Spring 5-20-2019

Human Activity Recognition Based on Multimodal Body Sensing

Anish Hemant Narkhede
San Jose State University

Human Activity Recognition Based on Multimodal Body Sensing

A Thesis

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

Of the Requirements for the Class

CS 298

By

Anish Hemant Narkhede

May 2019

The Designated Thesis Committee Approves the Thesis Titled

Human Activity Recognition based on Multimodal Body Sensing

by

Anish Hemant Narkhede

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

May 2019

Dr. Robert Chun, Department of Computer Science

Dr. Suneuy Kim, Department of Computer Science

Mr. Ashwin Vijaysai, Lumentum LLC.

ABSTRACT

In the recent years, human activity recognition has been widely popularized by a lot of smartphone manufacturers and fitness tracking companies. It has allowed us to gain a deeper insight into our physical health on a daily basis. However, with the evolution of fitness tracking devices and smartphones, the amount of data that is being captured by these devices is growing exponentially. This paper aims at understanding the process of dimensionality reduction such as PCA so that the data can be used to make meaningful predictions along with novel techniques using autoencoders with different activation functions. The paper also looks into how using autoencoders allows us to better capture the relations between features in the data. It also covers some of the classification techniques such as k-Nearest Neighbors, SVM and Random forest that are currently being used for activity recognition that have shown promising results.

***Keywords* – Classification, dimensionality reduction, neural networks, human activity recognition.**

ACKNOWLEDGEMENTS

I would like to thank Dr. Robert Chun for his continued support during the course of this project and for providing me with the necessary guidance for its successful completion. His suggestions and inputs during the course of this project have been crucial in its successful completion. I would also like to thank my committee members Dr. Suneuy Kim and Mr. Ashwin Vijaysai for their time and valuable inputs during the course of the project. Last but not least, I would like to express my gratitude towards my family for their constant support.

TABLE OF CONTENTS

LIST OF TABLES AND FIGURES

I. INTRODUCTION

Human activity recognition (HAR) is an important research topic in the healthcare domain. It aims at the identification of different activities ranging from simple to complex, which are usually detected by wearable devices having sensors (e.g. smartwatches, fitness trackers, smartphones). It serves as an extremely useful application in the field of health monitoring and development of human-machine interfaces [1].

Classical approaches towards tackling this problem included manually extracting features using time series data and training classification models using decision trees. This was a rather rudimentary approach towards solving an ever-evolving problem with the rapid evolution of data capturing devices in recent years. These devices can be used to collect vital data which can then be used to analyze and correctly classify each activity into a particular category. Some of the basic categories for HAR include lying, standing, climbing, walking and running [2]. Some of the important applications of HAR in healthcare are keeping track of the body vitals of patients in rehabilitation centers, people living in assisted living facilities and people that are diagnosed with chronic diseases [3].

Classifying human activities into categories with high-dimensional data is a challenging task. The devices used to monitor human activity measure vital information about a person using multiple sensors, thus generating high-dimensional features. A potential issue with having high dimensional data is over-fitting of data. This leads to a well-known issue commonly referred to as the curse of dimensionality [4]. As humans, it is not possible for us to visualize data in more than 3 dimensions. These device sensors collect data that has over 20 dimensions. Thus, dimensionality reduction helps us to compress this high dimensional data into lower dimensions

(2-D) while retaining important information from all 20 dimensions. A possible solution to this problem is reducing the number of dimensions to a manageable number such that it can be projected onto a 2-D plane. This would allow us to make sense of the data. This can be done using PCA, but since it is an unsupervised method, it fails to take into account the labelled data. In this project, we explore the potential challenges in the field of dimensionality reduction and its importance in successfully classifying different human activities.

In this project, we try to provide some insight on how human activity data can be used in order to make predictions for the future, how this data is collected and used. We also aim to answer the question of whether we can use autoencoders with different activation functions to perform better dimensionality reduction and capture non-linear relations between features that will help us better classify human activities.

II. ACTIVITY RECOGNITION USING MULTIMODAL SENSORS

Activity recognition using multimodal sensors involves identification of which activity is being performed at a particular moment based off of the data collected by multiple sensors fitted on the human body. Some of the processes in activity recognition include:

A. Data Acquisition
B. Data Preprocessing
C. Dimensionality Reduction
D. Training and Classification

A. Data Acquisition

Data acquisition is a critical part in HAR. The objective of HAR is analysis or interpretation of the data that is being tracked so as to derive insight from it. The study by Kumari et al. [5] suggests that acquiring data involves signal preprocessing which focuses on filtering and extracting the important features that will be required in order to train classifier. These features are extracted by placing sensors at predetermined places on the human body. The data that these sensors collect are then directed to two different systems, an online system that does the task of classifying the action immediately and an offline system which does not classify data instantly, typically used for applications that do not require immediate feedback and demand high computation.

A slightly different approach is suggested by Sebestyen et al. [6] which emphasizes on the use of smartphones and their capabilities of capturing data with the help of acceleration and localization sensors. These sensors measure acceleration in 3 directions which is useful in the state or type of activity that a person is performing. According to Sebestyen et al. [6] acceleration is an important feature that needs to be captured as it lets us determine whether a person is sitting or standing. It can also be used to determine whether the phone is in its usual place (pocket) and also the activity (walking, running or staying still) that the holder is doing.

An important factor that affects the quality of data that is captured is the sampling rate. The sampling rate is the frequency with which different sensors capture data. Thus, while a higher sampling rate increases the quality of measurement, it also causes the sensors to get overloaded with the activity data and consequently increases power consumption. Placement of sensors has been discussed frequently and how it affects the data that is captured. A study by He et al. [7] suggests that the most effective information is captured using sensors that are placed inside a person's trouser pocket. On the other hand, other studies suggest that the sensors placed on a person's arms and legs can be used to accurately predict the activity [8].

Efficient data acquisition is critical when it comes to mobile devices as they have a limited energy source. A potential solution to this problem is to use a technique called windowing, in which the sensors capture data only during a small timeframe. The length of each window is decided in such a way that it can accurately determine each activity and at the same time enable energy preservation.

B. Data Preprocessing

Often the data that is obtained from the real-world can be inconsistent or even incomplete. There is also a possibility of having errors in the data-types of the obtained data. Hence it is necessary to clean and preprocess data before it can be used to make predictions. One of the most commonly occurring error is that of missing values. There might be times where a particular sensor fails to register reading for a particular part of the human body.

There are times when multiple sensors might fail to register the readings as well. One of the common ways to deal with missing values is to drop a particular row that has a missing reading for a feature. However, dropping a row means loss of data. It is possible that there are hundreds of rows that contain missing values. Dropping all of these rows will cause us to lose important feature information that can otherwise be retained in order to train our models for better prediction. Another way of dealing with missing values is to calculate an arithmetic mean, median or mode of the missing feature and replace it with the missing value.

C. Dimensionality Reduction

Sensors used to record human activity data is usually high in dimension. In order to train classifiers for recognizing human activity, it is necessary to transform the data in such a way that it reduces the dimensionality while still retaining useful and important information necessary for HAR. Dimensionality reduction aims at reducing the random variables that are under consideration [9]. Data is collected by multiple sensors such as accelerometer, gyroscope and magnetometer which collect data for 3 different axes (x-axis, y-axis, z-axis). As a result of this,

the number of features that are collected is way too large to be used for training any machine learning classifier.

Even though this data can be used for training a classifier, it would most likely cause over-fitting of data. This leads to a well-known issue commonly referred to as the curse of dimensionality [4]. A study by Moudden et al. [9] demonstrates that the different features that are captured by sensors are of little interest and only a subset of the entire feature set contributes to the features that are worthy of interest. In order to obtain an effective and robust characterization of the domain, dimensionality reduction can prove to be a useful tool [10]. In order to make useful predictions regarding what activity is being done, the aim is to minimize the number of predictors and also make sure that these predictors are independent of each other. Xi et al. [11] also found that using only the most informative features that capture the maximum variance in data contribute towards higher accuracy of the model.

D. Training and Classification

In order to train a classifier that can predict a possible outcome, it needs to be trained using a supervised inference technique such that the classifier learns model parameters so that the classification error is minimized [11]. Once a classifier is trained on the training data, we can then use it in order to make predictions on unseen data. This is typically done by splitting the entire dataset into training and testing data. In this case, the testing data is used in order to validate the accuracy of the predictions made by our classifier. Classification can simply be defined as a task of identifying the class for a given set of data points. Classes are also commonly referred to as targets or labels. It is the task of learning a mapping function f(x) with the help of input variable x in order to predict the output variable y. In case of HAR, the input variables

would correspond to the various sensors (accelerometer, gyroscope, magnetometer) that are mounted on various parts of the body such as the chest, left-ankle and right-lower arm. Some of the commonly used algorithms for classification are decision trees, k-nearest neighbors [1], support vector machines [7], random forests [1].

III. DIMENSIONALITY REDUCTION APPROACHES

Dimensionality Reduction

Due to ease of availability of sensors in almost all electronic devices, the data that captured tends to be pretty high in dimensions and also more complex than conventional data [12]. Using this high dimensional data will result into noise and also the possibility of non-related entities of data. Dimensionality reduction is thus used in order to avoid the curse of dimensionality [4], so that captured data can be used to make sense of and be used to derive insights from it. The curse of dimensionality refers to the formation of a machine learning model that over-fits the data. As a result, such a model is incapable of generalizing well with unseen or new data. Once the dimensionality of a dataset has been reduced, this data can then be used to train machine learning classifiers.

A. Principal Component Analysis

Principal component analysis (PCA) is a technique that is used to transform a number of different and possible uncorrelated variables into a smaller set of variables that are uncorrelated [12]. It is a technique that uses variance in order to determine whether a certain feature is of interest to us or not, with the aim of finding vectors in the feature space that capture the maximum variance in the data.

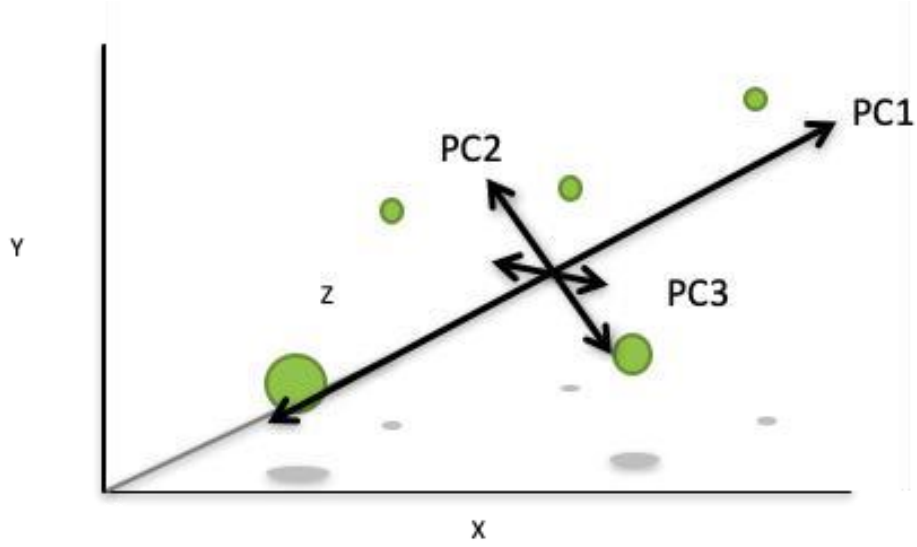


Fig 1. Principal component analysis [12]

PCA starts with standardization of the matrix. This is done because the values in certain columns might have values that are higher than the values in other attributes. This might cause such attributes to dominate the entire principal component matrix [13]. In order to achieve such standardization, by subtracting the mean of each column value from each of the attribute column and then dividing it by their standard deviation. Apart from this, standardization has additional benefits that allow for faster learning of neural networks. This can be extremely beneficial while training neural networks with huge amounts of data.

Next step in PCA is to obtain the covariance matrix from the standardized matrix. This is achieved by multiplying the standardized matrix (X) with its transform (X^T) and normalizing it by 1/n-1 where n is the total number of features that present in the standardized matrix. Following the covariance matrix calculation, the eigenvectors of the covariance matrix is calculated. This results in an eigenvector matrix which contains columns in descending order which correspond to the first principal component, the second principal component and so on.

Obtaining the PCA of a particular dataset, allows us to capture the maximum variability in the data without any loss of information [14]. PCA transformation is also a pretty convenient way of achieving dimensionality reduction since it extracts all of the meaningful feature information without us having to provide any additional information regarding the data source or domain knowledge regarding the problem that we are trying to solve [15].

However, PCA can also be restrained by its limitation of being able to capture only linear transformations. As a result, if a particular dataset consists of features that have non-linear relations, then those relations will not be captured using PCA.

B. Autoencoders

Autoencoders (AEs) are essentially Artificial Neural Networks that have a symmetric structure. They are also known as associative neural networks [16], replicator neural networks [17] or diabolo networks [18]. The AEs consist of an input layer, a middle layer and an output layer. The middle layer in the AE is responsible for holding the encoding from the input layer [19]. The data that enters the AE from the input layer is encoded and the AE is then trained to

reconstruct the input data from this encoding. However, this reconstruction process is restrained with certain restrictions so that the data is not simply copied across the neural network.

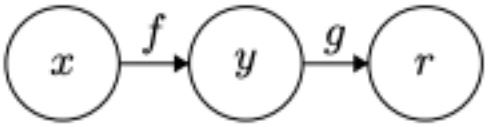


Fig 2: Autoencoder structure [19]

As shown in Fig. 1, we can see the basic structure of an autoencoder which consists of the input data which is represented by *x*, which maps to the encoding represented by *y*. The encoding *y* is then mapped to the output of autoencoder which is represented by *r*, where *f* is the encoding function and *g* is the decoding function.

The general structure of an autoencoder can also be captured in a feedforward neural network. The main objective of an autoencoder is to reproduce or replicate the original input data. As a result of which, the AE *x* and *r* are equal in the number of dimensions. The middle layer however, can be low-dimensional or high-dimensional depending on the objective. The layers within the AE are arranged in a symmetrical fashion on either side of the middle layer.

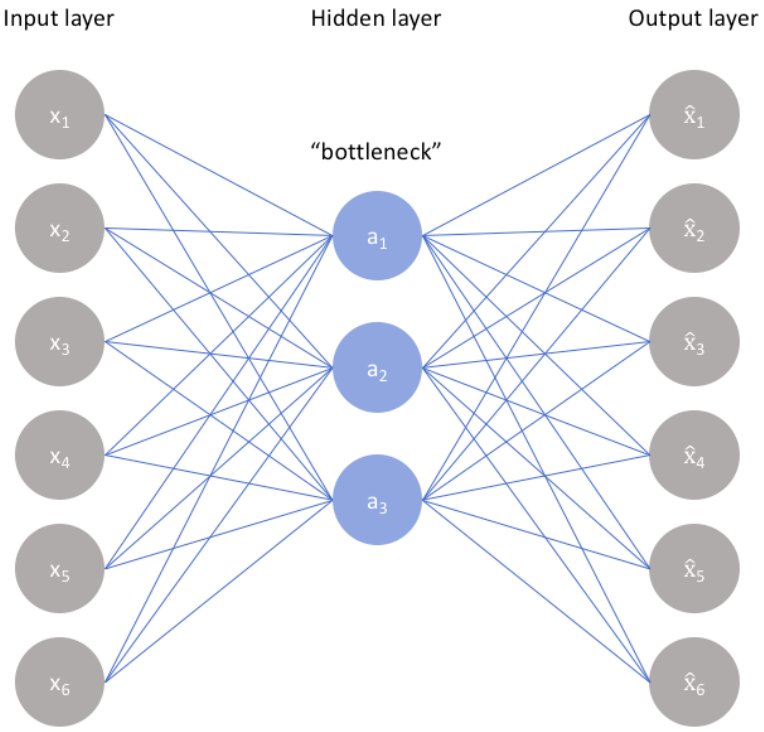


Fig 3: Neural network architecture for autoencoder [20]

Since autoencoders are capable of reconstructing the data that they are fed, they have a very obvious application in the task of data compression. Although AEs are capable of encoding data and then reconstructing the input data, they provide us with an approximate reconstruction of the input data. Thus, the data compression that AEs provide is lossy. Hence, AEs should only be used in cases where lossy data compression is tolerable. AEs were used with images, however since lossy reconstruction of images is not acceptable, standards such as JPEG [22] are more popular in image compression. Del Testa et al. [23] suggest that the use of autoencoders has been successful in the case of biometric data compression (blood pressure or heart rate data), which was obtained using wearables. According to Miao and Blunsom, language compression is another application where autoencoders have shown promise, that

allow us to draw sentences from a AE modelled on a particular language. Hsu [24] suggested that in the case of working with high-dimensional time series containing data related to electricity and water readings that were obtained from service grids, using autoencoders along with LSTM proved to be successful in compression.

a) Autoencoders with Linear Activation

Activation functions are biologically inspired by the human brain, where different neurons in our brain get activated by a certain stimulus. Activation functions are functions that are used in order to calculate the weighted sum of the inputs that the autoencoder receives, to which it adds a bias. This newly generated value is then used in order to decide whether a particular neuron will contribute towards the final solution (prediction). A neuron that is part of an artificial neural network is responsible for holding a value using which it can be decided whether or not the neuron should be activated. A neuron typically calculates the weighted sum of the inputs it receives from the source of the previous layer in the neural network. It then adds a bias to the value to decide whether the neuron should fire or not.

Linear function is a straight-line function which is effectively proportional to its input. The value of *Y* can be anything between negative infinity to positive infinity. The neuron is unaware of the bounds and also does not have an idea as to when it should be activated. Hence, we used activation functions for the purpose of helping a neuron decide whether it should activate or not.

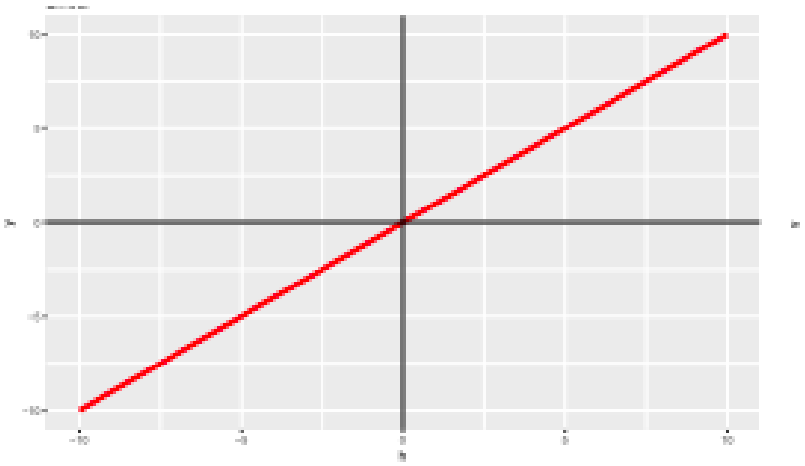


Fig 4: A linear activation function [19]

b) Autoencoders with Sigmoid Activation

The sigmoid activation function is another function that is commonly used in neural networks as an activation function. The sigmoid function is also commonly used for binary classification purposes. Thus, it performs well for indicating whether a particular neuron should be activated or not (a binary decision).

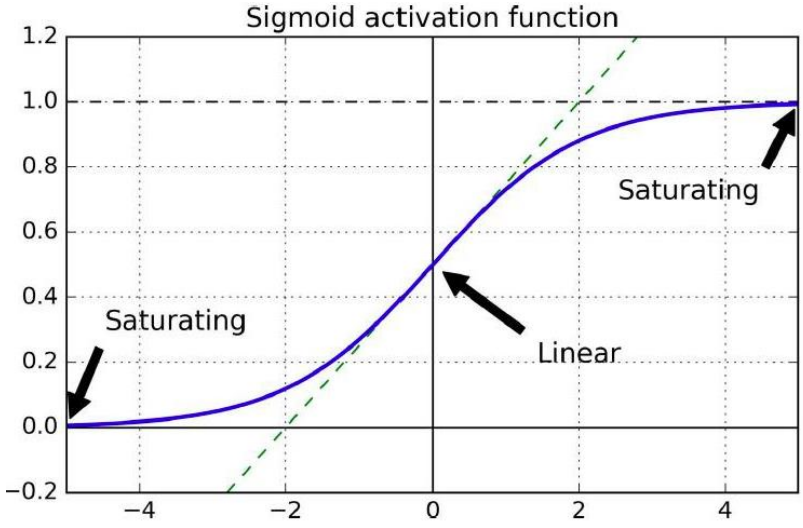


Fig 5: A sigmoid activation function [25]

Mathematically speaking, the sigmoid function gets as input a real-valued number from the source. This number is then squashed between the range of 0 and 1. As a result of this, very large positive numbers are become 1 and very large negative numbers become 0. As a result, all the possible values that the function receives as input, are converted to values within the range of 0 and 1. This allows the sigmoid function to have a good interpretation for firing rate of a neuron, i.e. the neuron fires if that output value is 1 and does not fire if the value is 0. However, an undesirable property of the sigmoid function is that of the gradients dying off due to saturation. As a result, during backpropagation if the gradient is very low, then there is a possibility that the neurons become saturated and the network does not learn.

The Vanishing Gradient Problem

Vanishing gradient is a problem that arises when the gradient of a particular loss function approaches zero. As a result of which no value propagates through the network causing no learning. As a result, it becomes hard to train a neural network [26].

During backpropagation, we calculate the derivative of a value in order to minimize the loss function. The sigmoid function when used, squishes any large or small value within the range of 0 and 1. As a result, for a particularly large or small value the derivative becomes close to zero. Thus, the value propagated back to the initial layers is very small causing no learning as weights and biases will not get updated during every epoch [27].

c) Autoencoders with Rectified Linear Unit

ReLU or Rectified linear unit is an activation function used in neural networks which became quite popular in recent years. It also gives a better performance since there is no complex math involved in the function, as a result of which it takes less time to train data [28].

When compared with the previously mentioned activation functions that include expensive operations such as the calculation of exponentials, ReLU was found to accelerate Stochastic Gradient Descent (SGD) and also train much faster, which might be because of its linear and non-saturating property [28]. ReLUs have been used a lot in activation functions for deep Convolutional Neural Networks (CNNs). One of the reasons that ReLU work well is due to the fact they perform well even with sparse data [29]. ReLUs have been used widely in Convolutional Neural Networks (CNNs) and have proved to have shown better performance during learning [30].

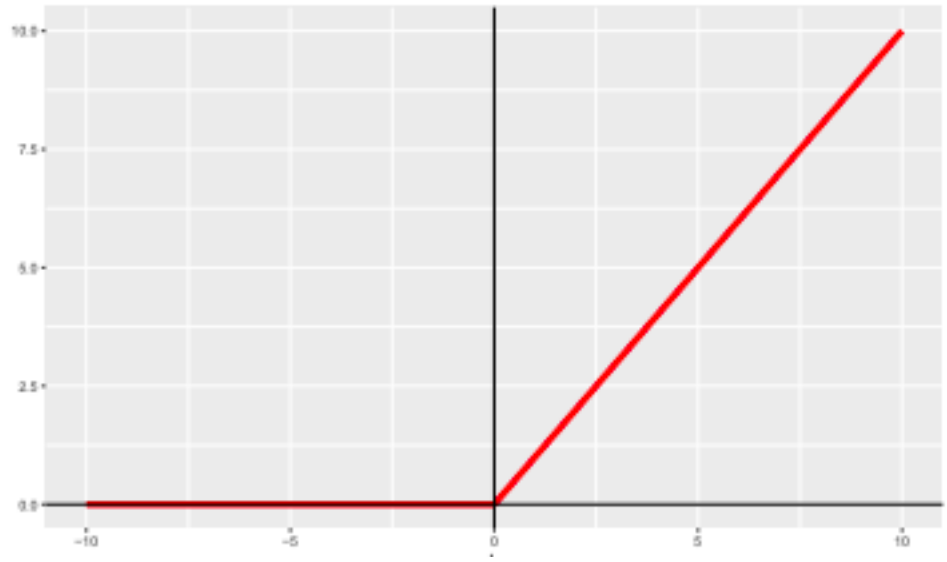


Fig 6: ReLU activation function [19]

IV. CLASSIFICATION TECHNIQUES

A. Support Vector Machine

Support Vector Machine (SVM) is a discriminative classifier that classifies by separating data with the help of a hyper-plane. In other words, given an input of labelled training data, the algorithm outputs an optimal hyper-plane that categorizes all of the input into different classes. It does so by finding an optimal boundary between different classes of data [31]. The separation should not pass too close to any data point on either side so as to generalize the classification better. The separation is based on the maximization of distance between separating classes.

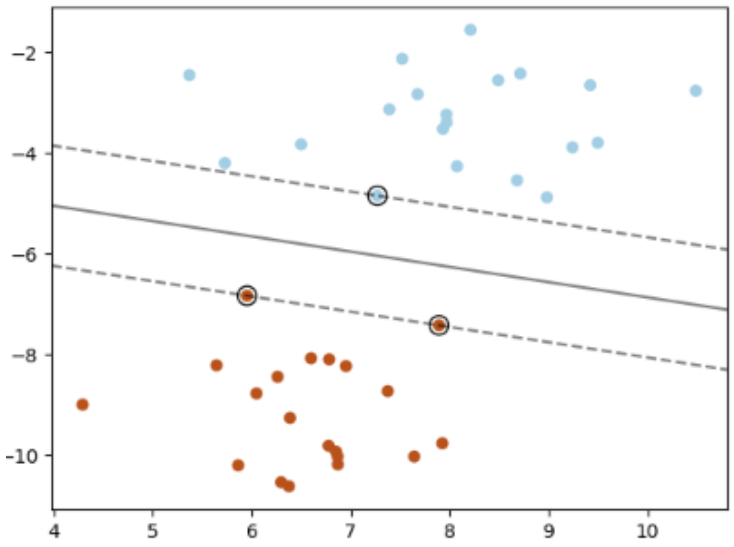


Fig 7: Support Vector Machine [33]

B. Random Forest Classifier

Random forest classifier (RFC) is a combination or ensemble of different decision tree predictors wherein the output of each predictor depends on feature vectors that are sampled independently [33]. The classifier is called random forest as it generates forests with random amounts of trees. Normal decision trees are based only on rules for prediction of any outcome. However, random forest classifier use information gain in order to split features from a particular node.

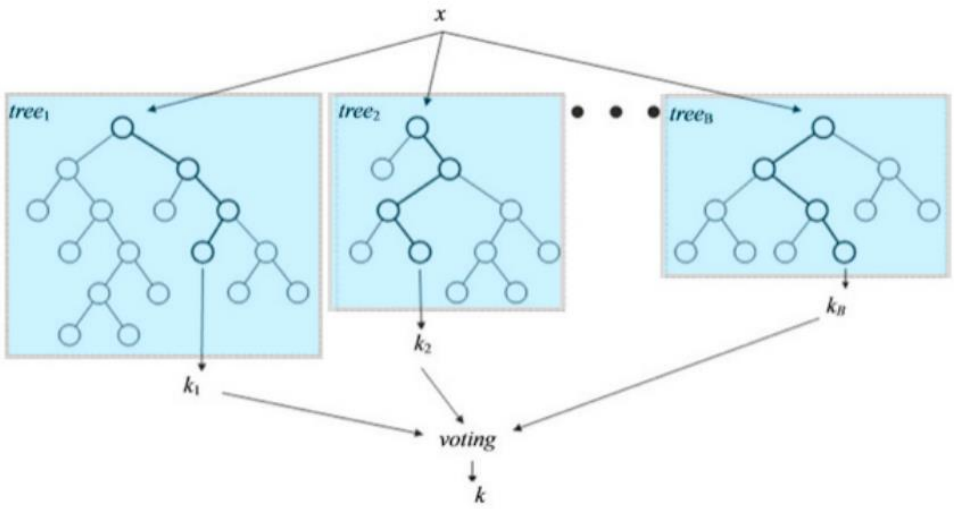
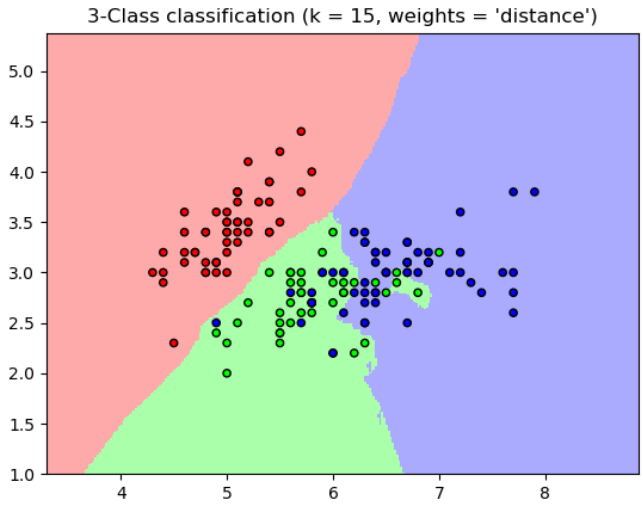


Fig 8: Random Forest Classifier [33]

It usually starts with the creation of many random decision trees where each one of those predicts a class from the features that are provided to it. These results are then used to get a majority vote to predict final class.

C. K Nearest Neighbors Classifier (K-NN)

K-Nearest neighbors is an algorithm that is used for classification that is based on the similarity of its neighbors. It is called lazy since there is no training that is involved, instead all of the data points will only be used at the time of making a prediction [33]. The data that the K-NN classifier gets can be scalars or even multidimensional vectors. All of the data points are assumed to be in the feature space, as a result of which there is a notion of distance. A commonly used metric for calculating distance between data points is Euclidean distance. K-NN can be used for binary classification as well as multi-label classification. A number K needs to be specified in order to decide the number of neighbors that will effectively influence the decision.



. Fig 9: K-Nearest Neighbor Classifier [33]

V. HYPOTHESIS AND CONTRIBUTION

Using the dimensionality reduction property of the autoencoders along with that of the ReLU activation function, we can prevent the obstacle of dying neurons during training the neural network. Furthermore, we also look at a technique called xavier initialization that helps us to ensure that the weights in the neural network stay within the correct bounds for better learning and results. Our contribution would be to combine the property of the ReLU wherein it avoids neurons from dying during training with the autoencoder in order to achieve better dimensionality reduction that can help us improve the accuracy and better generalize the data in order to predict human activities based on sensor readings.

The experiments performed will include performing PCA on the human activity data in order to perform dimensionality reduction. We will also perform the same task using autoencoders with linear, sigmoid, ReLU activations and xavier initialization and finally compare and contrast the results with an autoencoder with ReLU. The experiments will help us measure the accuracy and generalization with benchmark technique such as PCA.

VI. EXPERIMENT SETUP

A. Dataset

The dataset used for this experiment consists of body motion data collected from recordings of five log files containing data for individuals performing 12 different physical activities [37] [38] [39] consisting a total of 291,227 records. The data is gathered with the help of sensors that are placed on the subject's chest, right wrist, and left ankle in order to record the motion that is experienced by the different parts of the body. Obtaining sensor data using multiple sensors placed on different body parts allows us to capture body dynamics.

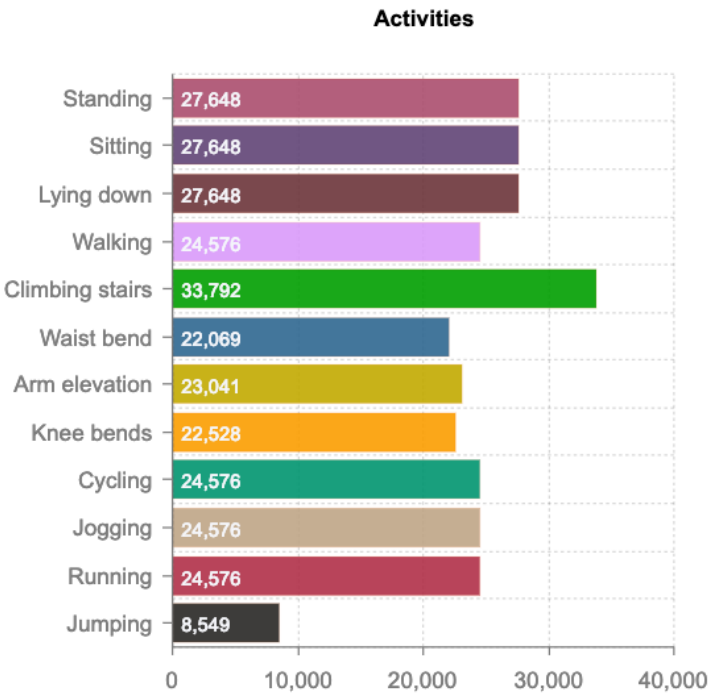


Fig 10: Number of distinct activities

Multiple sensors are placed on different parts of the subject's body so as to better capture the acceleration, rate of turn and the magnetic field orientation. All of the modalities are recorded

at a sampling rate of 50 Hz which is considered enough for the purpose of recording human activities. The data collected from these sensors is found to generalize well to the common activities that people carry out throughout the day capturing the intensity of all the actions. All of the activities are recorded in an out of lab environment thus enforcing no constraints on the actions or activities.

B. Choosing influential components

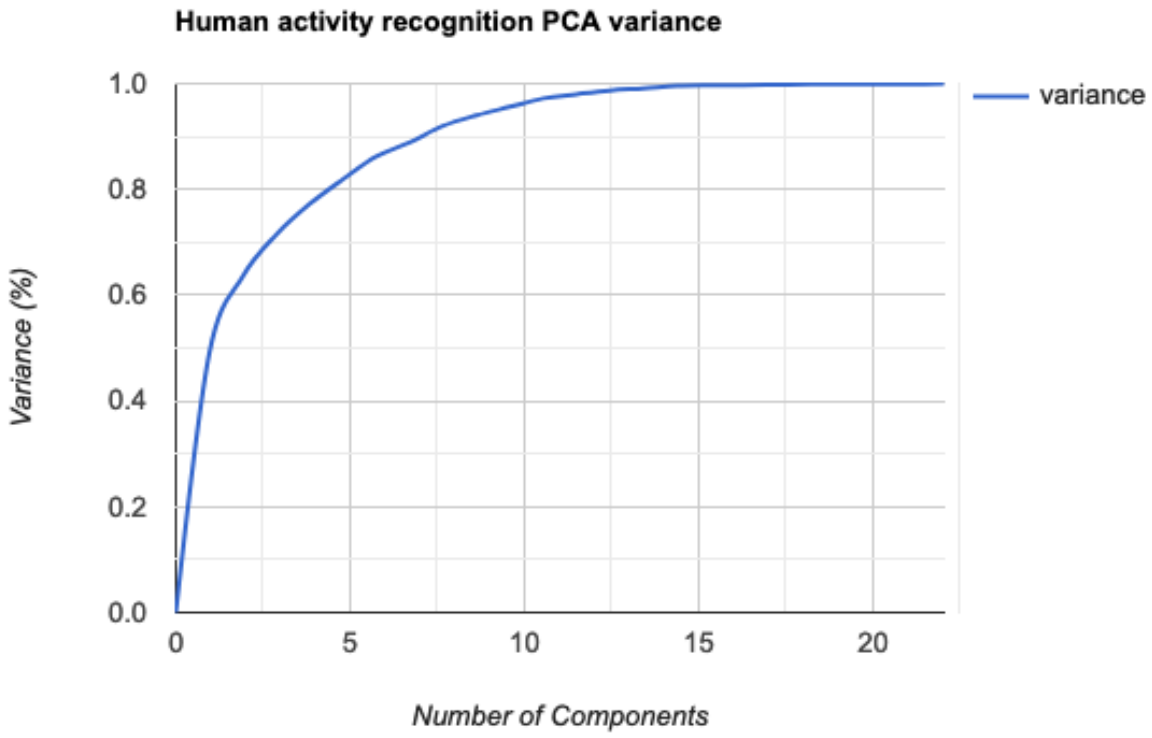


Fig 11: Influential components from PCA

Before starting the implementation, we need to identify the most influential components from the total components that we have. We do this by plotting the variance ratio (variance of data) of the different components at hand. From the above figure, we observe that selecting 10 components preserves around 95% of the total variance observed in the principal components. Thus, while calculating PCA for our data, we will only consider the first 10 dominant principal components.

C. Implementation Details

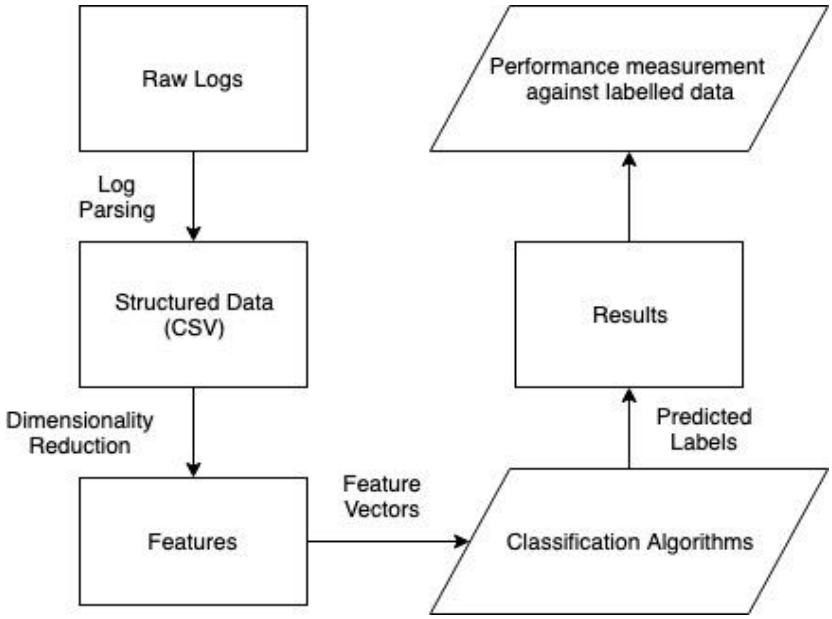


Fig 12: Experiment Implementation Flow

The first step in the experiment implementation would be parse the raw log data dumped by the sensors and convert it to a structured format. This can be done with the help of scripts that can be used in order to parse the log file row by row to collect all of the samples and then

convert it into a structured format (CSV file). Once we have obtained the data in a structured format, we can then pass on the data to dimensionality reduction algorithms in order to reduce their dimensions and extract important features from the original data. This reduced data which is in the form of feature vectors is then fed to different classification algorithms that are used to train a model. We then evaluate and compare the accuracy of the results provided by each classification model to verify if our hypothesis holds true.

D. Evaluation Metrics

The primary focus of the experiments is to obtain features (reduced) and then train classification models that are accurate and generalize well. Thus, to best evaluate the performance of these models we use 3 metrics. The results obtained from either of these models can be arranged in a confusion matrix M_{nxn} for problems that have *n* total classes [39].

True Positives (TP): Number of positive instances that were correctly classified as positive.

True Negative (TN): Number of negative instances that were correctly classified as negative.

False Positive (FP): Number of negative instances that were wrongly classified as positive.

False Negative (FN): Number of positive instances that were wrongly classified as negative.

Accuracy: Accuracy of a particular model is a standard metric for evaluating its classification performance, which is defined as

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Fig 13: Accuracy formula [40]

Precision: It is also referred to as a positive predictive value, precision is the ratio of correctly classified positive instances to the total instances that were classified as positive [40].

Recall: It is also referred to as true positive rate, which is the ratio of correctly classified positive instances to the total positive instances [40].

$$Precision - \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

Fig 14: Precision and recall formula [40]

Although these metrics are defined for binary classification, they generalize well and can also be used for multi-label classification with *n* classes. i.e. any particular instance can be said to be positive or negative depending on a particular class. e.g. positives would be all instances of standing and the rest of the instances would be negatives.

VII. EXPERIMENTS AND ANALYSIS

A. *Experiment 1: Dimensionality reduction using Principal Component Analysis*

In the first experiment, we use PCA as a baseline algorithm. The input data is initially scaled between the range of 0 and 1 using MinMaxScaler operation. The algorithm then runs PCA on the scaled data which gives us the principal components of the data. Since we know that the first 10 principal components capture almost 95% of the information in our dataset, we use the first 10 principal components and then train multiple classifier such as SVM, k-NN and Random Forest in order to obtain accuracy, precision and recall for each of these classifiers.

K- Nearest neighbor

K	Accuracy	Precision	Recall
10	0.66	0.68	0.66
20	0.71	0.72	0.7
30	0.76	0.77	0.75
40	0.84	0.83	0.81
50	0.85	0.85	0.83
60	0.85	0.85	0.82

Table 1: K-Nearest neighbor results

Random forest

No. of trees	Accuracy	Precision	Recall
2	0.69	0.67	0.68
4	0.78	0.77	0.78
8	0.83	0.82	0.81
16	0.87	0.86	0.85
32	0.87	0.83	0.84

Table 2: Random forest results

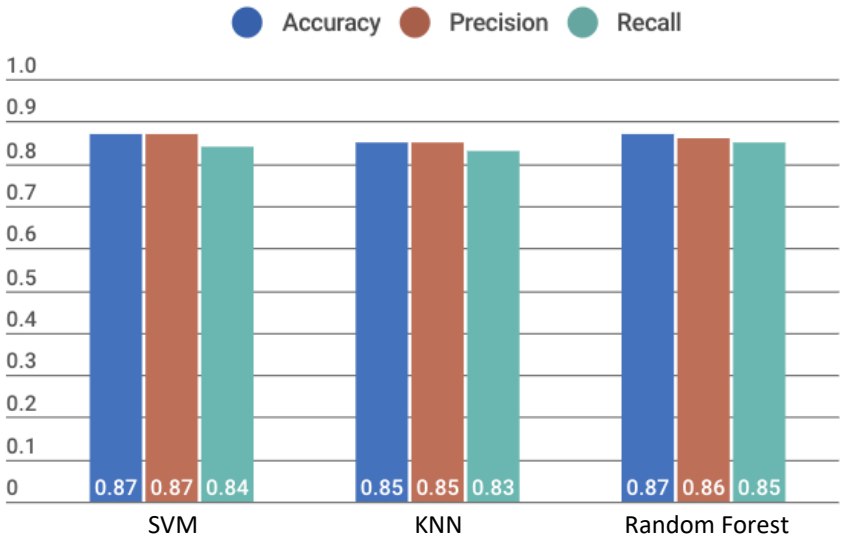


Fig 15: Classification results for PCA using SVM, KNN and Random Forest

Analysis

From the results, we can see that the SVM and random forest perform the best by achieving an accuracy of 87%. From fig.10, we know that the first 10 principal components are the most amount of information in the dataset. In order to find the right value for K in the KNN algorithm we iterate over values starting from 10 to 50. We choose 50 as a cut off value since from Table 1. we observe that the accuracy does not change beyond 50. Similarly, we iterate over different values for the number of trees used in random forest algorithm. From Table 2. We observe that the accuracy does not increase beyond 16 trees. This experiment, using PCA for dimensionality reduction will be a baseline for comparison with the following experiments.

B. *Experiment 2: Dimensionality reduction using linear Autoencoder*

Similar to the previous experiment, we first scale the input feature vectors using MinMaxScaler to scale the data between a range of 0 and 1. After we have scaled the feature vectors, the algorithm will then use these features to train a linearly activated autoencoder. The bottleneck layer of the autoencoder will consist of 10 neurons since the we need 10 most influential features from the input data. The encoded data that we obtain from the encoding phase of the autoencoder consists of reduced features that we then use to train multiple classifiers such as SVM, k-NN and Random Forest to obtain accuracy, precision and recall for each of the classifiers.

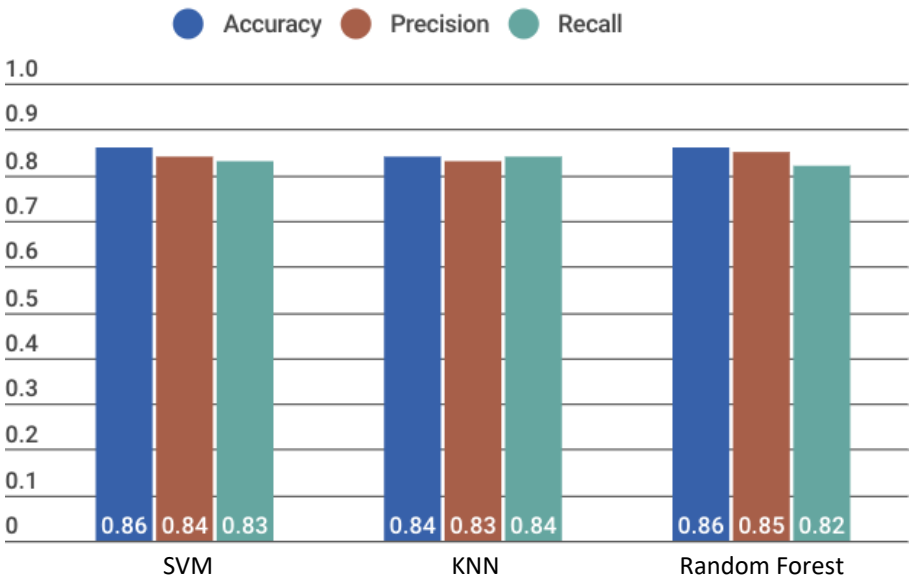


Fig 16: Classification results for Linear AE using SVM, KNN and Random Forest

Analysis

From fig.15 we can see that using the linearly activated autoencoders has not really improved the classification accuracy. The highest accuracy achieved is by the SVM and random

forest algorithm at 86% and the lowest accuracy at 84%. Using the linear activation function does not help us to capture the non - linear relations between features in the data, better than that of PCA. We observe that the result obtained using linear autoencoder are worse than that of PCA. This is because of the inherent non-linearity in the features obtained from the dataset. Next we will look at the effects of using non-linear functions in the autoencoder.

C. *Experiment 3: Dimensionality reduction using sigmoid Autoencoders*

We scale the data from input feature vectors using MinMaxScaler to scale the data between 0 and 1. After scaling the data, the algorithm will then use these features to train an autoencoder that uses a sigmoid activation function instead of a linear activation function. Sigmoid activation function allows the model to capture non-linear relations between the features better than that of linear activation. The bottleneck layer of the autoencoder will consist of 10 neurons as we need the 10 most influential features while training the autoencoder. The encoded data obtained after training the autoencoder consists of learnt and reduced features that we then use to train different classifiers in order to obtain accuracy, precision and recall.

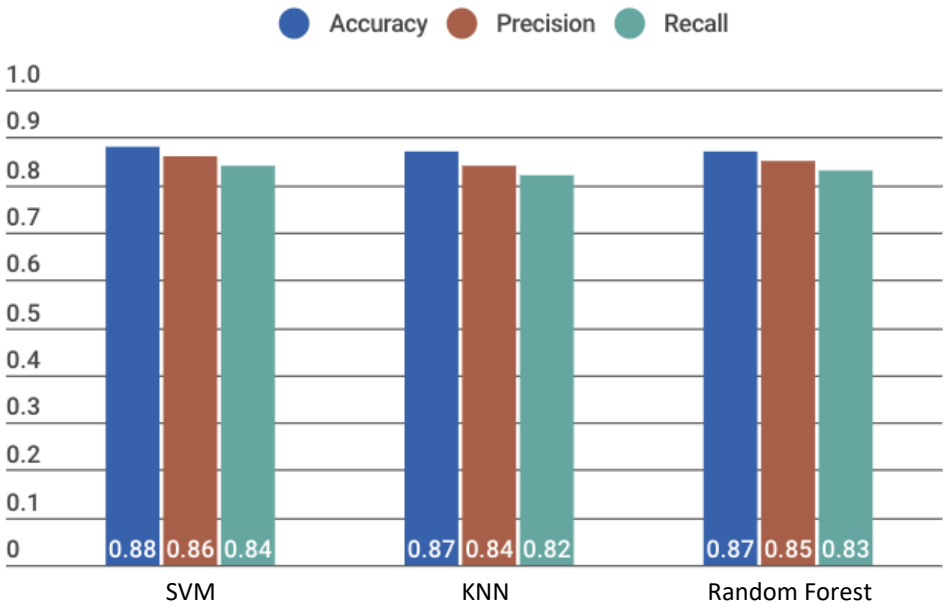


Fig 17: Classification results for Sigmoid AE using SVM, KNN and Random Forest

Analysis

From fig.16 we observe that the highest accuracy is achieved by SVM at 88% and the lowest by KNN at 87%. Using the sigmoid activation function, we try to learn the non-linear relations between the features in order to make better predictions. However, if the activation function receives a value closer to zero, then during back-propagation the updates to the weights in the previous layer will be extremely small. This can cause the neural network to stop learning as it induces dead neurons in the network.

D. *Experiment 4: Dimensionality reduction using ReLU Autoencoders*

We scale the data from the input feature vectors using MinMaxScaler to scale the data between 0 and 1. After scaling the data, the algorithm then uses these features to train an autoencoder that uses a ReLU activation function during training. The encoded data

obtained after training consists of learnt and reduced features that we then use in order to train classifiers in order to obtain accuracy, precision and recall.

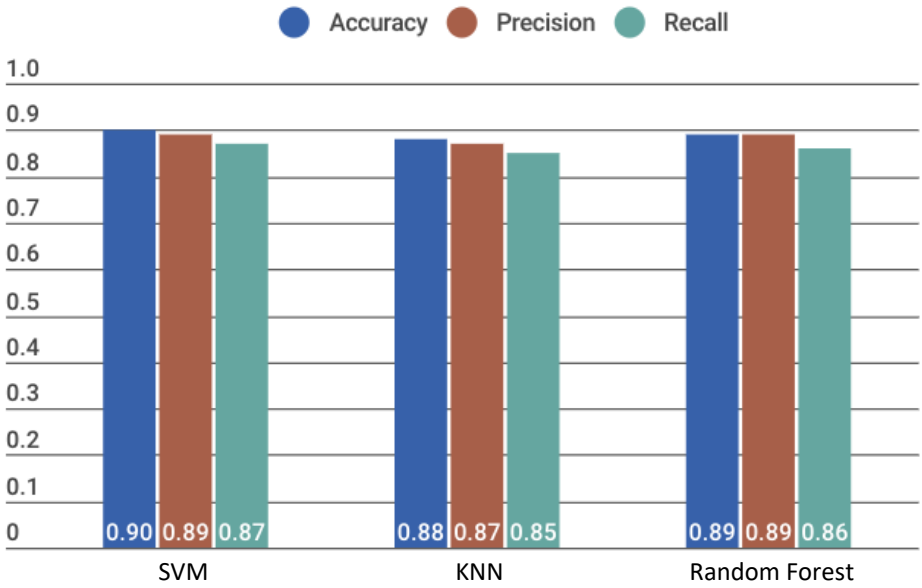


Fig 18: Classification results for ReLU AE using SVM, KNN and Random Forest

Analysis

From fig.17 we observe that we achieve the highest accuracy with SVM of 90% and the lowest accuracy of 88%. ReLU activation function has shown to improve performance in training CNNs [29]. We use the leaky ReLU (rectified linear unit) activation function which induces a small negative slope that helps in dealing with the vanishing gradient problem by not allowing the gradient value to become zero. This aims at addressing the dying neuron problem which stops the neural network from learning. Also, ReLUs in general converge faster compared to other activation functions as they are computationally efficient and simple.

E. *Experiment 5: Xavier initialization*

An important aspect that affects learning in neural networks are weights. Ideally, we want the weights to be initialized with random values such that they have a mean of 0 and a standard deviation of 1. This ensures that the weights do not saturate and hinder the learning process. However, this initialization causes problems further in the network as the standard deviation of neurons in next layer becomes the sum of the standard deviations of all neurons in the current layer, thus leading to the problem of exploding gradients. Xavier initialization [41] is a technique that allows us to mitigate this issue. Instead of initializing the weights with a standard deviation of 1, we initialize them to have a standard deviation of *1/n*, where *n* is the number of neurons in the previous layer.

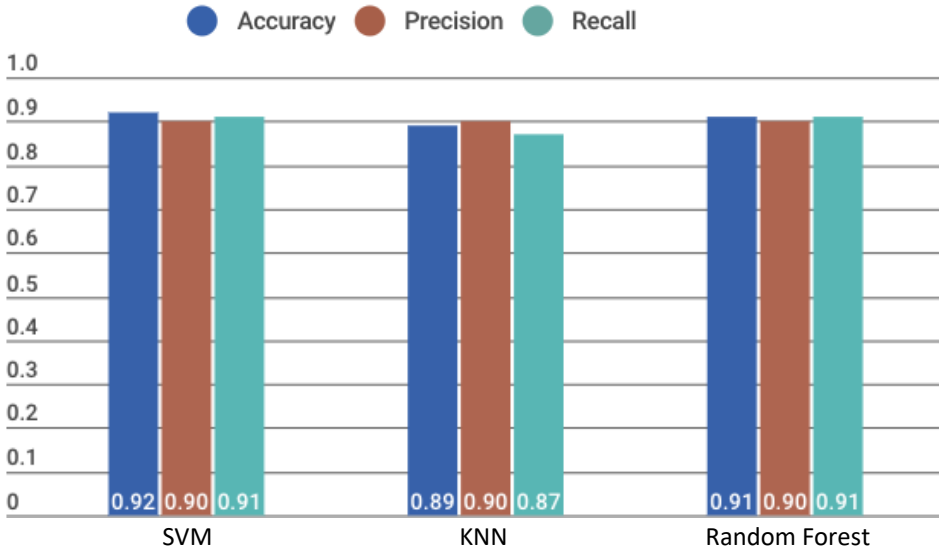


Fig 19: Classification results with Xavier initialization

Analysis

From fig.18, we observe that there is some improvement with the SVM and random forest classifier. Weights connecting neuron from one to the other have significant influence over the features that are learnt by the bottleneck layer. Using xavier initialization, we ensure that the value for each neuron has a mean of 0 and a standard deviation of 1. Using this sort of weight initialization helps improve accuracy in a significant way while using autoencoders that use ReLU activation function.

VIII. SUMMARY OF RESULTS

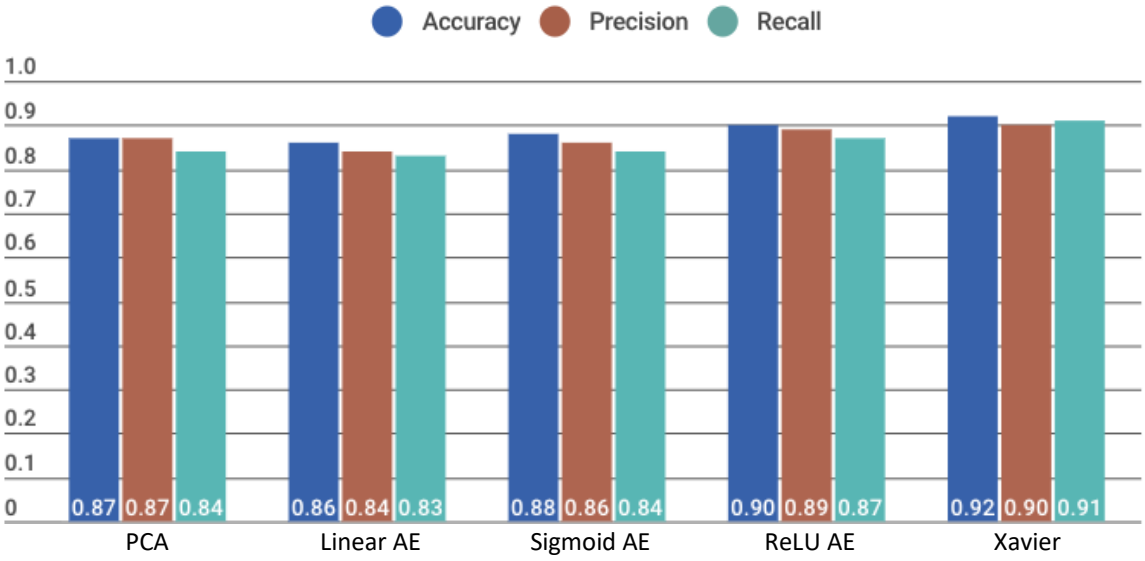


Fig 20: Comparison of overall results

From fig. 19, we can see that using autoencoders with ReLU activation functions helps us achieve an accuracy of 90% with SVM classifier. We initially started our experiments by using PCA for dimensionality reduction. From fig. 10, we see that the initial 10 principal components contribute towards almost 95% of the information obtained from the features. Thus using 10 principal components, we trained SVM, KNN and random forest classifier that we use as a baseline. We then proceeded with implementing the autoencoder with linear activation by encoding original features into a bottleneck layer of 10 neurons. However, linear activation functions fail to capture the non-linearity between the features. Using sigmoid activation for the autoenoder (Sigmoid AE) improves the accuracy over that of PCA, however it still suffers from the vanishing gradient problem caused due to values close to zero. ReLU AE, addresses this issue by inducing a small negative slope in the activation function, preventing the gradient from completely vanishing and thus prevents the network from aborting the learning process. The next possibility that we explored was that of xavier initialization which aims at correct weight

initialization that prevents the weights in the neural network to become very small or very large thus giving us better results.

IX. CONCLUSION AND FUTURE WORK

Over the course of this project, we have addressed the research question of whether autoencoders can be used in order to effectively reduce the dimensions of the different features from HAR data and to capture the non-linearity between different features. We have leveraged the properties of different activation functions, more specifically that of the ReLU (which are commonly used in CNNs) and leaky ReLUs in order to better learn the abstract features that have helped us in effective dimensionality reduction and better classification for HAR. We also looked at the effects of xavier initialization as a weight initialization technique that ensures better results. Effectively we have achieved a data saving of 57% since we end up using only 10 dominating features out of 23 to make our predictions. We also compared the results of the different experiments that we conducted with different activation functions with that of our baseline using PCA.

The primary aspect of the future work would be that of feature engineering i.e. deriving more meaningful features from the data with the help of domain knowledge. Furthermore, dimensionality reduction itself can be improved upon by experimenting with different autoencoder architectures using the engineered features to develop an optimal architecture. Future work will also include measuring the real-world effectiveness of experiments and optimizing them for real-world scenarios.

REFERENCES

[1] H. Jouhari, I. Moudden, S. ElBernoussi, M. Ouzir, “Learned model for human activity recognition based on dimensionality reduction” *2nd Int. Conf. on Smart Applications and Data Analysis for Smart Cities (SADASC).*

[2] M. A. Mannini and A. M. Sabatini, “Machine learning methods for classifying human physical activity from on-body accelerometers”, *IEEE Sensors J.*, 10(2), 2010, pp. 1154–1175.

[3] J. Wannenburg and R. Malekian "Body sensor network for mobile health monitoring, a diagnosis and anticipating system", *IEEE Sensors J.*, 15(12), 2015, pp. 6839-6852

[4] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality" in *Proc. 30th ACM Symp. on Theory of Computing*, (ACM), 1998, pp. 604–613.

[5] P. Kumari , L. Mathew, P. Syal , "Increasing trend of wearable and multimodal interface for human activity monitoring: A review", in *Biosensors and Bioelectronics.* vol. 90, 2017, pp. 298-307

[6] G. Sebestyen, A. Tirea, R. Albert , "Monitoring human activity through portable devices" in *Carpathian J. of Electronic and Computer Engineering*. vol. 5, 2012, pp. 101-106

[7] Z.-Y. He and L.-W. Jin, “Activity recognition from acceleration data using ar model representation and svm,” in International Conference on Machine Learning and Cybernetics, vol. 4, pp. 2245–2250, 2008.

[8] E. M. Tapia, S. S. Intille, W. Haskell, K. Larson, J. Wright, A. King, and R. Friedman, “Real-time recognition of physical activities and their intensities using wireless accelerometers and a heart monitor,” in Proc. International Symposium on Wearable Computers, 2007.

[9] I. Moudden, M. Ouzir, B. Benyacoub, S. Bernoussi, "Mining human activity using dimensionality reduction and pattern recognition", *Contemporary Engineering Sciences*, 9(21), 2016, pp. 1031-1041

[10] Bi, Jinbo, K. Bennett, M. Embrechts, C. Breneman and M. Song: “Dimensionality Reduction via Sparse Support Vector Machine”, *Journal of Machine Learning Research 3*, pp.1229-1243 (2003)

[11] R. Xi , M. Li, M. Hou, M. Fu , H. Qu , D. Liu , and C.R. Haruna, "Deep Dilation on Multimodality Time Series for Human Activity Recognition", *IEEE Access*, pp (99)1-1, 2018

[12] Statquest, https://statquest.org/2015/08/13/pca-clearly-explained/, Accessed 22 March 2019

[13] R. Sembiring, J. Zain, A. Embong, "Dimension Reduction of Health Data Clustering", *International Journal on New Computer Architectures and Their Applications (IJNCAA)* 1(3): 1041-1050

[14] T. Manning-Dahan, "PCA and Autoencoders", INSE 6220, Concorida University.

[15] K. Basterretxea, J. Echanobe, and I. del Campo, "A wearable human activity recognition system on a chip," in *Proc. Conf. Design Archit. Signal Image Process. (DASIP)*, Oct. 2014, pp. 1–8.

[16] S. Balli, E. Sagbas, M. Peker, "Human activity recognition from smart watch sensor data using a hybrid of principal component analysis and random forest algorithm", *Measurement and Control 2019*, Vol. 52(1-2) 37–45, DOI: 10.1177/0020294018813692

[17] M. A. Kramer, "Nonlinear principal component analysis using auto associative neural networks", *AIChE journal 37* (2) (1991) 233–243. DOI:10.1002/aic.690370209.

[18] R. Hecht-Nielsen, "Replicator neural networks for universal optimal source coding", Science (1995) 1860–1863, DOI:10.1126/ science.269.5232.1860.

[19] H. Schwenk, Y. Bengio, "Training methods for adaptive boosting of neural networks, in: Advances in neural information processing systems", 1998, pp. 647–653, DOI:10.1162/ 089976600300015178

[20] Introduction to Autoencoders, https://www.jeremyjordan.me/autoencoders/, Accessed on 22 March 2019

[21] D. Charte, F. Charte, S. Garcia, M. del Jesus, F. Herrera, "A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines", *Information Fusion 44* (2018) 78-96, DOI: 10.1016/j.inffus.2017.12.007

[22] G. K. Wallace, "The JPEG still picture compression standard", *IEEE transactions on consumer* electronics 38 (1) (1992) xviii– xxxiv. doi:10.1145/103085.103089.

[23] D. Del Testa, M. Rossi, "Lightweight lossy compression of biometric patterns via denoising autoencoders", *IEEE Signal Processing Letters* 22 (12) (2015) 2304–2308. DOI:10.1109/LSP. 2015.2476667.

[24] D. Hsu, "Time series compression based on adaptive piecewise recurrent autoencoder", arXiv preprint arXiv:1707.07961.

[25] Simpliliearn, https://br.simplilearn.com/training-deep-neural-nets-tutorial, Accessed 22 March 2019

[26] Y. Miao, P. Blunsom, "Language as a latent variable: Discrete generative models for sentence compression", in: *Proc of the 2016 Conf on Empirical Methods in Natural Language Processing*, 2016, pp. 319–328.

[27] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In AISTATS, 2010.

[28] A. Shah, E. Kadam, H. Shah, S. Shinde, S. Shingade, "Deep Residual Networks with Exponential Linear Unit" in *Proc on VisionNet'16 Proceedings of the Third International Symposium on Computer Vision and the Internet*

[29] A. Krizhevsky, I. Sutskever, G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", *Communications of the ACM*, Volume 60 Issue 6, June 2017 pp. 84-90

[30] K. He, et al., "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification". In Proc. of the international conf. on computer vision, pp. 1026-1034, 2015.

[31] B. Xu, N. Wang, T. Chen, M. Li, "Empirical Evaluation of Rectified Activations in Convolutional Network", *ICML Deep Learning Workshop*, Lille, France, 06-11 July 2015

[32] M. Zubair, K. Song, C. Yoon, "Human Activity Recognition Using Wearable Accelerometer Sensors" in *2016 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, Seoul, South Korea, 01-15 Jan 2017

[33] Scikit-learn documentation, https://scikit-learn.org/stable/modules/svm.html, Accessed 22 March 2019.

[34] D. Edla, K. Mangalorekar, G. Dhavalikar, S. Dodia, "Classification of EEG data for human mental state analysis using Random Forest Classifier" in *Intl Conf on Computational Intelligence and Data Science (ICCIDS 2018)*

[35] D. Saraswathi, E. Srinivasan, "Performance Analysis of Mammogram CAD System using SVM and KNN Classifier", in *Intl Conf on Inventive Systems and Control (ICISC-2017)*

[36] Scikit-learn documentation, https://scikit-learn.org/stable/auto_examples/neighbors/plot_classification.html, Accessed 22 March 2019.

[37] UCI Mhealth Dataset, https://archive.ics.uci.edu/ml/datasets/MHEALTH+Dataset, Accessed 22 March 2019.

[38] Banos, O., Garcia, R., Holgado, J. A., Damas, M., Pomares, H., Rojas, I., Saez, A., Villalonga, C. mHealthDroid: a novel framework for agile development of mobile health applications. Proceedings of the 6th International Work-conference on Ambient Assisted Living an Active Ageing (IWAAL 2014), Belfast, Northern Ireland, December 2-5, (2014)

[39] Banos, O., Villalonga, C., Garcia, R., Saez, A., Damas, M., Holgado, J. A., Lee, S., Pomares, H., Rojas, I. Design, implementation and validation of a novel open framework for agile development of mobile health applications. BioMedical Engineering OnLine, vol. 14, no. S2:S6, pp. 1-20 (2015).

[40] O. Lara, M. Labrador, "A Survey on Human Activity Recognition using Wearable Sensors" in *IEEE Communications surveys and tutorials*, Vol. 15, No. 3, Third Quarter 2013

[41] T. Sai, H. Lee, "Weight Initialization on Neural Network for Neuro PID Controller -Case study-" in *2018 Intl Conf on Information and Communication Technology Robotics (ICT-ROBOT)*, Nov 2018, DOI: 10.1109/ICT-ROBOT.2018.8549904