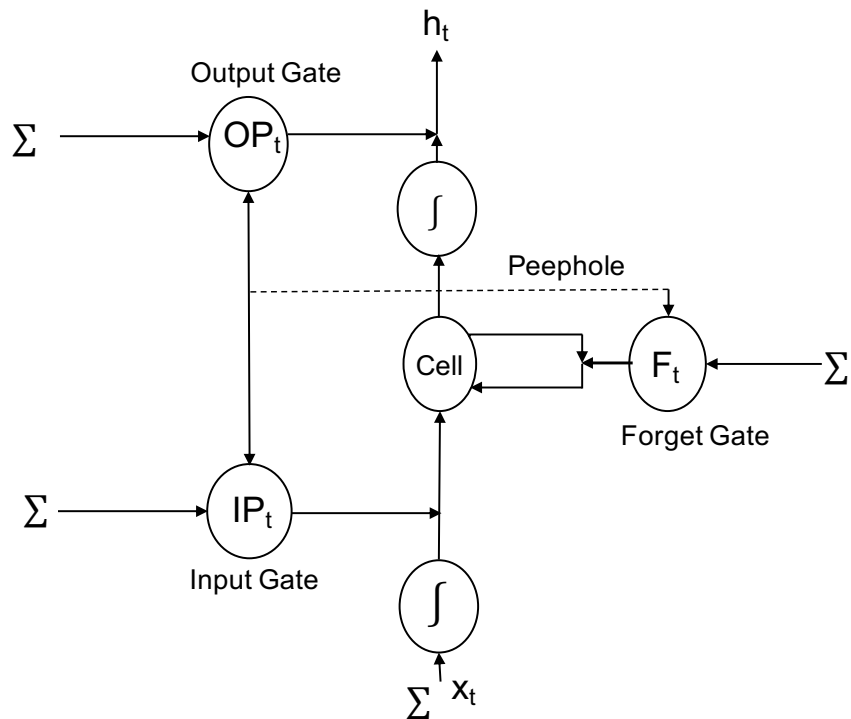


Figure 3.5: Schematic representation of the RNN-LSTM network

the connections form a directed cycle and the current time step t considers the state of the network in the previous time steps $t - 1, t - 2$ etc. [35]. Figure 3.5 shows the schematic diagram of the RNN-LSTM network. This network is made up of three different gates: (i) input gate (IP_t) which receives the previous output and the current input and passes to a sigmoid layer, (ii) output gate (OP_t) controls the output activation of the LSTM unit h_t , where the peephole connections provide the gates with the cell value, as an additional input for the gate activation [30] and (iii) forget gate (F_t) stores the information of its internal state in memory and determines the extent to which the information remains in the central cell. The content of the LSTM memory cell can be modified by the unit's input x_t and the simultaneous activation function of the input gate [67]. When these cells predict the output, the information in memory is reset depending on the present input and the information from the past internal states.

As the final modern architecture we investigate 1D-CNN. The difficulties of extracting hand crafted features is that feature engineering requires deep expertise of domain knowledge,

whereas with the deep 1D-CNNs the sequence can be convoluted iteratively by sliding the kernels over the input-sequence to map it to hidden vectors of features. A CNN is formed by using two distinct types of layers: (i) the convolutional layer which is a combination of both 1D convolutional layers and pooling layers and (ii) the fully connected layers which function like a standard MLP. Figure 3.6 displays the high level architecture of 1D-CNN model.

In 1D-CNN, the input layer receives raw 1D signals and the output layer is similar to an MLP. In the convolutional layers, an input sequence is analyzed by a set of filters to map the features with a pooling layer to reduce the size of the input space. The reduction of size of the learned features helps speed up the training process. The output of the pooling layer is fed into a set of fully connected layers which flatten the feature map and compare the probabilities of each feature occurring in conjunction with the others to achieve the highest classification rate. Since 1D-CNN performs a linear operation during the forward and back-propagation operations, it provides low computational complexity when classifying 1D data. In each 1D-CNN layer forward propagation can be represented as,

$$x_k^l = b_k^l + \sum_{i=1}^{N_{l-1}} \text{conv1D}(w_{ik}^{l-1}, s_i^{l-1}) \quad (3.1)$$

where x_k^l and b_k^l are the input and the bias of k^{th} neuron at layer l , s_i^{l-1} is the output of the i^{th} neuron at layer $l - 1$ and w_{ik}^{l-1} is the kernel from the i^{th} neuron at layer $l - 1$ to the k^{th} neuron at layer l . After the *conv1D* operation the dimension of x_k^l is less than the output s_i^{l-1} . The intermediate output can be obtained by passing the input through an activation function $f(x)$ as follows:

$$\begin{aligned} y_k^l &= f(x_k^l) \\ s_k^l &= y_k^l \downarrow \text{ss} \end{aligned} \quad (3.2)$$

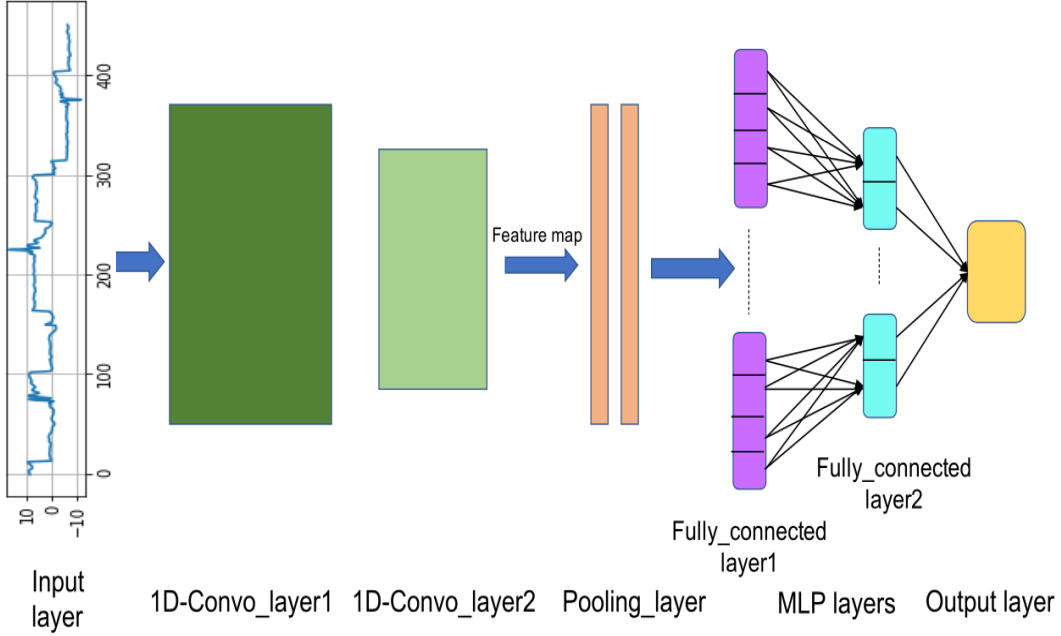


Figure 3.6: A high level representation of 1D-CNN classifier

where s_k^l is the output of the k^{th} neuron of layer l and $\downarrow ss$ stands for the down sampling operation with a scalar factor ss . The mean squared error of the back-propagation algorithm is defined by E_p in (3.3).

$$E_p = MSE(t^p, [y_1^L, \dots, y_{N_L}^L]') = \sum_{i=1}^{N_L} (y_i^L - t_i^p)^2 \quad (3.3)$$

Here, the input layer is considered as $l = 1$ and the output layer is considered as $l = L$. If N_L represents the number of classes, then for an input vector \mathbf{p} the target and the output vectors are t^p and $[y_1^L, \dots, y_{N_L}^L]'$ respectively [53].

3.2.3 Feature Extraction

Human activity recognition can be divided into two categories. The first one is the video-based HAR which includes monitoring and collecting data through a video camera [113]. The second one is the sensor-based HAR [57] where sensors collect high resolution time-series data by mobile accelerometers which could be mobile [92, 13], wrist-worn [100], waist-mounted [32] or gyroscopes and magnetometers [112]. In accelerometer-based systems,

the multi-axial sensors are used for data collection including normal activities like walking, standing, lying down, sitting, jumping, going up the stairs etc. These powerful sensors are small and cost effective enough to have become ubiquitous as they are placed inside mobiles, watches, laptops, etc. and collect the data in three physical dimensions.

The main focus of our work is the analysis of domain-specific features and how they compare to deep learning architectures running on raw temporal data. Feature engineering (FE) models have been incorporated using both datasets.

3.2.3.1 FE Using UniMiB-SHAR Dataset

In the deep learning model, each time domain sample of the dataset is fed as input to the algorithm which is expected to learn the features as a non-linear distribution between the hidden layers. In the feature-based model, 13 features are extracted manually from the time domain samples of the dataset and are provided as input to the classifiers.

Most of the features used in our study are domain-specific and well-regarded in the literature as motion-sensitive which includes the mean (on x, y and z), the variance (on x, y and z), pitch, roll, yaw, zero-crossing rate of the mean (on x, y and z) and the magnitude. The mean and variance as the first and second order statistics usually are the two most commonly used features in feature-engineering regardless of the application domain [43]. Figure 3.7 shows the mean and variance features of UniMiB-SHAR A-9 dataset for different samples in each class of activity. To consider the change of orientation during motion, three features, pitch, roll and yaw are used which are calculated as follows [52].

$$pitch = \arctan \left\{ \frac{x_N}{\sqrt{y_N^2 + z_N^2}} \right\} - \arctan \left\{ \frac{x_1}{\sqrt{y_1^2 + z_1^2}} \right\} \quad (3.4)$$

$$roll = \arctan \left\{ \frac{y_N}{\sqrt{x_N^2 + z_N^2}} \right\} - \arctan \left\{ \frac{y_1}{\sqrt{x_1^2 + z_1^2}} \right\} \quad (3.5)$$

$$yaw = \arctan \left\{ \frac{z_N}{\sqrt{x_N^2 + y_N^2}} \right\} - \arctan \left\{ \frac{z_1}{\sqrt{x_1^2 + y_1^2}} \right\} \quad (3.6)$$

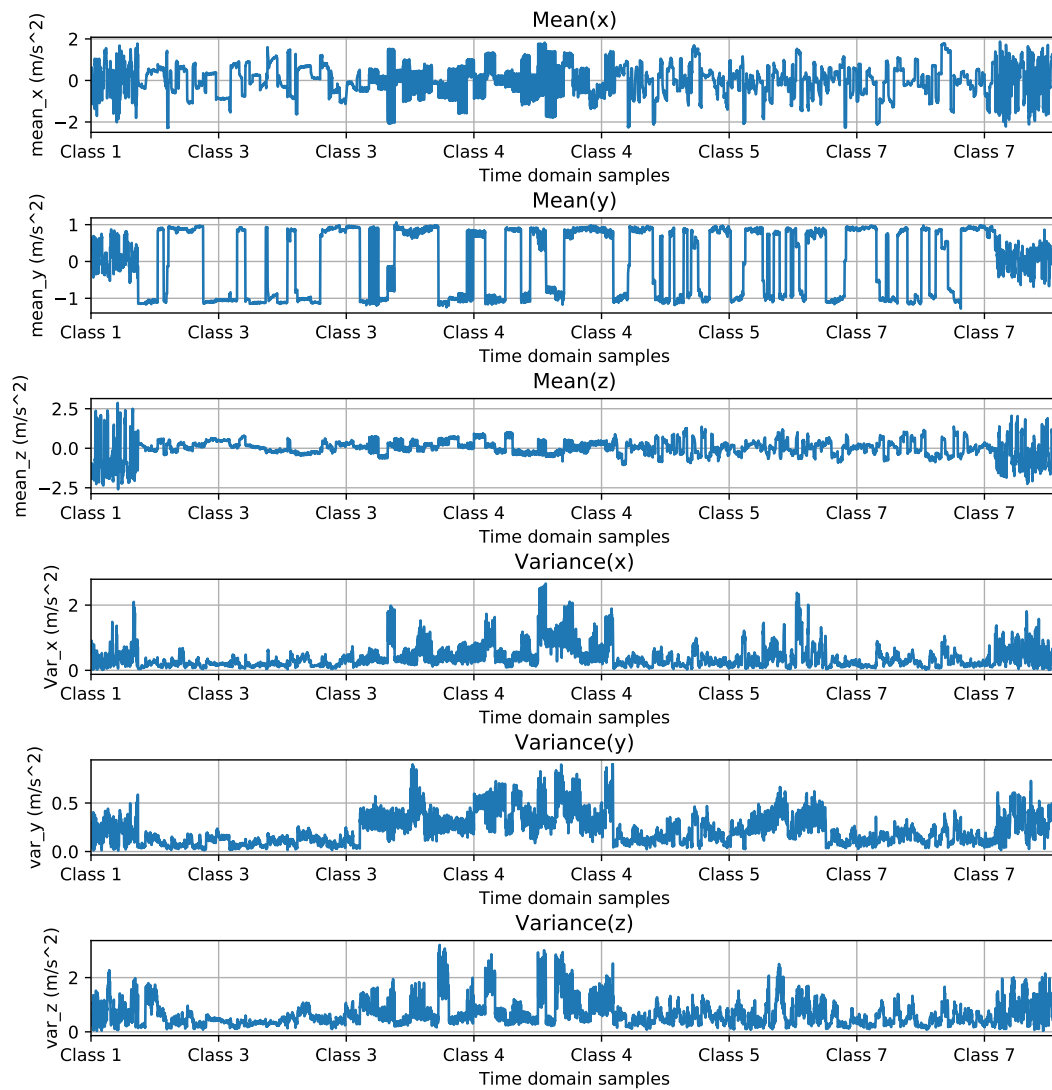


Figure 3.7: Mean and variance features for the UniMiB-SHAR dataset

In (3.4), (3.5) and (3.6) x_N, y_N and z_N represent the N^{th} column data-matrix of x, y and z axes and x_1, y_1 and z_1 represent the 1st column data-matrix of the corresponding axis. One of the most significant features which boosts up the model accuracy is the zero-crossing rate of the mean. It is the rate at which the mean changes from positive to negative and vice versa. The feature extraction is performed using the Librosa package of Python [66]. To round up the feature set, a commonly used acceleration feature, magnitude, is added which can be calculated from the mean accelerations on each axis as follows (3.7):

$$R = \sqrt{a^2 + b^2 + c^2} \quad (3.7)$$

where a, b and c represent the mean of x, y and z axes respectively.

The features pitch, roll, yaw, zero-crossing mean rate and magnitude are shown in Figure 3.8 for the A-9 dataset for different samples in each class of activity.

3.2.3.2 FE Using Modified ExtraSensory Dataset

For the modified ExtraSensory dataset, we extract the same thirteen domain specific features as the UniMiB-SHAR dataset. Figure 3.9 and 3.10 display the thirteen extracted features on the modified ExtraSensory dataset.

3.2.4 Feedforward ANN Setup

Different topologies are used for the neural network setup to cover a wide range of scenarios and to ensure a robust dynamic comparison in this study. For feedforward neural networks, the different topologies include: one hidden layer with 500 hidden neurons, two hidden layers with 500-500 hidden neurons, three hidden layers with 250-250-50 hidden neurons, four hidden layers with 250-250-50-20 hidden neurons and five hidden layers with 250-250-50-50-20 hidden neurons. In tables presenting the results, hidden layers are represented as HL1, HL2 etc. respectively.

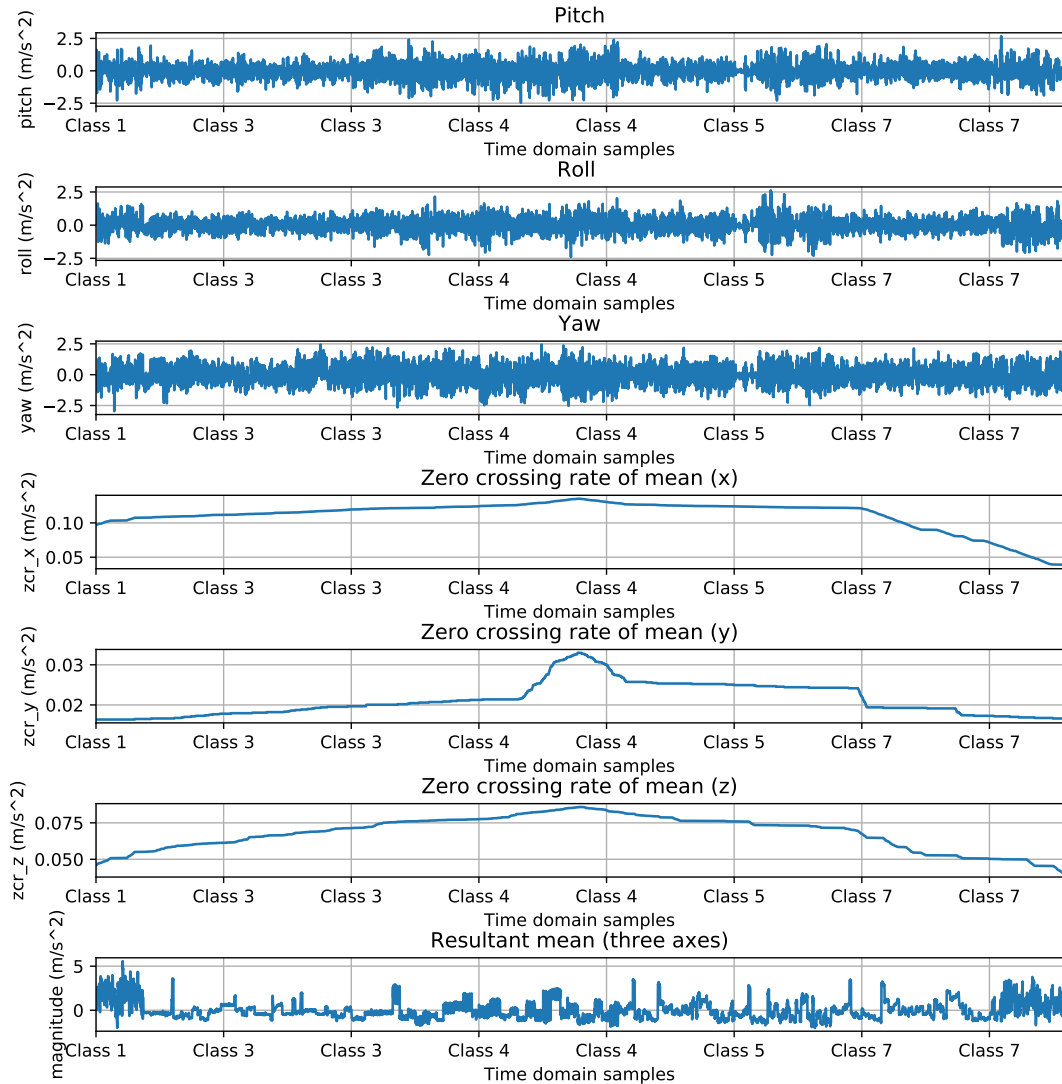


Figure 3.8: Pitch, roll, yaw, zero crossing rate of the mean and magnitude features for the UniMiB-SHAR dataset

3.2.5 RNN-LSTM Setup

In time-series classification, the advantage of a recurrent LSTM network is that it can directly learn without domain knowledge on the input features. In this chapter, the topologies of LSTM networks follow the same setup as the feedforward neural network classifier above.

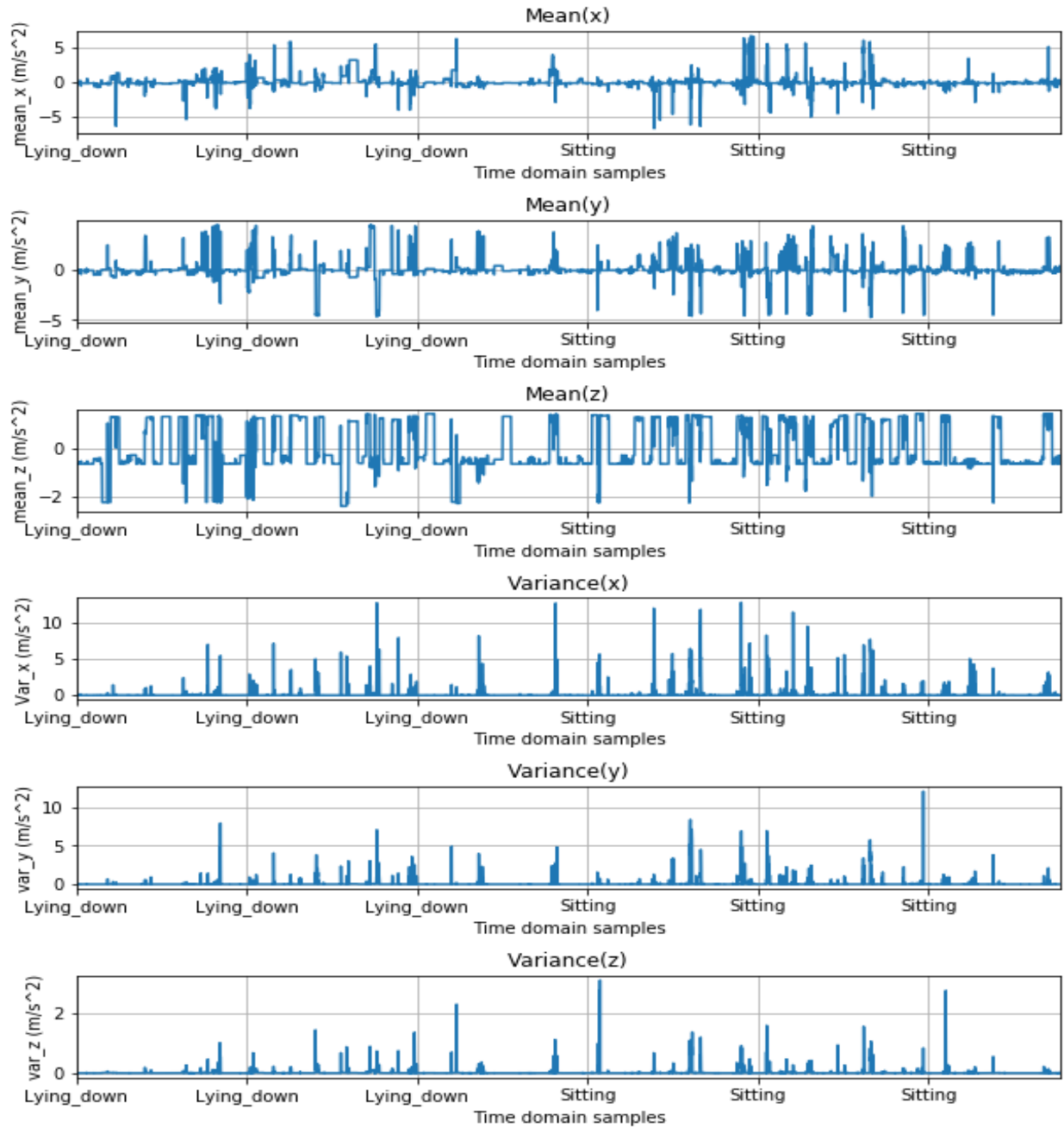


Figure 3.9: Mean and variance features for the modified ExtraSensory dataset

3.2.6 1D-CNN Setup

In order to improve the performance of the time-series classifier with lesser computational complexity, we train raw data and domain-specific features using the 1D-CNN. In this study two convolutional layers with filter sizes 64 and kernel sizes 3, followed by a dropout layer

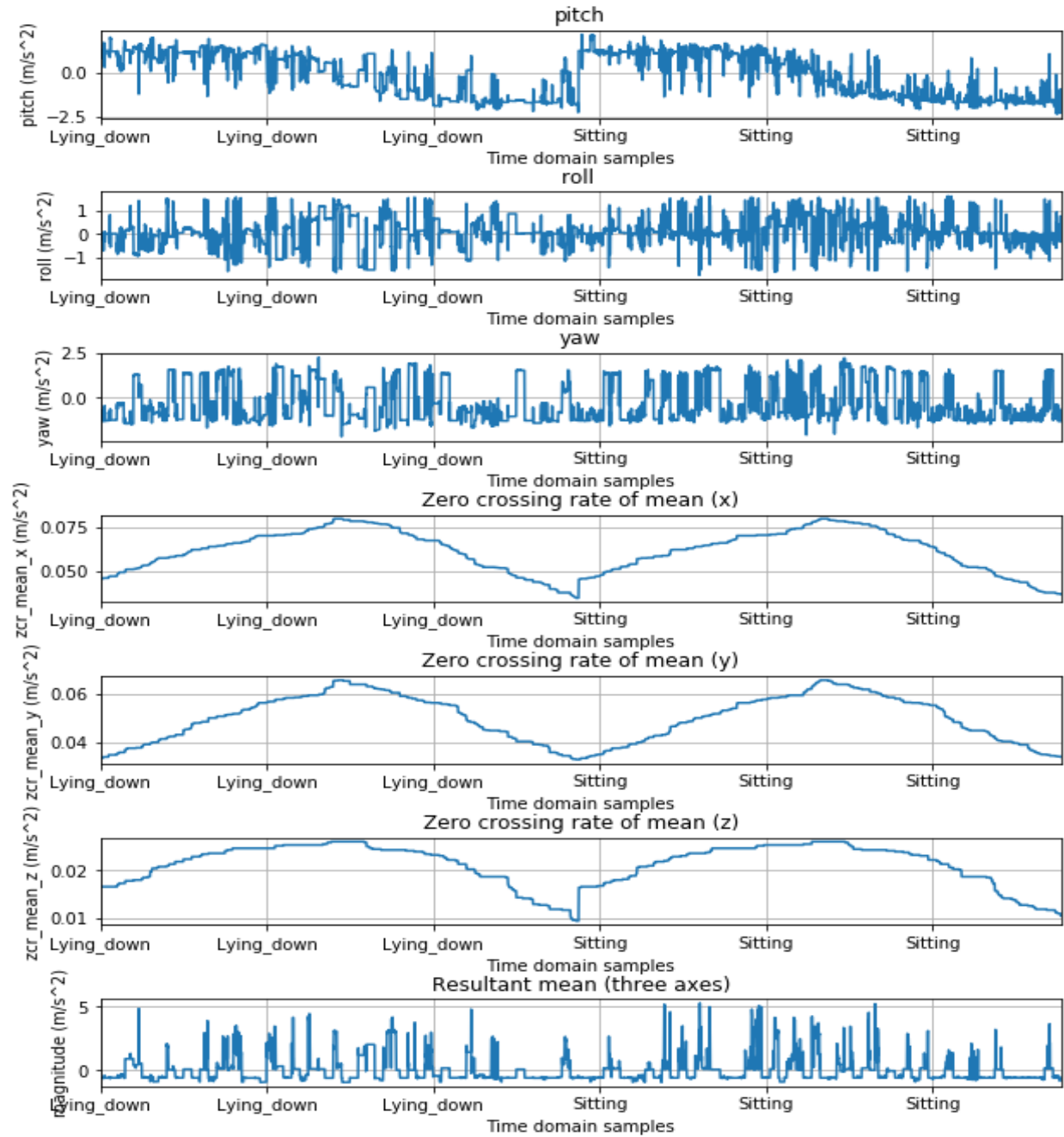


Figure 3.10: Pitch, roll, yaw, zero crossing rate of the mean and magnitude features for the modified ExtraSensory dataset

for regularization and a pooling layer of size 2 are considered for feature learning. After the pooling layer the learned features are flattened using the same five topologies of fully connected layers discussed above.

3.3 Results and Discussion

As briefly described before, we consider four different classification algorithms on two separate classification tasks in this study.

The two classification tasks are labeled as AF-2 and A-9, following the original paper which introduced the dataset for the first time. AF-2 includes 4192 samples which belong to the motion class labeled “fall” and 7579 samples which include a variety of observations from other motion classes (ADLs). The objective for AF-2 is the binary classification of samples into fall vs. no-fall categories. A-9 on the other hand includes 7579 samples from each of the 9 “non-fall” motion classes where the objective is standard multi-way classification. Four different algorithms are used for both AF-2 and A-9 in the form of SVM, a deep feedforward ANN, RNN-LSTM and 1D-CNN. After the normalization of data, we used partitioning to obtain statistically significant results for all configurations. For the SVM classifier, we applied a 10-fold cross-validation where the dataset was split into 10 partitions and the algorithm was trained with 90% of the dataset and tested with 10% of the dataset for each training cycle. As for the neural network architectures, we applied random sub-sampling for cross-validation where we randomly picked 20-30% of the dataset for testing and the remaining partition for training.

For all configurations, the data is represented in two formats: (i) low dimensional feature set extracted from the raw acceleration data as described in section 3.2.3 and (ii) high dimensional raw data with minimal preprocessing. For AF-2, the activation function used for the hidden layers is ‘relu’ where the output layer uses a ‘sigmoid’ function. Since the model trained for A-9 is a multiclass classifier, the output activation function is ‘softmax’. The stochastic gradient decent and Adam optimizers are used in these experiments with a learning rate of 0.0001 and L2 regularization with probability 0.1 to prevent overfitting. For 1D-CNN a 0.5 probability Dropout is applied.

3.3.1 Experimental Results

3.3.1.1 Results of UniMiB-SHAR Dataset

Tables 3.2 and 3.3 include a detailed performance comparison between the four classifiers using raw data and domain-specific features for AF-2 and A-9 datasets respectively. In the case of binary classification, the accuracies are approximately the same regardless of the input representation. For binary classification, 1D-CNN gives the highest classification rate of 99.82 % and the raw data model outperforms the features. However, there is a significant difference for the multi-class classification problem, where raw temporal data outperforms domain-specific feature extraction by a wide margin. Table 3.3 shows that the RNN-LSTM network on A-9 raw data using 250-250-50 topology reports the highest mean accuracy of 98.02% which outperforms the feature-based models with the highest accuracy of 96.35%. It is worth noting that the raw temporal data not only outperforms the domain-specific features but in fact achieves the highest recorded accuracy in the literature, including this study, on this dataset.

Table 3.4 shows the analysis of computational time and complexity for all classifiers on both datasets. The computer used in this study is configured with the following specifications: Intel(R) core (TM) i7 4.20GHz CPU, NVIDIA GeForce GTX 1080 GPU and 32 gigabyte memory (RAM). For the binary classification task, the SVM classifier displays fast convergence in training, especially when using features, and similarly the quickest on-demand test times compared with the other classifiers. Among the neural network setups, as expected, the RNN-LSTM has the slowest training and testing times of any classifier due to the recurrent feedback loops which require forward and back propagation. Surprisingly, 1D-CNN demonstrates comparable training/testing times with ANN. It is worth noting however, if one were to ignore the significant differences in the offline training times, the on-demand testing times for all classifiers are all very short.

Table 3.2: Mean accuracies of four classifiers on AF-2 UniMiB-SHAR dataset

Classifier	HL1	HL2	HL3	HL4	HL5	Mean Acc raw data (%)	Mean Acc features (%)
SVM	—	—	—	—	—	98.73	96.87
ANN	500	—	—	—	—	98.76	97.32
	500	500	—	—	—	98.64	99.02
	250	250	50	—	—	98.89	99.21
	250	250	50	20	—	99.01	99.30
	250	250	50	50	20	98.51	98.92
RNN-LSTM	500	—	—	—	—	99.38	99.41
	500	500	—	—	—	99.52	99.12
	250	250	50	—	—	99.51	99.32
	250	250	50	20	—	99.54	99.13
	250	250	50	50	20	99.46	99.32
1D-CNN	500	—	—	—	—	99.77	99.20
	500	500	—	—	—	99.77	98.57
	250	250	50	—	—	99.82	98.96
	250	250	50	20	—	99.72	99.04
	250	250	50	50	20	99.78	98.81

Table 3.3: Mean accuracies of four classifiers on A-9 UniMiB-SHAR dataset

Classifier	HL1	HL2	HL3	HL4	HL5	Mean Acc raw data (%)	Mean Acc features (%)
SVM	—	—	—	—	—	85.47	63.14
ANN	500	—	—	—	—	87.79	91.61
	500	500	—	—	—	87.86	92.07
	250	250	50	—	—	88.20	93.79
	250	250	50	20	—	85.75	91.87
	250	250	50	50	20	84.83	92.05
RNN-LSTM	500	—	—	—	—	94.13	95.78
	500	500	—	—	—	97.89	95.84
	250	250	50	—	—	98.02	96.35
	250	250	50	20	—	97.96	95.84
	250	250	50	50	20	97.11	95.75
1D CNN	500	—	—	—	—	97.46	95.44
	500	500	—	—	—	97.25	94.74
	250	250	50	—	—	97.18	94.69
	250	250	50	20	—	96.94	95.35
	250	250	50	50	20	97.14	95.02

Table 3.4: Computational complexity of classifiers on AF-2 and A-9 UniMiB-SHAR dataset

Data	Classifier	Accuracy (%)	Computational Time (Sec)	Train Time (Sec)	Test Time (Sec)	Epochs/ Training	Training Parameters
AF-2	SVM Raw	98.73	91.18	91.18	0.0001	10 Fold CV	—
	SVM Feature	96.87	11.7	11.7	0.001	10 Fold CV	—
	ANN Raw	99.29	55.67	55.41	0.27	100	2600493
	ANN Feature	99.01	167.23	167.17	0.06	500	79841
	RNN-LSTM Raw	99.7	555.67	553.99	1.69	52	3010501
	RNN-LSTM Feature	99.45	572.29	571.24	0.97	350	825251
	1D-CNN Raw	99.83	75.43	75.16	0.27	150	2349993
	1D-CNN Feature	99.26	71.26	70.99	0.26	200	141609
A-9	SVM Raw	85.47	184.65	184.65	0.001	10 Fold CV	—
	SVM Feature	63.14	13.81	13.81	0.001	10 Fold CV	—
	ANN Raw	87.14	157.57	157.48	0.09	800	189259
	ANN Feature	94.53	977.64	977.59	0.05	5000	79259
	RNN-LSTM Raw	97.82	4617.31	4615.25	2.01	800	841269
	RNN-LSTM Feature	96.57	1018.65	1017.75	0.906	1000	825659
	1D-CNN Raw	97.14	37.06	36.78	0.28	100	2354001
	1D-CNN Feature	96.09	88.36	88.17	0.19	300	145617

For the A-9 multi classification task, the SVM begins to fall behind both in accuracy and computational times as expected from an otherwise binary classifier. Among the networks, 1D-CNN has the best performance/speed trade-off compared with the other two classifiers where ANN has significantly lower performance and RNN-LSTM has significantly higher training times. In fact, when using raw data, the performance difference between the 1D-CNN and RNN-LSTM is almost insignificant whereas the difference in training times is noticeably large.

Figure 3.11 and 3.12 represent the confusion matrices of AF-2 and A-9 raw data classification achieved with 1D-CNN and RNN-LSTM classifiers respectively. Figure 3.11 shows the classification accuracy as 99.86% and Figure 3.12 shows the classification rate as 97.82%. It is interesting to note that the highest classification performances on the binary classification task, AF-2, when using SVMs, are statistically the same between the original paper and this one (98.71% versus 98.73%). However, the highest accuracy reported for A-9 in the original paper was 88.41% using a random forest classifier whereas the highest accuracy in this study

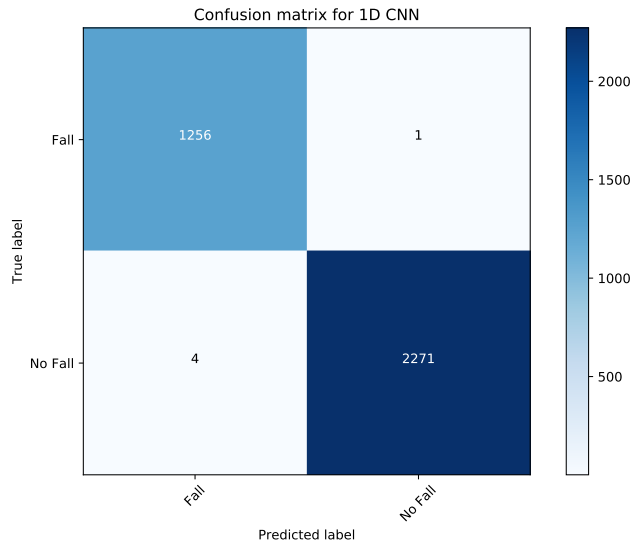


Figure 3.11: Confusion matrix of AF-2 raw data for 1D-CNN classifier

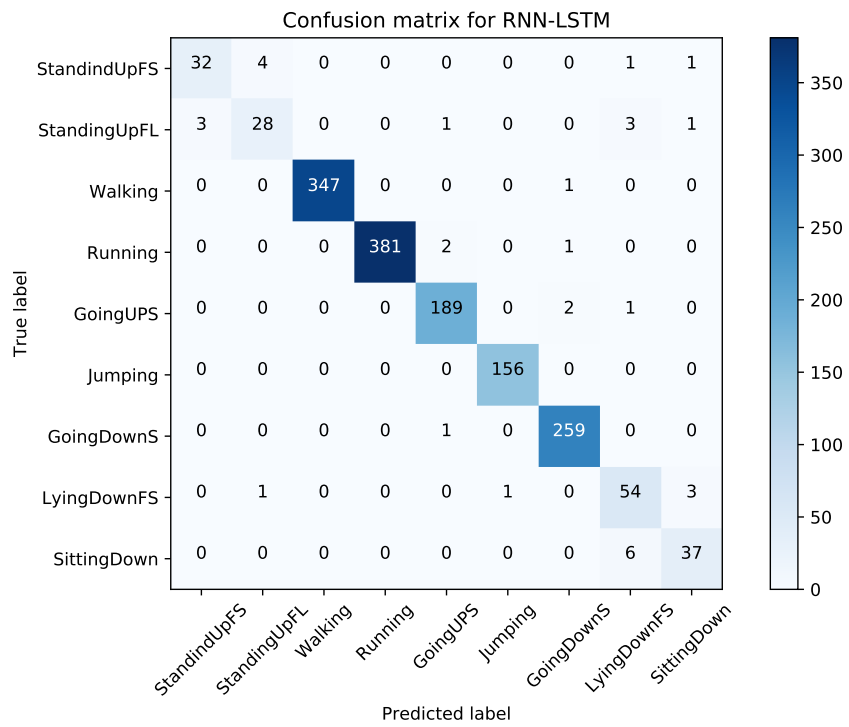


Figure 3.12: Confusion matrix of A-9 raw data for RNN-LSTM classifier

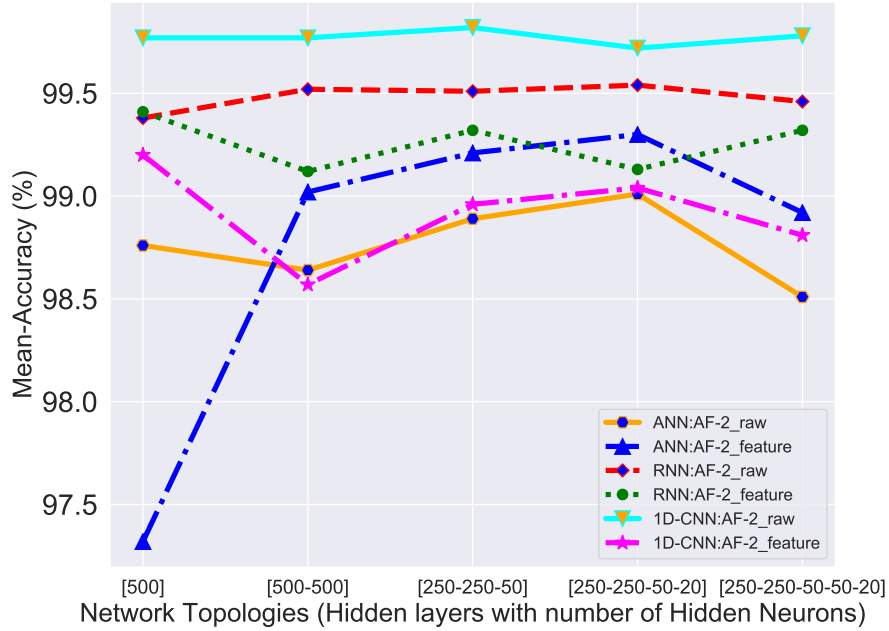


Figure 3.13: Mean accuracy plot of raw and feature-based models of UniMiB-SHAR AF-2 dataset based on various topologies

is obtained using an RNN-LSTM on raw data with a 98.02% classification rate. This is to the best of our knowledge the highest classification rate reported on this dataset in the literature so far. It also highlights the potential improvement in using complex non-linear models for learning.

Comparing tables 3.2 and 3.3, another interesting observation is in how the accuracy gaps between features and raw data change for different classification tasks. For example, in the case of AF-2 (table 3.2), the features demonstrate a competitive performance when it comes to binary classification accuracy for fall vs. no fall categorization. However, in the multiclass problem (A-9, table 3.3) the gaps become significantly larger where the neural networks seem to create a rich latent space from raw data to outperform engineered features for the RNN-LSTM and 1D-CNN configurations. Most interestingly, the same observation is not true for ANN where features outperform raw data representation. We conclude that as long as the network topology allows for constructing a rich latent feature space from

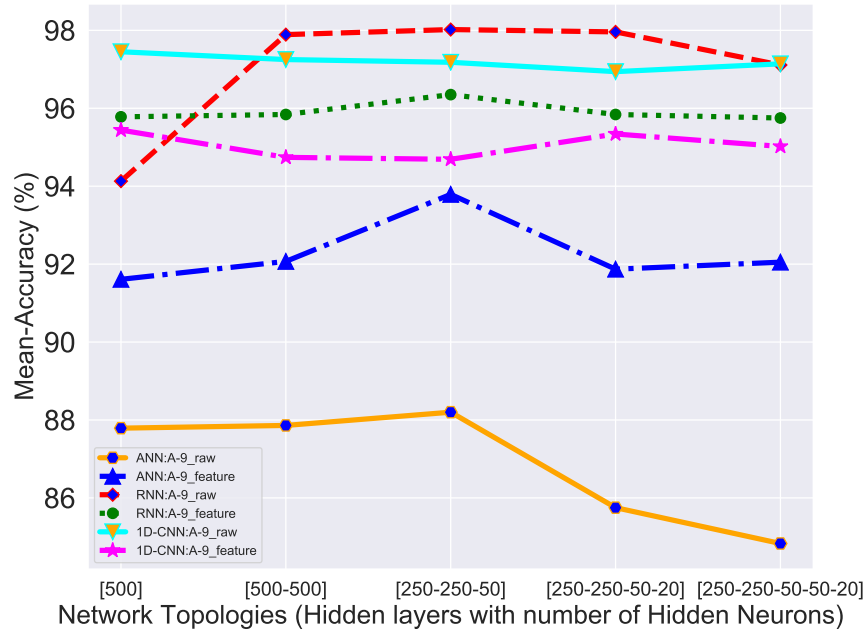


Figure 3.14: Mean accuracy plot of raw and feature-based models of UniMiB-SHAR A-9 dataset based on various topologies

the raw input data like the 1D-CNN, they can outperform expert features provided that there is sufficient data to learn from. Figure 3.13 and 3.14 show how the average accuracies change across topologies running on the AF-2 and A-9 datasets respectively for ANN, RNN-LSTM and 1D-CNN classifiers as the complexity of the network changes. The figures show that as the network gets deeper and more complex the accuracies generally increase as well. Moreover, as the network becomes larger with a greater number of parameters to model the nonlinear relationship, they begin to better utilize the richer information contained in the two complex datasets.

3.3.1.2 Results of Modified ExtraSensory Dataset

For the modified ExtraSensory dataset, we extract the same motion-related features like the UniMiB-SHAR dataset and follow the same network topologies & model parameters for the ANN model as implemented for the UniMiB-SHAR dataset.

Table 3.5: Mean accuracies of four classifiers on modified ExtraSensory dataset

Dataset	HL1	HL2	HL3	HL4	HL5	Mean Acc raw data (%)	Mean Acc features (%)
SVM	—	—	—	—	—	85.47	63.14
5775 Observations	500	—	—	—	—	68.99	74.89
	500	500	—	—	—	73.86	77.01
	250	250	50	—	—	76	78.54
	250	250	50	20	—	81.28	88.46
	250	250	50	50	20	84.37	89.39
55938 Observations	500	—	—	—	—	66.92	80.28
	500	500	—	—	—	66.13	85.61
	250	250	50	—	—	84.26	92.51
	250	250	50	20	—	83.74	91.12
	250	250	50	50	20	86.76	93.10

Table 3.5 shows that for the low dimensional dataset with 5775 observations, the highest classification accuracies are 84.37% and 89.39% for the raw data and extracted features respectively. As the size of the data is increased to 55938 observations, the accuracies of both the raw data and the feature set increase yet with significant differences. The classification accuracy for the raw data is increased to 86.76% whereas the domain specific features now achieve 93.1% accuracy. When we train several RNN-LSTM topologies on these two ExtraSensory datasets we still observe that regardless of the number of parameters, the highest accuracy achieved by the LSTM model is 84.23%, i.e., lower than the highest accuracy achieved by the less complex feedforward ANN model. Figure 3.15 shows the mean accuracy plots for raw and feature based models using feedforward ANN algorithm on the two modified extrasensory datasets to provide a clear overview of how the classification performance changes with the change in dataset size. Surprisingly, as the number of observations is increased, the performance of feature engineering gets significantly better compared to raw data which one would otherwise expect to fare better with increased data dimensionality.

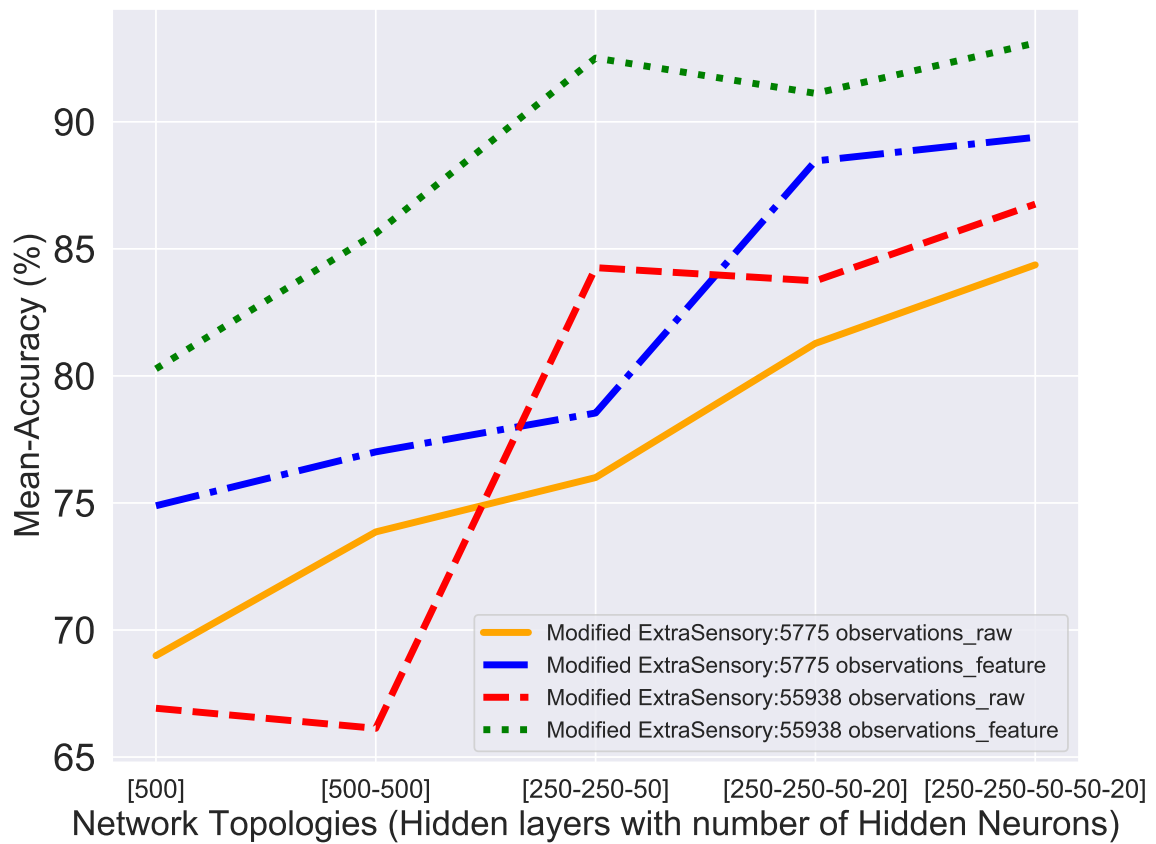


Figure 3.15: Mean accuracy plot of raw and feature-based models of modified ExtraSensory dataset based on various topologies

3.4 Summary and Conclusions

In this chapter, we show that in an activity of daily living recognition task using UniMiB-SHAR dataset, which consists of 9 different motion classes, a black-box deep learning approach using raw temporal data on a recurrent LSTM model outperforms the most recent results reported in the literature and achieves a state-of-the-art classification rate of 98.02%. However, on a different, larger and more complex modified ExtraSensory dataset where conventional logic would suggest the deep learning model would perform even better, domain specific features prove to have higher accuracy by a significant margin. What is more fascinating is that even as the number of observations increases tenfold, the accuracy advantage of feature engineering improves even further not to mention the more operational advantages of reduced computational complexity and significantly faster training and testing (when using only 13 features compared to the sample time vector of size 2400). This could be due to a number of reasons such as the reduced complexity of the classification task (binary versus multi-class) where a lower dimensional feature vector is able to better capture the key differences between the two classes but could otherwise be insufficient as more class labels and statistics get added to the mix. So we look forward to another well reputed dataset which proposes subject agnostic features for state-of-the-art performance and decided to compare this performance with deep learning models.

Chapter 4: Empowering Deep Feature Learning in HAR

Parts of this chapter have been communicated to the Journal of Biomedical Informatics, Elsevier for publication and currently, the paper is in its 2nd revision stage. Copyright permission for this chapter is attached to Appendix A

4.1 Introduction

In the previous chapter, we include an in-depth study on a publicly available dataset of human activities to compare the performances of feature engineering and deep learning algorithms running on raw data where we have found that while the former has commendable performance, the latter has better accuracy throughout a wide range of scenarios [48]. In this chapter, we further expand the portfolio of technologies and consider a well-established publicly available dataset from a research group at Stanford University which included subjects of different ages, genders with different sensor placements. The observations were obtained when looking into the sensor positions at different parts of the body and with persons of different ages where the accelerometer sensors were placed on the ankle and wrist positions of both young and adult subjects. Specifically, we investigate feature-engineering as applied by the authors of the original paper and compare it to using raw data [64]. We find that in most instances, raw data provides a better performance than even the most finely engineered features. Furthermore, we explore a data-augmentation technique to boost the performance of the feature-learner on the raw data and achieve a significant improvement in its performance on one of the more challenging datasets. The rest of the chapter is organized as follows. In the next sections we describe the design and methodology and the experimental setup in detail including the specifications of the dataset as well as the classification algorithms and

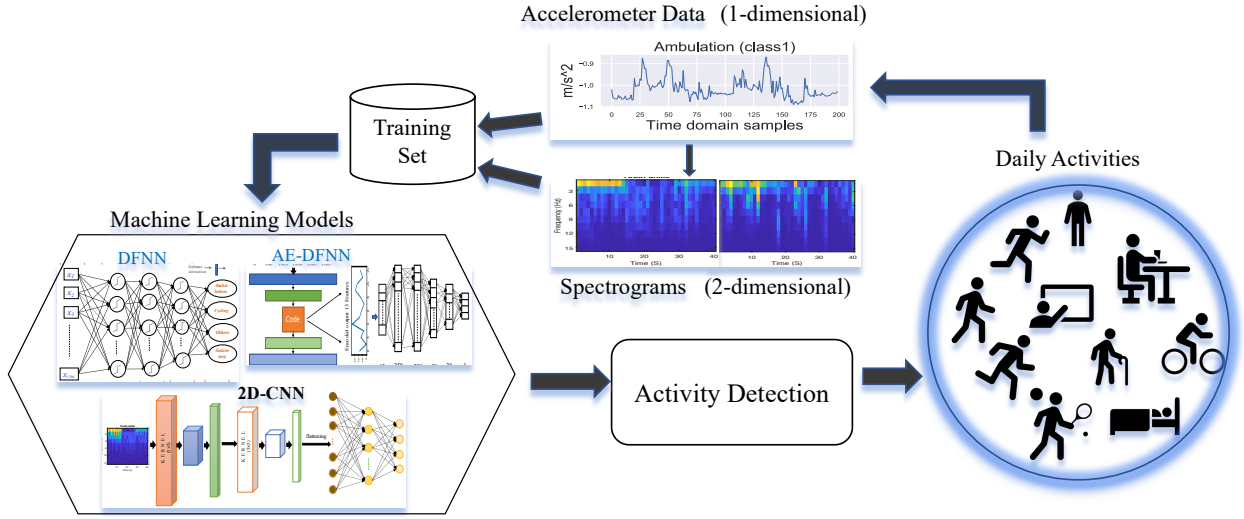


Figure 4.1: Human activity recognition using unsupervised feature learning

data augmentation such as synthetic minority over sampling technique (SMOTE) used in the experiments. The results are provided in section 4.4 followed by a detailed discussion of performance comparisons.

4.2 Design and Methodology

The framework of human activity recognition (HAR) system with unsupervised automatic feature learning technique is described in Figure 4.1. The details of this process is discussed in the rest of this chapter.

4.2.1 Dataset

In this research, we wanted to expand our analysis to subject agnostic features as described by a recent paper from the research group at Stanford University on a dataset of different ages, genders and sensor-locations [63, 64]. The overall data collection process includes 53 participants with 33 adults and 20 youths. The suits of synchronized sensors were placed on ankles and wrists of the participants who were directed to perform a sequence of common physical activities, lasting 3-5 minutes each. These common daily activities are

shown in table 4.1 [64] and categorized into four main classes: ambulation, cycling, other activities and sedentary. In this data collection process, Triaxial Wocket accelerometers [64, 45] which are small and lightweight devices, were placed on the wrist and ankle positions of the participants using custom Velcro bands. Based on the ages of the participants and the locations of the sensors on the body, the datasets are classified into four distinct groups: (i) adult-ankle, (ii) adult-wrist, (iii) youth-ankle and (iv) youth-wrist. Table 4.2 provides the sample sizes of four different datasets. Our research includes twenty-five experiments on four datasets when considering six classification algorithms for each dataset.

4.2.2 Data Preprocessing

In the data preprocessing stage, we re-sample the non-uniformly sampled data with a uniform fixed rate. More specifically, the data is resampled at 900 samples per second and filtered using a 4th order lowpass digital Butterworth filter with a cut-off frequency at 15 Hz and sampling rate at 90 Hz to limit the bandwidth and eliminate non-motion like noise in the raw data. Figure 4.2 shows some example observations of resampled (in blue) and filtered (in green) signals.

4.2.3 Classification Algorithms

To study the contributions of feature learning versus feature engineering, we employ six different popular and state-of-the-art classifiers including deep feedforward neural network (DFNN), auto encoder deep feedforward neural network (AE-DFNN), one-dimensional convolutional neural network (1D-CNN), two-dimensional convolutional neural network (2D-CNN), recurrent neural network with long short-term memory (RNN-LSTM) and feature engineering using deep feedforward neural network (FE-DFNN). Moreover, to boost up the performance of 2D-CNN classifier which has a larger number of trainable parameters and subsequently requires more data to train, we augment the number of observations using SMOTE specifically for the youth-ankle dataset where the performance lagged behind sub-

Table 4.1: Activities of daily living categorized into four classes

Class	Activities of Adult Dataset	Activities of Youth Dataset
Ambulation	Walking: carrying load Stairs: inside and down Stairs: inside and up Treadmill: 3 mph 0% incline Treadmill: 3 mph 6% incline Treadmill: 3 mph 9% incline Treadmill: 2 mph 0% incline Treadmill: 4 mph 0% incline Walking, natural	Walking, natural Treadmill walking: 2 Treadmill walking: 3 - 4 mph Treadmill running:4.5 - 5 mph – – – – –
Cycling	70rpm. 50W. 0.7kg Cycling outdoor level Cycling outdoor uphill Cycling outdoor downhill	70 rpm 50 watts Outdoor cycling – –
Others	Painting: roller Painting: brush Sweeping with broom – – – –	Basketball:- dribbling Basketball:- passing Basketball:- shortshots Clean room Soccer:- dribbling Soccer:- passing Tennis-ball:- fielding Tennis-ball:- throwing-catching
Sedentary	Sitting, internet search Sitting, computer typing Sitting: writing Sitting: reading Sorting files / paperwork Lying: on back Lying on left side Lying on-right-side Sitting: legs straight Standing still	Sitting: reading Play-computer-game Play-on-gameboy Wii:-boxing Wii:-tennis Lying: on back Sitting: legs straight Standing still – –

Table 4.2: Number of observations of four Adult-Youth datasets

Dataset	Observations
Adult-ankle	12812
Adult-wrist	12618
Youth-ankle	7910
Youth-wrist	7869

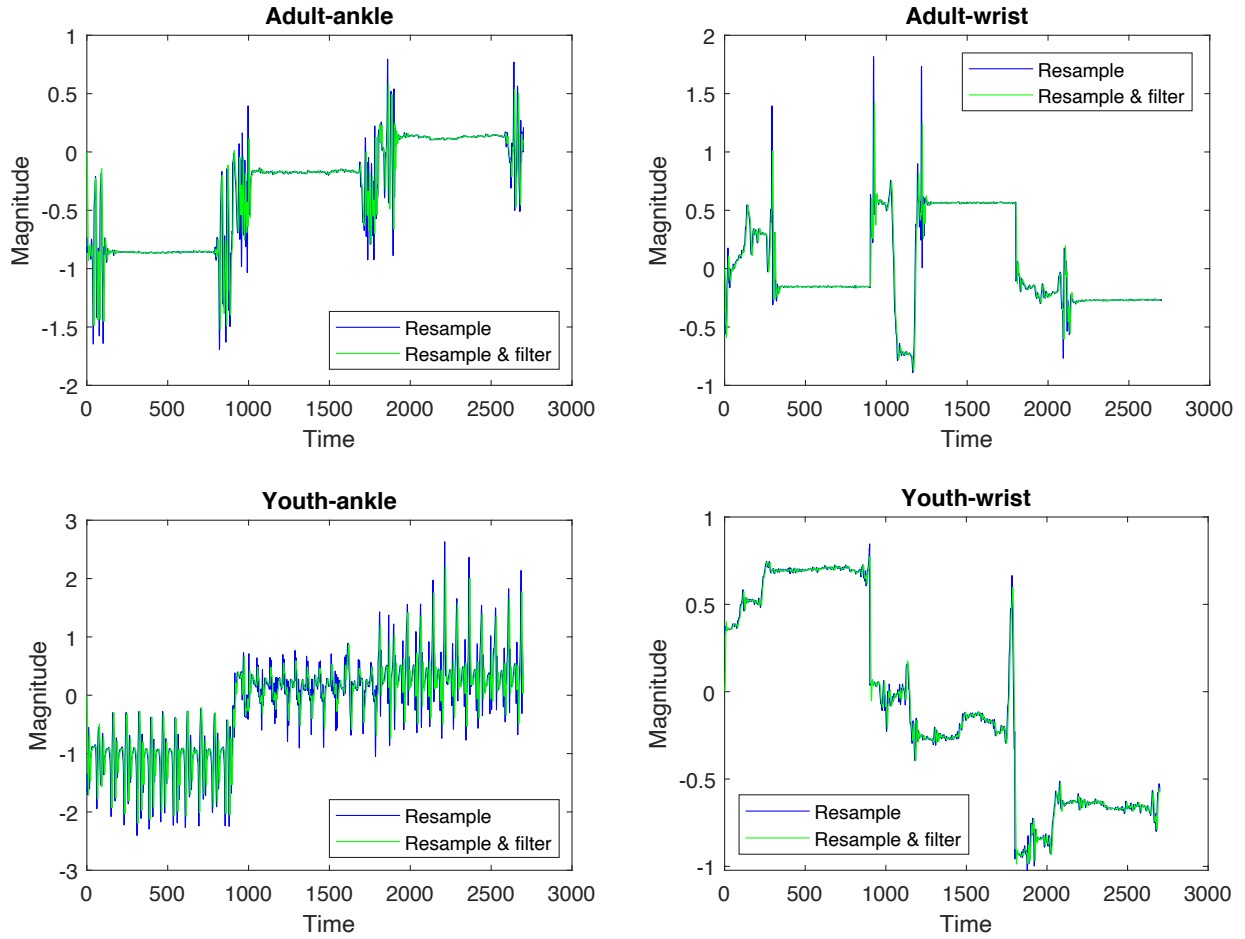


Figure 4.2: Example observations from different subjects and sensor placements for both re-sampled (marked by blue) and filtered (marked by green) signals

ject agnostic features. In this study, we perform a total of twenty-five experiments to classify four different labels which are the group of activities of daily living (ADLs). Figure 4.3 provides some representative samples for the single-axis acceleration data for these 4 groups of ADLs for the adult dataset. Each subplot in the figure is a representative sample of raw temporal acceleration data for one of the three axes (x, y or z) which happens to demonstrate the biggest dynamic range of motion as the dominant axis.

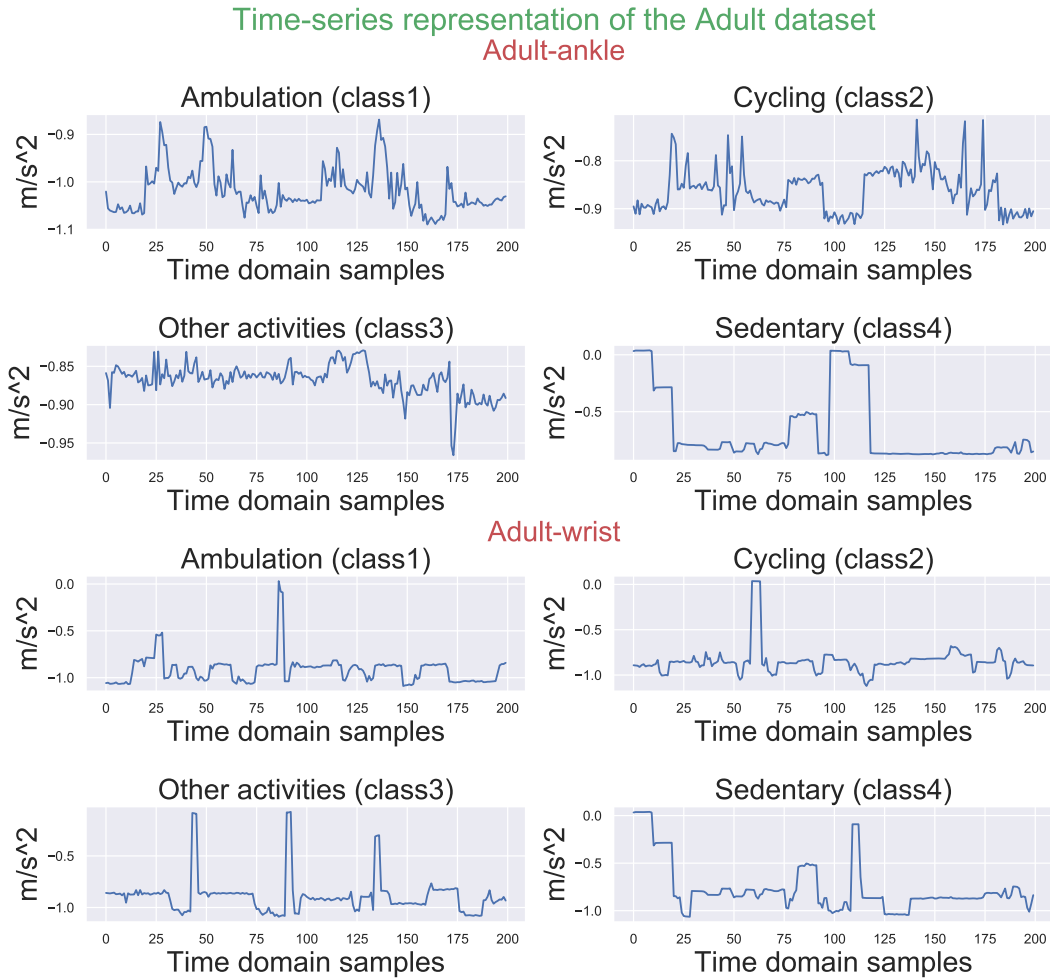


Figure 4.3: Time-series representations of single-axis acceleration data for four classes of activities of daily living for the adult-ankle and adult-wrist datasets

4.2.3.1 DFNN

Feed forward neural networks help realize a non-linear relationship between a set of predictors and a desired set of outcomes. In this particular study, the sensor data collected by the accelerometers is mapped into one of the four class labels. This is accomplished by learning a linear model on the transformed version of the input space, more specifically, to learn the nonlinear functions over the input x , the linear model is applied not to x itself but to a transformed input $\phi(x)$. ϕ is nonlinear transformation and can be defined as a set of

latent features which represents the input vector. In very broad terms, a feedforward neural network model can be represented as,

$$y = f(x, \theta, w) = \phi(x, \theta)^T w \quad (4.1)$$

where ϕ is a hidden layer and θ is used to learn ϕ from a broad class of functions and weight parameters w . A deep feedforward neural network is basically a multilayer perceptron with stacked hidden layers where each layer is represented as,

$$y = f\left(\sum_i w_i x + b\right) \quad (4.2)$$

where f is the nonlinear activation function, w_i are the set of parameters or connection weights for neuron i , x is the input vector which is often the output of the previous layer and b is the bias vector [31].

Figure 4.4 represents a simple architecture of the DFNN which was the best topological representation we arrived at for our specific experiment. As shown in the figure, in a DFNN, the information flows in one (forward) direction, from an input layer (x) to the output layer (y) through the intermediate computations which is defined as f in the dense layers or hidden layers to finally produce a value that is close to y (y_{pred}). During training, the forward propagation generates a cost function from which the backpropagation algorithm can compute the direction and size of change in weight parameters to reduce the cost function with the gradient of the network.

4.2.3.2 AE-DFNN

The AE-DFNN topology has two parts: (i) autoencoder and (ii) deep feedforward neural network. Autoencoder is a generative model which is trained to reproduce its input at its output. The autoencoder has two components: (i) encoder and (ii) decoder. The encoder

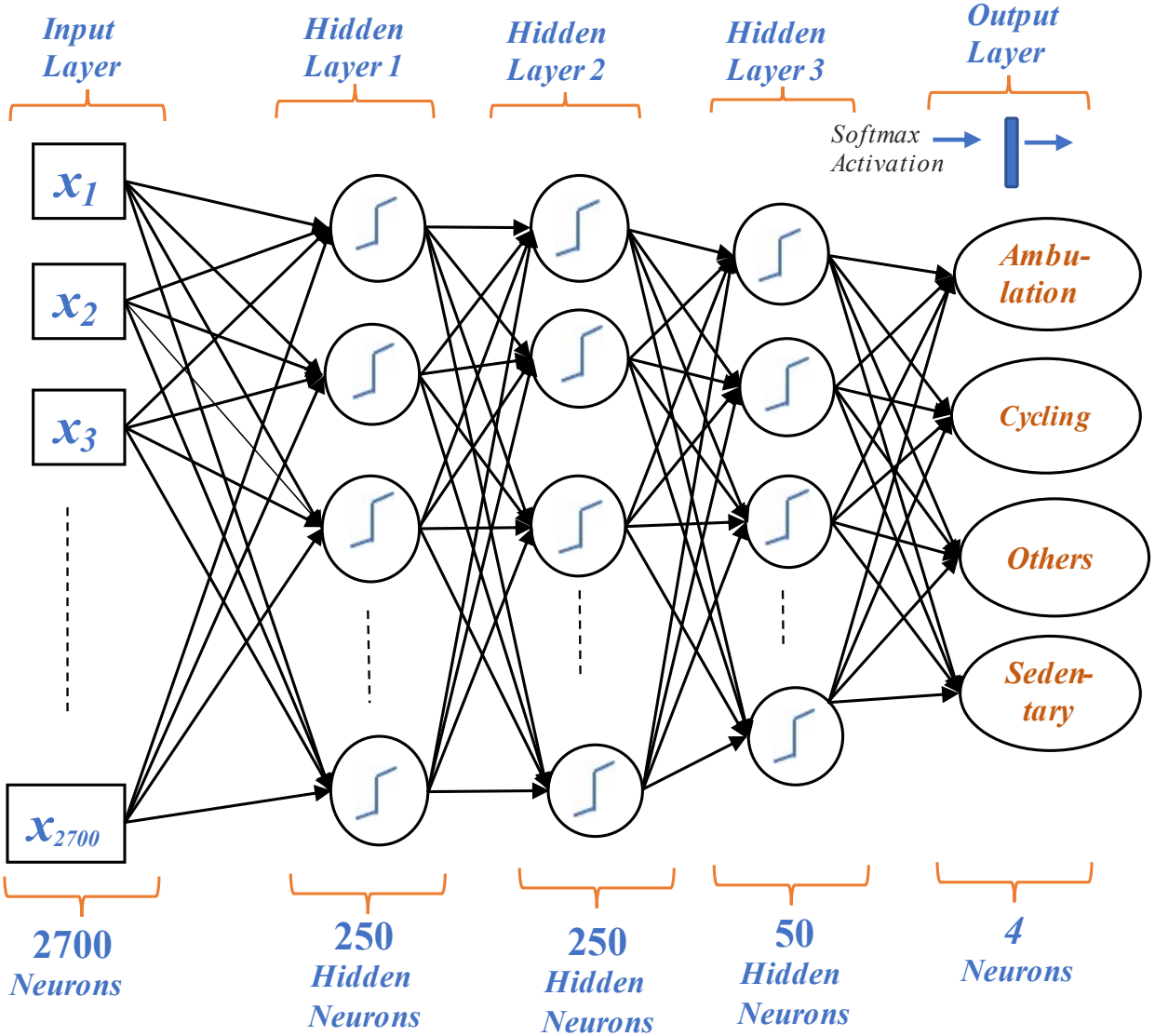


Figure 4.4: Architecture of a simple DFNN used in the experiments

function is denoted by ϕ which maps the input vector x to a latent space F , which is presented at the bottleneck. The encoding operation can be denoted as $\phi : x \rightarrow F$.

On the other hand, the decoder function is represented by ψ which maps the latent space F at the bottleneck to the output and can be shown as $\psi : F \rightarrow \bar{x}$. In (4.3) the encoding-decoding operation of the autoencoder can be defined as,

$$\phi, \psi = \underset{\phi, \psi}{\operatorname{argmin}} \|X - (\phi \circ \psi)\|^2 \quad (4.3)$$

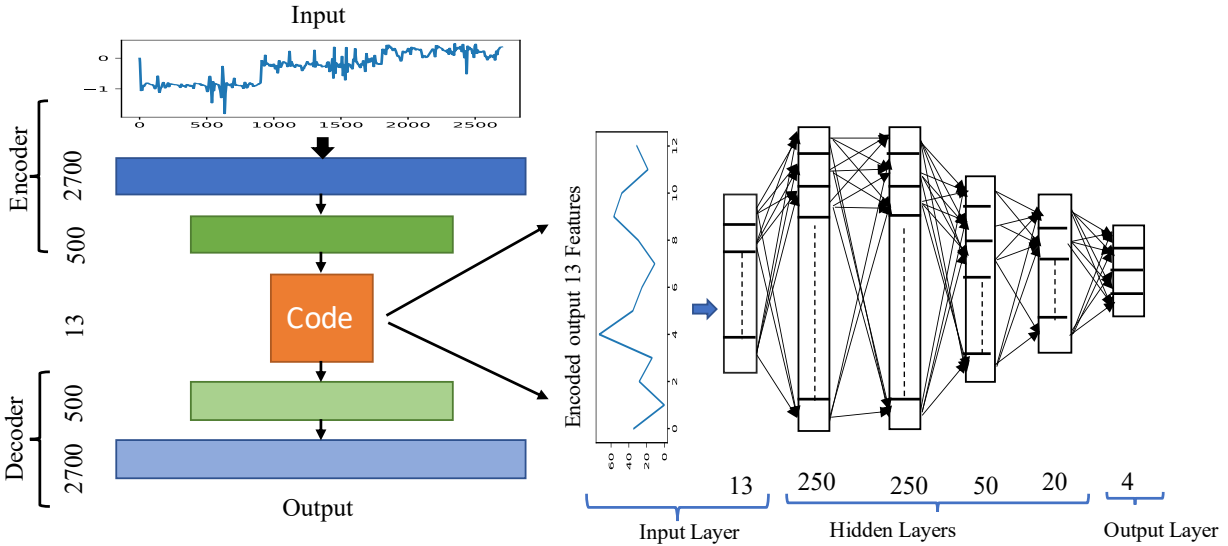


Figure 4.5: Autoencoder deep feedforward neural network model and training of learned features (encoder output) using the DFNN

For HAR, a deep autoencoder allows for automated end-to-end feature learning and thus, largely alleviates the need for domain-specific feature engineering procedures. In this regard, we have used an AE-DFNN model shown in Figure 4.5, where the input is raw data with 2700 features where each feature is sample of acceleration in time. The autoencoder framework includes an encoding layer with 500 and 13 neurons which automatically extracts 13 features using unsupervised feature learning. These 13 features are fed into the DFNN as input vectors to be trained through four hidden layers with 250, 250, 50 and 20 hidden neurons respectively to classify the four labels of daily activities.

4.2.3.3 1D-CNN

In time-series classification with feature learning, 1D-CNN provides a high classification rate with lesser computational complexity compared to other topologies [102]. 1D-CNN uses one-dimensional convolution of the input signal via kernels to extract specific characteristics from the raw signals [26]. Recent research on one-dimensional temporal data has proved that a 1D-CNN is superior for 1D signals when compared to their more popular 2D counterparts [41]. The architecture of a 1D-CNN combines the two major tasks, namely feature mapping

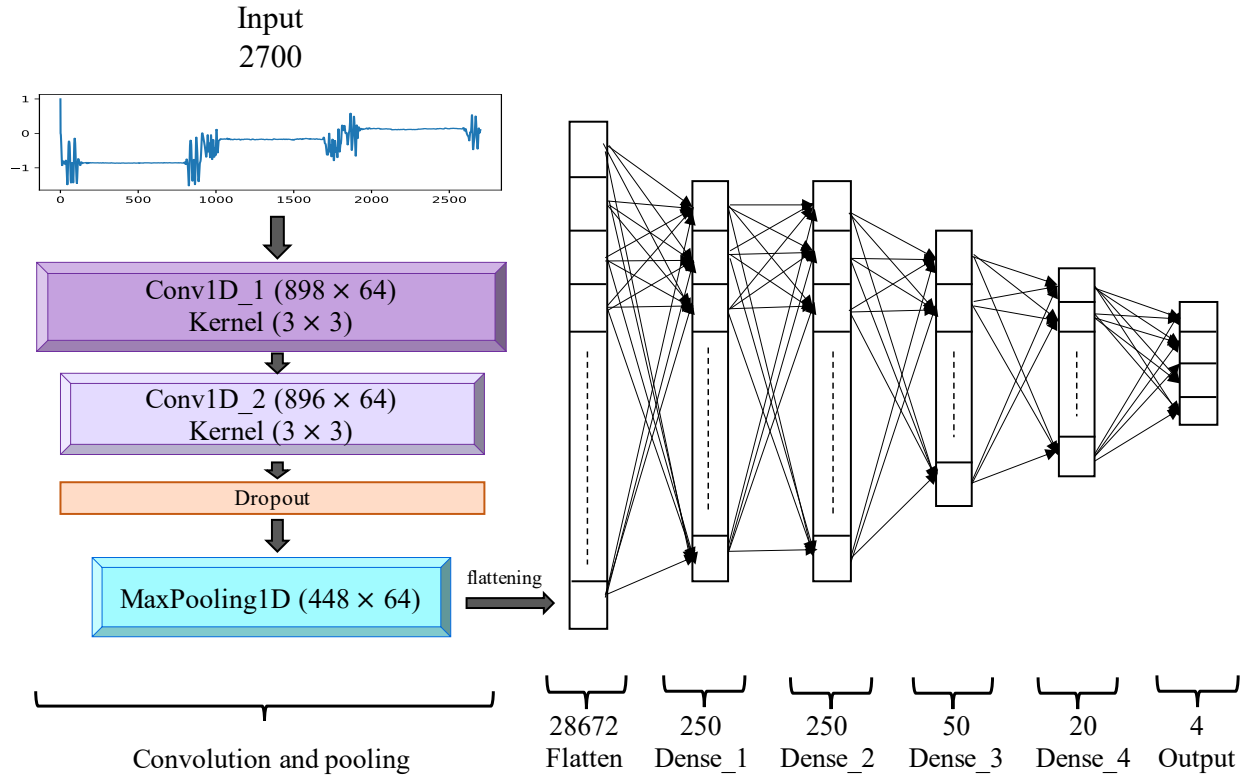


Figure 4.6: Feature learning using a one-dimensional convolutional neural network

and prediction of a 1D signal into a single learning system. The input layer receives the raw 1D signal which is forwarded to the convolutional layers. The convolution layers are used for feature mapping from the input, where every convolution layer consists of multiple kernels of the same size, followed by the pooling layer. Pooling performs average or max pooling operations to reduce the size of the input space to help speed up the training process and then send the output to the fully connected layers for conventional input-output mapping [26, 34]. Mathematically, if l is a convolution layer:

$$x_j^l = f\left(\sum_{i=1}^M x_i^{l-1} \otimes k_{ij}^l + b_{ij}^l\right) \quad (4.4)$$

where k represents the convolution kernels, j denotes the number of kernels, M represents the channel number of the input x^{l-1} , b is the bias corresponding to the kernel, $f()$ is the activation function and \otimes is the convolution operator [41].

Figure 4.6 shows the proposed deep 1D-CNN architecture for the adult-ankle dataset with 10080 observations in 2700 dimensions to be fed into the model as input. Two 1D-convolutional layers with 64 filters of kernel size (3, 3) perform the forward-backward operation by adjusting the layer-weights and extract the features to provide a high classification rate with lesser complexity. Here, a ‘Dropout’ layer with probability = 0.5 has also been added to prevent the overfitting of the training process. After the convolution layer, a pooling layer (of size 2) performs max pooling operation which is used to process each feature map to reduce the data dimensionality to 448 by selecting the maximum parameters within the range of the predetermined window as the output value. In this experiment, our model consists of 4 fully connected layers 250, 250, 50, 20 neurons respectively to map the output from the convolutional layers into the predicted class labels.

4.2.3.4 2D-CNN

A two-dimensional convolutional neural network automatically learns rich feature information from a 2D input representation to provide high classification accuracy. In a time-series classification task, the one-dimensional data naturally needs to be transformed into a 2D representation before utilizing a 2D-CNN. In this study, we generate spectrogram images from the accelerometer data which models both time and frequency fluctuations in the signals.

In a 2D-CNN, the input data is a matrix or tensor with a 3D spatial structure, where (H, W) , (H', W') , and (H'', W'') represent the size of the spatial dimension of input data, convolution kernel, and output data, respectively. The number of convolution kernel feature channels is represented by D , and D'' represents the 3D data.

$$\begin{aligned}
x &\in \mathbb{R}^{H \times W \times D} \\
f &\in \mathbb{R}^{H' \times W' \times D \times D''} \\
y &\in \mathbb{R}^{H'' \times W'' \times D''}
\end{aligned} \tag{4.5}$$

where x is the input data, f is the convolution filter, and y is the output data. The 1D signal x is convoluted by filter f to calculate signal y as follows:

$$y_{i'',j'',d''} = b_{d''} + \sum_{i'=1}^{H'} \sum_{j'=1}^{W'} \sum_{d'=1}^D f_{i'',j'',d''} \times x_{i''+i'-1,j''+j'-1,d',d''} \tag{4.6}$$

Equation (4.6) states that $b_{d''}$ is the neuron offset and $f_{i'',j'',d''}$ is the convolution kernel matrix of the d^{th} $i' \times j'$ (i'^{th} neuron and j'^{th} connection).

The pooling layer performs max pooling operation which calculates the maximum response of each feature channel in the $H' \times W'$ region [29].

$$y_{i'',j'',d''} = \max_{1 \leq i' \leq H', 1 \leq j' \leq W'} x_{i''+i'-1,j''+j'-1,d} \tag{4.7}$$

In this chapter, 1D data is first transformed into spectrogram images of dimension (13×13) and applied as input to the convolution layers. Figure 4.7 displays that one-dimensional temporal data is transformed into a two-dimensional frequency domain representation considering a 'kaiser' window with vector size as (128,18) which divides the input into segments of equal size and activates the window operation. 64 overlapping samples are used between the adjoining segments to compute a 64-point discrete Fourier Transform where the sampling rate is set at 90Hz. Figure 4.8 shows the spectrograms of representative samples from the adult-ankle, adult-wrist, youth-ankle and youth-wrist datasets respectively.

Figure 4.9 shows a framework of the specific 2D-CNN model using youth-ankle dataset where two 2D convolution layers with 'LeakyReLU' activation functions perform the convolution operation with 64 filters of kernel size as (5, 5) to extract the feature map. Here, two

max pooling layers with pool size (2, 2) and stride = 1 compute the maximum information from the feature map and send this to a fully connected layer of 500 neurons. The output layer with the 'Softmax' activation function predicts the class labels for the four daily activities.

4.2.3.5 RNN-LSTM

RNN-LSTM network has the repeating-module or chain like architecture where each module has four interacting neural network layers. In LSTM network, the information flows through the network's memory cell by the specialized control units which are called gate units such as forget, input and output units [10]. Figure 4.10 displays the RNN-LSTM model where four LSTM hidden layers are considered with 250, 250, 50 and 20 hidden neurons to build the recurrent network on youth-ankle dataset. At first, we reshape the youth-ankle input of dimension (6098, 2700) to represent as (6098, 900, 3) which is then forwarded to the LSTM layers. In LSTM layer, every LSTM cell is connected back to the previous time step to form the recurrent connection. Here, the information flows through the LSTM layers and the new information is stored in the memory unit of each LSTM cell. Then in the output layer, the output activation of LSTM units process the stored information to determine the new states h_T .

This model is trained for certain iterations and tested to classify the four activity-labels. However, the major drawback of RNN-LSTM network is the expensive computational time to train the model due to its recurrent structure and for this reason this classifier does not perform well for our Adult-Youth dataset with 2700 features.

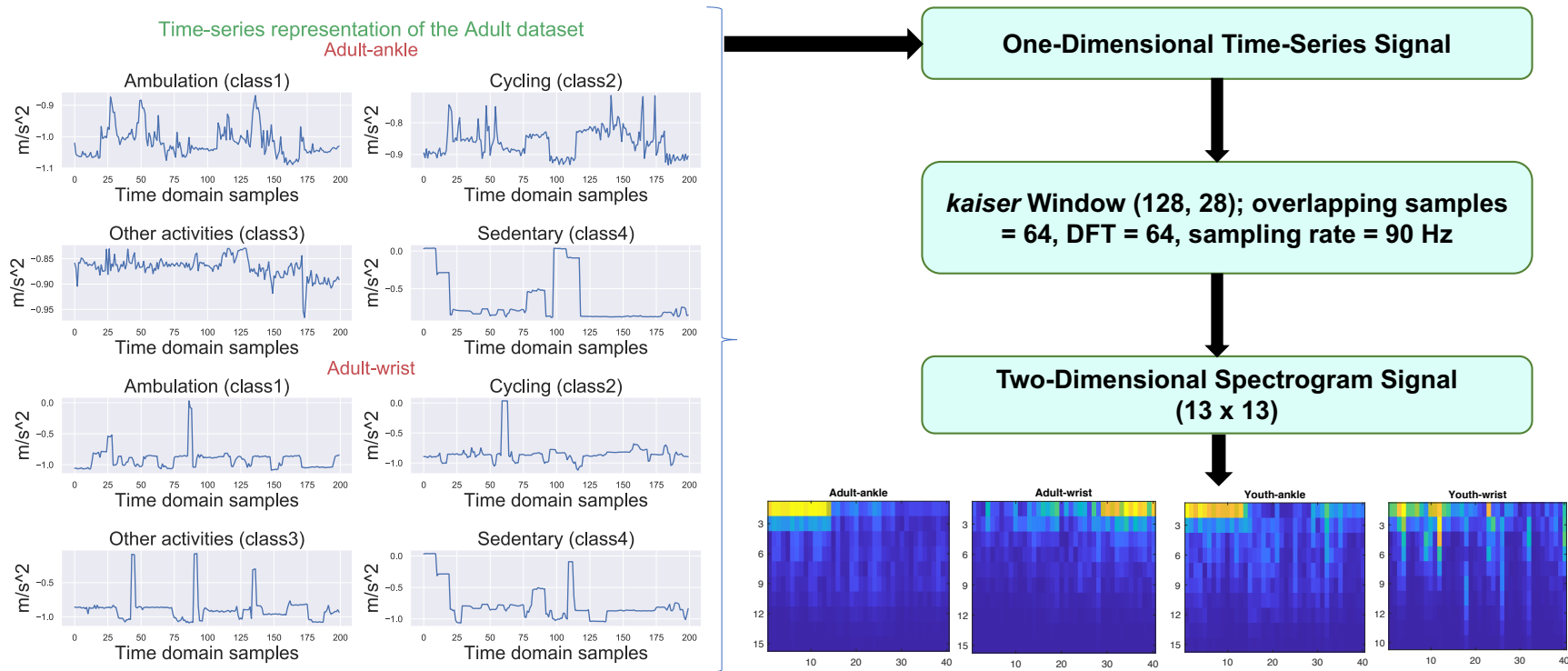


Figure 4.7: Transformation of 1D time domain signal to 2D frequency domain representation

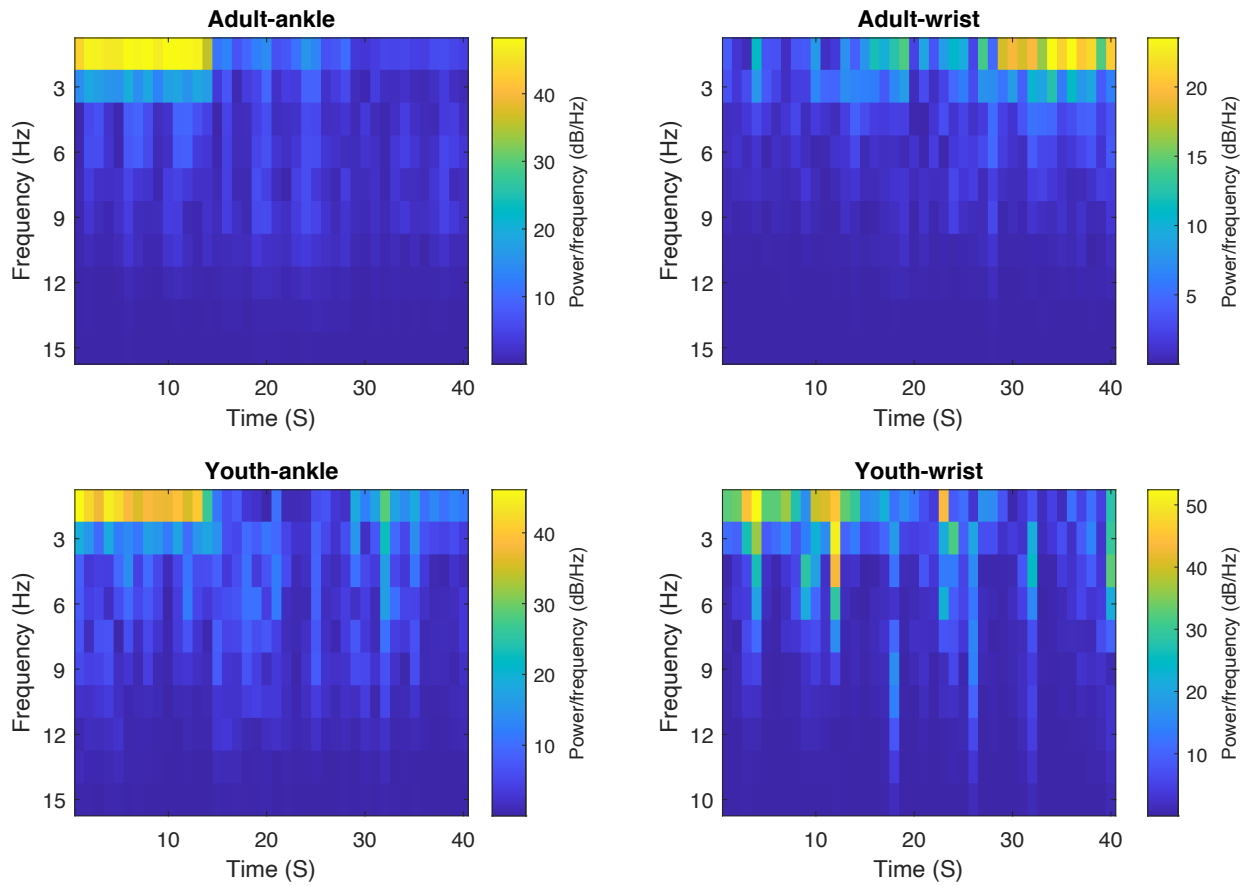


Figure 4.8: Frequency domain representation of sample observations from the four datasets

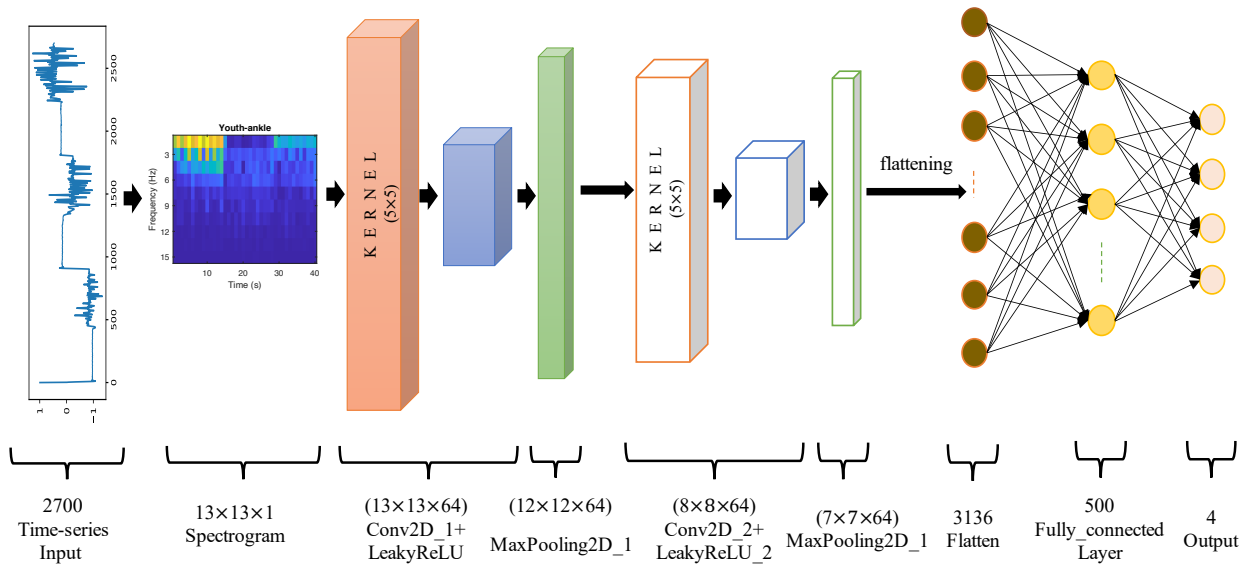


Figure 4.9: Feature learning using a two-dimensional convolutional neural network

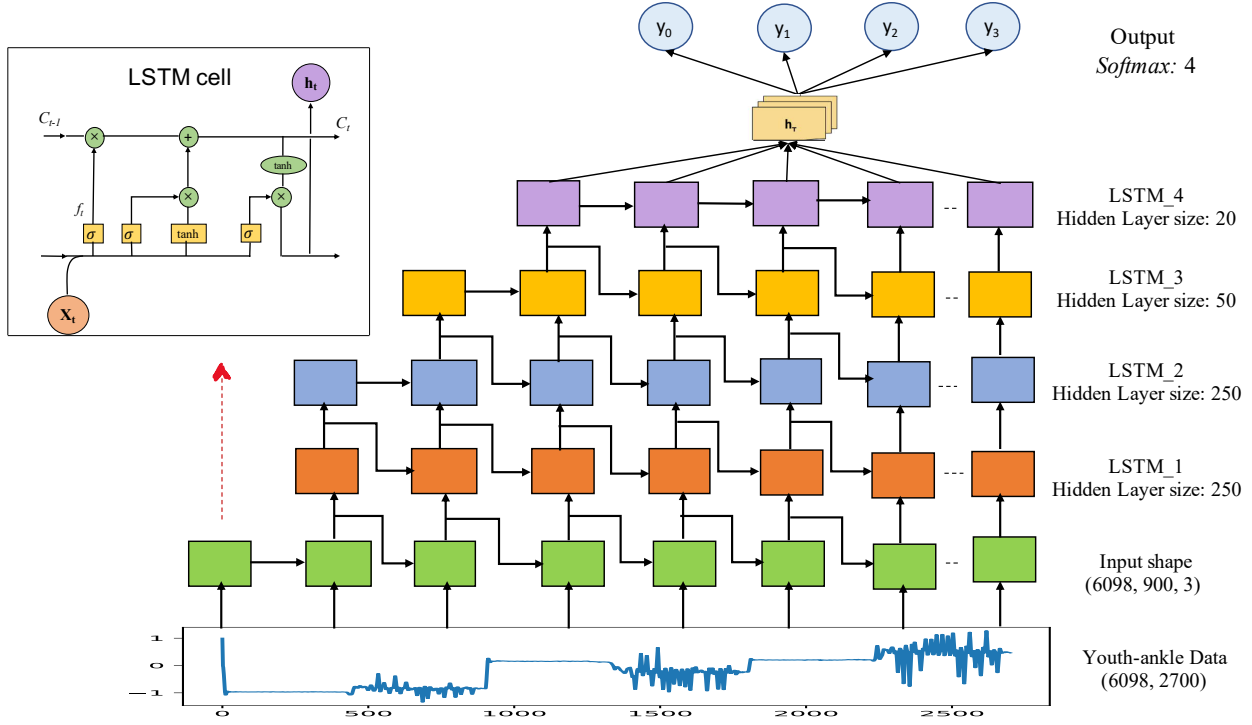


Figure 4.10: Schematics of RNN-LSTM network used in this experiment

4.2.3.6 Data Augmentation: SMOTE

For three out of the four datasets the 2D-CNN provides better classification accuracies in the raw data-space than the feature space. On the contrary, the feature learner does not perform as well on the youth-ankle dataset. This could be due to a number of reasons including the entropy of the dataset and the number of observations available to train the large number of trainable parameters. In these instances, synthetic data augmentation techniques such as SMOTE can substantially improve performance as evident from modern image classification applications [78]. SMOTE oversamples the minority class by taking each class sample and generating synthetic samples along the line segments joining its k nearest neighbors. Depending upon the desired level of over-sampling, neighbors are randomly selected from the k nearest neighbors as follows:

$$S' = S + \text{rand}(0, 1) * |S - S_k| \quad (4.8)$$

Table 4.3: The topologies for the different classifiers using feature learning for human activity recognition

Classifier	HNs (HL1)	HNs (HL2)	HNs (HL3)	HNs (HL4)
DFNN	250	250	50	–
AE-DFNN	250	250	50	20
1D-CNN	250	250	50	20
2D-CNN	500	–	–	–
RNN-LSTM	250	250	50	20
FE-DFNN	250	250	50	–

where S' denotes the new set of synthetic samples, S is the set of original samples for which k-nearest neighbors are being identified and S_k is the set of randomly selected k-nearest neighbor samples. To generate new samples, this process is repeated N number of times where N is the oversampling rate commonly provided as a percentage [19]. In this chapter, we look at four different levels of oversampling and demonstrate diminishing returns as the synthetic samples begin to contribute less and less meaningful information to the training process.

4.3 Experimental Setup

4.3.1 Leave-One-Subject-Out Cross-validation

Here, a wide array of topologies is tested using grid hyperparameter search, and the best topology has been chosen for each classifier to train the raw data using leave-one-subject-out (LOSO) cross-validation with a random ‘shuffle’ to estimate an unbiased and accurate model performance for all experiments. For two adult datasets we consider LOSO with k-fold = 33 and for the remaining two youth datasets, k-fold = 20 is used to implement cross-validation during the training process. Table 4.3 summarizes the best topologies used to train the six classifiers and shows the number of hidden neurons (HNs) for each hidden layer (HL) such as HL1, HL2 etc.

4.3.2 Model Hyperparameters

For the DFNN model, two dropout layers of probability 0.4 have been added after the first two hidden layers and another dropout layer (probability = 0.1) is added after third hidden layer to prevent overfitting of the network. In addition, a stochastic gradient descent optimizer with learning rate $lr = 0.0001$ is used to make the model learn and optimize faster. For the nonlinear model, categorical cross-entropy loss or softmax loss function has been considered as a natural cost function which maximizes the likelihood of predicting the output classes correctly.

The autoencoder model consists of two encoders with 500 and 13 neurons and two decoders with 500 and 2700 neurons. Using automatic feature learning technique, the model is trained for 200 epochs with ‘Adam’ optimizer and the encoders learn 13 unsupervised features from the original 2700 features which is then applied as input to the DFNN model. The hypermeters used to train the DFNN are as mentioned above.

In the case of 1D-CNN model, the two convolutional layers along with a dropout layer (probability = 0.5) reduce the dimension of the input time-series and extract the unsupervised features by adding a pooling layer of size 2. After the pooling operation, the output is flattened and passed through a group of fully connected layers using ‘Adam’ optimizer and categorical cross-entropy loss for training. For the 2D-CNN, we use the same hyperparameters mentioned above except the kernel size which is (5, 5) and only a single fully connected layer with 500 hidden neurons.

To design RNN-LSTM model, we consider the best topology which consists of four LSTM hidden layers with 250, 250, 50 and 20 hidden neurons. The LSTM layers are optimized using ‘Adam’ optimizer and ‘l2’ regularizer with probability 0.01 which is used in each layer to prevent overfitting.

4.3.3 Feature Engineering

In this dissertation, we aim to provide a robust performance comparison of the deep learning classification algorithms with feature engineering approaches. To accomplish this, we provide a one-to-one comparison of the deep learning models with the feature engineering results of both [64] and [48]. To do this, thirteen domain-specific motion sensitive features are extracted from raw temporal datasets including the means and variances of x, y and z axes of the accelerometer data, pitch, roll, yaw, zero-crossing rate of the mean (on x, y and z) and the magnitude [48].

4.4 Results and Discussion

As previously discussed, in order to conduct a fair comparison with the original paper from the Stanford group we apply the same LOSO cross-validation as we train & test six different classification algorithms on raw data and features separately for adult and youth datasets where the feature engineering is represented by the prior results reported in [64]. Moreover, we perform the classification experiments on these four datasets using other and more recent state-of-the-art features published in 2021 [48] to provide a robust comparison between deep learning and feature engineering. For DFNN, 1D-CNN, 2D-CNN and RNN-LSTM, the resampled and filtered data is represented at the input layer as-is. The AE-DFNN on the other hand is first trained to learn the unsupervised features from the latent space of raw data where the extracted latent features are subsequently used to train another dense FNN.

Table 4.4 represents the average classification accuracies with standard deviations across the four datasets for all models with LOSO cross-validation. We observe that even though the best result in each dataset is observed for the 2D-CNN feature learning deep neural network, the significance of proper learning representation is highlighted by the fact that human engineered features of [64] can actually outperform deep learning when the topology is not carefully selected (i.e., both for regular feedforward neural network and for the autoencoder

Table 4.4: Average accuracies for the different classifiers using feature learning versus the subject agnostic feature engineering used in the latest work on this dataset (Mannini *et al.* [64])

Classifier	Average classification accuracy (%)			
	Adult-ankle	Adult-wrist	Youth-ankle	Youth-wrist
DFNN	91.17 ± 3.47	87.43 ± 7.87	85.86 ± 10.43	88.30 ± 8.3
AE-DFNN	88.33 ± 4.81	81.31 ± 7.74	82.68 ± 10.78	80.81 ± 8.04
1D-CNN	95.21 ± 2.38	88.02 ± 6.29	88.49 ± 8.98	90.54 ± 4.98
2D-CNN	95.57 ± 2.54	93.45 ± 3.55	93.38 ± 2.67	93.13 ± 3.5
Mannini <i>et al.</i> [64]	94.8	87	92.4	91
FE-DFNN [48]	92 ± 4.38	83.78 ± 8.45	85.79 ± 9.82	85.86 ± 8.47

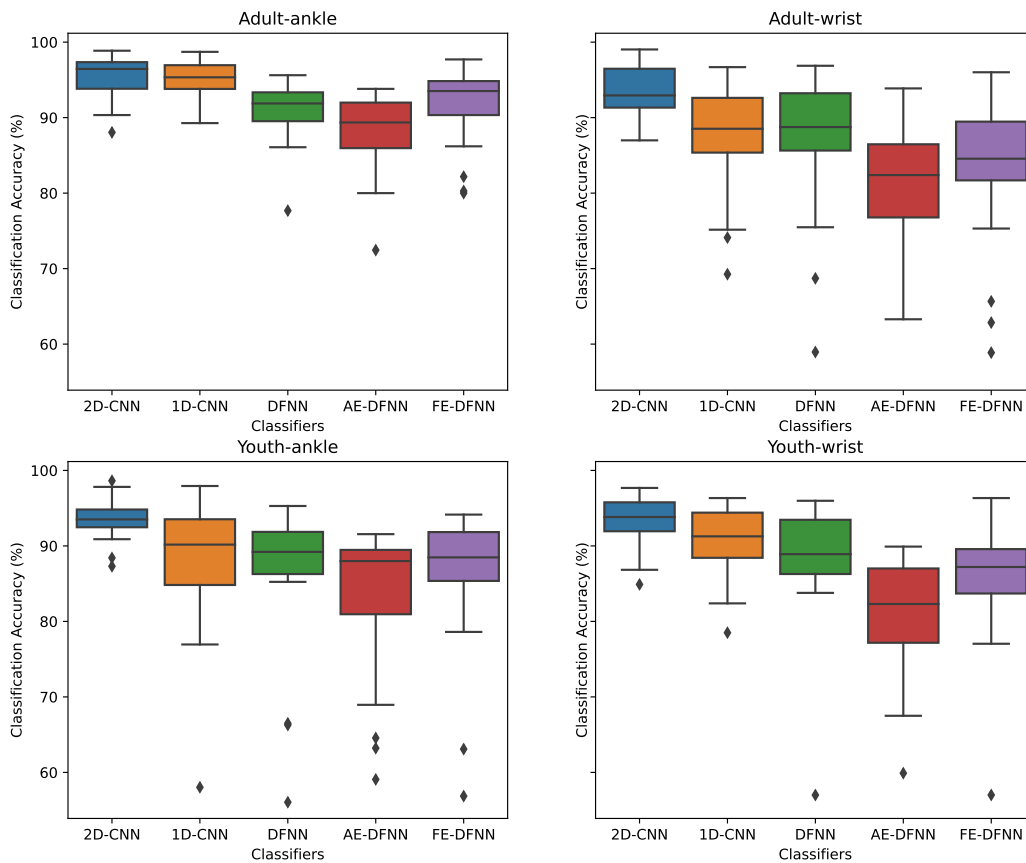


Figure 4.11: Boxplot distributions of classification accuracies across different classifiers on all 4 datasets

feature learner). Ultimately, results indicate state-of-the-art performance using a 2D-CNN model on raw data which outperforms the most recent results reported on the dataset using feature-based models. The table also provides a one-to-one comparison of the best accuracies reported in this study with the most recent results published in the literature [64] on this dataset as well as the performances of FE-DFNN models using more recent state-of-the-art features for motion data (not particularly tuned for this dataset) reported in [48]. Our findings from these experiments reveal that the subject agnostic features in [64] perform much better compared to the FE-DFNN classifier. However, the performances of convolutional feature learners, especially with spectrotemporal features, outperform the feature engineering classifiers. Figure 4.11 shows the boxplots for the accuracies of five classification algorithms including feature engineering on four datasets for a better statistical comparison. It is important to note that these results are all statistically significantly relevant as discussed further in the next subsection.

Another important observation is the requirement to support the training of the more complex 2D-CNN algorithm via a synthetic data augmentation method called SMOTE at different levels of oversampling. Table 4.5 displays the average classification accuracies with standard deviations of the 2D-CNN classifier on the youth-ankle dataset with and without data augmentation. As one can see, without any augmentation, the accuracy of the 2D-CNN classifier is actually slightly worse than the subject agnostic features reported in [64], 90.74% versus 92.4%. It is only through augmentation that the 2D-CNN model manages to outperform the current state-of-the-art reported in the literature. Figure 4.12 clearly shows the significant increases in performance as more synthetic samples are included in the training process. The diminishing returns are also apparent as when more samples are included in the training process, the classification accuracy no longer displays the same level of improvement.

In the recently published article [64], the highest accuracies on domain-specific features reported for adult-ankle, adult-wrist, youth-ankle and youth-wrist datasets are 94.8%, 87%,

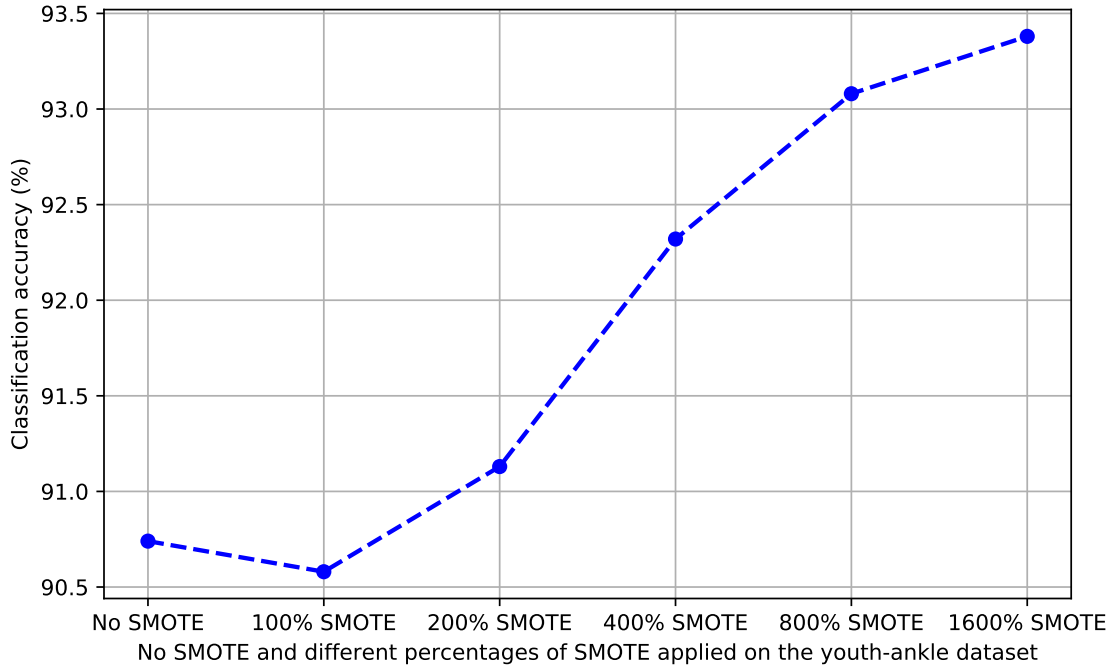


Figure 4.12: Average classification accuracies of 2D-CNN classifier with no SMOTE and different percentages of SMOTE on the youth-ankle dataset

Table 4.5: Average classification accuracies with and without data augmentation on the youth-ankle dataset

Average classification accuracy (%)					
Without SMOTE	100% SMOTE	200% SMOTE	400% SMOTE	800% SMOTE	1600% SMOTE
90.74 ± 3.57	90.58 ± 3.26	91.13 ± 3.1	92.32 ± 2.3	93.08 ± 2.51	93.38 ± 2.67

92.4% and 91% respectively using domain specific features with a support vector machine classifier. The highest classification accuracies mentioned in the chapter are achieved using 2D-CNNs on raw temporal data on the same datasets with 95.57%, 93.57%, 93.38% and 93.13% respectively. These results suggest that raw data can compete and often outperform feature engineering and that the latent features extracted by a 2D-CNN on rich spectrogram images perform better than both the human engineered features and the fully unsupervised feature learning using an autoencoder. Although it is difficult to make definitive conclusions on empirical observations, our previously published work as well as the new findings in this

research clearly demonstrate at least the competitiveness of feature learning especially when coupled with data rich input representations such as spectrograms which can be obtained from temporal activity recognition signals.

4.4.1 Statistical Analysis of the Performances of Classifiers

To compare the statistical significance of the classification results, we perform a statistical significance test between each pair of classifiers [9]. Since the outcomes of the goodness of fit test and the variance test using subject based classification accuracies of deep learning classifiers do not follow the conventional assumptions on the data distribution (such as Gaussian (normal) distribution) and the data samples cannot necessarily be assumed as independent, we have to follow a non-parametric approach for statistical testing [17]. More specifically, as the subject-based classification accuracies of two classifiers come from the same population and the data is paired, we perform a non-parametric Wilcoxon signed-rank test, a popular statistical test for paired data which does not come from a normal distribution, to investigate the null hypothesis [14]. The data distributions of all the proposed models (with or without feature engineering) are dependent as the features are still drawn from the raw data of the same subjects. Hence, we employ the same Wilcoxon signed-rank test for hypothesis test between the deep learning models versus the feature-based model. Table 4.6 and 4.7 show that FE-DFNN model performs statistically significantly worse than the raw models where the null hypothesis is rejected ($h=1$) for every dual comparison on three out of four datasets except on the youth-ankle dataset, where all algorithms have statistically the same performance except for 2D-CNN which is statistically significantly better than all the other models. Furthermore, the 2D-CNN model which utilizes spectrotemporal features automatically learned from the raw data, statistically significantly outperforms every other model in every single scenario except one. On the adult-ankle dataset, both convolutional models display the same performance, though still significantly better than the competing approaches. However, for this specific case, it is still important to note that even a small

performance difference (in this case 0.36%) when properly cross-validated across the subject-specific dataset still holds value for the machine learning community at large. Finally, we find that the test fails to reject the null hypothesis ($h=0$) for 1D-CNN vs. DFNN on adult-wrist, youth-ankle and youth-wrist datasets which further demonstrates that spectral properties of the signal should be taken into account for classification purposes.

4.4.2 Computational Complexity

Table 4.8 shows the computational complexities and computational run times of the five classifiers. The computer used in this study is configured with the following specifications: Intel(R) core (TM) i7 4.20GHz CPU, NVIDIA GeForce GTX 1080 GPU and 32 gigabyte memory (RAM). From the table, the feature learning algorithms (AE-DFNN and 2D-CNN) display fast convergence in training. However, comparing the high classification rates and the low computational time for testing among all the classifiers, 2D-CNN provides the highest performance with the lowest computational times. Since our previous research [48] on time-series data provided the highest accuracies for RNN-LSTM classifier, we investigate the classification rate of RNN-LSTM classifier for this dataset as well. Table 4.9 displays the RNN-LSTM classification-accuracies including computational time for three subjects randomly selected from each dataset and a comparison between the classification accuracies of 2D-CNN and RNN-LSTM algorithms for those subjects.

In table 4.9, the most interesting observation is that RNN-LSTM has significantly higher training time because of its recurrent feedback loop and for this reason the classification using this algorithm for LOSO cross-validation on this dataset is computationally expensive where it takes approximately 2-days to complete training only for a single subject partition out of the 53 subjects in the adult and youth datasets. More importantly however, it does not provide a better classification rate compared to 2D-CNN, so it was not considered for the full experimental analysis in this study.

Table 4.6: Statistical significance analysis between the performances of the classifiers on Adult dataset

Classification Algorithms	Null Hypothesis (h)	p-value (p)
Adult-ankle Dataset		
2D-CNN vs. 1D-CNN	0	0.272
2D-CNN vs. DFNN	1	3.54E-06
2D-CNN vs. AE-DFNN	1	7.11E-07
2D-CNN vs. FE-DFNN	1	3.53E-05
1D-CNN vs. DFNN	1	5.39E-07
1D-CNN vs. AE-DFNN	1	5.91E-07
1D-CNN vs. FE-DFNN	1	4.99E-06
DFNN vs. AE-DFNN	1	0.001
DFNN vs. FE-DFNN	1	0.009
AE-DFNN vs. FE-DFNN	1	9.45E-05
Adult-wrist Dataset		
2D-CNN vs. 1D-CNN	1	2.98E-06
2D-CNN vs. DFNN	1	4.58E-06
2D-CNN vs. AE-DFNN	1	1.02E-06
2D-CNN vs. FE-DFNN	1	5.39E-07
1D-CNN vs. DFNN	0	0.598
1D-CNN vs. AE-DFNN	1	4.45E-05
1D-CNN vs. FE-DFNN	1	1.69E-04
DFNN vs. AE-DFNN	1	0.0001
DFNN vs. FE-DFNN	1	1.47E-05
AE-DFNN vs. FE-DFNN	1	0.023

Table 4.7: Statistical significance analysis between the performances of the classifiers on Youth dataset

Classification Algorithms	Null Hypothesis (h)	p-value (p)
Youth-ankle Dataset		
2D-CNN vs. 1D-CNN	1	0.001
2D-CNN vs. DFNN	1	0.0001
2D-CNN vs. AE-DFNN	1	8.86E-05
2D-CNN vs. FE-DFNN	1	2.54E-04
1D-CNN vs. DFNN	0	0.03
1D-CNN vs. AE-DFNN	1	0.001
1D-CNN vs. FE-DFNN	0	0.145
DFNN vs. AE-DFNN	0	0.025
DFNN vs. FE-DFNN	0	0.737
AE-DFNN vs. FE-DFNN	0	0.247
Youth-wrist Dataset		
2D-CNN vs. 1D-CNN	1	0.01
2D-CNN vs. DFNN	1	0.0003
2D-CNN vs. AE-DFNN	1	8.86E-05
2D-CNN vs. FE-DFNN	1	2.19E-04
1D-CNN vs. DFNN	0	0.021
1D-CNN vs. AE-DFNN	1	8.86E-05
1D-CNN vs. FE-DFNN	1	2.54E-04
DFNN vs. AE-DFNN	1	0.0002
DFNN vs. FE-DFNN	1	0.03
AE-DFNN vs. FE-DFNN	1	5.93E-04

Table 4.8: Computational complexities of five classifiers for a single subject on four datasets

Classifier	Accuracy (%)	Epochs	Training Time (Sec)	Test Time (Sec)	Computational Time (Sec)	Computational Time/epoch (Sec)	Training Parameters
Adult-ankle Dataset, Subject - 14							
DFNN	95.15	200	159	0.11	159.11	0.8	750754
AE-DFNN	90.61	1700	1528.95	1.607	1530.56	0.9	2796617
1D-CNN	97.58	1000	15589.27	0.25	15589.53	15.59	7257646
2D-CNN	98.49	500	368.2	0.18	368.38	0.74	1674632
RNN-LSTM	99.7	100	246182.98	13.77	246196.74	2461.97	815404
Adult-wrist Dataset, Subject - 7							
DFNN	82.25	1500	1132.2	0.11	1132.32	0.76	750754
AE-DFNN	53.84	1700	1018.30	1.55	1019.85	0.6	2796617
1D-CNN	82.61	1000	15099.22	0.24	15099.46	15.1	7257646
2D-CNN	85.87	800	577.52	0.204	577.72	0.72	1674632
RNN-LSTM	64.86	150	386677.11	10.42	386687.54	2577.92	815404
Youth-ankle Dataset, Subject - 3							
DFNN	85.08	1200	575.75	0.11	575.86	0.48	750754
AE-DFNN	76.82	1700	637.42	0.86	638.28	0.38	2796617
1D-CNN	81.5	1000	9566.86	0.25	9567.12	9.57	7257646
2D-CNN	87.16	800	371.64	0.21	371.84	0.47	1674632
RNN-LSTM	92.24	100	156094.16	13.43	156107.59	1561.08	815404
Youth-wrist Dataset, Subject - 4							
DFNN	93.4	1500	699.62	0.1	699.72	0.47	750754
AE-DFNN	78.72	1700	623.009	0.859	623.868	0.367	2796617
1D-CNN	90.81	1500	14197.6	0.13	14197.73	9.47	7257646
2D-CNN	94.25	630	280.7	0.21	280.91	0.45	1674632
RNN-LSTM	89.99	100	137337.16	11.17	137348.33	1373.48	815404

Table 4.9: Classification accuracies and computational time of RNN-LSTM classifier based on three random subjects of four datasets

Subject	2D-CNN Accuracy (%)	RNN-LSTM Accuracy (%)	Epochs	Training Time (Sec)	Test Time (Sec)	Computational Time (Sec)
Adult-ankle Dataset						
1	95.52	92.41	53	123101.61	13.88	123115.49
3	92.86	90	50	123071.49	12.98	123084.47
14	98.86	99.7	100	246182.98	13.77	246196.74
Adult-wrist Dataset						
1	93.96	93.46	100	259166.59	12.77	259179.36
7	89.86	82.68	135	348009.39	10.42	348019.81
10	98.54	93.21	100	289893.70	13.64	289907.34
Youth-ankle Dataset						
1	92.76	88.69	100	150019.68	12.1	156106.26
2	94.41	93.01	100	158000.1	13.87	179271.62
3	91.54	92.24	100	156094.16	13.43	156107.59
Youth-wrist Dataset						
1	84.91	84.71	150	214672.83	11.467	214684.3
3	91.05	66.57	100	137337.16	11.17	137348.33
7	97.68	95.07	150	216252.6	13.11	216265.71

4.4.3 Subject-based Classification Accuracies on Four Datasets

For the interested readers, a detailed breakdown of subject specific classification accuracies across the three models on all four datasets are presented in tables 4.10, 4.11, 4.12 and 4.13 respectively.

4.4.4 Deep Learning vs. Feature Engineering on Reduced Size Data

According to the results regarding the superiority of deep learning, we substantiate an argument that deep learning could be valid in some instances, but not all. Moreover, to make the argument stronger, we explore what real world situations would warrant choosing human engineered features over feature learning. Specifically, we look into whether feature engineering can perform better in a scenario where data collection is limited (which is true in most medical applications of motion recognition). To test this hypothesis, we artificially reduce the sizes of all four datasets by randomly selecting 100 samples from each subject

Table 4.10: Subject-based classification accuracies for four classifiers on adult-ankle dataset

Subject	DFNN	AE-DFNN	1D-CNN	2D-CNN	FE-DFNN
1	89.65	82.67	94.14	95.52	86.2
2	95.58	91.99	96.21	91.17	94.65
3	89.53	93.08	93.06	92.86	90.31
4	91.9	89	94.17	98.38	95.47
5	94.18	91.56	96	97.45	94.9
6	93.9	85.21	94.31	93.09	95.52
7	88.5	80	90.22	88.04	89.85
8	92.03	92.33	94.4	96.17	91.15
9	92.06	92.75	95.47	97.03	94.9
10	93.19	88.88	93.49	96.45	92.6
11	91.06	91.59	94.52	95.1	91.35
12	91.22	87.64	96.95	96.95	91.98
13	88.99	91.05	92.05	93.58	89.6
14	93.93	89.36	98.18	98.86	94.84
15	95.58	93.79	98.24	96.47	93.52
16	89.88	85.96	97.92	98.81	91.96
17	86.08	72.45	91.88	93.65	80.28
18	92.87	90.97	97.52	98.14	96.9
19	90.12	90	93.42	93.83	93.82
20	87.22	86.75	93.15	90.34	90.34
21	94.78	93.81	98.7	97.08	97.71
22	93.92	85.46	96.66	96.66	94.83
23	90.88	81.5	95.88	93.65	93.52
24	91.69	92.84	97.13	97.71	91.4
25	89.43	88	94.85	97.35	94.3
26	87.32	86.78	93.8	94.94	91.26
27	89.13	88.2	95.34	97.93	89.75
28	77.68	80.93	89.28	93.94	80
29	93.35	93	98.01	97.67	96.34
30	91.02	90.01	94.87	94.87	93.58
31	92.28	93.12	98.71	97.11	94.21
32	92.72	84.51	96.73	96.01	96.36
33	95.62	89.71	96.56	96.88	82.18
Average accuracy (%)	91.17	88.33	95.21	95.57	92

Table 4.11: Subject-based classification accuracies for four classifiers on adult-wrist dataset

Subject	DFNN	AE-DFNN	1D-CNN	2D-CNN	FE-DFNN
1	85.1	75.01	86.60	93.96	83.98
2	93.23	80.13	87.38	92.13	88.45
3	95.34	73.33	90.00	98.46	92.39
4	94.49	81.2	90.61	99.03	89.47
5	87.27	74.33	82.25	88.04	81.69
6	58.95	63.29	69.26	89.11	62.85
7	86.45	83.6	84.06	89.86	82.34
8	93.09	82.39	94.13	97.36	86.72
9	89.65	76.33	85.59	96.48	81.69
10	91.52	88.44	90.67	98.54	93.47
11	80.61	69.84	86.17	92.22	75.33
12	90.17	63.48	93.96	95.09	83.12
13	80.59	76.78	83.69	92.05	75.31
14	94.87	83.14	91.81	99.02	88.61
15	89.77	86.79	94.05	97.02	91.39
16	85.64	84.63	75.16	86.99	80.27
17	87.35	85.61	89.91	95.10	84.17
18	93.42	83.49	95.98	98.04	89.56
19	89	74.55	84.40	88.40	84.56
20	88.75	86.47	86.08	95.15	86.77
21	94.88	92.83	95.14	91.41	88.36
22	83.45	77.69	91.67	92.86	80.07
23	85.55	78.89	90.29	92.94	81.02
24	88.03	80.52	85.01	89.34	82.91
25	86	88.53	85.37	91.33	82.02
26	89.61	89.56	92.68	92.55	89.11
27	68.71	77.2	74.11	87.73	58.87
28	75.48	73.39	88.45	94.22	65.66
29	93.67	92.39	92.62	96.31	92
30	94.91	83.99	94.97	93.62	90.12
31	96.87	93.48	96.69	96.69	96.01
32	86.59	84.11	87.27	92.73	86.45
33	86.12	93.87	88.52	90.16	89.88
Average accuracy (%)	87.43	81.31	88.02	93.45	83.78

Table 4.12: Subject-based classification accuracies for four classifiers, with and without SMOTE on youth-ankle dataset

Subject	DFNN	AE-DFNN	1D-CNN	2D-CNN (Without SMOTE)	1600% SMOTE+ 2D-CNN	FE-DFNN
1	86.36	84.87	88.99	92.76	93.13	78.61
2	89.58	90.45	92.19	94.41	97.82	90.99
3	93	88.84	84.18	91.54	93.83	86.26
4	89.34	90.21	93.39	94.83	95.07	85.83
5	88.3	88.00	90.35	85.67	90.89	94.15
6	91.66	89.66	89.87	85.33	92.52	92
7	89.08	87.08	95.94	89.26	95.62	93.62
8	56.05	59.07	58.03	85.56	87.30	63.09
9	85.24	64.56	84.94	87.45	92.34	83.97
10	92.46	87.7	92.28	92.67	93	91.96
11	93.75	87.99	95.61	95.45	93.98	87.31
12	95.29	91.56	97.94	96.10	98.63	88.65
13	86.97	89.24	89.66	87.69	92.7	89.27
14	90.14	85.11	97.43	93.42	95.26	94.15
15	87.89	68.95	84.50	91.01	94.02	90.35
16	89.69	89.41	93.96	92.07	94.74	88.21
17	86.02	88.67	91.79	91.83	91.35	91.79
18	66.28	69.2	76.95	93.74	93.57	88.31
19	93.57	90.33	90.00	86.44	93.45	80.43
20	66.5	63.21	81.71	87.55	88.41	56.86
Average accuracy (%)	85.86	82.68	88.49	90.74	93.38	85.79

Table 4.13: Subject-based classification accuracies for four classifiers on youth-wrist dataset

Subject	DFNN	AE-DFNN	1D-CNN	2D-CNN	FE-DFNN
1	83.78	75	82.39	84.91	77.04
2	88.6	77.68	90.12	96.71	89.22
3	57	59.9	78.51	91.04	57.01
4	94.63	79.87	95.11	93.68	86.95
5	92	89.83	93.57	92.69	96.34
6	95	85.04	92.80	95.70	88.53
7	93.33	89.91	94.10	97.68	90.65
8	88.75	82.15	90.96	92.24	85.5
9	86.37	69.84	88.14	88.14	83.51
10	84	81.59	83.60	86.84	78.45
11	86.01	87.55	91.18	95.26	89.21
12	93.85	88.49	94.87	92.82	94
13	85.4	78.41	89.27	88.96	86.97
14	90.73	87	95.04	96.37	91.57
15	88.06	67.5	84.69	93.46	81.23
16	90	86.36	88.52	94.52	86.4
17	87.86	84.98	91.36	95.10	87.45
18	89.07	82.46	94.25	93.98	87.64
19	95.98	87.06	96	96	95.71
20	95.34	75.65	96.34	96.58	83.76
Average accuracy (%)	88.3	80.81	90.54	93.13	85.86

which generates 3300 samples for the adult-ankle and adult-wrist datasets and 2000 samples for the youth-ankle and youth-wrist datasets. This corresponds to approximately a 10-fold reduction in sample size. We extract 13 domain specific features which we have used in our previously published feature engineering model (FE-DFNN) on a different dataset to examine raw data versus feature engineering on this reduced size dataset. The results of the experiments are presented in table 4.14 and 4.15.

From table 4.14 and 4.15, it is clear that the feature engineering now performs significantly better than feature learning on all four reduced size datasets. In other words, in a real-life scenario where data is limited, feature engineering still has the upper hand when it comes to proper inference from the data. Finally, we want to once again stress the point that as shown in table 4.4, the expert features which are hand-crafted by the Stanford group, actually did outperform deep learning in almost every scenario except the 2D-CNN model – once again highlighting how important it is to properly represent the data and sometimes augment it to support the training of large parameter sized and complex models (since 2D-CNN did perform worse without augmentation on youth-ankle dataset).

4.5 Summary and Conclusions

Our most significant contribution is the definitive performance analysis of feature engineering versus feature learning and the subsequent development of the framework which utilizes novel feature learning technique to outperform the state-of-the-art algorithm in recognizing human activity using cost-effective sensors placed on human body. In this chapter, we investigate six classification algorithms including raw and feature-based models and based on the results, we claim that automatic unsupervised feature learning outperforms feature engineering by introducing a novel learning representation of temporal data into a two dimensional input space. However, we want to stress the fact that feature engineering being outperformed by deep learning is never a forgone conclusion. In this context, we conduct an auxiliary experiment where we have created a likely real-life scenario where “access to robust

Table 4.14: Subject-based classification accuracies for DFNN and FE-DFNN classifiers on reduced size adult dataset

Subject	Adult-ankle Dataset		Adult-wrist Dataset	
	DFNN (Raw)	FE-DFNN (Feature)	DFNN (Raw)	FE-DFNN (Feature)
1	86.	89.61	79.52	83.98
2	94	96.77	85.71	88.45
3	87.53	92.	81.34	92.39
4	90	96	88.93	89.47
5	92.18	94.31	84.73	81.69
6	91.89	96.93	65.95	65.41
7	86.05	90	78.52	81.28
8	91.	93	90.34	93.47
9	91.06	96.91	83.59	89.63
10	90.19	94.84	88.62	92.65
11	89.86	94.08	75.61	80
12	91.1	93.66	80.05	85.61
13	86.69	91.87	75.66	78.69
14	92.3	96.19	84.37	90.05
15	94.97	93.68	78.6	95.55
16	88.88	92.74	80.07	84.68
17	85.12	83.39	75.71	84.81
18	92.01	98.59	90	90.04
19	90	93.18	89	86.77
20	85.69	92.68	81.15	86.77
21	93.55	96.81	86.37	91.23
22	90.91	95.33	80.47	82.54
23	91.09	92.89	78.98	84
24	89.89	92.01	88.03	82.91
25	87.13	94.87	86.3	78.83
26	84.79	93.23	85.13	87.71
27	85.08	88.34	56.71	65.5
28	74.88	83.83	69.39	75.43
29	92.47	95.89	81	86.71
30	90.86	93.64	80.7	86
31	91	94	83.81	90
32	90.12	97.73	88	86.68
33	92.48	84.07	79.3	86.97
Mean accuracy (%)	86.6 ± 15.82	93.12 ± 3.79	81.26 ± 7.17	85.03 ± 6.81

Table 4.15: Subject-based classification accuracies for DFNN and FE-DFNN classifiers on reduced size youth dataset

Subject	Youth-ankle Dataset		Youth-wrist Dataset	
	DFNN (Raw)	FE-DFNN (Feature)	DFNN (Raw)	FE-DFNN (Feature)
1	78.91	80.61	75.34	75.73
2	86.61	92.99	77.15	90.63
3	87.32	91.26	63.73	64.88
4	83.49	84.63	87.29	89.04
5	85.17	93.71	83.65	90.74
6	84.12	92.71	73.11	83.55
7	86.3	91.88	93.16	89.37
8	48.05	70.14	87.13	89.74
9	74.81	82.92	63.84	83.16
10	90.88	83.85	70	76.37
11	89.27	89.29	90.02	85.6
12	90.13	92.41	87.53	92.23
13	83.22	91.69	78.81	93.27
14	86.93	96.13	78.55	85.75
15	84.11	94.04	85.67	82.34
16	87.94	84	87.47	84.61
17	80.45	90.61	85.05	86.46
18	60.84	80.81	91.24	94.39
19	90.36	81.92	90.37	94
20	61.26	64.62	86.68	87.01
Mean accuracy (%)	81.01 ± 11.45	86.51 ± 8.22	81.79 ± 8.83	85.94 ± 7.18

data is limited” to show that feature engineering can in fact outperform feature learning. In fact, we demonstrate that in some cases the state-of-the-art human engineered features outperform deep learning. In other words, we stress the importance of choosing the right learning representation and the right topology.

Chapter 5: Sub-transfer Learning for Retuning the Outlier User Accuracy

The sections of this chapter have been communicated to IEEE Sensors Journal for publication and currently, the paper is in its 1st revision stage.

5.1 Introduction

Human activity recognition (HAR) has become one of the most active areas of sensory healthcare research due to its potential in improving the quality of life both for patients in healthcare settings who are susceptible to falls or undergo physical therapy and the public to promote a more active lifestyle. A variety of state-of-the-art machine learning and artificial intelligence algorithms are applied to ever growing datasets of sensory data collected from many users to achieve unprecedented performance in recognizing activities of daily living. Majority of the research focuses on the average performance across the population of users however, it is evident from these studies that the performance distribution across different users is anything but uniform. In fact, the outlier users can demonstrate performance degradations of as much as 30% compared to the median accuracy reported on any given dataset. In this chapter, we study several approaches in determining the impact of outlier users on HAR performance and propose our novel sub-transfer learning approach which follows the principles of transfer learning within the same dataset when coupled with augmentation techniques.

5.2 Methodology

In [49], we explored feature learning techniques for HAR including both one and two dimensional convolutional neural networks (1D-CNN and 2D-CNN) to investigate the classifi-

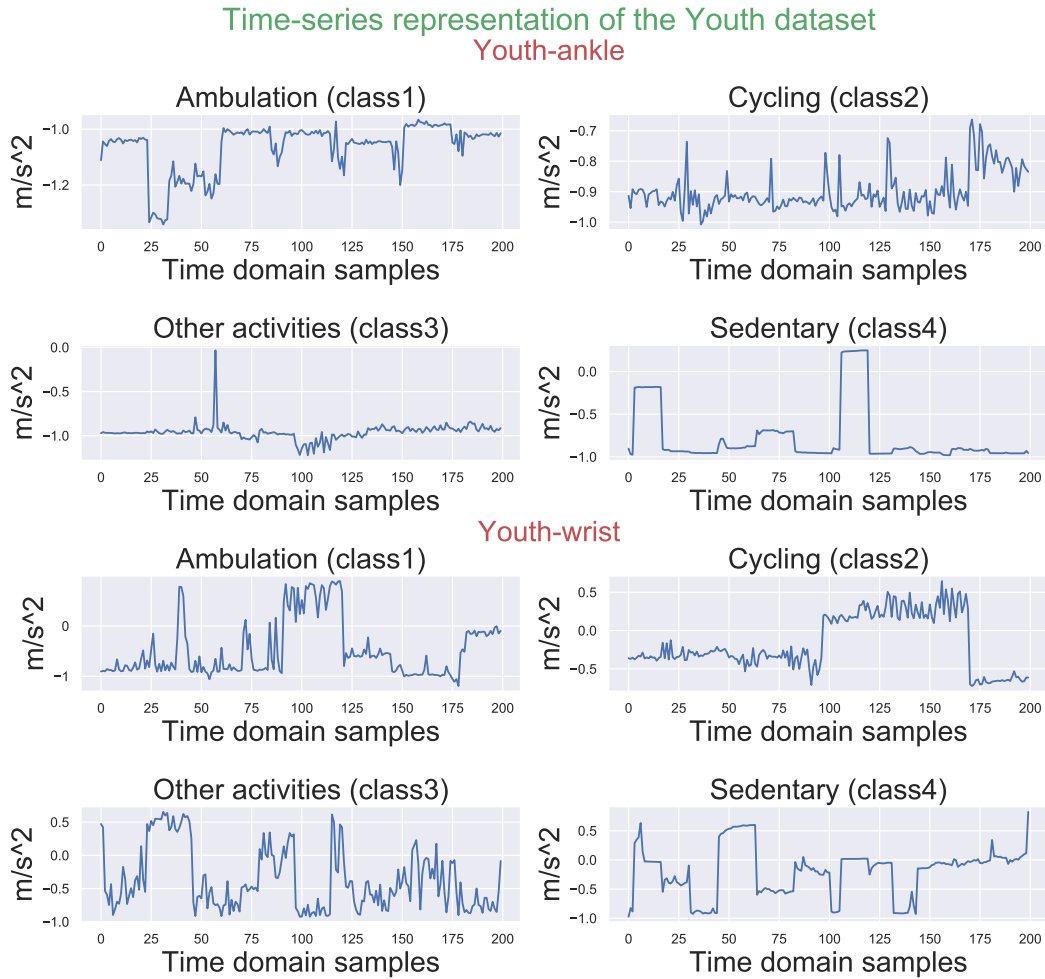


Figure 5.1: Time-series representations of single-axis acceleration data for four classes of activities of daily living for the youth-ankle and youth-wrist datasets

cation performances of feature learning from temporal and spectrotemporal data respectively versus feature engineering using leave-one-subject-out (LOSO) cross validation on Stanford Adult-youth dataset [64]. As previously discussed, some subjects have demonstrated significantly lower classification accuracies than others regardless of the algorithm being used. We will begin by describing the dataset followed by the specific learning algorithms used in the chapter.

5.2.1 Dataset

To boost the classification performances of the subjects with lower accuracies for subject-oriented training model using LOSO, we test our proposed learning representations named sub-transfer learning model (STLM) and subject-specific learning model (SSLM) on some subjects of the Stanford Adult-youth dataset. The dataset is categorized into four distinct datasets based on the combinations of the subject being an adult or youth and where the sensor is placed such as, (i) adult-ankle, (ii) adult-wrist, (iii) youth-ankle and (iv) youth-wrist dataset. The dataset includes 53 subjects with 33 adults and 20 youths [64]. In the preprocessing stage, the dataset is resampled and filtered considering the same parameters which we used in our previous research on this dataset [49]. In this dataset, twenty-six activities of daily living (ADLs) of the adult dataset such as walking: carrying load, stairs: inside and down, cycling outdoor uphill, painting: roller etc. and twenty-two daily ADLs of the youth dataset such as walking natural, outdoor cycling, basketball: dribbling etc. are categorized into four different classes: (i) ambulation, (ii) cycling, (iii) other activities and (iv) sedentary. Figure 5.1 represents a representative 200 samples from each of these 4 classes for the youth dataset for easy visualization. Each subplot in the figure displays the single-axis acceleration data for one of the three axes (x, y or z) which characterizes a more dynamic representation of motion as the dominant axis.

To study sub-transfer and subject-specific learning, we select three subjects from each of these four datasets based on the maximum, minimum and median accuracies that we have achieved among the 33 subjects for adult-ankle and adult-wrist datasets and the 20 subjects for youth-ankle and youth-wrist datasets and implemented a 1D-CNN classifier using LOSO cross validation.

5.2.2 One-Dimensional Convolutional Neural Network

Since HAR data is a one-dimensional time-series sequence, collected from accelerometer sensors, we implement a one-dimensional convolutional neural network as a sequence clas-

sifier. 1D-CNN model learns the feature vector in unsupervised fashion in its latent space from the sequences of raw temporal dataset and map the internal features to different ADLs. 1D-CNNs includes two distinct layers: i) convolutional layers where both 1D convolutions, activation functions and pooling are included, and ii) fully connected or dense layers which are similar to a standard multi-layer perceptron. While performing the convolution operation on raw temporal data, 1D-CNN learns to extract the rich features used in the classification task performed by the dense layers. The performance of the 1D-CNN classifier is optimized to achieve the maximum classification accuracy by combining both automatic feature extraction and classification operation into a single process which includes linear forward and back-propagation operations. Due to the linear operations between the convolutional layers, this algorithm results in a lower computational complexity compared to others. The 1D forward propagation (1D-FP) in each CNN layer is described as [54],

$$x_k^l = b_k^l + \sum_{i=1}^{N_{l-1}} \text{conv1D}(W_{ik}^{l-1}, S_i^{l-1}). \quad (5.1)$$

where, x_k^l represents the input, b_k^l is the bias of the k^{th} neuron at layer l , S_i^{l-1} is the output of the i^{th} neuron at layer $l-1$ and W_{ik}^{l-1} is the kernel from the i^{th} neuron at layer $l-1$ to the k^{th} neuron at layer l . $\text{conv1D}(\cdot, \cdot)$ is used to perform 1D convolution without zero-padding. The intermediate output, y_k^l can be defined as a function of the input x_k^l where the input passes through the activation function $f(\cdot)$.

$$y_k^l = f(x_k^l) \quad \text{and} \quad S_k^l = y_k^l \downarrow \text{ss}. \quad (5.2)$$

In (5.2), S_k^l is the output of the k^{th} neuron of the layer l and $\downarrow \text{ss}$ is defined as the down-sampling operation with a scalar factor, ss . During training, 1D-FP generates an error and the backpropagation (BP) of the error starts from the output of the fully connected layers by computing the direction and the size of change in weight parameters to reduce the error with the gradient of the network. The mean square error is shown as,

$$E_p = MSE(t^p, [y_1^L, \dots, y_{N_L}^L]') = \sum_{i=1}^{N_L} (y_i^L - t_i^p). \quad (5.3)$$

Let $l = 1$ for the input layer, $l = L$ for the output layer and N_L is the number of classes in the dataset. In (5.3), the target and the output vectors are defined as t^p and $[y_1^L, \dots, y_{N_L}^L]'$ respectively for an input vector p . To find the derivative of E_p by each network parameter, the delta error is calculated as $\Delta_k^l = \frac{\delta E}{\delta x_k^l}$. For BP from the next layer, $l + 1$, to the current layer, l , the input delta of the CNN layer l , is Δ_k^l which is found as,

$$\begin{aligned} \Delta_k^l &= \frac{\delta E}{\delta y_k^l} \times \frac{\delta y_k^l}{\delta x_k^l} = \frac{\delta E}{\delta us_k^l} \times \frac{\delta us_k^l}{\delta y_k^l} \times f'(x_k^l) \\ &= up(\Delta S_k^l) \times \beta \times f'(x_k^l). \end{aligned} \quad (5.4)$$

where, zero order up-sampled map, $us_k^l = up(S_k^l)$ and $\beta = (ss)^{-1}$. The BP from the 1st layer to the last CNN can be computed as,

$$\frac{\delta E}{\delta S_k^l} = \Delta S_k^l = \sum_{i=1}^{N_{l+1}} \frac{\delta E}{\delta x_i^{l+1}} \frac{\delta x_i^{l+1}}{\delta S_k^l} = \sum_{i=1}^{N_{l+1}} \Delta_i^{l+1} W_{ki}^l. \quad (5.5)$$

Then, BP of the delta error can be represented as,

$$\delta S_k^l = \sum_{i=1}^{N_{l+1}} conv1Dz(\Delta_i^{l+1}, rev(W_{ki}^l)). \quad (5.6)$$

where $rev(\cdot)$ is used to reverse the array and $conv1Dz(\cdot, \cdot)$ is used to perform 1D convolution with zero-padding. The weight and bias can finally be updated as follows:

$$\begin{aligned} \frac{\delta E}{\delta W_{ik}^l} &= conv1D(S_k^l, \Delta_i^{l+1}) \\ \frac{\delta E}{\delta b_k^l} &= \sum_n \Delta_k^l(n). \end{aligned} \quad (5.7)$$

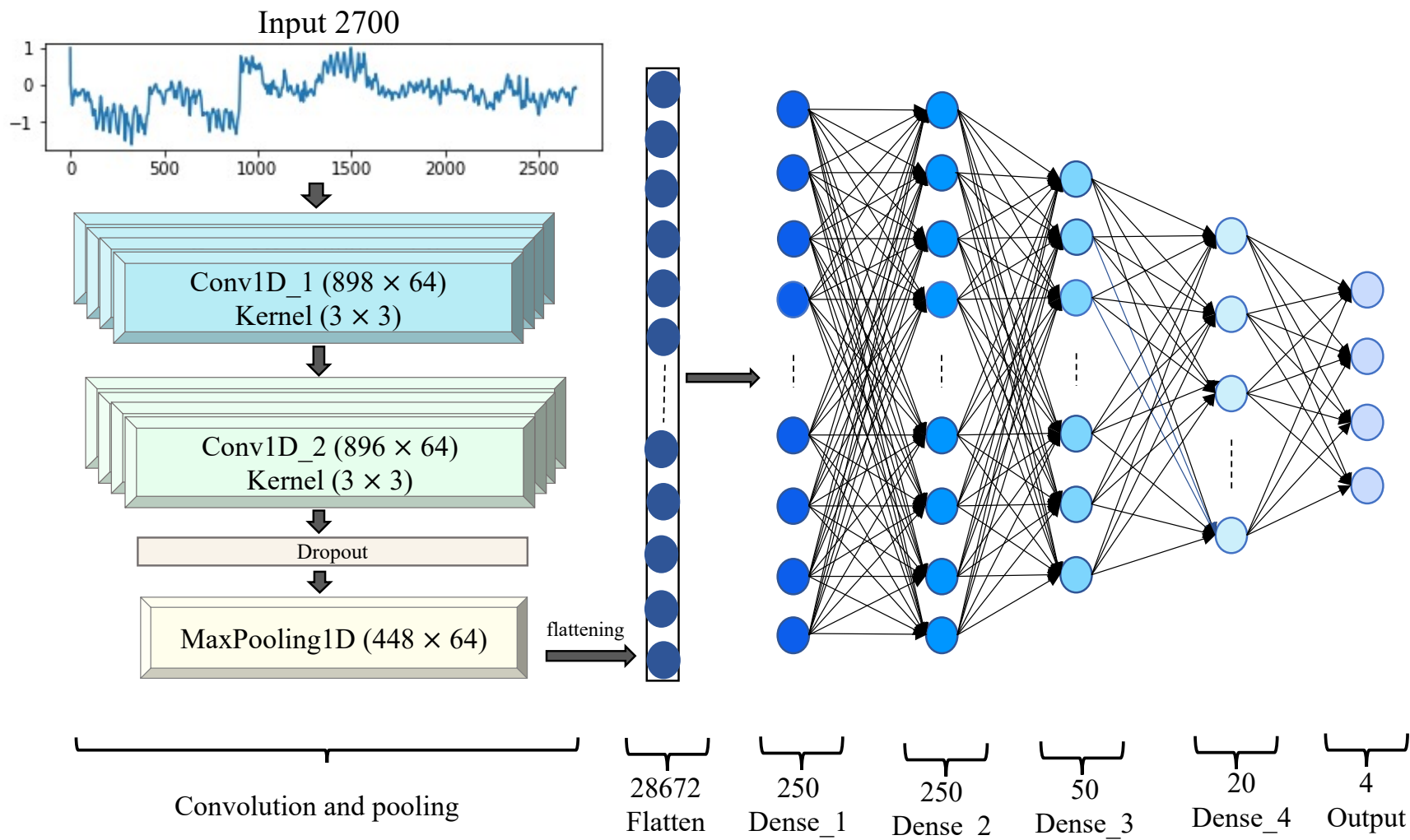


Figure 5.2: Framework of 1D-CNN classifier

Figure 5.2 represents the framework of the deep 1D-CNN classifier used in this chapter. The figure shows that the adult-ankle dataset with 10080 observations in 2700 dimensions is fed as the input to the model where two 1D-convolutional layers with 64 filters of kernel size (3, 3) perform the forward-backward operation and optimize the layer-weights to extract the features in the latent space of the deep neural network. In this architecture, a ‘Dropout’ layer with probability = 0.5 is added to prevent overfitting during training. After the convolution operation, a pooling layer (of size 2) maps each feature to 448 dimensions which is then flattened to 28672 dimensions by selecting the maximum parameters within the range of the predetermined window as the output value. The output of the flattened layer is processed through four fully connected layers with 250, 250, 50, 20 neurons respectively to map the output from the input.

5.2.3 Synthetic Minority Oversampling Technique

Deep learning requires a sufficient number of samples to properly train the thousands of parameters included in the many layers of its topology. Even for a medium sized dataset such as the one used in this study with tens of thousands of observations, unsupervised feature learning demands additional data to be able to appropriately represent the most critical features of raw data in the latent space. A popular technique to overcome the imbalance problem is the “minority oversampling method,” or SMOTE [28, 78]. Its main contribution is readjusting the size of the original training data set by increasing the minority class with random sampling. When the SMOTE oversamples the minority class, it produces new data instead of replicating the actual samples. This feature sets the SMOTE aside from other oversampling techniques and actually allows it to be used for other applications as well, such as data augmentation which we have used in this study. Obtaining new data is done with interpolation between similar examples which belong to the same class. This procedure works on the “feature (or latent) space” rather than the entire dataset. For instance, it

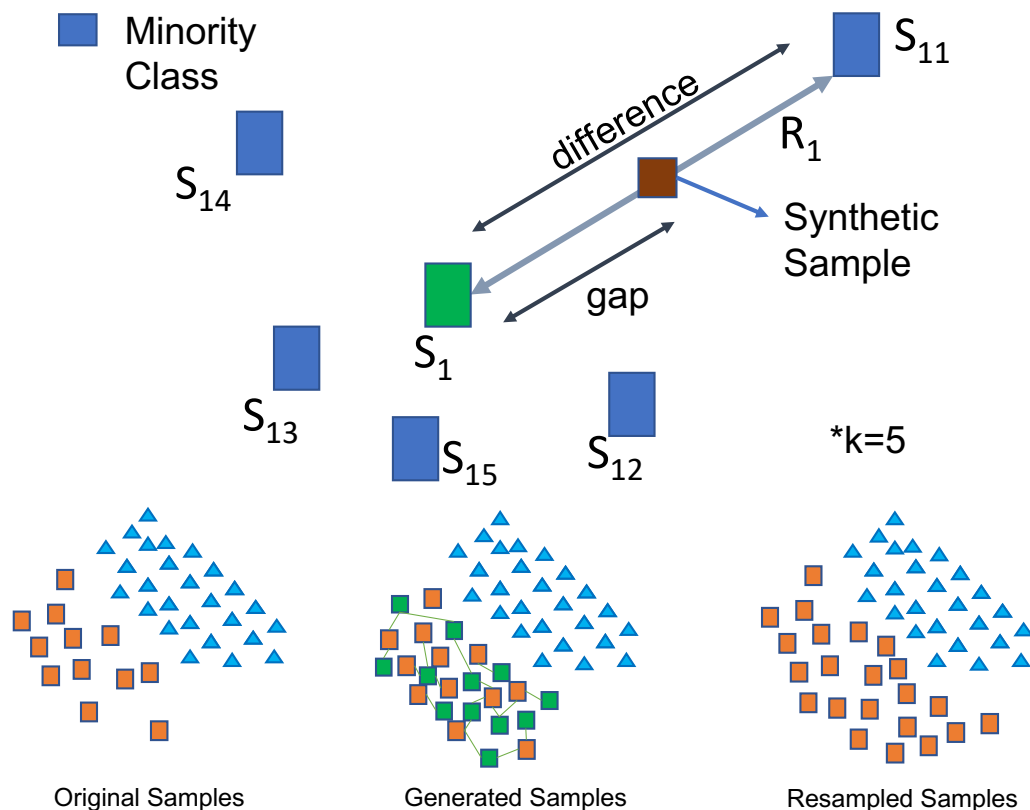


Figure 5.3: Generating synthetic samples using SMOTE

creates a new sample between two samples that are in the same class. This new sample is called a ‘synthetic sample’, which carries characteristics from these two class samples.

The procedure of the SMOTE algorithm can be described more formally as follows. The amount of oversampling, N , which indicates how many new samples per original sample will be generated is an important hyperparameter of the model. This is followed by an iterative method where the SMOTE algorithm randomly selects a minority class sample in the training data. The K-Nearest neighbor method is applied with a default value of 5 (another hyperparameter). Finally, the k-value is chosen randomly based on the oversampling parameter N for the minority samples to produce a synthetic piece by interpolation [19]. First, the distance is measured between the feature vectors and their neighbors by utilizing the “Euclidean Distance”. Then, this measurement is multiplied randomly with a number in between $[0,1]$ and added to the original feature sample. Hence, different samples are generated

that carry a variety of previous samples along the line. This process can be mathematically defined as follows:

$$S' = S + \text{rand}(0, 1) \times |S - S_k| \quad (5.8)$$

where S' is the new set of synthetic data samples, and S_k denotes the randomly picked k-nearest neighbor distance samples. Figure 5.3 shows the basic operational principle of SMOTE graphically as new samples are generated between the original samples S_1 and S_{11} for instance.

5.3 Design and Experimental Setup

In case of subject-agnostic datasets such as handwritten digits and speech recognition, the performance of the classification algorithm depends on the richness of the subjects selected for training. HAR is another example where the datasets are motion-sensitive and everyone’s movement signature is quite different where some people can be considered outliers compared to the general population. In this case, the classification accuracies for these subjects typically drop statistically significantly below the average accuracy levels when using the conventional training and testing methods. To address this important problem, we investigate several learning representations such as STLM and SSLM to i) study their impact on subject-specific accuracy and ii) potentially boost the performance for these outlier subjects and compare the performances with the state-of-the-art methods.

5.3.1 Sub-transfer Learning Model

We introduce STLM as a powerful alternative to fine tune the accuracy of subject oriented classification problems such as HAR in healthcare. In sub-transfer learning, we apply similar principles of pre-training the model from transfer learning with the main difference in using the same dataset but with other users to boost the performance on new users with fine tuning unlike as in transfer learning where training is performed on a completely different

domain dataset. In other words, we reuse a previously trained model with different subjects as the starting point for another model which can be trained/fine-tuned with only a handful of samples from the test subject and tested on the remaining majority samples of the test subject from the same dataset. The model which is trained using conventional training is repurposed on the test data to optimize the training performance and allow rapid updates to the fine-tuning process when modeling on the test subject of the same dataset. In order to achieve a fair comparison with subject-specific learning and enhance the classification accuracy, SMOTE is applied on the dataset to increase the number of test subject samples which are much fewer in number than the main dataset used in training the source model (SM).

5.3.2 Subject-specific Learning Model

As a baseline approach, we employ SSLM as a trivial approach where the training of the model starts from scratch, without transferring any knowledge from the pre-trained model. In this learning representation, the training is performed only on the training samples from the test subject and tested on the remaining samples. Since SSLM is trained and tested only on the test subjects individually, the size of the training data is augmented by generating the synthetic samples with artificial oversampling algorithm like SMOTE for a fair comparison with STLm and to boost the classification accuracy.

5.3.3 Experimental Setup and Model Design

Figure 5.4 shows the framework of our experimental setup and proposed model along with SM and SSLM on the adult-ankle dataset with 33 subjects. The figure displays the two learning representations with the 1D-CNN classifier used in this chapter. First, the train-test-split is performed on the adult-ankle dataset using 33-folds to implement leave-one-subject-out (LOSO) cross-validation to generate the *train data_SM* from 32 subjects and

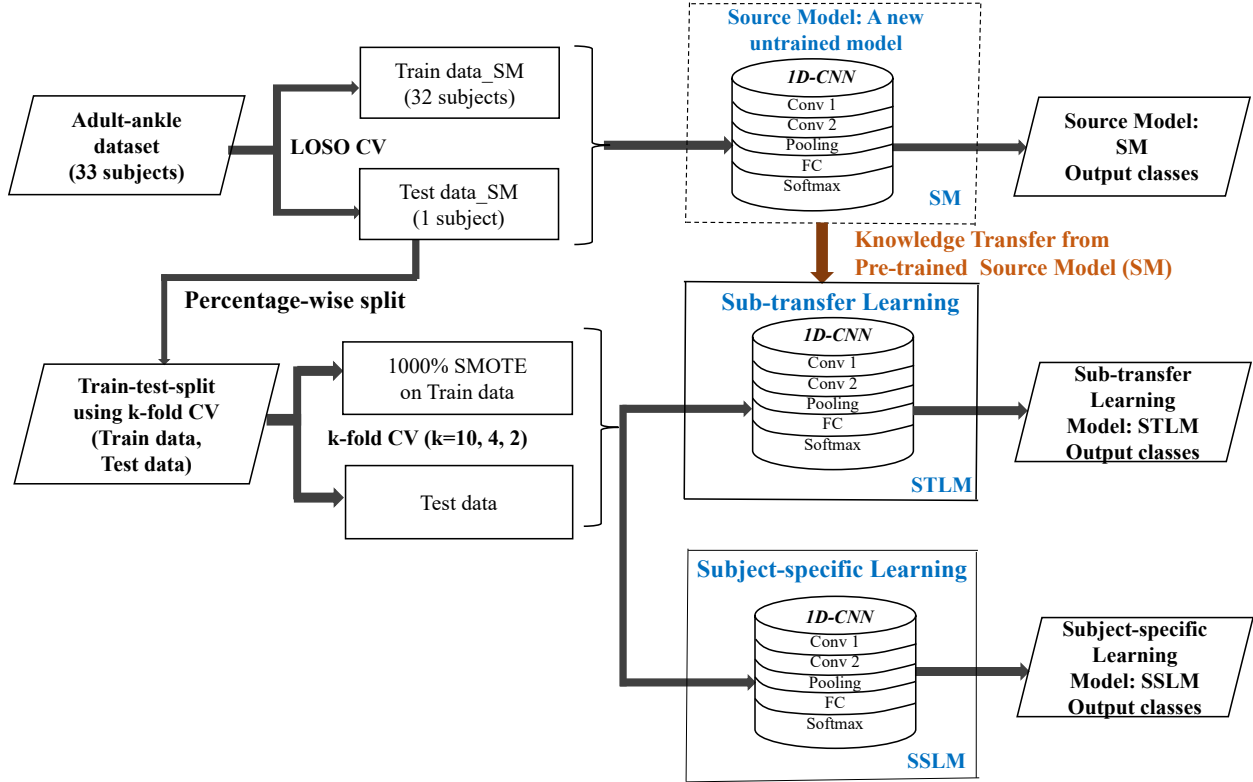


Figure 5.4: Block diagram of sub-transfer learning and subject-specific learning models

the *test data_{SM}* from the remaining subject. The training and validation of the models are then completed using the following steps:

- A new untrained 1D-CNN model is trained on the 32 subject training data to create a large SM which is then tested on *test data_{SM}* to record the subject-oriented classification accuracies.
- *Test data_{SM}* from the previous step is split into five combinations to train STLM and SSLM using 10, 4 and 2-fold cross validations to create the following training & testing split: 10% , 25%, 50%, 75% and 90% for training and the rest for testing respectively. As shown in Figure 5.4, these splits are called *train* and *test data*.
- To avoid the imbalanced classification problem and boost the performance of the classifier, we augment the size of the training data by 11-folds by applying SMOTE on *train data*. Moreover, we use random oversampling as a data augmentation technique

for the cases where the number of samples from the minority class is ≤ 1 . The $N = 11$ parameter is chosen experimentally such that the augmented data is comparable in size to the data used to train the SM without too much repetition.

- Then STLM which includes a deep 1D-CNN classifier is trained on *train data* using the knowledge (previous parameters) from the SM to classify the ADLs on test data.
- To do a fair comparison, we use the exact same data (*Train data*, *Test data*) as the input to the SSLM. The only difference between STLM and SSLM is that SSLM is trained from scratch whereas STLM is trained by retuning of the SM parameters using the augmented data from the subject.

The operational algorithms for both the source and the subsequent sub-transfer learning models are shown in Algorithm 1 and 2.

5.4 Results and Discussion

The experimental setup described in the previous section is applied to three different subjects from each dataset with the training of a new untrained classification source model. The subjects are specifically chosen at the different outlier levels in the classification space (i.e., the highest, the median, and the lowest accuracies) to explore the performance of sub-transfer and subject-specific learning for a range of challenging tasks. Table 5.1 shows a robust comparison of the classification performances of STLM (where the knowledge from the previously trained model is transferred during additional training on some samples from the test dataset) and SSLM (where the model performs training only on the samples from the test subject without doing a previous training) on twelve subjects (three for each of the four datasets) and finally SM without any additional training using LOSO CV.

The advantages of sub-transfer learning using previous experience from training on other subjects become apparent when we compare the classification performance of this learning representation with the subject-specific learning with no prior training. We observe that

Algorithm 1: Source Model

Input : $\Omega_{in} = (X_{S_k}^{f_n}, Y_{S_k})$;
 f_n : number of features for $n = 1, 2, \dots, m$;
 S_k : number of subjects for $k = 1, 2, \dots, l$;
Output: Labels of the time-series data;
 $labels : \{0, 1, 2, 3\}$;

- 1 **for** $i \leftarrow 1$ **to** l **do**
- 2 **Compute** Leave-one-subject-out cross-validation to generate *Test data_SM*
 (observations for 1 subject) and *Train data_SM* (observations for remaining
 subjects);
- 3 $Train_x[i] = \{x_{S_k}^{f_0}, x_{S_k}^{f_1}, \dots, x_{S_k}^{f_n}\}, \forall k \neq i$;
- 4 $Test_x[i] = \{x_{S_i}^{f_0}, x_{S_i}^{f_1}, \dots, x_{S_i}^{f_n}\}$, for $k = i$;
- 5 $Train_data_SM[i] = (X_{S_k}^{f_n}, Y_{S_k})$;
- 6 $Test_data_SM[i] = (X_{S_i}^{f_n}, Y_{S_i})$;
- 7 **for** $iteration \leftarrow 1$ **to** num **do**
- 8 Fit 1DCNN model using *Train data_SM*[i]; Back-propagate error and adjust
 model parameters;
- 9 **Compute** accuracy and loss function;
- 10 **end for**
- 11 **if** *convergence* **then**
- 12 Predict the labels of *Test data_SM*[i] using trained network;
- 13 **Compute** test accuracy;
- 14 **Save** source model;
- 15 **end if**
- 16 **end for**

Algorithm 2: Sub-transfer Learning Model

```
Input :  $\Omega_{in} = (X_{S_k}^{f_n}, Y_{S_k})$ ;  
          $f_n$ : number of features for  $n = 1, 2, \dots, m$ ;  
          $S_k$ : number of subjects for  $k = 1, 2, \dots, l$ ;  
Output: Labels of the time-series data;  
          $labels : \{0, 1, 2, 3\}$ ;  
1 Repeat the steps of the source model;  
2  $accuracyList = [ ]$ ;  
3  $train\_size = [10, 25, 50, 75, 90]$ ;  
4 for  $j \leftarrow 1$  to  $len(train\_size)$  do  
5    $tr\_size = train\_size[j]$ ;  
6   if  $(tr\_size == 10) \text{or} (tr\_size == 90)$  then  
7      $cross\_val = 10$ ;  
8   end if  
9   if  $(tr\_size == 25) \text{or} (tr\_size == 75)$  then  
10     $cross\_val = 4$ ;  
11  end if  
12  if  $(tr\_size == 50)$  then  
13     $cross\_val = 2$ ;  
14  end if  
15   $acclist = [ ]$ ;  
16   $testx = Test\_x[i]$ ;  
17   $testy = Test\_y[i]$ ;  
18  for  $tr\_id, ts\_id$  in  $cross\_val.split(testx, testy)$  do  
19     $tr\_x, ts\_x = testx[tr\_id], testx[ts\_id]$ ;  
20     $tr\_y, ts\_y = testy[tr\_id], testy[ts\_id]$ ;  
21     $Train\_data = [tr\_x, tr\_y]$ ;  
22     $Test\_data = [ts\_x, ts\_y]$ ;  
23     $tr\_smote = SMOTE(Train\_data, N, K)$ ; //  $N = 1000$ : amount of  
        oversampling (%),  $K = 5$ : number of nearest neighbors to  
        consider.  
24    for  $iteration \leftarrow 1$  to  $num$  do  
25      Load source model;  
26      Update source model using training  $tr\_smote$  iteration;  
27      Compute accuracy and loss function;  
28    end for  
29    if convergence then  
30      Predict the labels of  $Test\_data$  using trained network;  
31      Compute  $test\_accuracy$ ;  
32    end if  
33     $acclist.append(test\_accuracy)$ ;  
34  end for  
35   $Acc = np.array(acclist)$ ;  
36   $accuracyList.append(Acc)$ ;  
37 end for
```

Table 5.1: Experimental results of sub-transfer and subject-specific learning models

Adult-ankle Dataset									
	Subject-28 (Minimum)			Subject-31 (Maximum)			Subject-27 (Median)		
	STLM	SSLM	SM	STLM	SSLM	SM	STLM	SSLM	SM
Training data (%)	ACC \pm STD (%)	ACC \pm STD (%)	ACC \pm STD (%)	ACC \pm STD (%)	ACC \pm STD (%)	ACC \pm STD (%)	ACC \pm STD (%)	ACC \pm STD (%)	ACC \pm STD (%)
10	95.65\pm1.08	71.65 \pm 9.78		99.03\pm0.74	73.56 \pm 9.25		96.75\pm1.73	71.50 \pm 11.22	
25	96.62\pm1.08	88.11 \pm 4.06		98.92\pm0.83	86.08 \pm 4.72		96.78\pm2.14	87.38 \pm 5.11	
50	96.82\pm2.01	88.96 \pm 3.42	88.41	99.04\pm0.45	96.14 \pm 0.89	98.71	97.22\pm3.93	90.69 \pm 2.55	95.65
75	98.28\pm2.20	91.93 \pm 3.83		99.68\pm0.63	95.52 \pm 2.16		99.08\pm1.17	94.45 \pm 3.83	
90	98.29\pm2.39	95.10 \pm 3.60		99.70\pm0.96	98.35 \pm 2.34		1\pm0.0	92.18 \pm 6.34	
Adult-wrist Dataset									
	Subject-6 (Minimum)			Subject-31 (Maximum)			Subject-33 (Median)		
10	79.35\pm10.85	68.47 \pm 7.03		96.06\pm1.43	78.63 \pm 3.84		97.92\pm1.77	70.55 \pm 5.85	
25	87.50\pm8.65	73.15 \pm 6.11		96.13\pm1.17	86.32 \pm 2.18		98.03\pm2.22	92.25 \pm 2.77	
50	93.02\pm9.87	78.20 \pm 4.52	66.15	96.70\pm1.84	91.40 \pm 1.79	96.03	99.02\pm1.39	89.86 \pm 11.54	90.82
75	96.97\pm6.07	80.22 \pm 4.13		97.04\pm2.44	92.41 \pm 5.63		99.35\pm1.30	95.13 \pm 5.01	
90	97.78\pm5.84	85.97 \pm 3.75		99.38\pm1.32	93.79 \pm 3.84		1\pm0.0	98.98 \pm 2.71	
Youth-ankle Dataset									
	Subject-8 (Minimum)			Subject-12 (Maximum)			Subject-19 (Median)		
10	83.53\pm11.52	56.06 \pm 10.62		96.46\pm1.57	66.24 \pm 11.21		84.99\pm6.02	72.10 \pm 6.72	
25	89.18\pm6.07	70.69 \pm 5.85		96.89\pm1.85	83.20 \pm 6.57		89.79\pm4.01	81.12 \pm 5.62	
50	92.97\pm6.74	78.31 \pm 0.48	58.87	96.91\pm2.92	85.05 \pm 10.93	95.88	90.30\pm3.96	84.57 \pm 0.93	89.14
75	96.39\pm5.24	79.70 \pm 1.85		97\pm4.76	87.78 \pm 6.10		95.50\pm5.1	87.13 \pm 0.84	
90	98.33\pm2.99	80.28 \pm 6.48		98.29\pm2.56	89.07 \pm 3.90		97.83\pm4.38	84.19 \pm 7.09	
Youth-wrist Dataset									
	Subject-3 (Minimum)			Subject-20 (Maximum)			Subject-11 (Median)		
10	89.37\pm5.54	74.76 \pm 4.84		97.50\pm1.14	84.52 \pm 3.90		92.29\pm1.21	70.29 \pm 4.92	
25	91.73\pm4.82	84.88 \pm 3.18		98.10\pm1.32	87.66 \pm 3.53		94.43\pm2.68	80.06 \pm 6.35	
50	91.96\pm10.52	90.46 \pm 5.03	78.21	98.01\pm2.81	92.31 \pm 0.37	95.56	92.71\pm8.91	90.20 \pm 0.14	90.2
75	93.80\pm8.66	88.98 \pm 3.62		99.16\pm1.69	92.87 \pm 3.04		95.19\pm9.62	90.20 \pm 2.74	
90	98.86\pm3.61	91.12 \pm 5.59		99.17\pm1.84	92.57 \pm 5.29		96.75\pm8.59	92.27 \pm 7.05	

in most instances especially for the subjects with the lowest accuracies, STLM provides a significant boost to the classification accuracy compared to both SM & SSLM. When the subject specific accuracies are already good or median, the sub-transfer learning still provides a better performance albeit less so when compared to the SM. Statistically speaking (more on this in the next subsection), we can find statistically significant differences when the accuracy is low, but not when the accuracy is high. More importantly, these gains in performance are realized only when using a very small percentage of the subject specific samples (i.e., 10% which generally means around 10 to 15 samples per subject) where SSLM fails to compete with either approach. For the adult-wrist dataset, the most difficult dataset in this study based on all previously published [49] results, STLM outperforms SM with LOSO by a significant margin (almost 80% compared to 66%) for the lowest accuracy subject 6, a robust margin (98% to 91%) for the median accuracy subject 33 while having the same performance for the subject where the accuracy is already very high (96% in both cases)

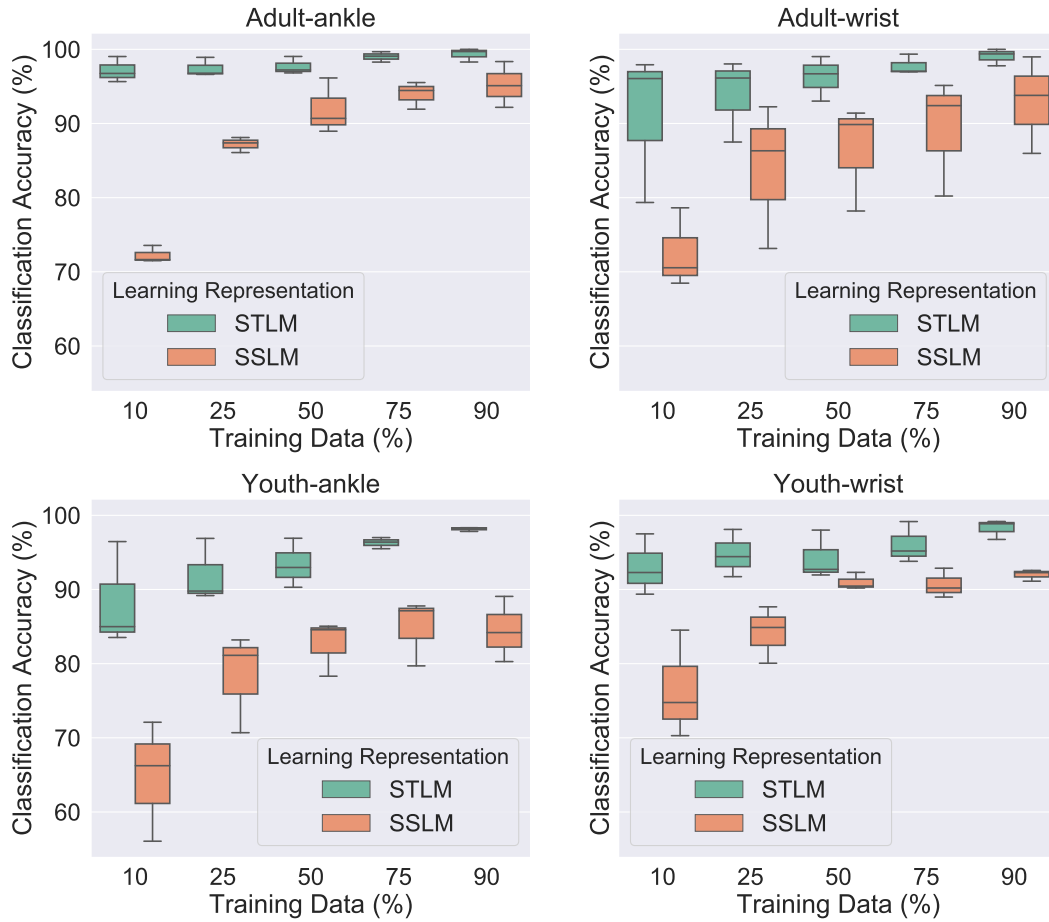


Figure 5.5: Boxplot distribution to compare the classification accuracies of sub-transfer learning and subject-specific learning models on Adult-youth dataset

even when using only 10% of the samples from the test set. Interestingly, SSLM displays a competitive performance with the SM for the lowest accuracy subjects, which highlights the insufficiency of a large model trained on the general population on outlier subjects. For these instances, subject-specific training has its own advantages as it is a much less resource-intensive way to train the learning algorithm on a specific subject with associated trade-offs in accuracy. Figure 5.5 displays the boxplot distributions for the classification accuracies of STLM and SSLM for the outlier subjects on four datasets for a better statistical comparison.

Table 5.2: Statistical significance analysis between the performances of the learning representations on twelve subjects of four datasets

Model	Null Hypothesis (h)	p-value (p)	Interpretation
SM vs. STLM	1	0.0093	Significant
STLM vs. SSLM	1	4.88E-04	Significant
SM vs. SSLM	1	9.77E-04	Significant

5.4.1 Statistical Significance Analysis of the Classification Models

To compare the statistical significance of the performance of our proposed model compared to the SM and SSLM, we conduct a statistical hypothesis test using the Wilcoxon signed-rank test. To perform any statistical test, one should investigate the independence of the observations, the homogeneity of the variance and the normality of the data tests to make common assumptions about the experimental data which constitutes the average classification accuracies of the STLM, SSLM and SM models on the 10% train data. Since the distributions of classification rates for these three models do not necessarily satisfy the above assumptions for a standard statistical significance test, we apply a popular non-parametric approach using a Wilcoxon signed-rank test for paired data. For two groups of dependent paired data, the Wilcoxon signed-rank test ranks the absolute values of the differences between the paired observations and realizes whether the two dependent samples are chosen from populations having the same distribution [25].

Table 5.2 clearly shows that for the Wilcoxon signed-rank test, the null hypotheses (h) are statistically rejected in case of every paired comparison. Since the average classification accuracies of $STLM > SM > SSLM$, we conclude that the STLM is statistically significantly better than SM which in turn is better than SSLM.

5.5 Summary and Conclusions

In this chapter, we explore several approaches of learning representations to boost the classification accuracies of HAR, especially for the low performing subjects and introduce a novel learning technique, sub-transfer learning which reuses the knowledge from the previ-

ously trained source model. We demonstrate that our novel STLM can enhance the classification accuracies of the state-of-the-art 1D-CNN classifier especially for the outlier users using a handful samples from the test data. Moreover, we provide a robust comparison of our sub-transfer learning technique with subject-specific learning which has no prior training experience and show that STLM performs statistically significantly better than SM which is better than SSLM.

Chapter 6: Conclusions, Contributions and Future Work

The contributions to science detailed in each chapter is summarized below, followed by a discussion on potential future work.

6.1 Deep Learning versus Feature Engineering

In this chapter, we have addressed some of the factors behind the inexact science of choosing deep learning versus feature engineering specifically when it comes to temporal time series for human activity recognition (HAR) from acceleration data. We develop an extensive experimental framework, which uses a binary and a multiclass datasets (UniMIB SHAR which is a publicly available and commonly used benchmark dataset) with different characteristics to compare state-of-the-art domain specific features and raw temporal data to train four different learning architectures across various topologies to ultimately arrive at the conclusion that a recurrent neural network with long short term memory (RNN-LSTM) classifier provides the highest recognition accuracy on a 9 activity class dataset whereas a one-dimensional convolutional neural network (1D-CNN) classifier provides the highest classification rate on a binary fall detection dataset both representing states-of-the-art in the literature.

More specifically, we show that in an activity of daily living recognition task which consists of 9 different motion classes, a black-box deep learning approach using raw temporal data on a recurrent LSTM model outperforms the most recent results reported in the literature using domain specific features and achieves a state-of-the-art classification rate of 98.02%. On the contrary, for a deep feedforward neural network where conventional logic would suggest better performance on raw temporal data, we observe that the domain-specific

features have a higher accuracy by a significant margin. This may suggest that to truly take advantage of the time dependencies expressed in the raw temporal form of a moderately sized dataset, the topology should be properly selected. Another conclusion is that if the network topology allows for constructing a rich latent feature space from the raw input data like the 1D-CNN, they can outperform expert features provided that there is sufficient data to learn from. In this case, the front ends of both a 1D-CNN and an RNN-LSTM are able to successfully adapt to these dependencies to achieve a better classification rate than using expert features.

6.1.1 Contribution 1

We have provided an exhaustive framework to study the performance of human engineered features (feature engineering) and latent features (feature learning) and demonstrated that state-of-the-art performance can only be achieved on datasets with different characteristics using different topologies and/or feature representations.

6.2 Empowering Deep Feature Learning in HAR

Our contribution in this chapter is an in-depth look into how two fundamentally different approaches to HAR: using state-of-the-art, subject agnostic and human engineered features and automated deep feature learning compare on a real life dataset. We have demonstrated for the first time that a two-dimensional convolutional neural network (2D-CNN) classifier provides the highest recognition accuracy on all four partitions of the Stanford motion dataset by outperforming the previous state-of-the-art using subject agnostic features. Furthermore, we have found out that not all feature learners perform at the same level as the learning representation takes an important role where a spectrotemporally represented data on a 2D-CNN outperforms the standard unsupervised feature extraction technique using an autoencoder with deep feedforward neural network model. Finally, null hypothesis tests to study the statistical significance comparison of the results for deep learning and feature engi-

neering algorithms have clearly demonstrated a statistically significantly higher classification performance for our proposed learning technology using spectrograms.

6.2.1 Contribution 2

We have introduced a novel feature learning representation for human activity recognition using spectrograms to demonstrate state-of-the-art performance on a foundational dataset where the best previously reported algorithm is using human engineered subject agnostic features.

6.3 Sub-transfer Learning for Retuning the Outlier User Accuracy

The main contribution of this chapter is the analysis and significant boosting of the classification accuracies for the outlier subjects for HAR using a black-box deep learning model for motion-sensitive applications. We perform an in-depth analysis of two different learning representations while using a data augmentation technique to artificially increase the training data and provide an extensive comparison between the classification performances of these two training methods. For subject-based classification tasks, we clearly and statistically significantly demonstrate that our novel sub-transfer learning representation can enhance the classification accuracies of the state-of-the-art 1D-CNN classifier substantially especially for the outlier subjects only using a handful samples from the test data. Here, the knowledge transferred from a source model to the sub-transfer model helps to optimize the model parameters which results in higher classification accuracies. In real-world applications of HAR, a user can train the sub-transfer learning model using just 10-15 observations to achieve the best classification performance compared to the state-of-the-art model. On the other hand, subject-specific learning with no prior experience will provide a competitive performance for a classification task with inadequate resources. To explore the statistical significance between the performances of each learning representations, we have performed a non-parametric Wilcoxon signed-rank test which demonstrates that our proposed sub trans-

fer learning model using prior training knowledge provides the statistically significantly better classification performance compared to the other learning representations.

6.3.1 Contribution 3

We have introduced sub-transfer learning model to improve the practical performance by boosting the limited sample space of outlier and difficult subjects and sub-transferring from a source model to achieve unprecedented performance on human activity recognition with demonstrable statistical significance.

6.4 Future Work

Although this dissertation has made significant contributions to knowledge in how HAR applications benefit from different learning representations, exciting work remains in properly formulating several rules-of-thumb when it comes to choosing correct topologies and learning representations. Specifically, future work should look at the different data repositories with different sizes, number of classes and observations per class labels to identify the complex statistical relationship between the performance of the learning algorithm and the specifications of the dataset. Although our study reinforces a significant argument that the performance of deep learning algorithms can be improved by applying the data augmentation techniques specifically when the accuracy is lower than conventional approaches, future work can investigate other learning representations such as including the augmented samples in the training of the first source model for a more challenging comparison. Practically speaking, this is impossible for a real-world application since the developers do not have access to the user data prior to marketing the product. However, as more outlier users are added to the system, their data can potentially be used to further boost the performance of the algorithm with and without augmentation.

References

- [1] What is machine learning: Definition, types, applications and examples. Technical report.
- [2] Technologies for long-term care and home healthcare: Global markets. Technical Report HLC079D, November 2018.
- [3] Global ai in healthcare market: Analysis and forecast, 2021-2030. Technical Report 6028794, February 2021.
- [4] Maryam Abo-Tabik, Nicholas Costen, John Darby, and Yael Benn. Towards a smart smoking cessation app: a 1d-cnn model predicting smoking events. *Sensors*, 20(4):1099, 2020.
- [5] Sharif Abuadbba, Kyuyeon Kim, Minki Kim, Chandra Thapa, Seyit A Camtepe, Yansong Gao, Hyounghick Kim, and Surya Nepal. Can we use split learning on 1d cnn models for privacy preserving training? *arXiv preprint arXiv:2003.12365*, 2020.
- [6] Sizhe An, Ganapati Bhat, Suat Gumussoy, and Umit Ogras. Transfer learning for human activity recognition using representational analysis of neural networks. *arXiv preprint arXiv:2012.04479*, 2020.
- [7] Itamar Arel, Derek C Rose, and Thomas P Karnowski. Deep machine learning-a new frontier in artificial intelligence research [research frontier]. *IEEE computational intelligence magazine*, 5(4):13–18, 2010.

- [8] James Bartlett, Vinay Prabhu, and John Whaley. Acctionnet: A dataset of human activity recognition using on-phone motion sensors. In *Proceedings of the 34th International Conference on Machine Learning (Sydney, Australia, 2017)*, 2017.
- [9] Smaranda Belciug. *Artificial intelligence in cancer: Diagnostic to tailored treatment*. Academic Press, 2020.
- [10] JD Bermúdez, P Achanccaray, ID Sanches, L Cue, P Happ, and RQ Feitosa. Evaluation of recurrent neural networks for crop recognition from multitemporal remote sensing images. In *Anais do XXVII Congresso Brasileiro de Cartografia*, pages 800–804, 2017.
- [11] Daniel Berrar and Werner Dubitzky. *Deep learning in bioinformatics and biomedicine*, 2021.
- [12] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.
- [13] 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*, pages 796–799. Springer, 2009.
- [14] Jason Brownlee. *How to calculate nonparametric statistical hypothesis tests in python*, 2018.
- [15] Jason Brownlee. Understand the impact of learning rate on neural network performance. *Machine Learning Mastery*, 2019.
- [16] Vitaly Bushaev. Adam — latest trends in deep learning optimization. *Towards Data Science*, 2018.
- [17] Jacinto Carrasco, Salvador García, MM Rueda, Swagatam Das, and Francisco Herrera. Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: Practical guidelines and a critical review. *Swarm and Evolutionary Computation*, 54:100665, 2020.

- [18] Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014.
- [19] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.
- [20] Heeryon Cho and Sang Min Yoon. Divide and conquer-based 1d cnn human activity recognition using test data sharpening. *Sensors*, 18(4):1055, 2018.
- [21] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223, 2011.
- [22] L Minh Dang, Kyungbok Min, Hanxiang Wang, Md Jalil Piran, Cheol Hee Lee, and Hyeonjoon Moon. Sensor-based and vision-based human activity recognition: A comprehensive survey. *Pattern Recognition*, 108:107561, 2020.
- [23] Li Deng and Dong Yu. Deep learning: methods and applications. *Foundations and trends in signal processing*, 7(3–4):197–387, 2014.
- [24] Gulustan Dogan, Sinem Sena Ertas, and İremnaz Cay. Human activity recognition using convolutional neural networks. In *2021 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, pages 1–5. IEEE, 2021.
- [25] A Durango and C Refugio. An empirical study on wilcoxon signed rank test. *J. Negros Orient. State Univ.,(December)*, 2018.
- [26] Levent Eren, Turker Ince, and Serkan Kiranyaz. A generic intelligent bearing fault diagnosis system using compact adaptive 1d cnn classifier. *Journal of Signal Processing Systems*, 91(2):179–189, 2019.

- [27] Muhammad Farooq and Edward Sazonov. Feature extraction using deep learning for food type recognition. In *International Conference on Bioinformatics and Biomedical Engineering*, pages 464–472. Springer, 2017.
- [28] Alberto Fernández, Salvador Garcia, Francisco Herrera, and Nitesh V Chawla. Smote for learning from imbalanced data: progress and challenges, marking the 15-year anniversary. *Journal of artificial intelligence research*, 61:863–905, 2018.
- [29] Hongmin Gao, Shuo Lin, Yao Yang, Chenming Li, and Mingxiang Yang. Convolution neural network based on two-dimensional spectrum for hyperspectral image classification. *Journal of Sensors*, 2018, 2018.
- [30] Felix A Gers and Jürgen Schmidhuber. Recurrent nets that time and count. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, volume 3, pages 189–194. IEEE, 2000.
- [31] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.
- [32] Piyush Gupta and Tim Dallas. Feature selection and activity recognition system using a single triaxial accelerometer. *IEEE Transactions on Biomedical Engineering*, 61(6):1780–1786, 2014.
- [33] Sojeong Ha and Seungjin Choi. Convolutional neural networks for human activity recognition using multiple accelerometer and gyroscope sensors. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 381–388. IEEE, 2016.
- [34] Ali Haidar and Brijesh Verma. Monthly rainfall forecasting using one-dimensional deep convolutional neural network. *IEEE Access*, 6:69053–69063, 2018.

- [35] Nils Y Hammerla, Shane Halloran, and Thomas Plötz. Deep, convolutional, and recurrent models for human activity recognition using wearables. *arXiv preprint arXiv:1604.08880*, 2016.
- [36] Jeff Heaton. An empirical analysis of feature engineering for predictive modeling. In *SoutheastCon 2016*, pages 1–6. IEEE, 2016.
- [37] Vincent Hernandez, Tomoya Suzuki, and Gentiane Venture. Convolutional and recurrent neural network for human activity recognition: Application on american sign language. *PloS one*, 15(2):e0228869, 2020.
- [38] Yaoshiang Ho and Samuel Wookey. The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling. *IEEE Access*, 8:4806–4813, 2019.
- [39] Chaoqun Hong, Jun Yu, Dacheng Tao, and Meng Wang. Image-based three-dimensional human pose recovery by multiview locality-sensitive sparse retrieval. *IEEE Transactions on Industrial Electronics*, 62(6):3742–3751, 2014.
- [40] Chaoqun Hong, Jun Yu, Jian Wan, Dacheng Tao, and Meng Wang. Multimodal deep autoencoder for human pose recovery. *IEEE Transactions on Image Processing*, 24(12):5659–5670, 2015.
- [41] Shuzhan Huang, Jian Tang, Juying Dai, and Yangyang Wang. Signal status recognition based on 1dcnn and its feature extraction mechanism analysis. *Sensors*, 19(9):2018, 2019.
- [42] Faisal Hussain, Muhammad Basit Umair, Muhammad Ehatisham-ul Haq, Ivan Miguel Pires, Tânia Valente, Nuno M Garcia, and Nuno Pombo. An efficient machine learning-based elderly fall detection algorithm. *arXiv preprint arXiv:1911.11976*, 2019.

- [43] 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*, pages 159–163, 2005.
- [44] Chie Ikeda, Karim Ouazzane, Qicheng Yu, et al. A new framework of feature engineering for machine learning in financial fraud detection. 2020.
- [45] Stephen S Intille, Fahd Albinali, Selene Mota, Benjamin Kuris, Pilar Botana, and William L Haskell. Design of a wearable physical activity monitoring system using mobile phones and accelerometers. In *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 3636–3639. IEEE, 2011.
- [46] Lishuai Jiang, Yang Zhao, Naser Golsanami, Lianjun Chen, and Weichao Yan. A novel type of neural networks for feature engineering of geological data: case studies of coal and gas hydrate-bearing sediments. *Geoscience Frontiers*, 11(5):1511–1531, 2020.
- [47] Yang Jiang, Nigel Bosch, Ryan S Baker, Luc Paquette, Jaclyn Ocumpaugh, Juliana Ma Alexandra L Andres, Allison L Moore, and Gautam Biswas. Expert feature-engineering vs. deep neural networks: which is better for sensor-free affect detection? In *International Conference on Artificial Intelligence in Education*, pages 198–211. Springer, 2018.
- [48] Ria Kanjilal and Ismail Uysal. The future of human activity recognition: Deep learning or feature engineering? *Neural Processing Letters*, 53(1):561–579, 2021.
- [49] Ria Kanjilal and Ismail Uysal. Rich learning representations for human activity recognition: How to empower deep feature learning for biological time series. *Journal of Biomedical Informatics*, 2022 - Under Revision.
- [50] Jack Kelly. New survey shows that up to 47% of u.s. healthcare workers plan to leave their positions by 2025. Technical report, April 2022.

- [51] Udayan Khurana, Horst Samulowitz, and Deepak Turaga. Feature engineering for predictive modeling using reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [52] Ozsel Kilinc, Alexander Dalzell, Ismail Uluturk, and Ismail Uysal. Inertia based recognition of daily activities with anns and spectrotemporal features. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 733–738. IEEE, 2015.
- [53] Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, and Daniel J Inman. 1d convolutional neural networks and applications: A survey. *arXiv preprint arXiv:1905.03554*, 2019.
- [54] Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, and Daniel J Inman. 1d convolutional neural networks and applications: A survey. *Mechanical systems and signal processing*, 151:107398, 2021.
- [55] Muhammed Kucuk and Ismail Uysal. Performance analysis of neural network topologies and hyperparameters for deep clustering. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6, 2020.
- [56] Ilya Kuzovkin. Understanding information processing in human brain by interpreting machine learning models. *arXiv preprint arXiv:2010.08715*, 2020.
- [57] Jennifer R Kwapisz, Gary M Weiss, and Samuel A Moore. Activity recognition using cell phone accelerometers. *ACM SigKDD Explorations Newsletter*, 12(2):74–82, 2011.
- [58] Jeff Lagasse. Continuous monitoring tools could save hospitals \$20,000 per bed, report says. Technical report, August 2016.

- [59] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th annual international conference on machine learning*, pages 609–616, 2009.
- [60] Chih-Ting Liu, Yi-Heng Wu, Yu-Sheng Lin, and Shao-Yi Chien. Computation-performance optimization of convolutional neural networks with redundant kernel removal. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE, 2018.
- [61] Alexander Selvikvåg Lundervold and Arvid Lundervold. An overview of deep learning in medical imaging focusing on mri. *Zeitschrift für Medizinische Physik*, 29(2):102–127, 2019.
- [62] Sayf A Majeed, Hafizah Husain, Salina Abdul Samad, and Tariq F Idbeaa. Mel frequency cepstral coefficients (mfcc) feature extraction enhancement in the application of speech recognition: a comparison study. *Journal of Theoretical and Applied Information Technology*, 79(1):38, 2015.
- [63] Andrea Mannini, Stephen S Intille, Mary Rosenberger, Angelo M Sabatini, and William Haskell. Activity recognition using a single accelerometer placed at the wrist or ankle. *Medicine and science in sports and exercise*, 45(11):2193, 2013.
- [64] Andrea Mannini, Mary Rosenberger, William L Haskell, Angelo M Sabatini, and Stephen S Intille. Activity recognition in youth using single accelerometer placed at wrist or ankle. *Medicine and science in sports and exercise*, 49(4):801, 2017.
- [65] J McCarthy, M Minsky, N Rochester, and CL Shannon. The dartmouth summer research project on artificial intelligence. *Artificial intelligence: past, present, and future*, 1956.

- [66] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Bat-tenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, volume 8, 2015.
- [67] Richard Meyes, Johanna Donauer, Andre Schmeing, and Tobias Meisen. A recurrent neural network architecture for failure prediction in deep drawing sensory time series data. *Procedia Manufacturing*, 34:789–797, 2019.
- [68] Daniela Micucci, Marco Mobilio, and Paolo Napoletano. Unimib shar: A dataset for human activity recognition using acceleration data from smartphones. *Applied Sciences*, 7(10):1101, 2017.
- [69] Shubham Mittal and Yasha Hasija. Applications of deep learning in healthcare and biomedicine. In *Deep Learning Techniques for Biomedical and Health Informatics*, pages 57–77. Springer, 2020.
- [70] Abdulmajid Murad and Jae-Young Pyun. Deep recurrent neural networks for human activity recognition. *Sensors*, 17(11):2556, 2017.
- [71] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.
- [72] Ronald Mutegeki and Dong Seog Han. A cnn-lstm approach to human activity recog-nition. In *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIIC)*, pages 362–366. IEEE, 2020.
- [73] Godwin Ogbuabor and Robert La. Human activity recognition for healthcare using smartphones. In *Proceedings of the 2018 10th international conference on machine learning and computing*, pages 41–46, 2018.
- [74] Jangsoo Park, Jongseok Lee, and Donggyu Sim. Low-complexity cnn with 1d and 2d filters for super-resolution. *Journal of Real-Time Image Processing*, pages 1–12, 2020.

- [75] Schalk Wilhelm Pienaar and Reza Malekian. Human activity recognition using lstm-rnn deep neural network architecture. In *2019 IEEE 2nd Wireless Africa Conference (WAC)*, pages 1–5. IEEE, 2019.
- [76] Syed Jahanzeb Hussain Pirzada and Ayesha Siddiqui. Analysis of edge detection algorithms for feature extraction in satellite images. In *2013 IEEE International Conference on Space Science and Communication (IconSpace)*, pages 238–242. IEEE, 2013.
- [77] Mikaela Pisani. Machine learning – chapter 1 – introduction. Technical report, 2011.
- [78] Yinghui Quan, Xian Zhong, Wei Feng, Jonathan Cheung-Wai Chan, Qiang Li, and Mengdao Xing. Smote-based weighted deep rotation forest for the imbalanced hyper-spectral data classification. *Remote Sensing*, 13(3):464, 2021.
- [79] Sebastian Raschka. Chapter 1: Introduction to machine learning and deep learning, 2020.
- [80] Kenneth D Roe, Vibhu Jawa, Xiaohan Zhang, Christopher G Chute, Jeremy A Epstein, Jordan Matelsky, Ilya Shpitser, and Casey Overby Taylor. Feature engineering with clinical expert knowledge: A case study assessment of machine learning model complexity and performance. *PloS one*, 15(4):e0231300, 2020.
- [81] Alireza Roshanzamir, Hamid Aghajan, and Mahdieh Soleymani Baghshah. Transformer-based deep neural network language models for alzheimer’s disease risk assessment from targeted speech. *BMC Medical Informatics and Decision Making*, 21(1):1–14, 2021.
- [82] William R Rowley, Clement Bezold, Yasemin Arikan, Erin Byrne, and Shannon Krohe. Diabetes 2030: insights from yesterday, today, and future trends. *Population health management*, 20(1):6–12, 2017.

- [83] Furqan Rustam, Aijaz Ahmad Reshi, Imran Ashraf, Arif Mehmood, Saleem Ullah, Dost Muhammad Khan, and Gyu Sang Choi. Sensor-based human activity recognition using deep stacked multilayered perceptron model. *IEEE Access*, 8:218898–218910, 2020.
- [84] Ruslan Salakhutdinov and Hugo Larochelle. Efficient learning of deep boltzmann machines. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 693–700, 2010.
- [85] Arthur L Samuel. Some studies in machine learning using the game of checkers. ii—recent progress. *IBM Journal of research and development*, 11(6):601–617, 1967.
- [86] Guto Leoni Santos, Patricia Takako Endo, Kayo Henrique de Carvalho Monteiro, Elisson da Silva Rocha, Ivanovitch Silva, and Theo Lynn. Accelerometer-based human fall detection using convolutional neural networks. *Sensors*, 19(7):1644, 2019.
- [87] Allah Bux Sargano, Xiaofeng Wang, Plamen Angelov, and Zulfiqar Habib. Human action recognition using transfer learning with deep representations. In *2017 International joint conference on neural networks (IJCNN)*, pages 463–469. IEEE, 2017.
- [88] Jonathan Schmidt, Mário RG Marques, Silvana Botti, and Miguel AL Marques. Recent advances and applications of machine learning in solid-state materials science. *npj Computational Materials*, 5(1):1–36, 2019.
- [89] Fatemeh Serpush, Mohammad Bagher Menhaj, Behrooz Masoumi, and Babak Karasfi. Wearable sensor-based human activity recognition in the smart healthcare system. *Computational Intelligence and Neuroscience*, 2022, 2022.
- [90] Muhammad Hameed Siddiqi, Rahman Ali, Md Rana, Een-Kee Hong, Eun Soo Kim, Sungyoung Lee, et al. Video-based human activity recognition using multilevel wavelet decomposition and stepwise linear discriminant analysis. *Sensors*, 14(4):6370–6392, 2014.

- [91] Ahsan Raza Siyal, Zuhaibuddin Bhutto, Syed Muhammad Shehram Shah, Azhar Iqbal, Faraz Mehmood, Ayaz Hussain, and Saleem Ahmed. Still image-based human activity recognition with deep representations and residual learning. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 11(5):471–477, 2020.
- [92] Lin Sun, Daqing Zhang, Bin Li, Bin Guo, and Shijian Li. Activity recognition on an accelerometer embedded mobile phone with varying positions and orientations. In *International conference on ubiquitous intelligence and computing*, pages 548–562. Springer, 2010.
- [93] Muhammad Syafrudin, Ganjar Alfian, Norma Latif Fitriyani, and Jongtae Rhee. Performance analysis of iot-based sensor, big data processing, and machine learning model for real-time monitoring system in automotive manufacturing. *Sensors*, 18(9):2946, 2018.
- [94] Wensi Tang, Guodong Long, Lu Liu, Tianyi Zhou, Jing Jiang, and Michael Blumenstein. Rethinking 1d-cnn for time series classification: A stronger baseline. *arXiv preprint arXiv:2002.10061*, 2020.
- [95] Amirsina Torfi, Rouzbeh A Shirvani, Yaser Keneshloo, Nader Tavvaf, and Edward A Fox. Natural language processing advancements by deep learning: A survey. *arXiv preprint arXiv:2003.01200*, 2020.
- [96] Farman Ullah, Asif Iqbal, Ajmal Khan, Rida Gul Khan, Laraib Malik, and Kyung Sup Kwak. An image-based human physical activities recognition in an indoor environment. In *2020 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 588–593. IEEE, 2020.
- [97] Yonatan Vaizman, Katherine Ellis, and Gert Lanckriet. Recognizing detailed human context in the wild from smartphones and smartwatches. *IEEE pervasive computing*, 16(4):62–74, 2017.

- [98] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 1999.
- [99] Alireza Abedin Varamin, Ehsan Abbasnejad, Qinfeng Shi, Damith C Ranasinghe, and Hamid Reza Tofighi. Deep auto-set: A deep auto-encoder-set network for activity recognition using wearables. In *Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pages 246–253, 2018.
- [100] Vijay Kumar Verma, Wen-Yen Lin, Ming-Yih Lee, and Chao-Sung Lai. Levels of activity identification & sleep duration detection with a wrist-worn accelerometer-based device. In *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 2369–2372. IEEE, 2017.
- [101] Abdul Wahab, Hilal Tayara, Zhenyu Xuan, and Kil To Chong. Dna sequences performs as natural language processing by exploiting deep learning algorithm for the identification of n4-methylcytosine. *Scientific reports*, 11(1):1–9, 2021.
- [102] Huan Wang, Zhiliang Liu, Dandan Peng, and Yong Qin. Understanding and learning discriminant features based on multiattention 1dcnn for wheelset bearing fault diagnosis. *IEEE Transactions on Industrial Informatics*, 16(9):5735–5745, 2019.
- [103] Zhiping Wang, Cundong Tang, Xiuxiu Sima, and Lingxiao Zhang. Research on application of deep learning algorithm in image classification. In *2021 IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC)*, pages 1122–1125. IEEE, 2021.
- [104] Zachary J Ward, Sara N Bleich, Angie L Cradock, Jessica L Barrett, Catherine M Giles, Chasmine Flax, Michael W Long, and Steven L Gortmaker. Projected us state-level prevalence of adult obesity and severe obesity. *New England Journal of Medicine*, 381(25):2440–2450, 2019.

- [105] Przemyslaw Woznowski, Rachel King, William Harwin, and Ian Craddock. A human activity recognition framework for healthcare applications: Ontology, labelling strategies, and best practice. In *IoTBD*, pages 369–377, 2016.
- [106] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. Convolutional neural networks: an overview and application in radiology. *Insights into imaging*, 9(4):611–629, 2018.
- [107] Rui Yao, Guosheng Lin, Qinfeng Shi, and Damith C Ranasinghe. Efficient dense labelling of human activity sequences from wearables using fully convolutional networks. *Pattern Recognition*, 78:252–266, 2018.
- [108] Jun Yu, Min Tan, Hongyuan Zhang, Dacheng Tao, and Yong Rui. Hierarchical deep click feature prediction for fine-grained image recognition. *IEEE transactions on pattern analysis and machine intelligence*, 2019.
- [109] Lei Yu and Huan Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 856–863, 2003.
- [110] Bin Zeng, Ivan Sanz-Prieto, and Ashish Kr Luhach. Deep learning approach to automated data collection and processing of video surveillance in sports activity prediction. *Annals of Operations Research*, pages 1–20, 2021.
- [111] Jian Zhang, Jun Yu, and Dacheng Tao. Local deep-feature alignment for unsupervised dimension reduction. *IEEE transactions on image processing*, 27(5):2420–2432, 2018.
- [112] Mi Zhang and Alexander A Sawchuk. Motion primitive-based human activity recognition using a bag-of-features approach. In *Proceedings of the 2nd ACM SIGHT International Health Informatics Symposium*, pages 631–640, 2012.

- [113] Yong Zhang, Yu Zhang, Zhao Zhang, Jie Bao, and Yunpeng Song. Human activity recognition based on time series analysis using u-net. *arXiv preprint arXiv:1809.08113*, 2018.
- [114] Kai Zhao. Feature extraction using principal component analysis — a simplified visual demo. Technical report, 2019.
- [115] Naifan Zhuang, Guo-Jun Qi, The Duc Kieu, and Kien A Hua. Differential recurrent neural network and its application for human activity recognition. *arXiv preprint arXiv:1905.04293*, 2019.

Appendix A: Copyright Permissions

The copyright permission below is for the reproduction of material in Chapter 1, Chapter 3 and Chapter 6 [48]



Permissions

Get permission to reuse Springer Nature content

Springer Nature is partnered with the Copyright Clearance Center to meet our customers' licensing and permissions needs.

Copyright Clearance Center's RightsLink® service makes it faster and easier to secure permission for the reuse of Springer Nature content to be published, for example, in a journal/magazine, book/textbook, coursepack, thesis/dissertation, annual report, newspaper, training materials, presentation/slide kit, promotional material, etc.

Simply visit [SpringerLink](#) and locate the desired content;

Go to the article or chapter page you wish to reuse content from. (Note: permissions are granted on the article or chapter level, not on the book or journal level). Scroll to the bottom of the page, or locate via the side bar, the "Reprints and Permissions" link at the end of the chapter or article.

Select the way you would like to reuse the content;

Complete the form with details on your intended reuse. Please be as complete and specific as possible so as not to delay your permission request;

Create an account if you haven't already. A RightsLink account is different than a SpringerLink account, and is necessary to receive a licence regardless of the permission fee. You will receive your licence via the email attached to your RightsLink receipt;

Accept the terms and conditions and you're done!

For questions about using the RightsLink service, please contact Customer Support at Copyright Clearance Center via phone +1-855-239-3415 or +1-978-646-2777 or email springernaturesupport@copyright.com.

How to obtain permission to reuse Springer Nature content not available online on SpringerLink

Requests for permission to reuse content (e.g. figure or table, abstract, text excerpts) from Springer Nature publications currently not available online must be submitted in writing. Please be as detailed and specific as possible about what, where, how much, and why you wish to reuse the content.

Your contacts to obtain permission for the reuse of material from:

- books: bookpermissions@springernature.com
- journals: journalpermissions@springernature.com

Author reuse

Please check the Copyright Transfer Statement (CTS) or Licence to Publish (LTP) that you have signed with Springer Nature to find further information about the reuse of your content.

Authors have the right to reuse their article's Version of Record, in whole or in part, in their own thesis. Additionally, they may reproduce and make available their thesis, including Springer Nature content, as required by their awarding academic institution. Authors must properly cite the published article in their thesis according to current citation standards.

Material from: 'AUTHOR, TITLE, JOURNAL TITLE, published [YEAR], [publisher - as it appears on our copyright page]'

If you are any doubt about whether your intended re-use is covered, please contact journalpermissions@springernature.com for confirmation.

Self-Archiving

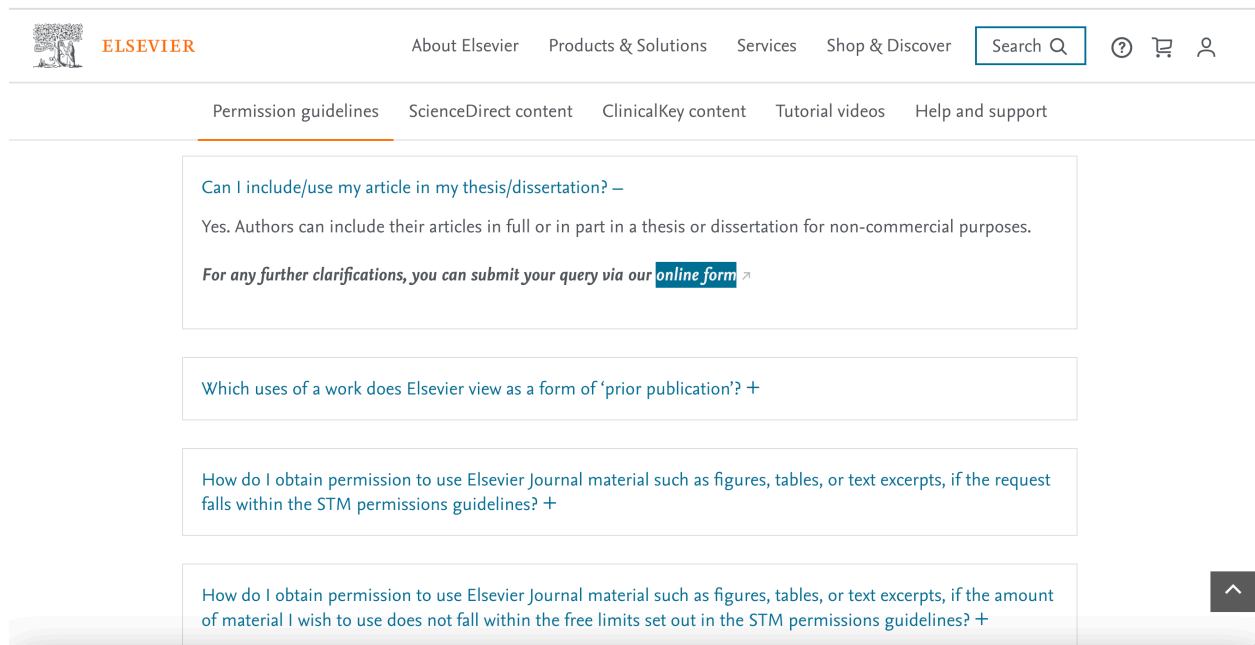
- Journal authors retain the right to self-archive the final accepted version of their manuscript. Please see our self-archiving policy for full details:

<https://www.springer.com/gp/open-access/authors-rights/self-archiving-policy/2124>

- Book authors please refer to the information on this link:

<https://www.springer.com/gp/open-access/publication-policies/self-archiving-policy>

The copyright permission below is for the reproduction of material in Chapter 1, Chapter 4 and Chapter 6 [49].



The screenshot shows the Elsevier website's navigation and content area. At the top left is the Elsevier logo. To its right are links for 'About Elsevier', 'Products & Solutions', 'Services', and 'Shop & Discover'. A search bar with a magnifying glass icon is on the right. Below these are icons for help, shopping cart, and user profile. A secondary navigation bar contains 'Permission guidelines' (underlined), 'ScienceDirect content', 'ClinicalKey content', 'Tutorial videos', and 'Help and support'. The main content area features four expandable FAQ items:

- Can I include/use my article in my thesis/dissertation? –**
Yes. Authors can include their articles in full or in part in a thesis or dissertation for non-commercial purposes.
For any further clarifications, you can submit your query via our [online form](#)
- Which uses of a work does Elsevier view as a form of 'prior publication'? +**
- How do I obtain permission to use Elsevier Journal material such as figures, tables, or text excerpts, if the request falls within the STM permissions guidelines? +**
- How do I obtain permission to use Elsevier Journal material such as figures, tables, or text excerpts, if the amount of material I wish to use does not fall within the free limits set out in the STM permissions guidelines? +**

A dark grey button with a white upward-pointing arrow is located in the bottom right corner of the content area.

About the Author

Ria Kanjilal is a Ph.D. candidate in Electrical Engineering at the University of South Florida (USF), FL, USA. Since 2019, she has been a Research Assistant with the USF, Florida, USA. She received her B.S and M.S. degrees in Electronics and Communication Engineering from West Bengal University of Technology, West Bengal, India in 2009 and 2014. Her research interests include supervised and unsupervised feature learning using deep learning algorithms, specifically the applications of machine learning in healthcare, food supply chains and computational bioinformatics.