MEASURING STROKE REHABILITATION BY POSTURE DETECTION USING NEURAL NETWORKS

A MAJOR PROJECT REPORT

Submitted by

MRUDHULA RAYA	**[RA1811003020322]**
MUKHIL SUNDARARAJ	
GOWTHAMAN	**[RA1811003020276]**
S. DHAARINI	**[RA1811003020299]**

Under the guidance of

Ms. SWATHI R

(Assistant Professor, Department of Computer Science and Engineering)

in fulfillment for the award of the degree

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

RAMAPURAM CAMPUS, CHENNAI -600089

MAY 2022

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Deemed to be University U/S 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this project report titled **"MEASURING STROKE REHABILITATION BY POSTURE DETECTION USING NEURAL NETWORKS"** is the bonafide work of **Mrudhula Raya [RA1811003020322], Mukhil Sundararaj Gowthaman [RA1811003020276], S. Dhaarini [RA1811003020299]** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award wasconferred on an occasion on this or any other candidate.

SIGNATURE

Ms. Swathi R
Assistant Professor,
Computer Science and Engineering,
SRM Institute of Science and Technology,
Ramapuram Campus, Chennai.

SIGNATURE

Dr. K RAJA, ME., Ph.D.,
Professor and Head
Computer Science and Engineering,
SRM Institute of Science and Technology,
Ramapuram Campus, Chennai.

Submitted for the project viva-voce held on .....at SRM Institute of Science and Technology , Ramapuram Campus, Chennai -600089.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

RAMAPURAM, CHENNAI – 89

DECLARATION

We hereby declare that the entire work contained in this project report titled "**MEASURING STROKE REHABILITATION BY POSTURE DETECTION USING NEURAL NETWORKS**" has been carried out by **Mrudhula Raya [RA1811003020322], Mukhil Sundararaj Gowthaman [RA1811003020276], S. Dhaarini [RA1811003020299]** at SRM Institute of Science and Technology, Ramapuram Campus, Chennai- 600089, under the guidance of **Ms. Swathi R, Assistant Professor**, Department of Computer Science and Engineering.

Place: Chennai

Date:

MRUDHULA RAYA
MUKHIL SUNDARARAJ GOWTHAMAN
S. DHAARINI

Own Work Declaration

Department of Computer Science and Engineering

SRM Institute of Science & Technology

Own Work Declaration Form

This sheet must be filled in (each box ticked to show that the condition has been met). It must be signed and dated along with your student registration number and included with all assignments you submit – work will not be marked unless this is done.

To be completed by the student for all assessments

Degree/ Course	:	B. Tech
Student Name	:	Mrudhula Raya, Mukhil Sundararaj Gowthaman, S.Dhaarini
Registration Number	:	RA1811003020322, RA1811003020276, RA1811003020299
Title of Work	:	Measuring Stroke Rehabilitation By Posture Detection Using Neural Networks

I / We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism**, as listed in the University Website, Regulations, and the Education Committee guidelines.

I / We confirm that all the work contained in this assessment is my / our own except where indicated, and that I / We have met the following conditions:

- Clearly references / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website

I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:
I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above.
If you are working in a group, please write your registration numbers and sign with the date for every student inyour group. RA1811003020276 RA1811003020299 RA1811003020322

ABSTRACT

More than 795 thousand people in the US experience a stroke, per year. Out of this, around 610 thousand are RST or new strokes. [2] Strokes are a leading factor of mortality for Americans with 140 thousand people dying annually. [3] Incidences of stroke have increased dramatically in the young populace with over 20% of the affected population being under the age of 45. Hemiparesis, which occurs after a stroke, is a major motor impairment and can affect up to 65% of stroke victims as a large number of stroke patients can experience muscle weakness. [4] Loss of strength in muscle post stroke is a common symptom and is a major factor in slowing the recovery of stroke victims. This can result in the lower physical activity or even immobilization in patients. 30% to 60% of stroke victims report reduced physical range in the affected body part when using it for daily activities. [5] The feasibility of a 4 by 4 excible sensor matrix using paper as a base material is explored and a paper-based pressure sensor is developed. This helps reduce the cost and fabrication time. The static and dynamic characteristics of the matrix are also presented. The sensor can be effectively used to recognize and assess the recovery of stroke rehabilitation patients despite some variation in response of different sensors due to the properties of the structural material. The matrix is analyzed by using a neural network that is trained to detect the position of a load based on sensor values.

ACKNOWLEDGEMENT

We place on record our deep sense of gratitude to our lionized Chairman **Dr.R.SHIVAKUMAR** for providing us with the requisite infrastructure throughout the course.

We take the opportunity to extend our hearty and sincere thanks to our Dean, **Dr.M.MURALI KRISHNA, B.E., M.Tech., Ph.D. MISTE,FIE,C.Engg.,** for manoeuvring us into accomplishing the project.

We take the privilege to extend our hearty and sincere gratitude to the Professor and Head of the Department, **Dr.K.RAJA,M.E.,PhD.,** for his suggestions, support and encouragement towards the completion of the project with perfection.

We express our hearty and sincere thanks to our guide **Ms. Swathi R, Assistant Professor,** Department of Computer Science and Engineering for her encouragement, consecutive criticism and constant guidance throughout this project work.

Our thanks to the teaching and non-teaching staff of the Computer Science and Engineering Department of SRM Institute of Science and Technology, Ramapuram Campus, for providing necessary resources for our project.

MRUDHULA RAYA

MUKHIL SUNDARARAJ GOWTHAMAN

S. DHAARINI

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Overview

Strokes mostly occur in combination with other medical diagnoses. Current stroke rehabilitation evidence, however, is not focused on the co morbidities and therefore does not align with the experiences of the current population. The motivation behind this study is to decide the degree and nature of the ongoing randomized and controlled preliminary of stroke restoration with an emphasis on patients with multimorbidity. Stroke recovery is often incomplete, but rehabilitation training can help recovery outcomes by engaging endogenous neuroplasty.

1.2 Problem Statement

High doses of training are needed in pre-clinical models of stroke rehabilitation in order to recover the physical activity of the impacted appendages in animals. However, in human body the required training to help recouping rate is unknown. The lack of pragmatic and objective approaches to measuring the rehabilitation training doses adds to this ignorance. To develop a proper measurement approach, the primitives of activities are to be identified. Examples of primitives include running, jumping, and walking. In our review, forty-eight individuals with chronic stroke are chosen to perform a variety of stroke rehabilitation training activities. To capture the upper body movements the individuals, use inertial measurement units or IMUs. Human labellers are used to identify the primitive activities and to label and segment the IMU data. A recurrent neural network (RNN) is designed that outperforms existing techniques. To compute the separate embeddings of the various physical quantities of the sensor data an initial module is included in the proposed algorithm. The proposed technique swaps the batch normalization, which uses measurements processed from training data to perform normalization, with instance normalization, that uses measurements processed from test data. The end result is increased robustness towards possible distributional shifts when encountering new patients.

1.3 Objective

A common side-effect of stroke is muscle weakening which often results in the reduction of the range of movement of the impacted body part. A wide scope of physical training is included in stroke rehabilitation which can help the patients re-establish and develop body fortitude, endurance, balance, and stability. A 4 4 excible pressure matrix is used to analyse and identify the movement of the body while performing the training exercises. The sensor can be used to measure the performance of a stroke rehabilitation patient and mark their progress. An Artificial Intelligence (AI) based algorithm is presented to measure the positioning accuracy of stroke sufferers. Assessment shows that the proposed calculation has a mean blunder of 0.103 cm in recognizing load while numerical investigation gives a mean mistake of 0.704 cm. The pressure sensor can be used to calculate the mistake in the training exercise positioning and can also calculate the duration to complete the exercises.

1.4 Domain Overview

Artificial intelligence seeks to simulate the problem-solving and decision-making abilities of the human mind by making use of computers and machines.

Although there have been a number of definitions of artificial intelligence, in his study, John McCarthy proposes the definition: "It is the art and science of creating intelligent devices, particularly intelligent computer programs. It's akin to the task of utilizing computers to study human intellect, but AI doesn't have to be limited to physiologically observable methods."

Alan Turing's landmark paper "Computing Machinery and Intelligence," published in 1950, is regarded as the start of the AI discussion in academia years before this concept. Turing is known as the "father of computer science". He asks the following question in his paper: "Can machines think?" To answer this, he developed a test, which is now known as the "Turing Test". In the “Turing Test”, a human interrogator is called in to try and tell the difference between the response of a computer and a human. This test has been criticized to a great deal, and yet it remains an essential element of the history of artificial intelligence. It is also considered to be a philosophical question as it also uses the concepts of linguistics.

Since deep learning and machine learning are often confused with one another, one needs to know the differences. Deep learning is a sub-field of machine learning, while both are sub-fields of artificial intelligence.

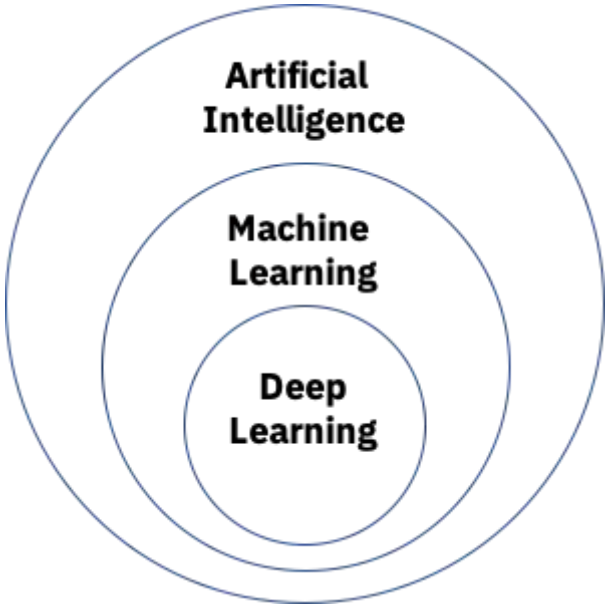


Fig 1.4.1 Venn diagram of AI, ML, and DL.

Neural networks are considered to be the “building blocks of deep learning”. A deep learning method, however, is a neural net with more than three layers, including the inputs and outputs. The following diagram is a general representation of this.

Deep learning and machine learning mainly differ in the way each algorithm learns. Deep learning removes the need for manual involvement of humans by automating a large portion of the feature extraction process, which allows for the use of larger data sets. Deep learning can be thought of as "scalable machine learning." Classical machine learning, which is also known as "non-deep" machine learning, is in contrast more reliant on manual intervention to learn. To understand the differences between the various inputs of data, human professionals develop a hierarchy of characteristics, which usually needs more structured data to learn.

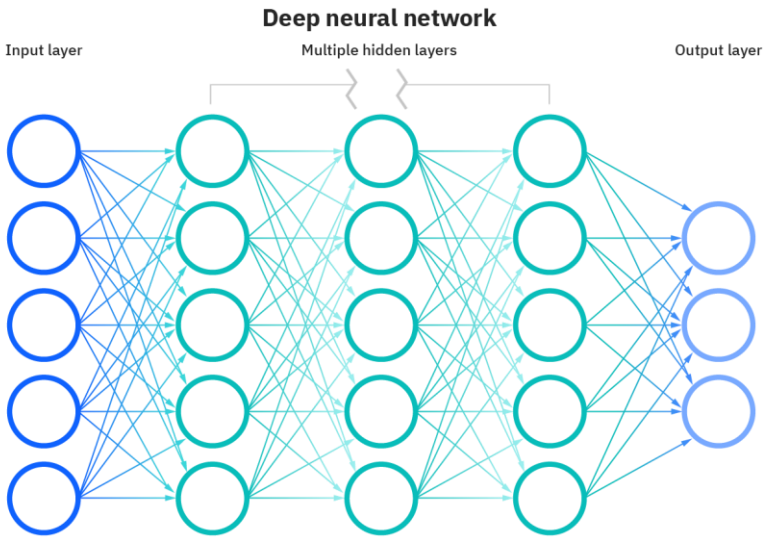


Fig 1.4.2 Diagrammatic representation of neural network.

Labeled datasets, also known as supervised learning, can be used to inform "deep" machine learning algorithms, but they aren't always used. It can use unstructured data in its raw form and discovers the features that separate various types of data automatically. It does not need manual intervention to interpret data, unlike machine learning, allowing users to scale it in more ways. Machine learning, deep learning, and neural networks imitate the working of the human brain. This lets programs spot patterns and solve problems.

A node layer generally is made up of an input layer, one or more hidden layers, and an output layer in artificial neural networks (ANNs). Every node links to another and is given a corresponding threshold and weight. If the output of a node exceeds a certain threshold value, then the node is activated. When this happens, the node data is sent to the subsequent tier of the network. If this is does not happen, then no data is carried on to the next tier of the network.

To increase the accuracy of the neural network as time goes on, the neural network makes use of training data to learn. Once the learning algorithms are fine-tuned to increase the accuracy, they become powerful tools in artificial intelligence, allowing users to quickly classify and cluster data. Speech recognition or image recognition can take hours when manually identified by human interpreters as opposed to the hours it takes for such algorithms to complete such tasks.

Different types of neural networks are employed for different purposes. Frank Rosenblatt invented the perceptron. The perceptron is the oldest neural network. It is also the simplest type of a neural network having only one neuron:

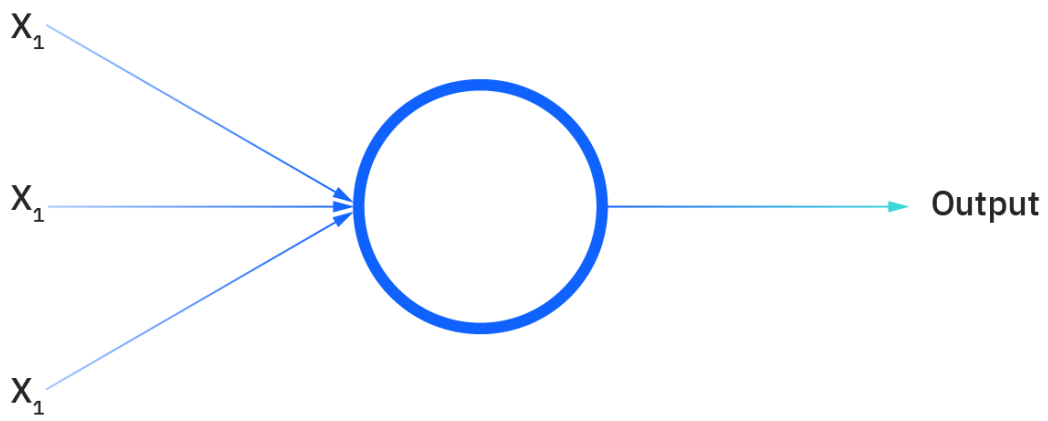


Fig 1.4.3 Single neuron.

Feed forward neural networks are also known as multi-layer perceptrons (MLPs). They consist of an input layer, a concealed layer or layers, and an output layer make up these layers. While these neural networks are also known as MLPs, it's vital to remember that they're made up of sigmoid neurons rather than perceptions because most real-world problems aren't linear. These models provide the cornerstone for computer vision, natural language processing, and other neural networks, and they are typically fed data to train them.

Like feed forward networks, convolutional neural networks (CNNs) are used for image recognition, pattern identification, and/or computer vision. These networks use linear algebra principles, notably matrix multiplication, to find patterns in images.

The feedback loops distinguish recurrent neural networks (RNNs). These learning algorithms are generally used to create predictions about future outcomes using time-series data, such as stock market projections or sales forecasting.

A recurrent neural network (RNN) is a type of artificial neural network that usually uses time series or sequential data. These algorithms are often used for issues like language

translation, natural language processing (NLP), speech recognition, and image captioning. Similar to feed forward networks and convolutional neural networks (CNNs), Recurrent neural networks learn from training input. They are distinguished by their "memory”. The memory lets the model affect the current inputs and outputs by making use of the knowledge from previous inputs. Recurrent neural networks' output is reliant on the sequence's prior elements as opposed to traditional neural nets which assume that all the inputs and outputs are independent of each other. While future occurrences may be useful in determining a sequence's output, single directional recurrent neural networks do not account for the future occurrences while making their predictions.

Recurrent networks are further separated in that that their parameters are shared within all of the network’s layers. While in a feed forward neural net, each node has a variable weight, in a RNN every layer has the same weight. For reinforcement learning, these weights are modified through back propagation and gradient descent processes.

In order to estimate the gradients, recurrent neural networks make use of the BPTT technique, which significantly differs from regular back propagation as it is specialized to sequence data. The normal back propagation uses the same concepts as BPTT, where the model trains itself by computing errors from its output layers to the input layers.

The computations enable users to accurately alter and fit the model's parameters. Because feed-forward neural nets tend to not share the parameters among layers, BPTT varies from the standard approach in that it adds the errors at every time step, whereas feed forward networks do not usually require totaling errors.

RNNs usually face two issues throughout the process: inflating gradients, vanishing gradients. The size of each gradient, which is calculated by using the slope of the loss function along the error curve, is the cause for these concerns. If the gradient is small, it can get smaller and smaller, and the weight parameters also change to reflect this. At a point, the parameters become insignificant, reaching zero. In such a case, the algorithm will stop learning. On the flip side if the gradient is too large, it can explode. This results in an unstable model. The model weights get excessively huge in this situation, and they will finally be represented as NaN. A solution to these concerns is to lower the number of hidden layers of the neural network, which in turn removes part of the complexity of the

RNN model.

1.5 Organization of the Report

The report is organized in the manner as follows: Chapter 1 presents the introduction about Machine Learning and Artificial Intelligence.

Chapter 2, Literature Survey provides an overview to the topics required for the system along with the state of the subject matter in the present. The paper consists of the detailed survey with a summary in the end.

Chapter 3, System Architectural Design, contains a high-level overview of the project, including its aim, scope and the purpose.

Chapter 4, System module provides a detailed description of the system architecture along with a summary.

Chapter 5, Module Implementation talks about the modules present in the system and how each module functions.

Chapter 6, Results and discussion contains the tools and methods that were used to implement the system, the code samples included.

Chapter 7, Conclusion and Future Work provides a summary of the content as a whole and discusses about the future enhancements if any.

CHAPTER 2

LITERATURE SURVEY

The various proposed ideas and concepts are studied and related with the proposed system with better understanding and explanation making the concepts clear and emphasizing on its importance.

2.1 Stroke Identification via Rough Sets

The authors Muhammad Salman et al in their paper “Identifying Stroke Indicators Using Rough Sets”, sought to rank various EHR (Electronic Healthcare Records) records in the order of importance that can be used to detect a stroke. [6] A novel rough-set technique was devised which, unlike the conventional techniques, can be used on any dataset that comprises binary feature sets. The proposed algorithm was then evaluated using a public dataset of EHR records. The conclusions that were drawn from the evaluation are that average glucose level, heart disease, hypertension, and age were the key attributes to help detect strokes in patients. The proposed algorithm was then benchmarked with popular feature-selection algorithms. The proposed technique obtained the best performance in ranking the importance of individual features to detect stroke.

2.2 Real time detection of rehabilitation

Stroke rehabilitation patients commonly employ compensations without therapist supervision, which leads to suboptimal recovery outcomes. Siqi Cai et al in their study “Real-Time Detection of Compensatory Patterns in Patients with Stroke to Reduce Compensation During Robotic Rehabilitation Therapy” [7] investigated the possibility of real-time monitoring of compensation in stroke rehab patients. Data from the pressure distribution and Machine Learning (ML) algorithms were utilized to do so. Trunk compensation reduction by combining online compensation detection and haptic feedback from a rehab robot was also a point of investigation. An SVR classifier (Support vector machine) was trained to classify the online compensatory patterns. The dataset used to train the classifier consisted of pressure distribution data recorded from six stroke patients who performed three different forms of reaching movements. A rehabilitation robot is used to

provide an assistive force to stroke patients to reduce compensatory movements. The high classification accuracies of the proposed algorithm confirmed the feasibility of using pressure distribution data to reduce compensatory movements in stroke patients. The proposed algorithm was merged with a rehabilitation assistive robot to reduce truck compensations in stroke patients.

2.3 Classification of strokes using medical images

A systematic analysis of the current computational systems for the segmentation, detection, and classification of strokes using medical images was proposed by Roger M. Sarmento et al. [8] To improve the accuracy of the diagnostic process and the interpretation consistency of such medical images, CAD (Computer-Aided Diagnosis) systems were developed using various technologies such as digital image processing and analysis and Artificial Intelligence. However, low sensitivity, reduction of false positives, optimization of the algorithms, and an improvement in the identification and classification of different shapes and sizes were some points that require more attention in the proposed techniques. Their research mainly focused on analysing the applied techniques for the development of CAD systems and verifying their efficiency for stroke detection, segmentation, and classification.

2.4 Existing System

The authors Anis Fatema et al in their paper “A Low-Cost Pressure Sensor Matrix for Activity Monitoring in Stroke Patients Using Artificial Intelligence”, studied the aftermath of stroke in recovering patients with regards to how it affects their motor controls and movement.[1] As muscle weakening is a common fallout of the stroke condition, the team of researchers developed a 4x4 pressure sensor matrix to quantize the performance and progress of rehabilitation of patients undergoing physiotherapy post stroke. A set series of movement exercises carried out with the pressure sensor matrix with the aim to estimate the force exerted and the error of movement

CHAPTER 3
SYSTEM ARCHITECTURAL DESIGN

The overall design is discussed and explained in this chapter. The high-level architecture provides clear details of the entities involved in the system and how they are dependent with other entities.

3.1 System Architecture

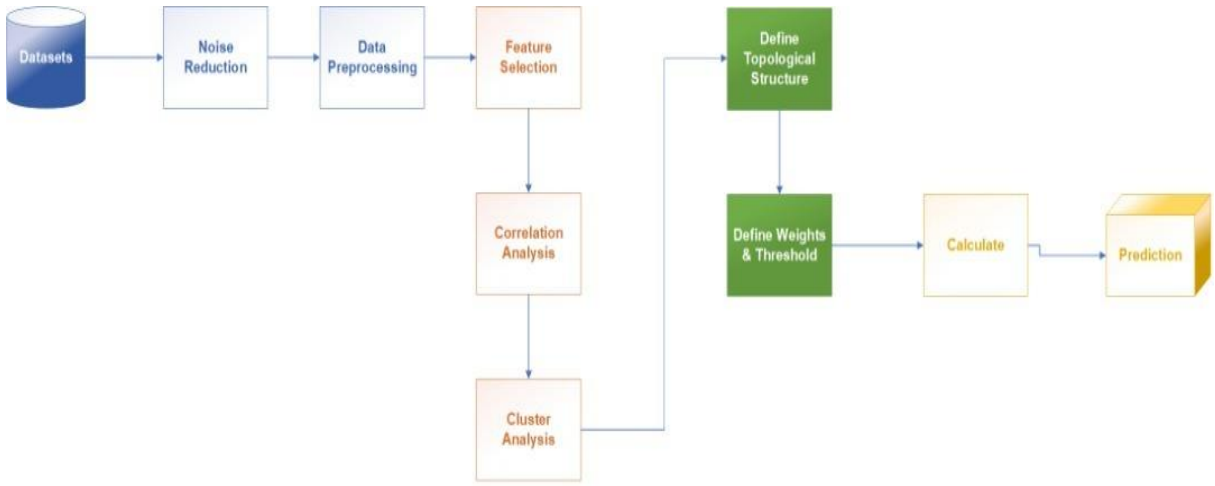


Fig 3.1.1 Architecture Diagram

The device architecture is a conceptual arrangement that describes the structure, behaviour, and perspectives of a system. A structure description is a proper description and illustration of a device, prepared in a manner that helps with reasoning approximately the systems and behaviours of the device. A device structure can encompass device additives and the sub-structures developed. There were efforts to formalize languages to explain device structure, together those are known as structure description languages (ADLs). The motivation at the back of device structure sporting events is to represent a whole association depending on standards, ideas, and homes coherently diagnosed with and are inter dependable. The answer structure has elements, homes, and qualities that fulfil, past what many could remember possible, the problem or opportunity communicated with the aid of using a

gaggle of framework necessities (recognizable to mission/enterprise and companion prerequisites) and existence cycle ideas (e.g., functional, backing) and which are implementable via innovations.

The System architecture which has been designed for this application use can be broken down into four sub domains namely,

- Data Imputation and Cleaning
- Feature Extraction
- Network Structurisation
- Prediction Calculation

Initially, the data is ingested from the data set and is cleaned for preprocessing by the removal of null values, noise and unwanted features present in it. After the data is cleaned, it is then put through a feature extraction process where exploratory data analysis (EDA) is performed on the said data to analyze and understand the patterns and shape of the dataset. Post feature extraction, the data is run through a correlation and cluster analysis where patterns and repetition of results leading to a clearer statistical analysis could prove to be beneficial to the description of the network structure and the threshold weights of the attributes or features which follow as the next stage of the system process. The network structure defines the speed and accuracy of the network as the number of layers is set in this phase of the process. Feature and attribute weights are then assigned to the model to define the prediction threshold for the network. As the last step in the process, the network is then run with the training data to verify and validate the prediction algorithm and fine tune the same if necessary for increased accuracy and minimal loss for the test data prediction.

CHAPTER 4
SYSTEM MODULE

4.1 Introduction

This chapter focuses on each module of the system in detail and explains its working.Each module is divided into separate subheadings and discussed accordingly. The pseudo code and working methodologies are also included and explained with diagrams wherever possible.

4.2 Imputation Details

Imputation of data is the handling of missing values in our dataset. One method to deal with missing data is to delete the records that contain the missing values, but this could lead to losing out on valuable information. There are various types of imputation. [9] Categorical imputations relate to the missing categorical values in the dataset. The missing data, in this case, is replaced by the most commonly occurring record in the dataset. Numerical imputation deals with missing numeric values in the dataset, the values are generally filled with the mean of the corresponding value in other records. From the imputation data collected we have consolidated a dataset which is used as the value source for the prediction of the model. Before proceeding further, we clean the dataset from possible noise that could have entered it, through means of standardization, duplicate deletion, and elimination of null values.

4.3 Feature Extraction

Selection methods like Feature extraction can be utilized in isolation or in union with each other to increase efficiency of the working of the model. It can be used to work on the precision, visualization, and comprehension of the learned information. Elements can be delegated relevant, irrelevant, or excess. [10] A subset of the accessible information is chosen, in this process. The best subset to pick is unified with the most un-number of aspects that add to learning exactness. [11] Information gain evaluates the gain of each individual feature with respect to the target variable. The larger the information gain, the larger will be the contribution of the characteristics to the text. The characteristics with

higher information gain are selected to be Features. Dimensionality reduction is a popular pre-processing method in data analysis, modeling, and visualization. This can be easily achieved using Feature Selection as we can only select the input characteristics that contain information relevant to solving the problem at hand. They can be further classified into Feature weighting methods and subset search methods. [12] This method has a low process expense and is faster. However they are also not very reliable in classification when compared to wrapper methods and are more suited to high dimensional data sets. Hybrid techniques have been created which consolidate the benefits of filters as well as wrapper strategies. In order to weigh the features to extract patterns and perform exploratory data analysis, we map labels to the different activities performed by the test subjects, recorded in the dataset. From the label map, we can further analyze the activities performed by each subject and further understand the data to draw patterns from it through means of multivariate feature analysis.

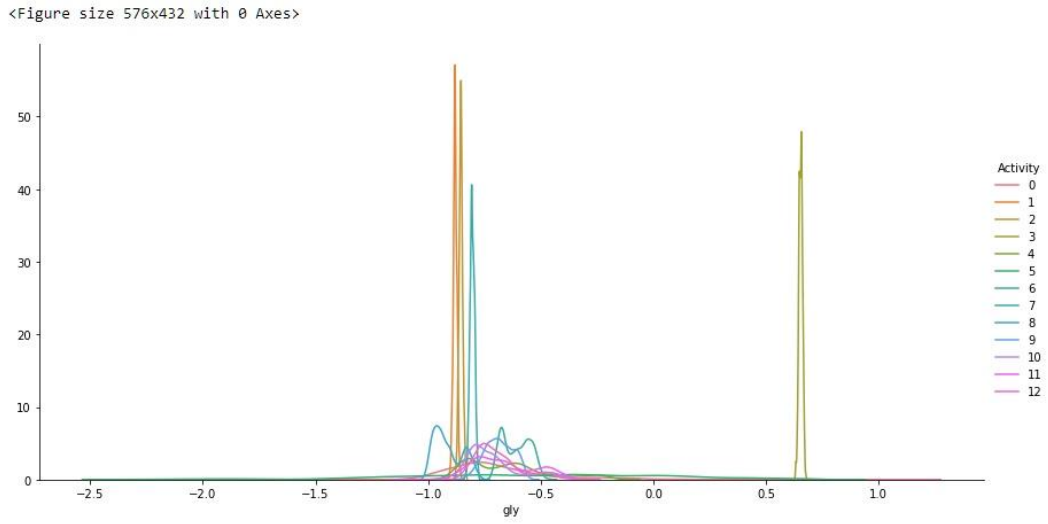


Fig 4.3.1 Activity frequency distribution for Y axis movement

4.4 Prediction

Neural Networks are a series of algorithms that aim to recognize underlying patterns in data in a way that mimics the human brain. It can be used for classification, prediction, and other such purposes. The number of neurons and hidden layers in the neural network is directly proportional to the complexity of the network. The same allows the neural network to learn complicated patterns that are hidden in the data. In a neural network, changing the weights of any one connection has a ripple impact on the wide range of various neurons and their initiations in the ensuing layers. A RNN (Recurrent Neural Network) is a kind of neural network that solves the problem of vanishing gradients as it incorporates the back propagation algorithm. [13] Therefore, it is useful for complicated sequence problems that occur in ML (Machine Learning) and can even attain SOTA (state-of-the-art) results. The RNN has memory blocks that are connected through layers, instead of neurons. The memory blocks have components that enable them to store memory for latest sequences. The blocks contain gates that maintain the state and output of the block. The gates use sigmoid activation units to act upon the input sequence to control if the block is triggered or not. [14] This enables the change of state of the block and any addition of information to the block to be conditional. Figure 3 shows the neural map of the network we have developed with a total of 50,797 parameters out of which 50,605 parameters are trainable. From this neural network we have achieved an accuracy of 95.4% during test scenarios as depicted by.

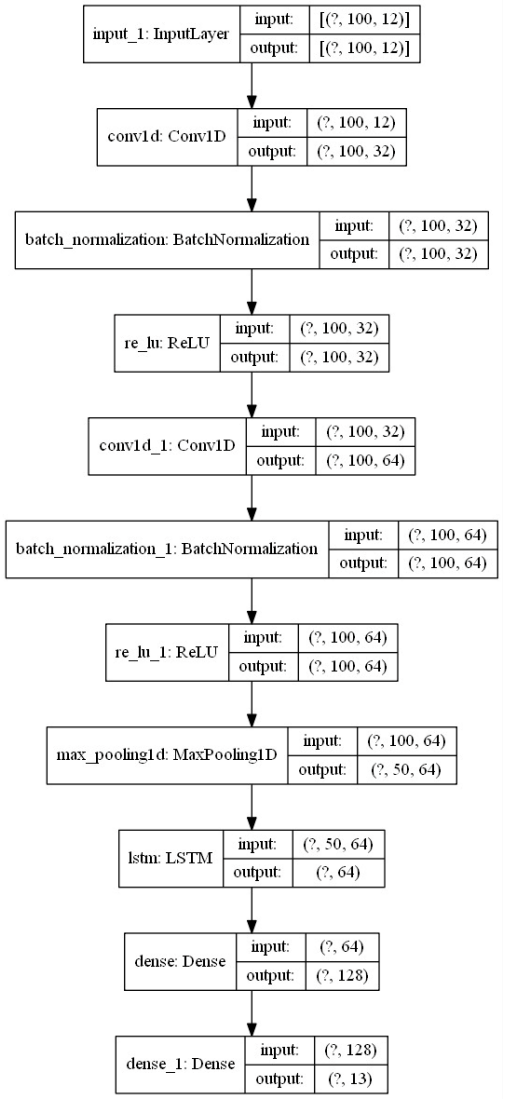


Fig 4.4.1 Neural Network Map

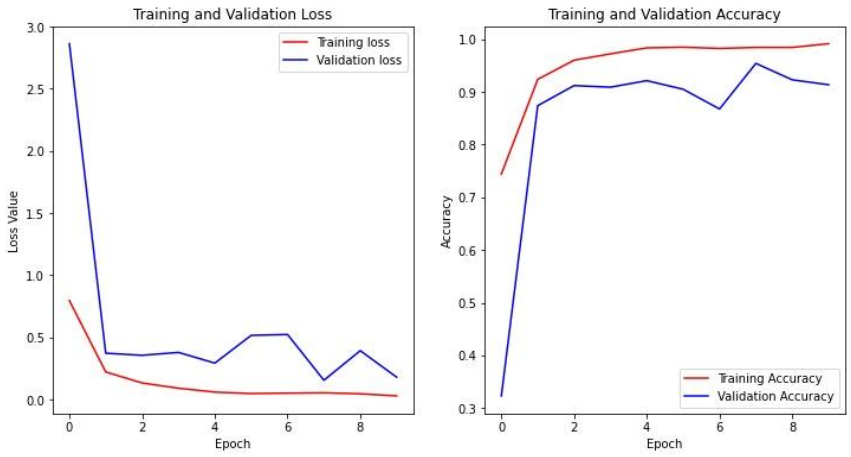


Fig 4.4.2 Model Loss and Accuracy visualization

CHAPTER 5
MODULE IMPLEMENTATION

5.1 Introduction

This chapter focuses on the different modules that comprise the proposed system. Each module is divided into separate subheadings and discussed accordingly. The pseudo code and working methodologies are also included and explained accordingly and diagrams are added whenever required.

The main system to be implemented is a Neural Network model consisting of 10 layers. Each layer serves its own unique purpose in the system, created to best fit the sensor data acquired. This model creation is done using the software TensorFlow. An optimizer is also added so as to reduce the losses incurred; the optimizer chosen for the proposed system is the Adam optimizer.

5.2 Overview of the Platform

The main system is implemented mainly using TensorFlow software. Other modules used are - NumPy, MatPlotLib, and Scikit-Learn. These are further expanded as follows.

5.2.1 TensorFlow

TensorFlow is a ML framework that is used for creating, designing, and training deep learning models. One can use the TensorFlow library to perform various numerical computations, which may not seem very unique in and of them, but the numerical computations are performed using data flow graphs. Nodes in the data flow graphs represent mathematical computations, while the edges represent the information that is exchanged between them, which are typically multidimensional data arrays or tensors. TensorFlow derives its name from the operations performed by neural networks on multidimensional data arrays, or tensors. Using tf.keras, one can quickly create, fit, assess, and use DL models to generate predictions. It facilitates the execution of popular deep learning tasks, such as classification and regression predictive modeling.

.

Another critical component is TensorFlow's scalability. You can write your code and then

execute it on a CPU, a GPU, or a cluster of these devices for the goal of training. Generally, a major portion of the computation is spent training the model. Additionally, the training procedure is repeated numerous times to address any issues that may develop. This procedure consumes more energy, necessitating the use of distributed computing. TensorFlow makes it simple to process massive amounts of data by running the code in a distributed fashion. GPUs, or graphical processing units, have grown in popularity in recent years. Nvidia is a market leader in this segment. It excels at mathematical operations such as matrix multiplication and is a critical component of deep learning.

TensorFlow also integrates with the C++ and Python APIs, which speeds up development significantly. A tensor is a mathematical entity that is represented by higher-dimensional arrays. These arrays of data of varying sizes and ranks are supplied into the neural network as input. This is the case with tensors. You can have one-dimensional arrays or vectors or two-dimensional matrices. However, tensors can be three, four, or five-dimensional. As a result, it aids in consolidating data in a single location and then performs all analysis around it.

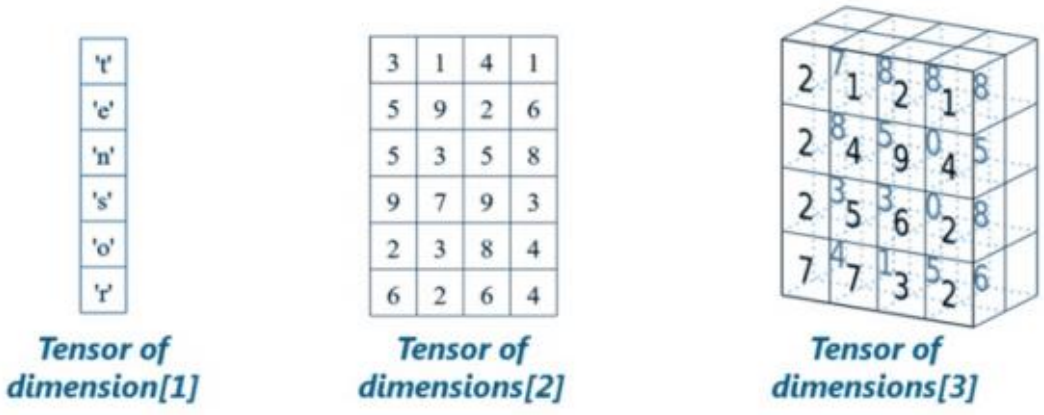


Fig 5.2.1.1 Tensors of different dimensions

5.2.2 Scikit-Learn

Scikit-learn (Sklearn) is one of Python's powerful and comprehensive machine learning library. It serves a set of efficient ML and statistical modeling methods, such as, regression classification, dimensionality reduction, and clustering , through a Python-based interface. This library is centered on NumPy, SciPy, and Matplotlib.

Rather than concentrating on data loading, manipulation, and summarization, the Scikit-learn toolkit focuses on data modeling. Sklearn's most popular model groupings include the following:

- Almost all prominent supervised learning methods, such as Linear Regression, Support Vector Machine (SVM), and Decision Tree, are included in scikit-learn.
- It also includes all the most prominent unsupervised learning algorithms, including factor analysis, PCA, clustering, and unsupervised neural networks.
- Clustering is a technique for grouping data that is not labeled.
- Cross Validation is a technique for determining the accuracy of supervised models on previously unknown data.
- Dimensionality Reduction is used to reduce the number of qualities in data so that it can be summarized, visualized, and selected for features.
- As the name implies, ensemble methods are used to combine the predictions of numerous supervised models.
- It is used to extract features from data in order to define characteristics in picture and text data.
- It is used to identify useful properties for the purpose of creating supervised models.
- It is an open-source library that is also commercially available under the BSD license.

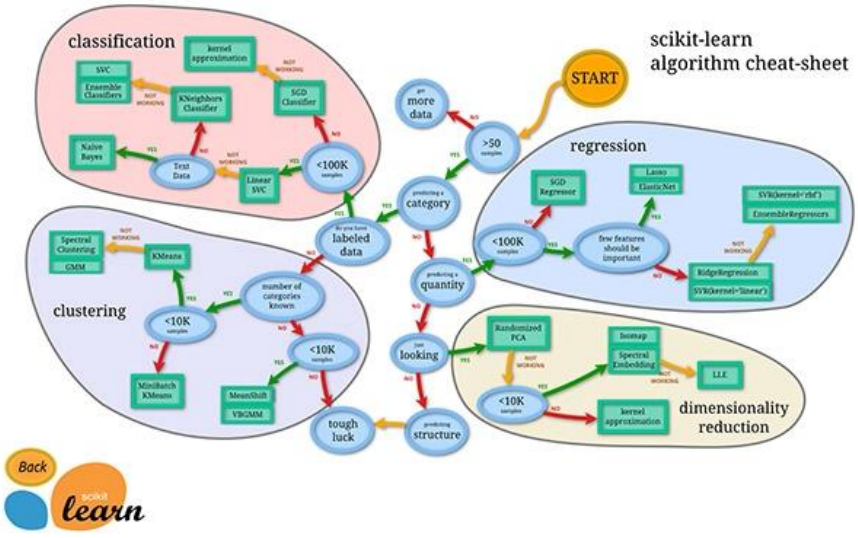


Fig 5.2.2.1 Scikit-Learn features

5.2.3 NumPy

NumPy is an abbreviation for 'Numerical Python'. It is a free Python library that may be used to accomplish a variety of numerical activities. It is a Python library that provides support for large, multi-dimensional arrays and matrices to the said programming language, as well as a vast variety of high-level mathematical functions for manipulating these arrays. Additionally, NumPy is key to the ML stack.

NumPy attempts to give a quicker array object than typical Python lists by up to 50x. NumPy's array object is named ndarray; it comes with a slew of accompanying methods that make dealing with ndarray a breeze. Arrays are commonly employed in data science, where performance and resource efficiency are critical.

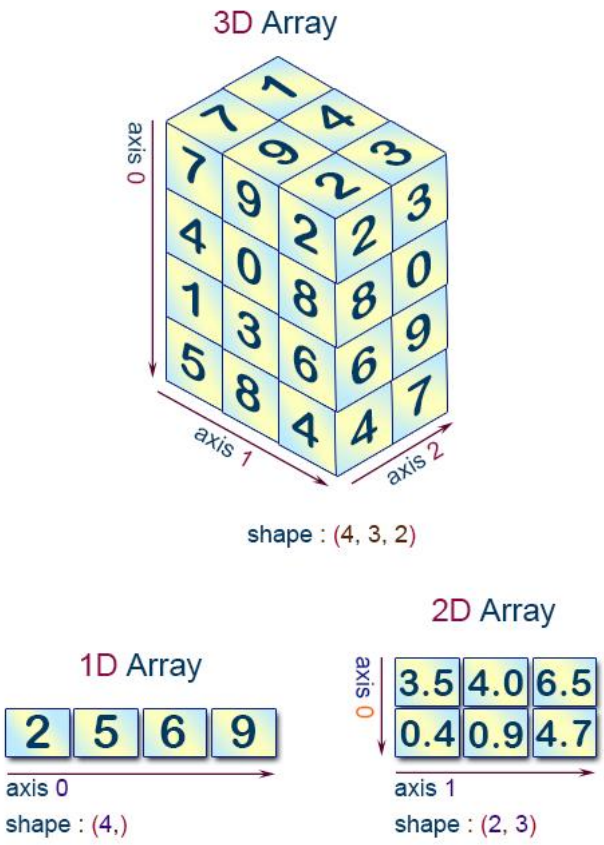


Fig 5.2.3.1 Arrays of different dimensions

5.3 Implementation Details

5.3.1 Introduction

The simulation of the proposed system was carried out using Anaconda and Jupyter Notebook. This is because to begin, Anaconda comes pre-installed with a number of data science packages, so one will be ready to begin working with data immediately. Second, by using conda to manage one's packages and environments, one will be able to avoid future conflicts with the numerous libraries one will be utilizing. Additionally, Jupyter Notebook provides support for more than 40 programming languages, including Python, R, and Julia, to mention a few. It enables users to download the notebook in a variety of file formats, including HTML, Python, PDF, Markdown, and .ipynb.

5.3.2 Anaconda Software

Anaconda is the data science platform of the future for data scientists, information technology professionals, and business executives. It is a Python, R, as well as other programming languages distribution. With over 300 data science packages, it has rapidly become one of the best platforms for any project. Anaconda contributes to the simplification of package management and deployment. It also has a variety of tools for collecting data from a variety of sources using a variety of machine learning and artificial intelligence techniques. It assists in setting up an easily controllable and intuitive environment capable of deploying any project.

Anaconda streamlines the deployment and management of packages. Additionally, it has a plethora of tools for collection of data via artificial intelligence and machine learning algorithms.

Anaconda Navigator is used for the desktop. It is a graphical user interface (GUI) that ships with Anaconda. It enables you to run programs and manage conda packages, environments, and channels without using a command-line interface. It may search for packages either locally or on Anaconda Cloud. You no longer need to input commands in a terminal; Navigator enables you to deal with packages and environments with a single click.

5.3.3 Jupyter Notebook

JupyterLab is a development environment that is interactive. It is used for notebooks, code, and data that is accessible through the internet. Users can configure workflows in data science, machine learning, and scientific computing using the versatile interface. The design, which is modular, encourages the inclusion of modules to extend functionality.

The main web application that is used for creating and sharing computational documents is the Jupyter Notebook. It provides a simplified, direct, and document-centric user experience. A notebook is a document that contains both code and its output. It contains visuals, narrative prose, mathematical formulae, and other rich media. In other words, it's a single document in which the user can execute code, view the output, and include explanations, formulas, and charts to help make your content more transparent, intelligible, repeatable, and shared.

Using Notebooks has become ingrained in the data science workflows of organizations the world over. If your objective is to work using data, adopting a Notebook will accelerate your productivity and make communicating and sharing your discoveries easier. The Jupyter Notebook App is a server-client application that enables the user to modify and execute their notebooks using a web browser. It can be run on a Workstation without Internet connectivity or on a remote server through which it can be accessed through the World Wide Web.

It is composed of two primary components: kernels and a dashboard:

- A kernel is a piece of software that executes and inspects the user's code. The Jupyter Notebook App comes with a kernel that is used for executing code written in Python programming language, but kernels for additional programming languages are also available.
- The application's dashboard not only displays and allows you to reopen notebook pages, but it could also be used to control kernels: you can see which ones are operating and close them down when needed.

5.4 Algorithms

Neural Networks are a series of algorithms that aim to recognize underlying patterns in data in a way that mimics the human brain. It can be used for classification, prediction, and other such purposes. Higher the number of hidden layers and therefore more number of neurons and connections a neural network has the more complex it is. What allows the neural network to learn complicated patterns that are hidden in the data is this complexity. In a neural network, changing the weight of anyone connection has a ripple impact on the wide range of various neurons and their initiations in the ensuing layers. A RNN (Recurrent Neural Network) is a kind of neural network that solves the problem of vanishing gradients as it incorporates the back propagation algorithm.

5.4.1 Recurrent Neural Network

A classic multi layered feed forward network is the basis behind the Recurrent Neural Network (RNN). The RNN is very similar to that of a single layer feed forward network, the difference being the existence of more than one intermediate layer of neurons between the input and output layers in tandem with a feedback loop from the output layer to the input layer of the network, driving the recurrent nature of the network. This also allows for the provision for self loops in the neural network, aiding the performance and accuracy of the network output.

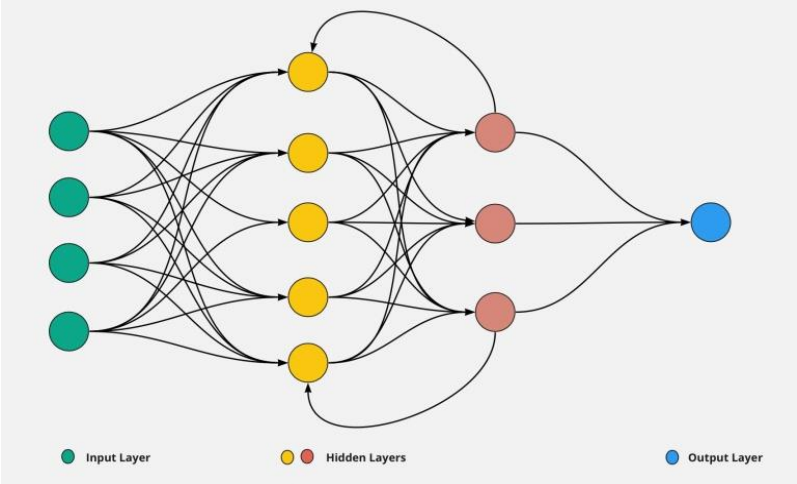


Fig 5.4.1.1 Recurrent Neural Network Architecture

5.5 Implementation Screenshots

This section contains screenshots of the code, graphs, and plots as proof of work.

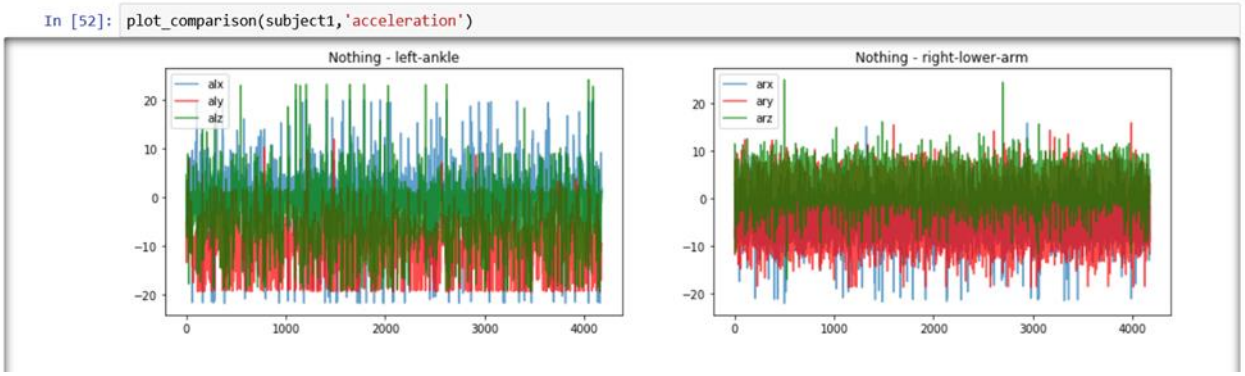


Fig 5.5.1 Acceleration sensor data – Activity: Nothing

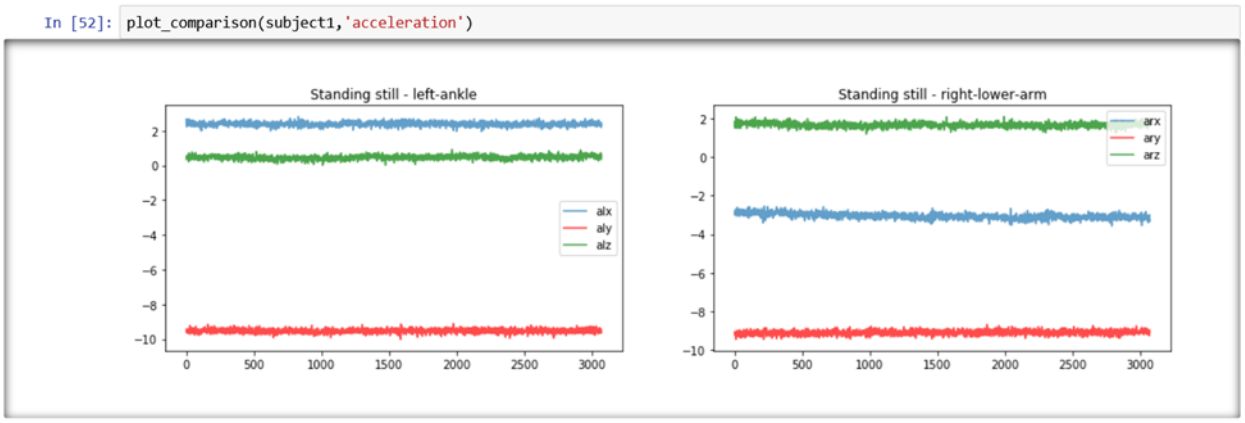


Fig 5.5.2 Acceleration sensor data – Activity: Standing Still

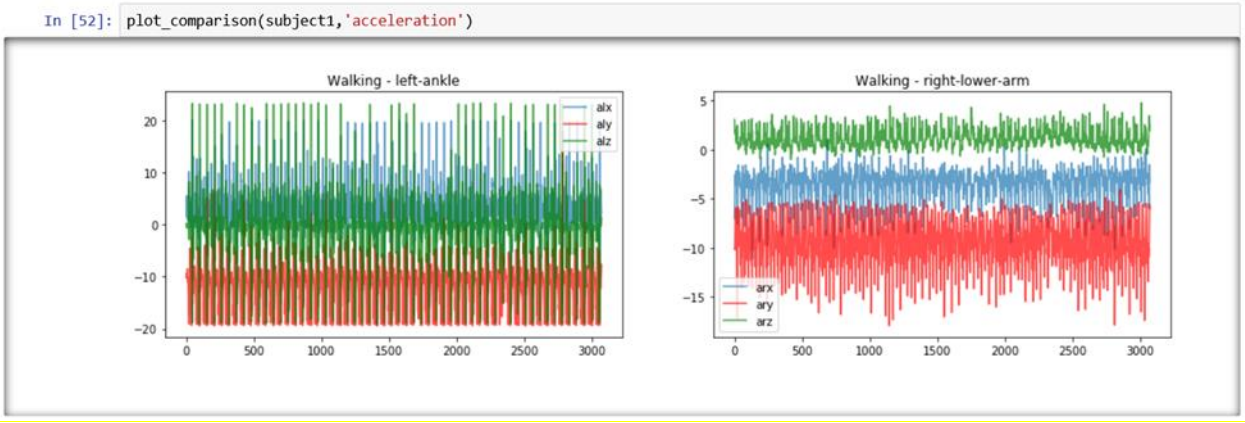


Fig 5.5.3 Acceleration sensor data – Activity: Walkin

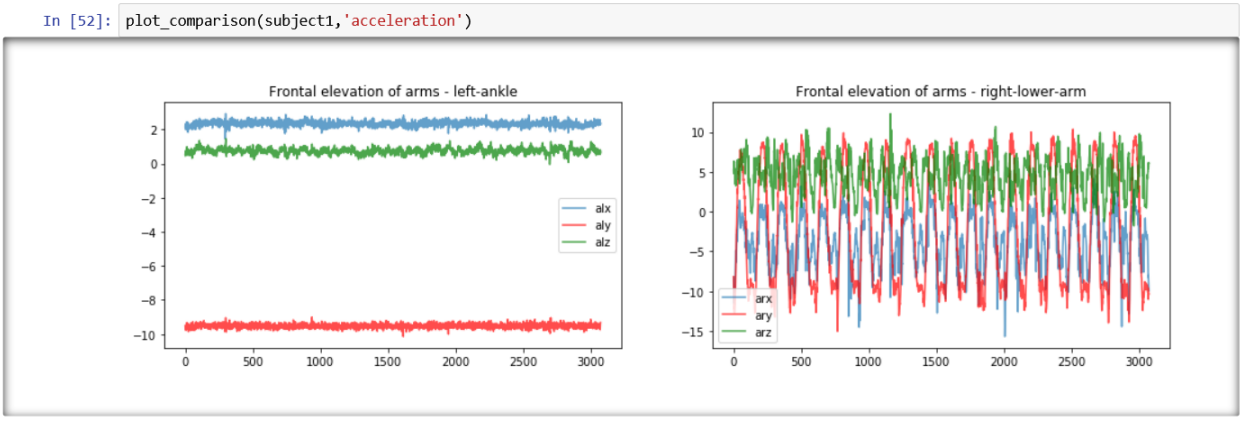


Fig 5.5.4 Acceleration sensor data – Activity: Frontal elevation of arms

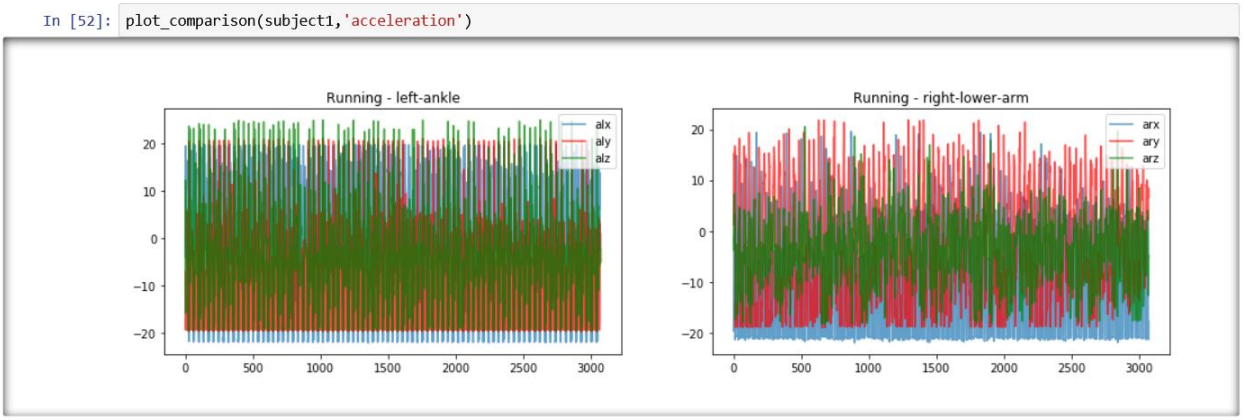


Fig 5.5.5 Acceleration sensor data – Activity: Running

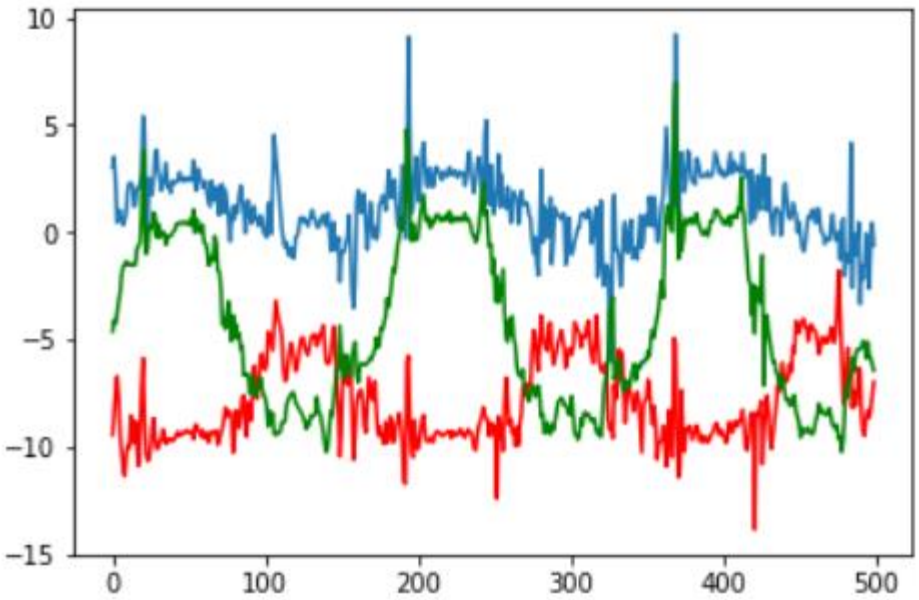


Fig 5.5.6 Acceleration sensor data – Subject 1

```
In [54]: plot_comparison(df)
```

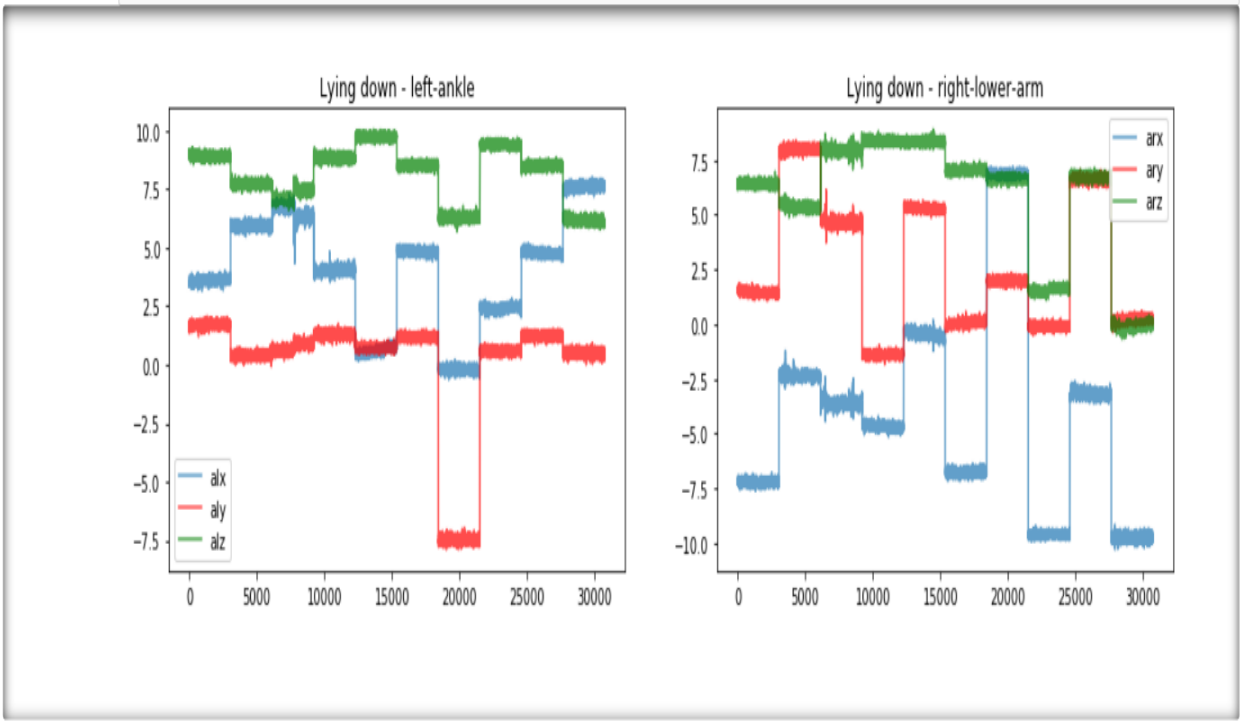


Fig 5.5.7 Sensor data for entire database

```
In [56]: plt.figure(figsize=(8,6))
sns.boxplot(data=df)
plt.show()
```

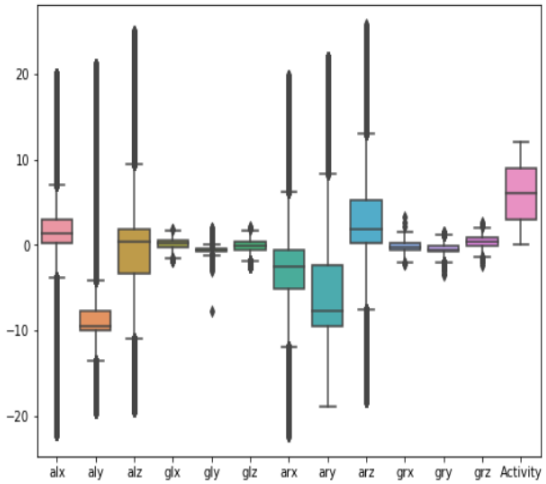


Fig 5.5.8 Box plot of the database

```
In [57]: df.describe().T
```

Out[57]:

	count	mean	std	min	25%	50%	75%	max
alx	372280.0	1.771027	4.172418	-22.1460	0.159087	1.365650	2.883700	20.0440
aly	372280.0	-9.127639	5.101975	-19.6190	-10.078000	-9.606300	-7.722375	21.1610
alz	372280.0	-0.726089	6.355493	-19.3730	-3.368625	0.299375	1.760300	25.0150
glx	372280.0	0.090394	0.463281	-1.8942	-0.354360	0.183670	0.484230	1.7941
gly	372280.0	-0.557061	0.427508	-7.7899	-0.810510	-0.690430	-0.493430	2.0038
glz	372280.0	-0.125207	0.555029	-2.6267	-0.571710	-0.110020	0.349710	2.1022
arx	372280.0	-3.477788	5.780065	-22.3450	-5.162625	-2.532800	-0.601955	19.8010
ary	372280.0	-5.784344	6.493884	-18.9720	-9.542800	-7.712850	-2.373800	21.9650
arz	372280.0	2.383474	4.147604	-18.2390	0.138650	1.830900	5.287000	25.7410
grx	372280.0	-0.209329	0.547731	-2.2392	-0.686270	-0.307840	0.252940	3.2588
gry	372280.0	-0.416437	0.547800	-3.5113	-0.837780	-0.599590	-0.053388	1.5565
grz	372280.0	0.368620	0.522133	-2.3362	-0.056034	0.435340	0.834050	2.6207
Activity	372280.0	5.687055	3.574002	0.0000	3.000000	6.000000	9.000000	12.0000

Fig 5.5.9 Database description

```
In [86]: model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 100, 32)           1184

 batch_normalization (BatchN  (None, 100, 32)          128
 ormalization)

 re_lu (ReLU)                (None, 100, 32)           0

 conv1d_1 (Conv1D)           (None, 100, 64)           6208

 batch_normalization_1 (Batc  (None, 100, 64)          256
 hNormalization)

 re_lu_1 (ReLU)              (None, 100, 64)           0

 max_pooling1d (MaxPooling1D  (None, 50, 64)           0
 )

 lstm (LSTM)                 (None, 64)                33024

 dense (Dense)               (None, 128)               8320

 dense_1 (Dense)             (None, 13)                1677

=================================================================
Total params: 50,797
Trainable params: 50,605
Non-trainable params: 192
```

Fig 5.5.10 Summary of the created model and its layers

```
sns.heatmap(conf_matrix, xticklabels= label_map.values(), yticklabels= label_map.values(), annot=True, fmt="d")
plt.show()
```

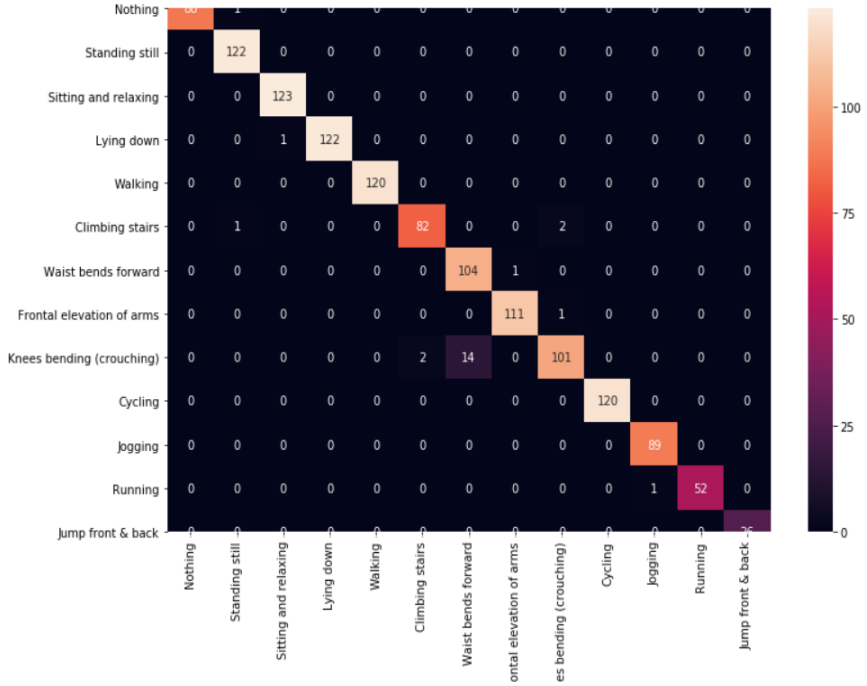


Fig 5.5.11 Confusion Matrix and Heat Map of the predicted data

5.6 Summary

The chapter covered the module implementation and elucidated upon the inner workings of our proposed model. An overview of the platforms used was explained. NumPy, TensorFlow, and Scikit-Learn software were expanded upon and introduced. Then we delved into the implementation of the proposed model starting with the software used to run the model. Namely, Anaconda software and Jupyter notebook were also introduced. The specifics of the algorithms used such as RNN was next explained. Finally, implementation screenshots were provided to further assist the exploration and as proof of work.

CHAPTER 6

RESULT AND DISCUSSION

6.1 Introduction

This chapter will be discussing about the performance analysis with the resultant output.

6.2 Analysis of Result and Output

A 2D hexagonal binning graph is plotted from the acceleration data from the left ankle sensor and the acceleration data from the right lower arm sensor. The data from when the subject is standing still is represented in Figure 5 and the data from when the subject is climbing stairs is represented in Figure 6. By examining both the graphs, it is clear that the static activities like standing still are different and can be separated from dynamic activities like jumping and climbing stairs.

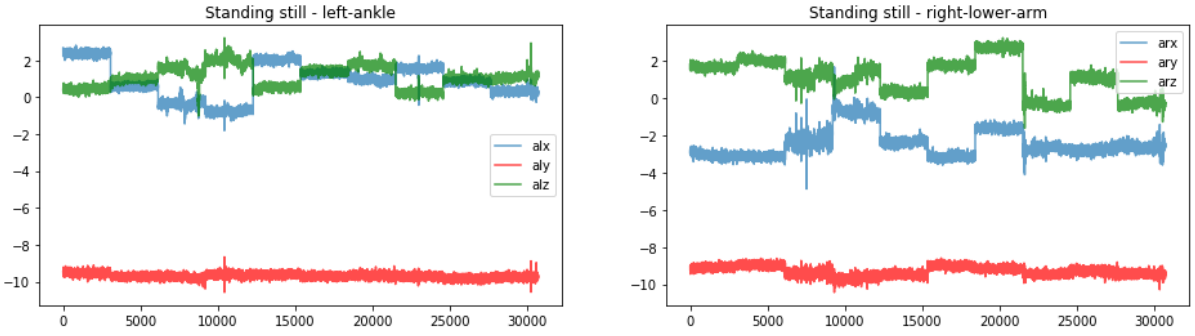


Fig 6.2.1 2D Line graph of data from the sensors while subject stands still

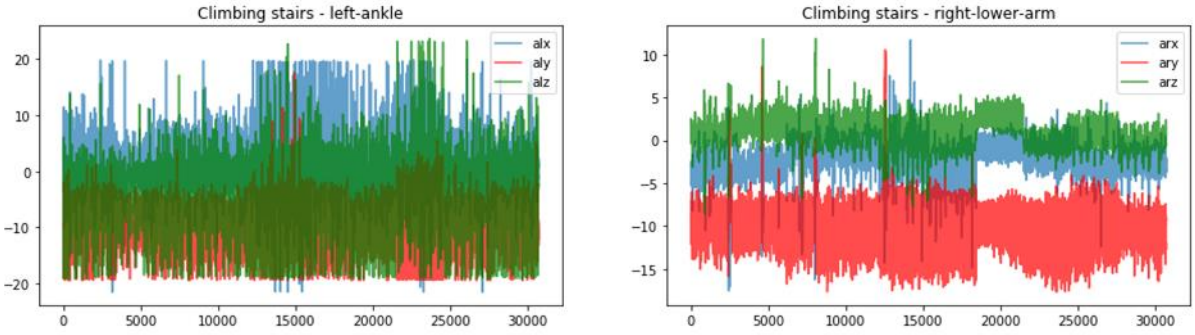


Fig 6.2.2 2D Line graphs of data from the sensors while subject climbs stairs

After understanding the data, it is then possible to apply the data to our model. A confusion matrix is then plotted to evaluate the model. A confusion matrix provides a visual representation of the performance of classification algorithms. The actual and predicted values are plotted against each other such that there are four different combinations - True positive, False positive, False negative and True negative. From the matrix, it is clear that the data is predicted to a good accuracy as the centre diagonal is darker than the surrounding boxes.

6.3 Confusion Matrix

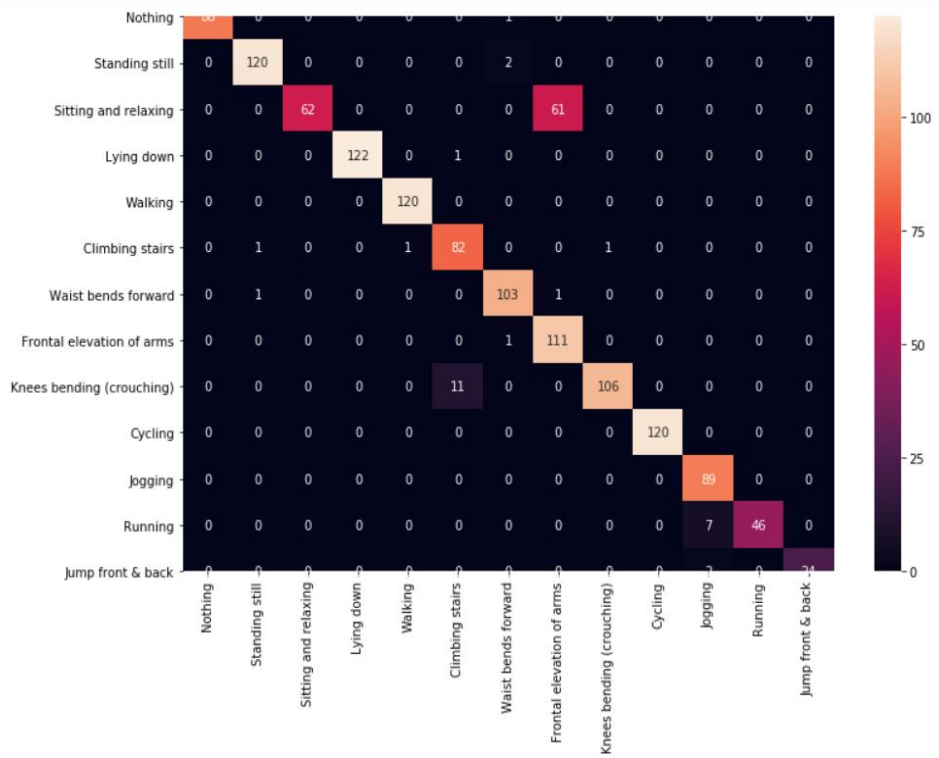


Fig 6.3.1 Confusion Matrix

To further evaluate the model, a classification report is generated. A classification report is a metric used for performance evaluation that displays the recall, f1-score, precision and support of the model. The following conclusions are drawn from the report –

- The accuracy of the model is 98% which indicates that the model performs well on our data. Support represents the number of actual observations of the specific class in the dataset.

- Precision gives the ratio of correctly predicted observations compared to the total predictions. We can see the general trend of precision for the various activities is close to 1, which is ideal.
- Recall is the ratio of correctly predicted observations to that of the total observations. From our results, it is seen that recall is generally higher than 0.8. This is good for this model.
- F1-score is a weighted average of precision and recall. This is generally considered to be a better measure than accuracy. A low F1 score shows poor precision and recall. From the classification report, it is seen that the average score is close to 1.

	precision	recall	f1-score	support
0	1.00	0.99	0.99	89
1	0.98	1.00	0.99	122
2	0.99	1.00	1.00	123
3	1.00	0.99	1.00	123
4	1.00	1.00	1.00	120
5	0.98	0.96	0.97	85
6	0.88	0.99	0.93	105
7	0.99	0.99	0.99	112
8	0.97	0.86	0.91	117
9	1.00	1.00	1.00	120
10	0.99	1.00	0.99	89
11	1.00	0.98	0.99	53
12	1.00	1.00	1.00	26
accuracy			0.98	1284
macro avg	0.98	0.98	0.98	1284
weighted avg	0.98	0.98	0.98	1284

Fig 6.3.2 Classification Report

CHAPTER 7
CONCLUSION AND FUTURE WORK

7.1 Conclusion

A novel dataset was introduced that can be used by benchmarking algorithms to detect and classify upper-limb mobility compensatory movements during stroke rehabilitation exercises. The benchmarking algorithms can be used for providing real-time automated stroke rehabilitation coaching and also can be merged with rehabilitation robots to assess and correct compensatory movements during training exercises in stroke victims. A baseline model was presented that is used to demonstrate the usefulness of the dataset. The angle of the trunk vector or the shoulder vector is compared to the compensatory movements to examine the results. The specificity and sensitivity of the classification were also shown.

7.2 Future Work

Our dataset focuses on the angle of the trunk vector and shoulder vector to compute compensatory movement. This can be expanded upon by including multiple other reference points as we move beyond upper limb compensatory movements. Another potential avenue to explore is the baseline algorithm. It can be further improved by using a novel algorithm or an ensemble model that is curated to the given stroke rehabilitation data. This would perform better and show improved accuracy and results.

REFERENCES

[1] J. H. van der Lee, R. C. Wagenaar, G. J. Lankhorst, T. W. Vogelaar, W. L. Deville, and L. M. Bouter, "Forced use of the upper extremity in chronic stroke patients: Results from a single-blind randomized clinical trial," Stroke, vol. 30, no. 11, pp. 23692375, Nov. 1999.

[2] C. Patten, J. Lexell, and E. Brown, "Weakness and strength training in persons with poststroke hemiplegia: Rationale, method, and efcacy," J. Rehabil. Res. Dev., vol. 41, pp. 293312, Dec. 2004.

[3] D. J. Lipomi et al., "Skin-like pressure and strain sensors based on transparent elastic lms of carbon nanotubes," Nature Nanotechnology., vol. 6, no. 12, pp. 788792, Dec. 2011.

[4] Khalid, Samina & Khalil, Tehmina & Nasreen, Shamila. (2014). A survey of feature selection and feature extraction techniques in machine learning. Proceedings of 2014 Science and Information Conference, SAI 2014. 372-378. 10.1109/SAI.2014.6918213.

[5] Khalid, Samina & Khalil, Tehmina & Nasreen, Shamila. (2014). A survey of feature selection and feature extraction techniques in machine learning. Proceedings of 2014 Science and Information Conference, SAI 2014. 372-378. 10.1109/SAI.2014.6918213

[6] E. J. Benjamin, M. J. Blaha, S. E. Chiuve, M. Cushman, S. R. Das, and R. Deo, "Heart disease and stroke statistics2017 update: A report from the American Heart Association," circulation, vol. 135, no. 10, pp. e146e603, 2017.

[7] husnejahan,"https://github.com/husnejahan/Multivariate-Time-series-Analysis-using-LSTM-ARIMA/blob/master/AirQualityEDA_LSTM.ipynb", 20 Sept 2019.

[8] Sarmento RM, Vasconcelos FFX, Filho PPR, Wu W, de Albuquerque VHC. Automatic Neuroimage Processing and Analysis in Stroke-A Systematic Review. IEEE Rev Biomed Eng. 2020;13:130-155. doi: 10.1109/RBME.2019.2934500. Epub 2019 Aug 23. PMID: 31449031.

[9]"https://www.projectpro.io/article/8-feature-engineering-techniques-for-machine-learning/423", 04 April 2022

[10]"https://www.maiango.com/post/forecasting-time-series-data-using-lstm-recurrent-neural-networks/", 01 May 2020

[11] Toğaçar, M., Z. Cömert, And B. Ergen, Classification Of Brain Mri Using Hyper Column Technique With Convolutional Neural Network And Feature Selection Method.

Expert Systems With Applications, 2020. 149: P. 113274.

[12] M. S. Pathan, Z. Jianbiao, D. John, A. Nag and S. Dev, "Identifying Stroke Indicators Using Rough Sets," in IEEE Access, vol. 8, pp. 210318-210327, 2020, doi: 10.1109/ACCESS.2020.3039439.

[13] Cai S, Li G, Su E, Wei X, Huang S, Ma K, Zheng H, Xie L. Real-Time Detection of Compensatory Patterns in Patients With Stroke to Reduce Compensation During Robotic Rehabilitation Therapy. IEEE J Biomed Health Inform. 2020 Sep;24(9):2630-2638. doi: 10.1109/JBHI.2019.2963365. Epub 2020 Jan 1. PMID: 31902785.

[14] Anis Fatema, Surya Poondla, Rishabh B. Mishra, and Aftab M. Hussain, “A Low-Cost Pressure Sensor Matrix for Activity Monitoring in Stroke Patients Using Artificial Intelligence”, IEEE Sensors Journal, Vol. 21, No. 7, April 1, 2021

APPENDICES

The project has covered and executed the concept of neural networks presented to measure the positioning accuracy of stroke sufferers. To begin, a thorough examination of the existing system's benefits and drawbacks is conducted. Second, the implementation module provides us with an overview of the processing operation. This gives a good sense of the neural network concept and the technology that go along with it.a baseline model was presented in order to demonstrate the dataset's utility. To analyze the findings, the angle of the trunk vector or the shoulder vector is compared to the compensatory motions. The classification's specificity and sensitivity were also demonstrated. Thus, the code and concept of the project has been executedand processed to get output successfully.

Sample Coding

```
In [1]: import warnings
        warnings.filterwarnings(action='ignore')

In [2]: import os

In [3]: import numpy as np

In [4]: import pandas as pd

In [5]: import time

In [6]: import matplotlib.pyplot as plt

In [7]: %matplotlib inline

In [8]: import seaborn as sns

In [9]: import plotly.express as px

In [10]: import plotly.graph_objects as go

In [11]: from sklearn.model_selection import train_test_split

In [12]: from sklearn.preprocessing import StandardScaler

In [13]: from sklearn.linear_model import LogisticRegression
```

In [18]:
```
from sklearn.metrics import confusion_matrix, classification_report
```

In [97]:
```
from tensorflow.keras.models import Sequential, load_model
```

In [20]:
```
from tensorflow.keras.layers import Input, BatchNormalization, Conv1D,ReLU, LSTM, MaxPool1D, Dense
```

In [21]:
```
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
```

In [22]:
```
tf.random.set_seed(42)
```

In [23]:
```
df = pd.read_csv('Stroke-Rehab-Dataset.csv')
```

In [24]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1215745 entries, 0 to 1215744
Data columns (total 14 columns):
alx         1215745 non-null float64
aly         1215745 non-null float64
alz         1215745 non-null float64
glx         1215745 non-null float64
gly         1215745 non-null float64
glz         1215745 non-null float64
arx         1215745 non-null float64
ary         1215745 non-null float64
arz         1215745 non-null float64
grx         1215745 non-null float64
gry         1215745 non-null float64
grz         1215745 non-null float64
Activity    1215745 non-null int64
subject     1215745 non-null object
dtypes: float64(12), int64(1), object(1)
memory usage: 129.9+ MB
```

In [25]:
```
df.dtypes
```

Out[25]:
```
alx         float64
aly         float64
alz         float64
glx         float64
gly         float64
glz         float64
arx         float64
ary         float64
arz         float64
grx         float64
gry         float64
grz         float64
Activity      int64
subject      object
dtype: object
```

In [26]:
```
df.head()
```

Out[26]:

	alx	aly	alz	glx	gly	glz	arx	ary	arz	grx	gry	grz	Activity	subject
0	2.1849	-9.6967	0.63077	0.103900	-0.84053	-0.68762	-8.6499	-4.5781	0.187760	-0.44902	-1.0103	0.034483	0	subject1
1	2.3876	-9.5080	0.68389	0.085343	-0.83865	-0.68369	-8.6275	-4.3198	0.023595	-0.44902	-1.0103	0.034483	0	subject1
2	2.4086	-9.5674	0.68113	0.085343	-0.83865	-0.68369	-8.5055	-4.2772	0.275720	-0.44902	-1.0103	0.034483	0	subject1
3	2.1814	-9.4301	0.55031	0.085343	-0.83865	-0.68369	-8.6279	-4.3163	0.367520	-0.45686	-1.0082	0.025862	0	subject1
4	2.4173	-9.3889	0.71098	0.085343	-0.83865	-0.68369	-8.7008	-4.1459	0.407290	-0.45686	-1.0082	0.025862	0	subject1

In [27]:
```
df.sample(10)
```

Out[27]:

	alx	aly	alz	glx	gly	glz	arx	ary	arz	grx	gry	grz	Activity	subject
422482	4.65890	-4.23580	-7.73810	-0.615960	-0.15947	-1.695500	1.345400	-7.4802	10.980000	-0.680390	-0.38604	2.05600	0	subject4
65360	2.74540	-9.32740	0.92567	0.176250	-0.64540	0.685660	-3.146200	-9.3833	0.666250	-0.862750	-0.56674	0.40948	0	subject1
881759	1.24880	-9.75490	-2.22530	0.886830	0.31520	0.058939	-8.010000	-2.4153	5.473000	-0.023529	1.08620	-0.56250	0	subject8
715941	2.27340	-9.50970	1.50560	0.458260	-0.55535	0.557960	-3.150200	-9.0033	1.674800	-0.276470	-0.62423	0.90302	0	subject6
546284	0.73388	0.60199	9.70410	0.087199	0.95122	0.809430	-0.553670	5.3733	8.331200	0.631370	0.44559	0.76724	0	subject5
787846	1.70340	-9.72890	-0.61525	0.647500	-0.58349	-0.449900	-5.631000	-3.8302	5.975200	0.123530	-0.59343	0.93103	0	subject7
513485	-1.00720	-8.90570	-1.46580	-0.645640	-0.61726	-0.630650	-2.459400	-10.1600	0.033684	-0.758820	-0.72279	-0.20259	0	subject4
496438	2.23760	-9.20460	2.31210	-0.502780	-0.83865	-0.220040	0.077416	-9.5307	1.929700	-0.247060	-0.98768	-0.32112	0	subject4
954619	4.01920	-10.38000	-2.33300	-0.320960	-0.81426	0.422400	-17.231000	-7.4516	-4.164200	-0.588240	-0.13758	-0.81897	0	subject8
686795	4.12890	-6.98420	-6.33920	-0.020408	-0.25328	-1.082500	-5.690300	-18.0730	2.700500	0.509800	-0.95072	-0.32974	0	subject6

In [28]:
```
df.isnull().sum()
```

Out[28]:
```
alx         0
aly         0
alz         0
glx         0
gly         0
glz         0
arx         0
ary         0
arz         0
grx         0
gry         0
grz         0
Activity    0
subject     0
```

In [29]:
```
df.duplicated().sum()
```

Out[29]:
```
0
```

In [30]:
```
vc = df["Activity"].value_counts()
```

In [31]:
```
vc
```

Out[31]:
```
0     872550
11     30720
10     30720
9      30720
5      30720
4      30720
3      30720
2      30720
1      30720
7      29441
8      29337
6      28315
12     10342
Name: Activity, dtype: int64
```

In [33]:
```
df_majority = df[df.Activity==0]
df_minorities = df[df.Activity!=0]
```

In [34]:
```
df_majority_downsampled = resample(df_majority,n_samples=30000, random_state=42)
df = pd.concat([df_majority_downsampled, df_minorities])
```

In [35]:
```
vc = df["Activity"].value_counts()
```

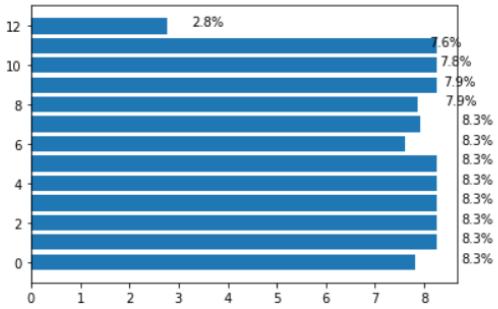
In [36]:
```
vc
```

Out[36]:
```
11    30720
10    30720
9     30720
5     30720
4     30720
3     30720
2     30720
1     30720
0     30000
7     29441
8     29337
6     28315
12    10342
Name: Activity, dtype: int64
```

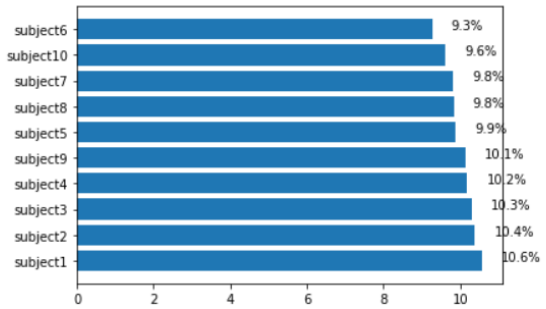
In [43]:
```
def plot_comparison(data, metric = 'acceleration'):
    metric = metric[0].lower()
    data = data

    for i in range(0,13):
        plt.figure(figsize=(16,4))

        plt.subplot(1,2,1)
        plt.plot(data[ data['Activity']==i ].reset_index(drop=True)[metric+'lx'], alpha=.7, label=metric+'lx')
        plt.plot(data[ data['Activity']==i ].reset_index(drop=True)[metric+'ly'],color='red', alpha=.7, label=metric+'ly')
        plt.plot(data[ data['Activity']==i ].reset_index(drop=True)[metric+'lz'],color='green', alpha=.7, label=metric+'lz')
        plt.title(f'{label_map[i]} - left-ankle')
        plt.legend()

        plt.subplot(1,2,2)
        plt.plot(data[ data['Activity']==i ].reset_index(drop=True)[metric+'rx'], alpha=.7, label=metric+'rx')
        plt.plot(data[ data['Activity']==i ].reset_index(drop=True)[metric+'ry'],color='red', alpha=.7, label=metric+'ry')
        plt.plot(data[ data['Activity']==i ].reset_index(drop=True)[metric+'rz'],color='green', alpha=.7, label=metric+'rz')
        plt.title(f'{label_map[i]} - right-lower-arm')
        plt.legend()

        plt.show()
        print()
```

In [44]:
```
def plot_category(data,cat):
    array = (data[cat].value_counts().sort_values(ascending=False)/len(data))*100
    plt.barh(array.index, width = array.values)
    for index, value in enumerate(array.values):
        plt.text(value + .5 , index, s= '{:.1f}%'.format(value))
    plt.show()
```

In [45]:
```
plot_category(df,'Activity')
plt.show()
```



In [46]:
```
plot_category(df,'subject')
```



In [57]:
```
df.describe().T
```

Out[57]:

	count	mean	std	min	25%	50%	75%	max
alx	372280.0	1.771027	4.172418	-22.1460	0.159087	1.365650	2.883700	20.0440
aly	372280.0	-9.127639	5.101975	-19.6190	-10.078000	-9.606300	-7.722375	21.1610
alz	372280.0	-0.726089	6.355493	-19.3730	-3.368625	0.299375	1.760300	25.0150
glx	372280.0	0.090394	0.463281	-1.8942	-0.354360	0.183670	0.484230	1.7941
gly	372280.0	-0.557061	0.427508	-7.7899	-0.810510	-0.690430	-0.493430	2.0038
glz	372280.0	-0.125207	0.555029	-2.6267	-0.571710	-0.110020	0.349710	2.1022
arx	372280.0	-3.477788	5.780065	-22.3450	-5.162625	-2.532800	-0.601955	19.8010
ary	372280.0	-5.784344	6.493884	-18.9720	-9.542800	-7.712850	-2.373800	21.9650
arz	372280.0	2.383474	4.147604	-18.2390	0.138650	1.830900	5.287000	25.7410
grx	372280.0	-0.209329	0.547731	-2.2392	-0.686270	-0.307840	0.252940	3.2588
gry	372280.0	-0.416437	0.547800	-3.5113	-0.837780	-0.599590	-0.053388	1.5565
grz	372280.0	0.368620	0.522133	-2.3362	-0.056034	0.435340	0.834050	2.6207
Activity	372280.0	5.687055	3.574002	0.0000	3.000000	6.000000	9.000000	12.0000

In [58]:
```
def preprocess_inputs(df):
    df = df.copy()
    df = df.drop('subject', axis=1)

    samples = []
    for category in df['Activity'].unique():
        category_slice = df.query("Activity == @category")
        samples.append(category_slice.sample(2000, random_state=1))
    df = pd.concat(samples, axis=0).sample(frac=1.0, random_state=1).reset_index(drop=True)

    y = df['Activity']
    X = df.drop('Activity', axis=1)
    X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, shuffle=True, random_state=1)
    scaler = StandardScaler()
    scaler.fit(X_train)
    X_train = pd.DataFrame(scaler.transform(X_train), index=X_train.index, columns=X_train.columns)
    X_test = pd.DataFrame(scaler.transform(X_test), index=X_test.index, columns=X_test.columns)

    return X_train, X_test, y_train, y_test
```

In [59]:
```
X_train, X_test, y_train, y_test = preprocess_inputs(df)
```

In [60]:
```
X_train
```

Out[60]:

	alx	aly	alz	glx	gly	glz	arx	ary	arz	grx	gry	grz
11053	0.188019	0.006462	-0.227579	-1.505898	-0.703738	0.614441	0.433045	-0.328323	-0.338820	-1.321804	-0.164647	-1.171589
12600	0.012763	-0.068942	-0.629326	0.902507	1.395344	-0.904193	0.003734	-1.373747	0.030758	1.371505	-0.640965	0.102497
7364	-0.741512	-0.330751	-0.072983	-0.122258	-0.650154	0.948743	1.096023	-0.167813	0.934700	1.605071	0.150408	1.140491
4806	-0.309668	-0.370427	0.017870	1.034105	-0.440246	0.578869	-0.095336	-0.487691	-0.255131	-0.810878	-0.438423	0.513637
19271	-0.037446	-0.133903	-0.039918	1.273336	-0.310725	0.244559	0.106195	0.075244	-0.344438	-0.051795	-0.678464	0.835195
...	...	...	...	...	...	...	...	...	...	...	...	...

In [70]:
```
for feature in new_df.columns[:-2]:
    lower_range = np.quantile(df[feature],0.01)
    upper_range = np.quantile(df[feature],0.99)
    print(feature,'range:',lower_range,'to',upper_range)
    new_df = new_df.drop(new_df[(new_df[feature]>upper_range) | (new_df[feature]<lower_range)].index, axis=0)
    print('shape',new_df.shape)
```

```
alx range: -11.481209999999999 to 19.234
shape (364839, 14)
aly range: -19.379 to 2.452431000000023
shape (359129, 14)
alz range: -18.95 to 14.181420000000042
shape (355387, 14)
glx range: -0.74212 to 0.80705
shape (348542, 14)
gly range: -1.0675 to 0.9662300000000001
shape (341987, 14)
glz range: -1.1061 to 0.82908
shape (336551, 14)
arx range: -21.492 to 9.104067000000057
shape (331475, 14)
ary range: -18.694000000000006 to 11.956630000000063
shape (325419, 14)
arz range: -10.38084 to 11.822
shape (322860, 14)
grx range: -1.0196 to 0.95686
shape (319392, 14)
gry range: -1.1417 to 0.9076
shape (314567, 14)
grz range: -0.69828 to 1.125
shape (310179, 14)
```

In [71]:
```
train = new_df[(new_df['subject'] != 'subject10') & (new_df['subject'] != 'subject9')]
```

In [74]:
```
X_train = train.drop(['Activity','subject'],axis=1)
```

In [75]:
```
y_train = train['Activity']
```

In [76]:
```
X_test = test.drop(['Activity','subject'],axis=1)
```

In [77]:
```
y_test = test['Activity']
```

In [78]:
```
X_train.shape,y_train.shape,X_test.shape,y_test.shape
```

Out[78]:
```
((245921, 12), (245921,), (64258, 12), (64258,))
```

In [79]:
```
def create_dataset(X, y, time_steps, step=1):
    Xs, ys = [], []
    for i in range(0, len(X) - time_steps, step):
        x = X.iloc[i:(i + time_steps)].values
        labels = y.iloc[i: i + time_steps]
        Xs.append(x)
        ys.append(stats.mode(labels)[0][0])
    return np.array(Xs), np.array(ys).reshape(-1, 1)
```

In [80]:
```
X_train,y_train = create_dataset(X_train, y_train, 100, step=50)
```

In [81]:
```
X_train.shape, y_train.shape
```

Out[81]:
```
((4917, 100, 12), (4917, 1))
```

In [82]:
```
X_test,y_test = create_dataset(X_test, y_test, 100, step=50)
```

In [83]:
```
X_test.shape, y_test.shape
```

Out[83]:
```
((1284, 100, 12), (1284, 1))
```

In [84]:
```
def create_model():
    model = Sequential()
    model.add(Input(shape=[100,12]))
    model.add(Conv1D(filters=32, kernel_size=3, padding="same"))
    model.add(BatchNormalization())
    model.add(ReLU())
    model.add(Conv1D(filters=64, kernel_size=3, padding="same"))
    model.add(BatchNormalization())
    model.add(ReLU())
    model.add(MaxPool1D(2))
    model.add(LSTM(64))
    model.add(Dense(units=128, activation='relu'))
    model.add(Dense(13, activation='softmax'))
    return model
```

In [85]:
```
model = create_model()
```

In [89]:
```
model.compile(optimizer="adam", loss="sparse_categorical_crossentropy", metrics=["sparse_categorical_accuracy"],)
```

In [90]:
```
model_history = model.fit(X_train,y_train, epochs= 10, validation_data=(X_test,y_test), callbacks=callbacks)
```

```
Epoch 1/10
154/154 [==============================] - 5s 22ms/step - loss: 0.7968 - sparse_categorical_accuracy: 0.7425 - val_loss: 2.2078
- val_sparse_categorical_accuracy: 0.3941
Epoch 2/10
154/154 [==============================] - 3s 18ms/step - loss: 0.1707 - sparse_categorical_accuracy: 0.9492 - val_loss: 0.1721
- val_sparse_categorical_accuracy: 0.9276
Epoch 3/10
154/154 [==============================] - 3s 18ms/step - loss: 0.1494 - sparse_categorical_accuracy: 0.9540 - val_loss: 0.1769
- val_sparse_categorical_accuracy: 0.9470
Epoch 4/10
154/154 [==============================] - 3s 18ms/step - loss: 0.0828 - sparse_categorical_accuracy: 0.9738 - val_loss: 0.3075
- val_sparse_categorical_accuracy: 0.8910
Epoch 5/10
154/154 [==============================] - 3s 18ms/step - loss: 0.0618 - sparse_categorical_accuracy: 0.9780 - val_loss: 0.3008
- val_sparse_categorical_accuracy: 0.9081
Epoch 6/10
154/154 [==============================] - 3s 18ms/step - loss: 0.0467 - sparse_categorical_accuracy: 0.9868 - val_loss: 0.1159
- val_sparse_categorical_accuracy: 0.9759
Epoch 7/10
154/154 [==============================] - 3s 18ms/step - loss: 0.0480 - sparse_categorical_accuracy: 0.9852 - val_loss: 0.1450
- val_sparse_categorical_accuracy: 0.9626
Epoch 8/10
154/154 [==============================] - 3s 19ms/step - loss: 0.0818 - sparse_categorical_accuracy: 0.9756 - val_loss: 0.0558
- val_sparse_categorical_accuracy: 0.9813
Epoch 9/10
154/154 [==============================] - 3s 18ms/step - loss: 0.0401 - sparse_categorical_accuracy: 0.9878 - val_loss: 0.2844
- val_sparse_categorical_accuracy: 0.9533
Epoch 10/10
154/154 [==============================] - 3s 18ms/step - loss: 0.0273 - sparse_categorical_accuracy: 0.9917 - val_loss: 0.3394
- val_sparse_categorical_accuracy: 0.8855
```

In [91]:
```
train_loss = model_history.history['loss']
```

In [92]:
```
val_loss = model_history.history['val_loss']
```

In [93]:
```
train_accuracy = model_history.history['sparse_categorical_accuracy']
```

In [94]:
```
val_accuracy = model_history.history['val_sparse_categorical_accuracy']
```

In [95]:
```
plt.figure(figsize=(12,6))

plt.subplot(1,2,1)
plt.plot(train_loss, 'r', label='Training loss')
plt.plot(val_loss, 'b', label='Validation loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss Value')
plt.legend()

plt.subplot(1,2,2)
plt.plot(train_accuracy, 'r', label='Training Accuracy')
plt.plot(val_accuracy, 'b', label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

plt.show()
```

In [98]:
```
model = load_model('./rehab_best.h5')
```

In [99]:
```
train_loss, train_acc = model.evaluate(X_train,y_train)
test_loss, test_acc = model.evaluate(X_test,y_test)
```

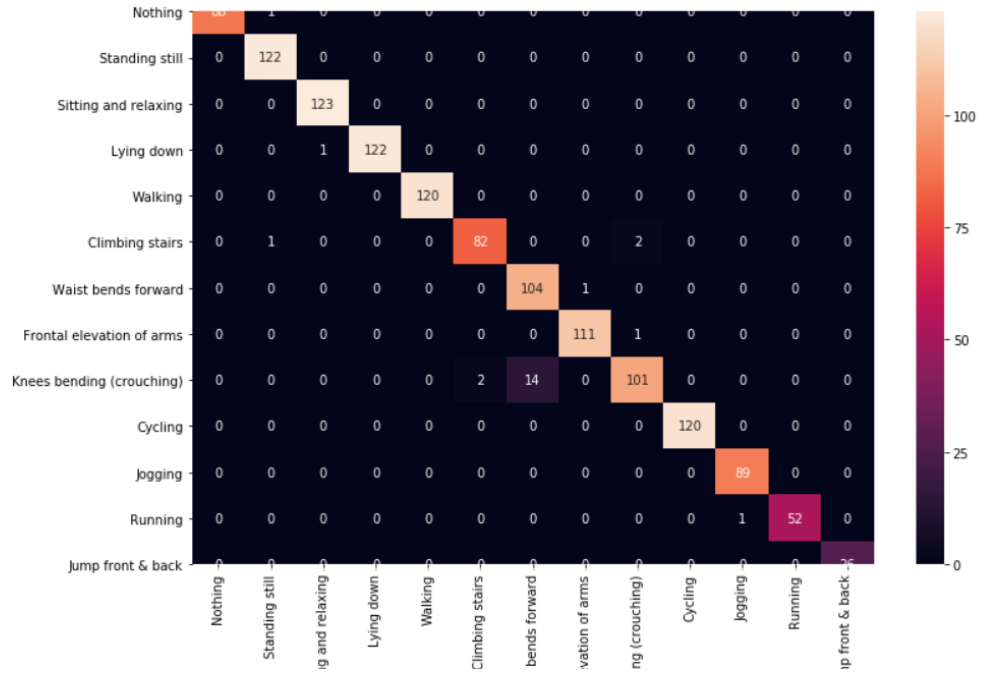
```
154/154 [==============================] - 2s 8ms/step - loss: 0.0192 - sparse_categorical_accuracy: 0.9957
41/41 [==============================] - 0s 7ms/step - loss: 0.0558 - sparse_categorical_accuracy: 0.9813
```

In [100]:
```
print("Train accuracy", round(train_acc*100, 2),'%')
print("Train loss", train_loss)
print("Test accuracy", round(test_acc*100, 2),'%')
print("Test loss", test_loss)
```

```
Train accuracy 99.57 %
Train loss 0.019175739958882332
Test accuracy 98.13 %
Test loss 0.055846549570560455
```

In [101]:
```
pred = model.predict(X_test)
pred = np.argmax(pred, axis = 1)
pred = pred.reshape(-1,1)
```

In [103]:
```
plt.figure(figsize=(12,8))
conf_matrix = confusion_matrix(y_test,pred)
sns.heatmap(conf_matrix, xticklabels= label_map.values(), yticklabels= label_map.values(), annot=True, fmt="d")
plt.show()
```



SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Deemed to be University u / s 3 of UGC Act, 1956)

Office of Controller of Examinations

REPORT FOR PLAGIARISM CHECK ON THE DISSERTATION / PROJECT REPORT FOR UG / PG PROGRAMMES
(To be attached in the dissertation / project report)

1	Name of the Candidate **(IN BLOCKLETTERS)**	MRUDHULA RAYA MUKHIL SUNDARARAJ GOWTHAMAN S.DHAARINI
2	Address of Candidate	Chennai 600091 Chennai 600087 Chennai 600125 **Mobile Number:** 9952921945, 8825890075, 8778026064
3	Registration Number	RA1811003020322, RA1811003020276, RA1811003020299
4	Date of Birth	03/02/2001, 17/01/2001, 27/12/2000
5	Department	Computer Science and Engineering
6	Faculty	Ms. SWATHI R
7	Title of the Dissertation / Project	MEASURING STROKE REHABILITATION BY POSTURE DETECTION USING NEURAL NETWORKS
8	Whether the above project / dissertation isdone by	Individual or group : Group (Strike whichever is not applicable) a) If the project / dissertation is done in group, then how many students together completed the project : #03 b) Mention the Name & Register number of other candidates : MRUDHULA RAYA [RA1811003020322] MUKHIL SUNDARARAJ GOWTHAMAN [RA1811003020276] S.DHAARINI [RA1811003020299]
9	Name and address of the Supervisor /Guide	Ms. SWATHI R, Assistant Professor, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Ramapuram Campus, Chennai 89 **Mail ID:** swathir@srmist.edu.in

		Mobile Number : 87542 30477		
10	Name and address of the Co-Supervisor /Guide	NA **Mail ID:** NA **Mobile Number:** NA		
11	Software Used	Turnitin		
12	Date of Verification	06/05/2022		
13	**Plagiarism Details : (to attach the final report from the software)**			
Chapter	**Title of the Report**	**Percentage of similarity index (including self citation)**	**Percentage of similarity index (Excluding self citation)**	**% of plagiarism after excluding Quotes, Bibliography, etc.,**
1	MEASURING STROKE REHABILITATION BY POSTURE DETECTION USING NEURAL NETWORKS	NA	NA	9%
Appendices		NA	NA	NA
I / We declare that the above information have been verified and found true to the best of my / our knowledge.				
Signature of the Candidate		**Name & Signature of the Staff (Who uses the plagiarism check software)**		
Name & Signature of the Supervisor / Guide		**Name & Signature of the Co-Supervisor / Co-Guide**		
Dr. K. Raja **Name & Signature of the HOD**				

pap

ORIGINALITY REPORT

9%	3%	2%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to ESC Rennes Student Paper	1%
2	Submitted to University of Greenwich Student Paper	1%
3	Submitted to Hong Kong University of Science and Technology Student Paper	1%
4	Roger M. Sarmento, Francisco F. X. Vasconcelos, Pedro P. Reboucas Filho, Wanqing Wu, Victor Hugo C. De Albuquerque. "Automatic Neuroimage Processing and Analysis in Stroke - A Systematic Review", IEEE Reviews in Biomedical Engineering, 2019 Publication	1%
5	Submitted to Cranfield University Student Paper	<1%
6	Prasad K.D.V. Yarlagadda, Eric Cheng Wei Chiang. "A neural network system for the prediction of process parameters in pressure	<1%

WhatsApp 7990172303 editor@ijcrt.org ijcrt.org

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS - IJCRT (IJCRT.ORG)

International Peer Reviewed & Refereed Journals, Open Access Journal

ISSN: 2320-2882 | Impact factor: 7.97 | ESTD Year: 2013

Scholarly open access journals, Peer-reviewed, and Refereed Journals, Impact factor 7.97 (Calculate by google scholar and Semantic Scholar | AI-Powered Research Tool) , Multidisciplinary, Monthly, Indexing in all major database & Metadata, Citation Generator, Digital Object Identifier(DOI)

Dear Author, Congratulation!!!

Your manuscript with Registration ID: **IJCRT_ 219062** has been **accepted** for publication in the **International Journal of Creative Research Thoughts (IJCRT) | www.ijcrt.org | ISSN: 2320-2882 | International Peer Reviewed & Refereed Journals, Open Access Journal. IJCRT is Peer Review Journal, Refereed Journal, Peer Reviewed Journal Referred Journal and Indexed Journal Open Access Journal Online and Print Journal**

IJCRT Impact Factor: 7.97 | UGC Approved Journal No: 49023 (18)

Check your paper status: **https://ijcrt.org/track.php or http://ijcrt.org/Authorhome/alogin.php**

Online Payment Link for Indian author: http://ijcrt.org/pay_form_2.php

Online Payment Link for Foreign/international author: https://ijcrt.org/pay_form_international.php

Your Paper Review Report :	
Registration ID:	IJCRT – 219062
Title of the Paper:	**MEASURING STROKE REHABILITATION BY POSTURE DETECTION USING NEURAL NETWORKS**
Accepted or Not:	**Accepted**

Criteria	Points out of 100%
Continuity	94%
Text structure	88%
References	85%
Understanding and Illustrations	87%
Explanatory power	85%
Detailing	94%
Relevance and practical advice	94%

Track Your Paper Details:	
Paper Track Link 1:	https://ijcrt.org/track.php
Paper Track Link 2:	http://ijcrt.org/Authorhome/alogin.php
Online Payment link Details:	
Indian author:	http://ijcrt.org/pay_form_2.php
Foreign/international author:	https://ijcrt.org/pay_form_international.php

Overall Assessment (Comments.) **Paper Accepted: YES**

Unique Contents: 95% Paper Accepted

IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Measuring stroke rehabilitation by posture detection using Neural Networks

[1]Swathi R, [2]Mukhil Sundararaj Gowthaman, [3]Dhaarini Srinivasan, [4]Mrudhula Raya

[1]Assistant Professor, [2, 3, 4]Student

[1]Department of Computer Science,

[1]SRM Institute of Science and Technology, Ramapuram, Chennai, India

Abstract: More than 795 thousand people in the US experience a stroke, per year. Out of this, around 610 thousand are RST or new strokes. [2] Strokes are a leading factor of mortality for Americans with 140 thousand people dying annually. [3] Stroke incidence has increased drastically in the young population, with over 20% of those affected being under the age of 45. Hemiparesis, which occurs following a stroke, is a significant motor impairment that affects up to 65 percent of stroke sufferers because to a high rate of muscle frailty. [4] Muscle pain is a significant role in delayed recovery following stroke in victims, and it is also a common symptom reported in patients. This can result in a patient's range of physical activities being limited or even immobilized. Between 30% and 60% of stroke victims report having less physical range in the damaged body portion when performing daily activities. [5] The feasibility of a 4 4 excible sensor matrix which uses paper as a base is explored and a pressure sensor based on paper is developed. This helps reduce the cost and fabrication time. Additionally, the matrix's static and dynamic properties are discussed. Despite considerable variance in response between sensors due to the structural material qualities, the sensor may be efficiently used to continuously evaluate stroke rehabilitation patients' recovery. The matrix's performance is evaluated using a neural network that has been designed to recognize the location of a load using the values of individual sensors.

Index Terms **- Artificial Intelligence, Stroke, Recurrent Neural Network, Sensors**

I. Introduction

Strokes mostly occur in combination with other medical diagnoses. Current stroke rehabilitation evidence, however, is not focused on the comorbidities and therefore does not align with the experiences of the current population. The motivation behind this study is to decide the degree and nature of the ongoing randomized and controlled preliminary of stroke restoration with an emphasis on patients with multimorbidity. Stroke recovery is often incomplete, but rehabilitation training can help recovery outcomes by engaging endogenous neuroplasty. High doses of training are needed in pre-clinical models of stroke rehabilitation to recover the physical activity of the impacted appendages in animals. However, in human body the required training to help recouping rate is unknown. The lack of pragmatic and objective approaches to measuring the rehabilitation training doses adds to this ignorance. To develop a proper measurement approach, the primitives of activities are to be identified. Examples of primitives include running, jumping, and walking. In our review, forty-eight individuals with chronic stroke are chosen to perform a variety of stroke rehabilitation training activities. To capture the upper body movements the individuals, use inertial measurement units or IMUs. Human labellers are used to identify the primitive activities and to label and segment the IMU data. A recurrent neural network (RNN) is designed that outperforms existing techniques. To compute the separate embeddings of the various physical quantities of the sensor data an initial module is included in the proposed algorithm. The proposed technique swaps the batch normalization, which uses measurements processed from training data to perform normalization, with instance normalization, that uses measurements processed from test data. The result is increased robustness towards possible distributional shifts when encountering new patients.

A common side-effect of stroke is muscle weakening which often results in the reduction of the range of movement of the impacted body part. A wide scope of physical training is included in stroke rehabilitation which can help the patients re-establish and develop body fortitude, endurance, balance, and stability. A 4 4 excible pressure matrix is used to analyse and identify the movement of the body while performing the training exercises. The sensor can be used to

blunder of 0.103 cm in recognizing load while numerical investigation gives a mean mistake of 0.704 cm. The pressure sensor can be used to calculate the mistake in the training exercise positioning and can also calculate the duration to complete the exercises.

II. Literature Survey

The authors Muhammad Salman et al in their paper "Identifying Stroke Indicators Using Rough Sets", sought to rank various EHR (Electronic Healthcare Records) records in the order of importance that can be used to detect a stroke. [6] A novel rough-set technique was devised which, unlike the conventional techniques, can be used on any dataset that comprises binary feature sets. The proposed algorithm was then evaluated using a public dataset of EHR records. The conclusions that were drawn from the evaluation are that hypertension, heart disease, average glucose level, and age were the most important attributes to help detect strokes in patients. The proposed algorithm was then benchmarked with popular feature-selection algorithms. The proposed technique obtained the best performance in ranking the importance of individual features to detect stroke.

Stroke rehabilitation patients frequently engage in compensatory behaviours without the supervision of a therapist, resulting in unsatisfactory recovery outcomes. Siqi Cai et al. studied the idea of real-time monitoring of compensation in stroke rehab patients in their paper "Real-Time Detection of Compensatory Patterns in Patients with Stroke to Reduce Compensation During Robotic Rehabilitation Therapy" [7]. This was accomplished using pressure distribution data and machine learning methods. Trunk compensation reduction was also investigated using a combination of online compensation detection and haptic input from a rehab robot. To classify the online compensatory patterns, an SVR classifier (Support vector machine) was trained. The training dataset consists of pressure distribution data from six stroke patients performing three distinct types of reaching movements. Stroke patients were assisted by a rehabilitation robot to lessen compensatory movements. The suggested algorithm's excellent classification accuracy demonstrated the viability of employing pressure distribution data to decrease compensatory movements in stroke patients. The proposed approach was combined with a rehabilitation assistive robot to help stroke sufferers with truck compensations.

Roger M. Sarmento et al. presented a systematic assessment of existing computational approaches for the detection, segmentation, and identification of strokes using medical images. [8] To increase the diagnosis process's accuracy and consistency of interpretation of such image data, CAD (Computer-Aided Diagnosis) systems have been developed using a variety of technologies, including digital image processing methods, and artificial intelligence. However, limited sensitivity, a reduction in false positives, algorithm optimization, and an increase in the identification and classification of varied shapes and sizes are some areas where the presented methodologies could be improved. The classification steps for various stroke types and subtypes, in particular, were cited as areas for improvement. Their research focuses mostly on examining the strategies used to construct CAD systems and validating their efficacy for stroke identification, segmentation, and categorization.

The authors Anis Fatema et al. examined the effects of stroke on recovering patients' motor controls and mobility in their work "A Low-Cost Pressure Sensor Matrix for Activity Monitoring in Stroke Patients Using Artificial Intelligence." [1] As muscle pain is a typical complication of stroke, the researchers created a 4 by 4 pressure sensor matrix to assess the performance and development of patients receiving post-stroke physiotherapy. A series of movement exercises using the pressure sensor matrix with the objective of estimating the force exerted and the movement inaccuracy.

III. METHODOLOGY

3.1 Architecture Diagram

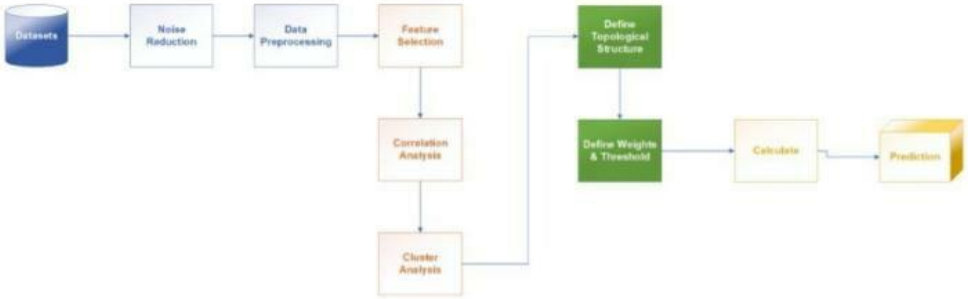


Figure 1 Architecture Diagram

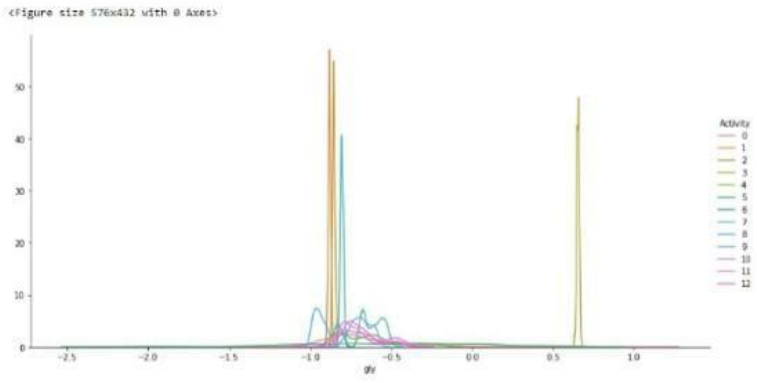
3.2 Imputation Details

Imputation of data is the handling of missing values in our dataset. One method to deal with missing data is to delete the records that contain the missing values, but this could lead to losing out on valuable information. There are various types of imputation. [9] Categorical imputations relate to the missing categorical values in the dataset. The missing data, in this case, is replaced by the most commonly occurring record in the dataset. Numerical imputation deals with missing numeric values in the dataset, the values are generally filled with the mean of the corresponding value in other records. From the imputation data collected we have consolidated a dataset which is used as the value source for the prediction of the model. Before proceeding further, we clean the dataset from possible noise that could have entered it, through means of standardization, duplicate deletion, and elimination of null values.

3.3 Feature Extraction

Selection methods like Feature extraction can be utilized in isolation or in union with each other to increase efficiency of the working of the model. It can be used to work on the precision, visualization, and comprehension of the learned information. Elements can be delegated relevant, irrelevant, or excess. [10] A subset of the accessible information is chosen, in this process. The best subset to pick is unified with the most un-number of aspects that add to learning exactness. [11] Information gain evaluates the gain of each individual feature with respect to the target variable. The larger the information gain, the larger will be the contribution of the characteristics to the text. The characteristics with higher information gain are selected to be Features. Dimensionality reduction is a popular pre-processing method in data analysis, modeling, and visualization. This can be easily achieved using Feature Selection as we can only select the input characteristics that contain information relevant to solving the problem at hand. They can be further classified into Feature weighting methods and subset search methods. [12] This method has a low process expense and is faster. However, they are also not very reliable in classification when compared to wrapper methods and are more suited to high dimensional data sets. Hybrid techniques have been created which consolidate the benefits of filters as well as wrapper strategies. To weigh the features to extract patterns and perform exploratory data analysis, we map labels to the different activities performed by the test subjects, recorded in the dataset. From the label map, we can further analyze the activities performed by each subject and further understand the data to draw patterns from it through means of multivariate feature analysis.

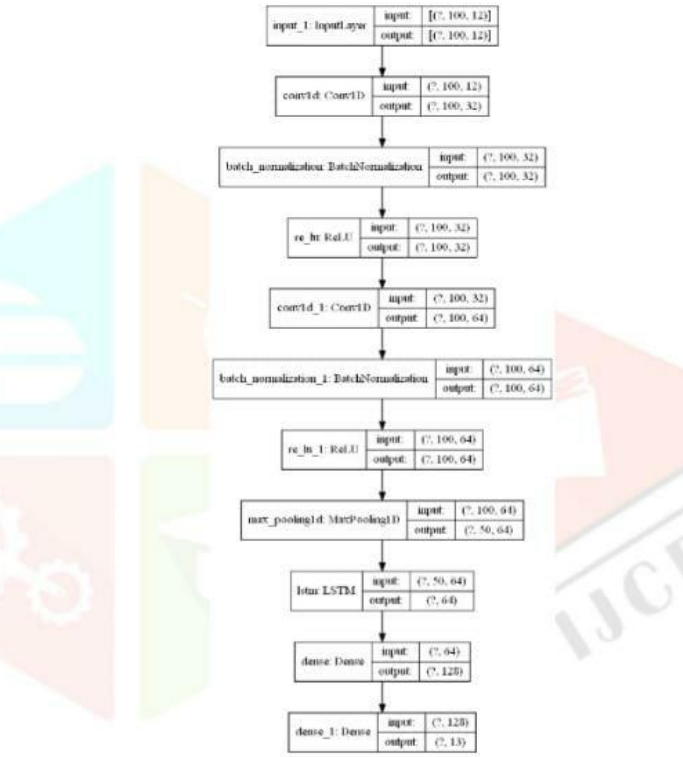
Figure 2 Frequency distribution of Activity for Y axis movement



3. Prediction

Neural Networks are a series of algorithms that aim to recognize underlying patterns in data in a way that mimics the human brain. It can be used for classification, prediction, and other such purposes. Higher the number of hidden layers and therefore a greater number of neurons and connections a neural network has, the more complex it is. What allows the neural network to learn complicated patterns that are hidden in the data is this complexity. In a neural network, changing the weight of anyone connection has a ripple impact on the wide range of various neurons and their initiations in the ensuing layers. A RNN (Recurrent Neural Network) is a kind of neural network that solves the problem of vanishing gradients as it incorporates the backpropagation algorithm. [13] Therefore, it is useful for complicated sequence problems that occur in ML (Machine Learning) and can even attain SOTA (state-of-the-art) results. The RNN has memory blocks that are connected through layers, instead of neurons. The memory blocks have components that enable them to store memory for latest sequences. The blocks contain gates that maintain the state and output of the block. The gates use sigmoid activation units to act upon the input sequence to control if the block is triggered or not. [14] This enables the change of state of the block and any addition of information to the block to be conditional. Figure 3 shows the neural map of the network we have developed with a total of 50,797 parameters out of which 50,605 parameters are trainable. From this neural network we have achieved an accuracy of 95.4% during test scenarios as depicted by Figure 4.

Figure 3 Neural Network Map



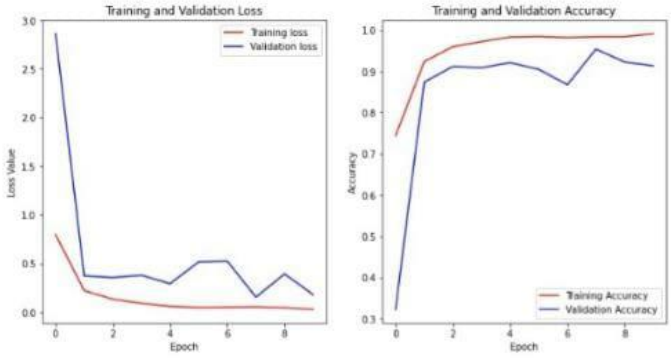


Figure 4 Model Loss and Accuracy visualization

IV. Results And Discussion

A 2D hexagonal binning graph is plotted from the acceleration data from the left ankle sensor and the acceleration data from the right lower arm sensor. The data from when the subject is standing still is represented in Figure 5 and the data from when the subject is climbing stairs is represented in Figure 6. By examining both the graphs, it is clear that the static activities like standing still are different and can be separated from dynamic activities like jumping and climbing stairs.

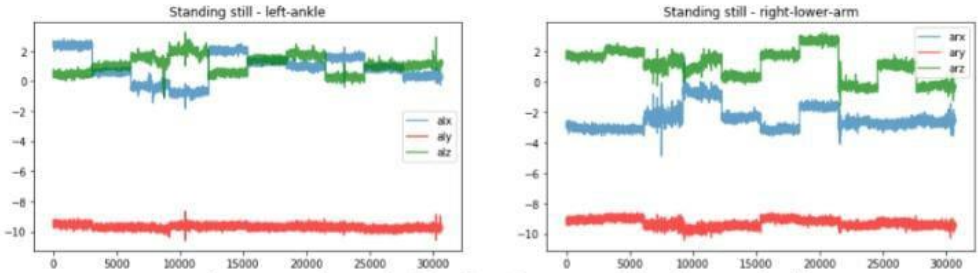


Figure 5 2D Line graph of data from the sensors while subject stands still

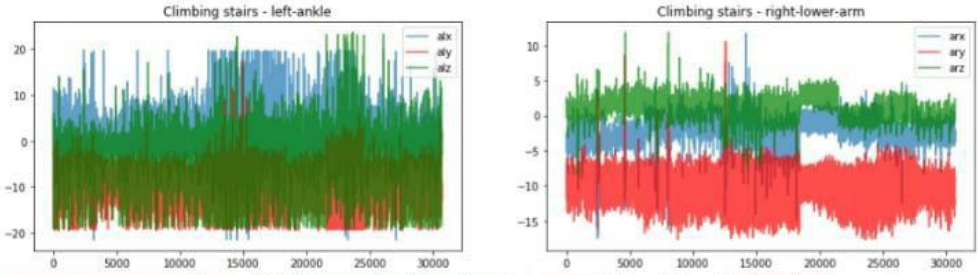


Figure 6 2D Line graph of data from the sensors while subject climbs stairs

After understanding the data, it is then possible to apply the data to our model. A confusion matrix is then plotted to evaluate the model. A confusion matrix provides a visual representation of the performance of classification algorithms. The actual and predicted values are plotted against each other such that there are four different combinations - False positive, True positive, False negative and True negative. From the matrix, it is clear that the data is predicted to a good accuracy as the centre diagonal is darker than the surrounding boxes.

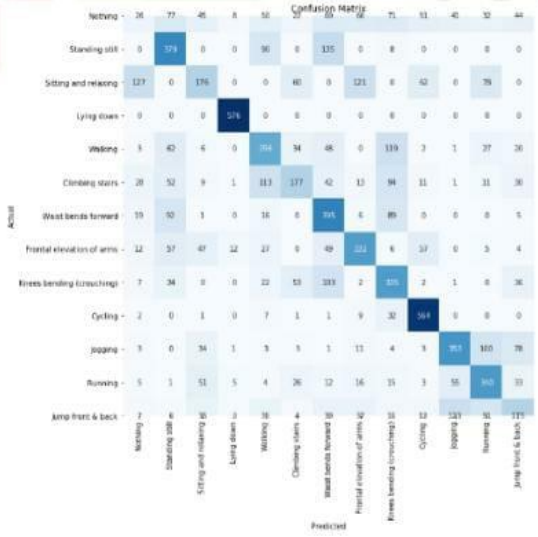


Figure 8 Confusion Matrix

To further evaluate the model, a classification report is generated. A classification report is a metric used for performance evaluation that displays the precision, recall, f1-score and support of the model. The following conclusions are drawn from the report –

- The accuracy of the model is 98% which indicates that the model performs well on our data. Support represents the number of actual observations of the specific class in the dataset.
- Precision gives the ratio of correctly predicted observations compared to the total predictions. We can see the general trend of precision for the various activities is close to 1, which is ideal.
- Recall is the ratio of correctly predicted observations to the total observations. From our results, it is seen that recall is generally higher than 0.8. This is good for this model.
- F1-score is a weighted average of precision and recall. This is generally considered to be a better measure than accuracy. A low F1 score shows poor precision and recall. From the classification report, it is seen that the average score is close to 1.

	precision	recall	f1-score	support
0	1.00	0.99	0.99	89
1	0.98	1.00	0.99	122
2	0.99	1.00	1.00	123
3	1.00	0.99	1.00	123
4	1.00	1.00	1.00	120
5	0.98	0.96	0.97	85
6	0.88	0.99	0.93	105
7	0.99	0.99	0.99	112
8	0.97	0.86	0.91	117
9	1.00	1.00	1.00	120
10	0.99	1.00	0.99	89
11	1.00	0.98	0.99	53
12	1.00	1.00	1.00	26
accuracy			0.98	1284
macro avg	0.98	0.98	0.98	1284
weighted avg	0.98	0.98	0.98	1284

Figure 9 Classification Report

V. CONCLUSION

A novel dataset was introduced that can be used by benchmarking algorithms to detect and classify upper-limb mobility compensatory movements during stroke rehabilitation exercises. The benchmarking algorithms can be used for providing real-time automated stroke rehabilitation coaching and can be merged with rehabilitation robots to assess and correct compensatory movements during training exercises in stroke victims. A baseline model was presented that is used to demonstrate the usefulness of the dataset. The angle of the trunk vector or the shoulder vector is compared to the compensatory movements to examine the results. The specificity and sensitivity of the classification were also shown.

References

[1] Anis Fatema, Surya Poondla, Rishabh B. Mishra, and Aftab M. Hussain, "A Low-Cost Pressure Sensor Matrix for Activity Monitoring in Stroke Patients Using Artificial Intelligence", IEEE Sensors Journal, Vol. 21, No. 7, April 1, 2021

[2] E. J. Benjamin, M. J. Blaha, S. E. Chiuve, M. Cushman, S. R. Das, and R. Deo, "Heart disease and stroke statistics2017 update: A report from the American Heart Association," circulation, vol. 135, no. 10, pp. e146e603, 2017.

[3] C. Patten, J. Lexell, and E. Brown, "Weakness and strength training in persons with poststroke hemiplegia: Rationale, method, and efcacy," J. Rehabil. Res. Dev., vol. 41, pp. 293312, Dec. 2004.

[4] J. H. van der Lee, R. C. Wagenaar, G. J. Lankhorst, T. W. Vogelaar, W. L. Deville, and L. M. Bouter, "Forced use of the upper extremity in chronic stroke patients: Results from a single-blind randomized clinical trial," Stroke, vol. 30, no. 11, pp. 23692375, Nov. 1999.

[5] D. J. Lipomi et al., "Skin-like pressure and strain sensors based on transparent elastic lms of carbon nanotubes," Nature Nanotechnology., vol. 6, no. 12, pp. 788792, Dec. 2011.

[6] M. S. Pathan, Z. Jianbiao, D. John, A. Nag and S. Dev, "Identifying Stroke Indicators Using Rough Sets," in IEEE Access, vol. 8, pp. 210318-210327, 2020, doi: 10.1109/ACCESS.2020.3039439.

[7] Cai S, Li G, Su E, Wei X, Huang S, Ma K, Zheng H, Xie L. Real-Time Detection of Compensatory Patterns in Patients With Stroke to Reduce Compensation During Robotic Rehabilitation Therapy. IEEE J Biomed Health Inform. 2020 Sep;24(9):2630-2638. doi: 10.1109/JBHI.2019.2963365. Epub 2020 Jan 1. PMID: 31902785.

[8] Sarmento RM, Vasconcelos FFX, Filho PPR, Wu W, de Albuquerque VHC. Automatic Neuroimage Processing and Analysis in Stroke-A Systematic Review. IEEE Rev Biomed Eng. 2020;13:130-155. doi: 10.1109/RBME.2019.2934500. Epub 2019 Aug 23. PMID: 31449031.

[9] "https://www.projectpro.io/article/8-feature-engineering-techniques-for-machine-learning/423", 04 April 2022

[10] Khalid, Samina & Khalil, Tehmina & Nasreen, Shamila. (2014). A survey of feature selection and feature extraction techniques in machine learning. Proceedings of 2014 Science and Information Conference, SAI 2014. 372-378. 10.1109/SAI.2014.6918213.
[11] Toğaçar, M., Z. Cömert, And B. Ergen, Classification Of Brain Mri Using Hyper Column Technique With Convolutional Neural Network And Feature Selection Method. Expert Systems With Applications, 2020. 149: P. 113274.
[12] Khalid, Samina & Khalil, Tehmina & Nasreen, Shamila. (2014). A survey of feature selection and feature extraction techniques in machine learning. Proceedings of 2014 Science and Information Conference, SAI 2014. 372-378. 10.1109/SAI.2014.6918213.
[13] husnejahan,"https://github.com/husnejahan/Multivariate-Time-series-Analysis-using-LSTM-ARIMA/blob/master/AirQualityEDA_LSTM.ipynb", 20 Sept 2019.
[14] "https://www.maiango.com/post/forecasting-time-series-data-using-lstm-recurrent-neural-networks/", 01 May 2020