ISSN 2320-9984 03
9 772320 998114
Volume 11 Issue 3 | September 2023 ISSN: 2320-9984 (Online)
International Journal of Modern Engineering & Management Research
Website: www.ijmemr.org

Analysis of Human Activity Recognition Based on Smartphone Data using Artificial Neural Networks

Vaishali Pasi
M.Tech. Research Scholar
Computer Science and Engineering
Takshshila Institute of Engineering and Technology
Jabalpur, (M.P.) India
Email: vaishalipasi9@gmail.com

Swati Soni
Assistant Professor
Department of Computer Science and Engineering
Takshshila Institute of Engineering and Technology
Jabalpur, (M.P.) India
Email: swatisoni@takshshila.org

Abstract*—The Aim of this work was to classify human activities based on smartphone sensor data. The dataset used for this study consisted of recordings from a group of 30 volunteers who performed six different activities while wearing a smartphone on their waist. The smartphone captured 3-axial linear acceleration and 3-axial angular velocity using its embedded accelerometer and gyroscope. The goal was to develop an Artificial Neural Network (ANN) model that could accurately classify the activities performed by the volunteers.

*The ANN model was trained using the training dataset, which consisted of 70% of the volunteer data. The training process involved 30 epochs and a batch size of 16. The model was evaluated using the remaining 30% of the data, known as the test dataset, to assess its performance on unseen instances. The evaluation metrics used were loss and accuracy.

The results of the model training and evaluation indicate that the developed ANN model is effective in classifying human activities based on smartphone sensor data. The model achieved a high ***Accuracy of 0.9474%, Loss of 0.1435%*** *and Model* ***Execution Time is 0.685025453*** *in Secon the test dataset, demonstrating its ability to generalize well and accurately predict the activities performed by individuals.

*The findings of this study have potential applications in various domains such as healthcare, fitness tracking, and behavior analysis. Accurate activity recognition using smartphone sensor data can provide valuable insights into individual's physical movements and behaviors, enabling personalized applications and interventions.

*Overall, this work highlights the successful application of an ANN model to classify human activities using smartphone sensor data, and it showcases the potential benefits and implications of accurate activity recognition in various real-world scenarios.

Keywords:—*Human Activity Recognition, Machine Learning Algorithm, ANN Classification, Tensorflow, Keras, Training - Validation Accuracy Loss.*

1. INTRODUCTION

Human activity recognition has become a significant research area in the field of machine learning and data science. The ability to accurately classify human activities based on smartphone sensor data has the potential to revolutionize various domains, including healthcare, fitness tracking, and behavior analysis. In this study, our aim was to develop an Artificial Neural Network (ANN) model capable of classifying human activities using smartphone sensor data.

The dataset utilized for this study consisted of recordings from a group of 30 volunteers who wore a smartphone on their waist while performing six different activities. The smartphone captured 3-axial linear acceleration and 3-axial angular velocity through its embedded accelerometer and gyroscope. By leveraging this dataset, we sought to train an ANN model that could effectively recognize and classify the activities performed by individuals.

To achieve this goal, the dataset was divided into a training set, which encompassed 70% of the volunteer data, and a test set, which contained the remaining 30% of the data. The ANN model was trained using the training dataset, employing 30 epochs and a batch size of 16. The model's performance was evaluated on the test dataset, measuring its accuracy and loss as evaluation metrics.

The results of the model training and evaluation revealed the efficacy of the developed ANN model in accurately classifying human activities based on smartphone sensor data. The model achieved an impressive accuracy of 94.74% on the test dataset, showcasing its ability to generalize well and predict activities with high precision.

The implications of this study extend beyond the realm of research, offering potential applications in various real-world scenarios. Accurate activity recognition using smartphone sensor data can provide valuable insights into individual's physical movements and behaviors. These insights can be harnessed to develop personalized applications and interventions in areas such as healthcare monitoring, fitness tracking, and behavior analysis.

A. Machine Learning

In the context of human activity recognition using smartphone sensor data, machine learning plays a crucial role in developing an accurate and effective classification model. Machine learning algorithms enable the model to learn patterns and relationships within the sensor data, allowing it to make predictions or classifications on new, unseen instances.

In this study, the aim was to develop an Artificial Neural Network (ANN) model, which is a type of machine learning model, capable of accurately classifying human activities based on smartphone sensor data.

In the context of human activity recognition using smartphone sensor data, there are several types of machine learning techniques that can be employed. Let's explore three key types:

1. Supervised Learning: In supervised learning, the model is trained using labeled data, where the input data (smartphone sensor data) is paired with the corresponding output labels (human activity categories). The goal is to learn a mapping between the input data and the output labels, enabling the model to make accurate predictions or classifications on new, unseen data. In this study, the development of an Artificial Neural Network (ANN) model falls under supervised learning, as the model was trained using labeled data consisting of sensor data and corresponding activity labels.

2. Unsupervised Learning: Unsupervised learning techniques can also be applied to human activity recognition. Unlike supervised learning, unsupervised learning does not rely on labeled data. Instead, it aims to discover patterns, structures, or relationships within the data without prior knowledge of the output labels. In the context of human activity recognition, unsupervised learning techniques like clustering can be used to group similar activities together based on patterns present in the sensor data, even without explicit labels. This approach can help uncover hidden insights and patterns within the data.

3. Reinforcement Learning: While not explicitly mentioned in the provided text, reinforcement learning is another type of machine learning that could potentially be applied to human activity recognition.

Reinforcement learning involves an agent interacting with an environment and learning to take actions to maximize rewards or minimize penalties. In the context of activity recognition, an agent could learn to perform activities by taking actions based on smartphone sensor data and receiving feedback on the correctness of its predictions. Reinforcement learning can be useful in scenarios where activities are sequential and involve decision-making processes.

B. Artificial Neural Network (ANN)

A basic Artificial Neural Network (ANN) can be used for multiclass classification by employing a specific architecture and activation functions. Here's a general overview of how a basic ANN works for multiclass classification:

1. Architecture:

Input Layer: The input layer consists of nodes/neurons corresponding to the features or input variables of the dataset. Each node represents a specific feature.

Hidden Layers: There can be one or more hidden layers in the ANN. Each hidden layer consists of multiple nodes/neurons. The number of nodes in each hidden layer and the number of hidden layers themselves can vary depending on the complexity of the problem.

Output Layer: The output layer represents the number of classes or categories to be predicted. In multiclass classification, the output layer will have nodes equal to the number of classes. Each node in the output layer represents a class and outputs a probability or score corresponding to that class.

2. Activation Functions:

Activation functions are applied to the nodes/neurons in the hidden layers and output layer to introduce non-linearity into the network. Common activation functions for multiclass classification include the **softmax** function in the output layer, which converts the scores into probabilities that sum up to 1, and activation functions like **ReLU** (Rectified Linear Unit) in the hidden layers, which introduce non-linearity and help in capturing complex patterns.

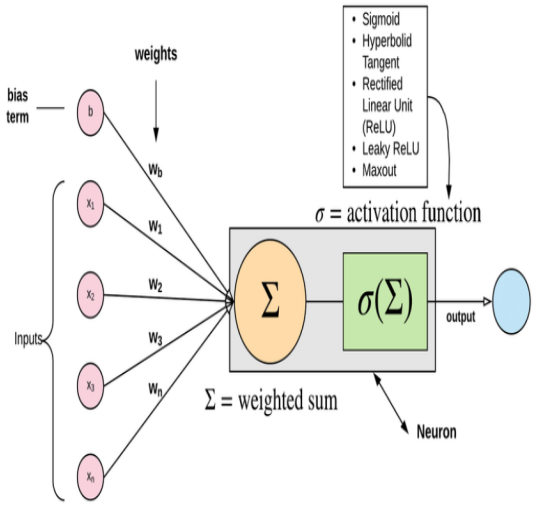

Figure 1: Basic Artificial Neural Network Structure

3. Forward Propagation:

During forward propagation, the input data is passed through the network. The input values are multiplied by the weights connecting the nodes and passed through the activation function of each node to compute the output values for each layer. This process is repeated until the output layer is reached, producing the final predicted probabilities for each class.

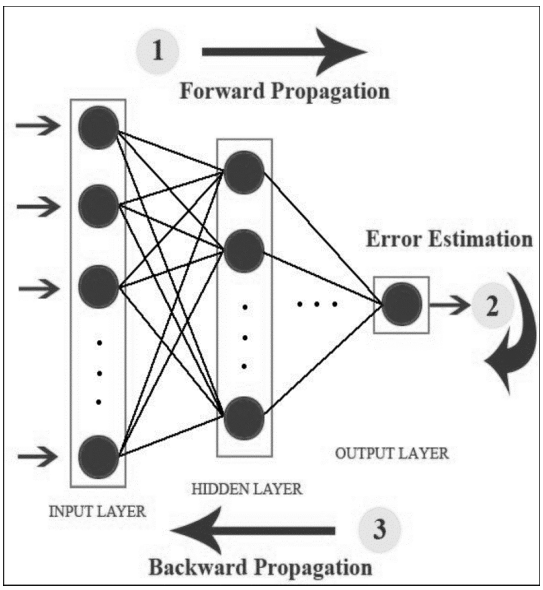

Figure 2: Forward-Backward Propagation in ANN

4. Backpropagation and Optimization:

Backpropagation is used to calculate the gradients of the loss function with respect to the weights and biases in the network. These gradients are then used to update the weights and biases iteratively through an optimization algorithm such as **Stochastic Gradient Descent (SGD) or Adam**, with the aim of minimizing the loss function and improving the model's performance.

5. Loss Function:

A suitable loss function is chosen to measure the difference between the predicted probabilities and the true labels of the training data. For multiclass classification, the commonly used loss function is **categorical cross-entropy**, which measures the dissimilarity between the predicted probabilities and the one-hot encoded true labels.

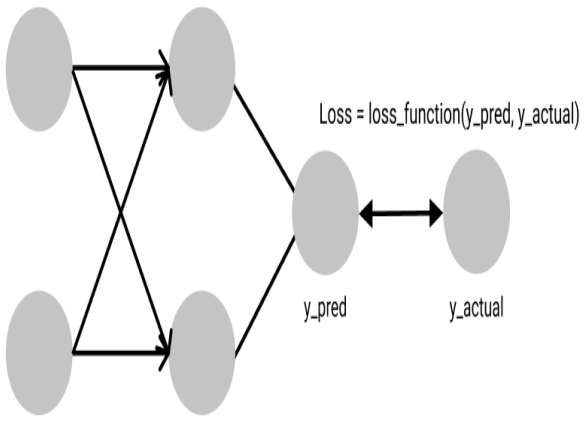

Figure 3: Loss function in ANN

6. Training and Evaluation:

The ANN is trained on the labeled training dataset by iteratively adjusting the weights and biases using the forward propagation and backpropagation steps. The model's performance is evaluated on a separate validation or test dataset, using metrics such as accuracy, precision, recall, or F1 score, to assess its ability to classify the unseen data accurately.

By optimizing the weights and biases through training, the ANN learns to recognize patterns and make predictions for multiclass classification tasks, enabling it to accurately classify new instances into one of the multiple classes.

C. Data Preprocessing

a). Separating Input Features and Output Labels

Separating the train data into input features and output labels involves splitting the dataset into two parts: one part containing the input data, which serves as the features, and another part containing the corresponding output data, which represents the labels or target values.

To separate the data into input features and output labels follow these steps:

1. Identify the features: Determine which columns or attributes in your dataset represent the input features. These could be numerical values, categorical variables, or any other relevant data that you want to use as input for your model.

2. Identify the labels: Identify the column or attribute that represents the output labels or target values. These are the values you want your model to predict or classify based on the input features.

3. Split the Dataset: Once you have identified the input features and output labels, you need to split the dataset into two separate variables, typically denoted as X and y:

- x: This variable represents the input features. It is a matrix or dataframe that contains the values of all the features for each sample in your dataset.
- y: This variable represents the output labels. It is a vector or array that contains the corresponding label or target value for each sample.

The splitting can be done by extracting the desired columns from the dataset or by indexing the data structure appropriately.

b) Label Encoding Conversion

In the given scenario, you have a dataset train_df with an "Activity" column containing six different classes: 'Laying', 'Sitting', 'Standing', 'Walking', 'Walking_ Downstairs', and 'Walking_Upstairs'. You want to convert these string labels into integer values using label encoding.

To perform label encoding, you can use the LabelEncoder class from the scikit-learn library. Import the preprocessing module from scikit-learn, which contains the LabelEncoder class. We create an instance of LabelEncoder and use the fit_transform method to fit the encoder on the training data (y_train) and transform the labels into encoded integer values. The resulting encoded labels are stored in y_train_encoder.

For the validation and test datasets (y_val and y_test), we only need to transform the labels using the transform method since the encoder has already been fitted on the training data. The transformed labels are stored in y_val_encoder and y_test_encoder, respectively.

Label encoding assigns a unique integer value to each class, starting from 0. In this case, the label encoder would assign the following integer values to the activity classes:

'Laying': 0

'Sitting': 1

'Standing': 2

'Walking': 3

'Walking_ Downstairs': 4

'Walking_Upstairs': 5

The encoded labels can be used as inputs for machine learning models that require numerical representations of the classes. Remember to use the same label encoder instance on new data during inference or prediction to ensure consistent encoding.

D. Tensorflowand Keras for Human Activity Recognition

TensorFlow is an open-source machine learning framework developed by Google. It provides a platform for building and deploying machine learning models, particularly deep learning models. TensorFlow offers a wide range of tools and libraries for various tasks in machine learning, including data preprocessing, model building, training, and deployment.

Keras, on the other hand, is a high-level neural networks API that is integrated into TensorFlow. It provides a user-friendly interface for building and training deep learning models. Keras allows you to define and customize neural network architectures using building blocks called layers.

In the case of multiclass classification for the "Human Activity Recognition Using Smartphones" dataset, the goal is to predict the activity performed by a person based on smartphone sensor data. The dataset consists of various sensor measurements, such as accelerometer and gyroscope readings, collected from smartphones while individuals were performing different activities.

E. Model Underfitting and Overfitting

Underfitting occurs when a model is not able to capture the underlying patterns in the data effectively. It typically results in high bias and low variance. In the context of the neural network trained on the provided dataset, underfitting would mean that the model is not learning enough from the training data and is performing poorly on both the training and test sets.

Looking at the training process, we can observe that the model's **training accuracy** steadily increases from around **62% to 98.32%** over the course of 30 epochs.

Similarly, the **validation accuracy** improves from **75.73% to 97.21%.** These high accuracy values suggest that the model is learning well from the training data and can generalize to the validation set.

However, when evaluating the model on the **test data**, the results show a lower accuracy of **93.96%** and a higher **loss** of **0.1529**. This discrepancy between the validation accuracy and the test accuracy indicates that the model may have overfit the training data.

Overfitting occurs when a model learns the training data too well and fails to generalize to new, unseen data. It typically exhibits low bias and high variance. In this case, the model's high accuracy on the training and validation sets suggests that it might have learned the specific patterns of the training data too well, resulting in overfitting.

2. LITERATURE REVIEW

This work focuses on Human Activity Recognition (HAR) systems, which are used to continuously observe human behavior in various fields such as environmental compatibility, sports injury detection, senior care, rehabilitation, entertainment, and intelligent home surveillance. The authors highlight the use of inertial sensors like accelerometers, linear acceleration, and gyroscopes, which are now commonly found in smartphones. They emphasize the need for activity data acquisition using smartphones due to their widespread use.

The paper presents the collection of smartphone sensor-based raw data called H-Activity using an Android-based application for accelerometer, gyroscope, and linear acceleration. In addition, the authors propose a **hybrid deep learning model called CNN-LSTM**, **which combines convolutional neural networks (CNN) and long-short term memory networks (LSTM)** and is empowered by the self-attention algorithm to enhance predictive capabilities.

The model is evaluated using the collected H-Activity dataset as well as benchmark datasets such as MHEALTH and UCI-HAR. The results show that the proposed model achieves high accuracy. It attains **99.93% accuracy on the H-Activity dataset**, **98.76% on the MHEALTH dataset**, and **93.11% on the UCI-HAR dataset**. The authors emphasize the efficacy of their model in recognizing human activity.

The paper also discusses the datasets used in the evaluation. The H-Activity dataset consists of data collected from ten volunteers performing four different physical activities. The MHEALTH dataset includes body signals and indicators recorded while participants engage in 12 physical activities. The UCI-HAR dataset consists of data collected from 30 individuals performing six different actions.

In conclusion, the work presents a deep CNN-LSTM model with self-attention for human activity recognition using smartphone sensor data. The proposed model achieves high accuracy and demonstrates robustness when compared to benchmark datasets. The authors plan to continue strengthening their dataset, adjust the network structure, and focus on real-time classification of elderly health issues and security systems. They believe that the developed framework could be applicable in clinical settings, and the collected data could be valuable for further research.[1]

Dataset	Type of sensor and location	Sampling Freq.	Samples	Volunteers	Placement of the sensor	Type of activities
H-Activity	Phone based accelerometer, gyroscope, and linear acceleration	10 Hz.	48920	10	Smartphone in trousers right pocket	4 activities. Sitting/standing, walking, jogging, and running
MHEALTH	accelerometer, gyroscope, and magnetometer	50 Hz	120	10	Sensors were implanted on the participant's chest, right wrist, and left ankle.	12 physical activities such as standing, sitting, lying, walking, climbing, cycling, jogging, running jumping.
UCI-HAR	Phone based accelerometer, and gyroscope	50 Hz	10299	30	Smartphone in Waist	6 activities such as: Walking, going upstairs/downstairs, sitting, standing, and lying

Figure 4:concise narration of the H-Activity, MHEALTH, and UCI-HAR datasets used [1]

This work focuses on human activity recognition (HAR) using smartphone sensor data. The authors highlight the growing research area of self-regulated HAR in smart and intelligent healthcare. Traditional machine learning (ML) approaches for HAR face challenges such as handcrafted feature extraction and separate dimensionality reduction modules. In this article, the authors propose a deep learning (DL) **approach that combines convolutional neural networks (CNNs), autoencoders (AEs), and long short-term memory (LSTM)** networks to overcome these challenges. The proposed architecture, called "ConvAE-LSTM," takes advantage of the complementary modeling capabilities of CNNs, AEs, and LSTMs. The CNNs automatically extract features, AEs perform dimensionality reduction, and LSTMs excel at temporal modeling. The authors evaluate the architecture on **four standard public datasets: WISDM, UCI, PAMAP2, and OPPORTUNITY**.

The experimental results demonstrate that the proposed approach outperforms existing state-of-the-art methods in terms of computational time, accuracy, F1-score, precision, and recall. For example, on the UCI dataset, the proposed method achieves an average precision of 97%, recall of 96.83%, F1-score of 97.67%, and accuracy of 98.14%. On the WISDM dataset, it achieves an average precision of 98.17%, recall of 98.33%, F1-score of 98.17%, and accuracy of 98.67%.

The authors also discuss the datasets used in the evaluation. The UCI dataset is a balanced dataset collected from 30 subjects performing six daily life activities. The WISDM Actitracker dataset includes data from 36 subjects performing six different physical activities. The OPPORTUNITY dataset contains recordings of naturalistic activities using multiple sensors, and the PAMAP2 dataset includes recordings from nine subjects performing various lifestyle tasks.

In conclusion, the proposed ConvAE-LSTM model demonstrates better performance and generalization compared to other DL and shallow ML approaches in HAR. The authors plan to compare their method with other DL-based methods, perform experiments on more datasets, and analyze the applicability of the proposed method in real-life applications in future work.[2]

The goal is to provide accurate information about human activities using smart phones and machine learning algorithms. The authors address the challenges of HAR, such as low classification accuracy and high computational overhead, and propose a method based on exploratory data analysis (EDA) to improve accuracy.

The study uses data from the **University of California, Irvine (UCI) database, collected from 30 individuals wearing smart phones on their waists**. **The phones capture accelerometer and gyroscope data to track the participant's activities.** The sensor signals are pre-processed using noise filtering and sampling techniques.

The methodology of the study involves checking the balance of the data, performing exploratory data analysis, applying feature engineering based on domain knowledge, and building models with the data. The authors analyze the results using confusion matrices, normalized confusion matrices, and other evaluation measures.

The experimental results show that the proposed HAR method based on EDA achieves a **high accuracy of 96.56%**. The authors compare different models and find that the **LinearSVC algorithm** has a higher correlation degree and accuracy. They suggest that future research should focus on improving the performance of HAR applications, enhancing health care functionality, quality, and safety.

Pre-processing sensor signals refers to applying specific operations to the raw data captured by the sensors (accelerometer and gyroscope) of the smart phones. The goal of this pre-processing step is to enhance the quality of the sensor data and make it suitable for further analysis and modeling.

The pre-processing steps mentioned in the information include two main techniques: applying a noise filter and sampling in a sliding window. Let's break down each step:

Noise filter: A noise filter is used to remove unwanted or irrelevant signals from the sensor data.

It helps in reducing the impact of noise, artifacts, or interference that might be present in the raw data. The specific type of noise filter used in this work is not mentioned.

Sliding window and sampling: Sliding window is a technique where a fixed-width window is moved over the sensor data in a sequential manner. In this case, the sliding window has a width of 2.56 seconds. The purpose of using a sliding window is to segment the continuous sensor data into smaller, overlapping segments or windows.

Within each window, sampling is performed to select a subset of readings. The information mentions that there are 128 readings per window. The sampling is done in such a way that there is a 50% overlap between consecutive windows. This means that each window will share half of its data with the previous window, providing some temporal context in the subsequent analysis. By applying the sliding window technique with sampling and a fixed width, the continuous sensor data is divided into smaller chunks, allowing for the extraction of features and analysis on a per-window basis. This segmentation helps capture local patterns and variations in the sensor data, which can be useful for activity recognition or other analysis tasks.[3]

In this work, we proposed iSPLInception, a deep learning (DL) model for human activity recognition (HAR) that combines high predictive accuracy with efficient resource utilization. The datasets used for evaluation include the **UCI HAR using smartphones dataset**, Opportunity activity recognition dataset, **Daphnet freezing of gait dataset**, and **PAMAP2 physical activity monitoring dataset**. Our experimental results demonstrated that the iSPLInception model outperforms existing DL architectures, such as vanilla long short-term memory (LSTM), stacked LSTM, convolutional neural network (CNN), **CNN-LSTM**, and bidirectional LSTM (BiLSTM) networks. The UCI HAR dataset consists of recordings of 30 subjects performing basic activities of daily life using a waist-mounted smartphone with inertial sensors. The Opportunity dataset includes recordings of 12 subjects performing various naturalistic activities using 72 environmental and body sensors. The Daphnet dataset focuses on freezing of gait detection in patients with Parkinson's disease using wearable acceleration sensors. The PAMAP2 dataset involves monitoring physical activities of 9 participants using inertial measurement units (IMUs).

On the UCI HAR dataset, our iSPLInception model achieved the highest test **accuracy of 95.09% and F1 score of 95%** compared to other models. **On the Opportunity dataset**, our model achieved an **F1 score of 88%**, outperforming the other models. Similarly, **on the Daphnet dataset**, our model achieved an **F1 score of 94%**, surpassing the performance of the other models. **On the PAMAP2 dataset**, our model achieved an **F1 score of 89%**, outperforming most of the other models.

In conclusion, our iSPLInception model demonstrated remarkable performance for HAR applications, outperforming existing DL architectures on multiple datasets. This work provides a benchmark for future research in deep learning on these four datasets and

suggests the potential for using an ensemble of DL models in HAR applications.[4]

3. METHODOLOGY

A. Artificial Neural Network (ANN) Model

The problem addressed in this work is the classification of human activities based on smartphone sensor data. The goal is to develop an Artificial Neural Network (ANN) model that can accurately classify the activities performed by individuals. The dataset used for this study consists of recordings from a group of 30 volunteers who performed six different activities while wearing a smartphone on their waist. The smartphone captured 3-axial linear acceleration and 3-axial angular velocity using its embedded accelerometer and gyroscope.

The specific objectives of this work are as follows:

1. Develop an ANN model for activity recognition using smartphone sensor data.
2. Train the ANN model using 70% of the volunteer data.
3. Evaluate the performance of the trained model using the remaining 30% of the data.
4. Assess the accuracy of the model in classifying activities performed by individuals.
5. Explore the potential applications of accurate activity recognition in domains such as healthcare, fitness tracking, and behavior analysis.

The expected outcome of this work is a highly accurate ANN model that can classify human activities based on smartphone sensor data. The model's performance will be evaluated using metrics such as loss and accuracy.

The dataset used in this study is publicly available and can be accessed from the provided link. The experiments were conducted with a group of 30 volunteers within an age bracket of 19-48 years, and each person performed six activities while wearing a smartphone on their waist. The data was labeled manually based on video recordings, and the dataset was randomly partitioned into training and test sets, with 70% of the data used for training and 30% for testing the model's performance.

The dataset used in this study is publicly available and can be accessed from the provided link. The experiments were conducted with a group of 30 volunteers within an age bracket of 19-48 years, and each person performed six activities while wearing a smartphone on their waist. The data was labeled manually based on video recordings, and the dataset was randomly partitioned into training and test sets, with 70% of the data used for training and 30% for testing the model's performance.

The tools and techniques employed in this work include machine learning algorithm ANN model for classification. Tensorflow and Keras libraries were utilized for implementing the ANN model. The training process involved 30 epochs and a batch size of 16. The model's performance was evaluated using the test dataset, and metrics such as accuracy and loss were used to assess its effectiveness.

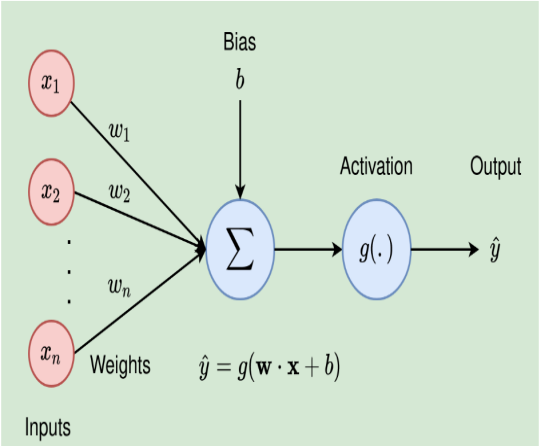

Figure 5: Artificial Neural Network Fundamentals

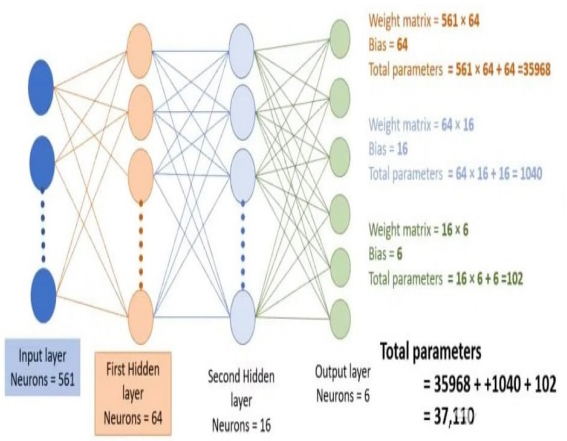

Figure 6: ANN Proposed Model for HAR

Input Layer:

The input layer is defined using keras.layers.InputLayer(input_shape=(561,)).

The input_shape parameter specifies the shape of the input data, which is a feature vector with 561 dimensions.

Hidden Layers:

- Two hidden layers are added to the model using keras.layers.Dense with different activation functions ('relu').
- The first hidden layer is added using model.add(keras.layers.Dense (units=64, activation='relu')).
- The second hidden layer is added using model.add(keras.layers.Dense (units=16, activation='relu')).
- Each hidden layer consists of a specified number of units (neurons) and applies the rectified linear unit (ReLU) activation function.

The mathematical calculations in these layers involve matrix multiplications, bias additions, and applying the activation function to the inputs.

Output Layer:

- The output layer is added to the model using model.add (keras.layers.Dense(units=6, activation='softmax')).
- The output layer has 6 units, which correspond to the 6 classes in the dataset.
- The activation function used in the output layer is softmax, which calculates the probabilities of the input belonging to each class.
- The softmax activation function involves exponentiation and normalization to produce the class probabilities.

Model Summary:

- The model.summary() function is used to display a summary of the model's architecture.
- The summary includes the layer type, output shape, and number of trainable parameters in each layer.
- The summary provides insights into the model's structure and helps in understanding the flow of data through the network.
- No significant mathematical calculations are involved in generating the model summary.

B. Compilation

The compilation step of a neural network model using the Keras library. Here's an explanation of the code:

Optimizer:

- The optimizer is responsible for updating the weights of the neural network during training, optimizing the model's performance.keras. optimizers.Adam(learning_rate= 0.0001).
- The learning rate determines the step size for weight updates during optimization.

Loss Function:

- The loss function quantifies the difference between the predicted output and the actual output, serving

as a measure of how well the model is performing.

- In this code, the loss function used is sparse categorical cross-entropy, denoted by keras.losses.Sparse CategoricalCrossentropy().
- The sparse categorical cross-entropy loss is commonly used for multi-class classification problems, where the classes are mutually exclusive.

Metrics:

- Metrics are used to evaluate the performance of the model during training and evaluation.
- In this code, the metric chosen is accuracy, specified as ['accuracy'].
- The accuracy metric measures the percentage of correctly classified instances out of the total instances.

Compilation:

- The model.compile() function is used to compile the neural network model, where the optimizer, loss function, and metrics are specified.
- The code snippet shows the compilation step of the model, with the optimizer, loss, and metrics defined as explained above.
- By compiling the model with the chosen optimizer, loss function, and metrics, the model is prepared for the subsequent training phase. During training, the optimizer will update the model's weights based on the specified loss function, and the chosen metrics will be used to evaluate the model's performance.

4. IMPLEMENTATION WORK

4.1 Data Source

Data Source: https://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones

About Dataset

The experiments have been carried out with a group of 30 volunteers within an age bracket of 19-48 years. Each person performed six activities (Walking, Walking_Upstairs, Walking_Downstairs, Sitting, Standing, Laying) wearing a smartphone (Samsung Galaxy S II) on the waist. Using its embedded accelerometer and gyroscope, It captured 3-axial linear acceleration and 3-axial angular velocity at a constant rate of 50Hz. The experiments have been video-recorded to label the data manually. The obtained dataset has been randomly partitioned into two sets, where 70% of the volunteers was selected for generating the training data and 30% the test data.

For each record in the dataset it is provided:

- Triaxial acceleration from the accelerometer (total acceleration) and the estimated body acceleration.
- Triaxial Angular velocity from the gyroscope.
- A 561-feature vector with time and frequency domain variables.
- Its activity label.
- An identifier of the subject who carried out the experiment.

Human Activity Recognition Dataset Contains – 2 csv Files

- train.csv, Size:45.9 MB, Shape:7352 rows × 563 columns
- test.csv, Size: 18. MB, Shape: 2947 rows × 563 columns

Outcome: Class variable (1 2 3 4 5 6)

Class Distribution:

- Represents: 'Laying'
- Represents: 'Sitting'
- Represents: 'Standing'
- Represents: 'Walking'

- Represents: 'Walking_Downstairs'
- Represents: 'Walking_Upstairs'

B. Platform and Methodology

- IDE - Google Collaborator
- Python–Python 3
- Applied Multiclass Classification on Labeled Data

C. Implementation

- Mounting the Drive
- Importing the Dependencies
- Find the Shape of the Tain DataFrame using test_df.shape
- Data Preprocessing
- Computing the sum of null values in the training and test datasets using train_df.isnull().sum() and test_df.isnull().sum().
- Separating the Train Data into Input = feature and Output = Label Variables
- Separating the Test Data into Input = feature and Output = Label Variables

The features mentioned in the dataset are derived from the accelerometer sensor measurementsand represent different statistical measures of the body acceleration signals in three dimensions (X, Y, and Z). Here is an explanation of each feature:

1. ***tBodyAcc-mean()-X:*** This feature represents the mean (average) value of the body acceleration signal along the X-axis.
2. ***tBodyAcc-mean()-Y:*** This feature represents the mean value of the body acceleration signal along the Y-axis.
3. ***tBodyAcc-mean()-Z:*** This feature represents the mean value of the body acceleration signal along the Z-axis.
4. ***tBodyAcc-std()-X:*** This feature represents the standard deviation of the body acceleration signal along the X-axis. It measures the variability or spread of the signal values.
5. ***tBodyAcc-std()-Y:*** This feature represents the standard deviation of the body acceleration signal along the Y-axis.
6. ***tBodyAcc-std()-Z:*** This feature represents the standard deviation of the body acceleration signal along the Z-axis.
7. ***tBodyAcc-mad()-X:*** This feature represents the median absolute deviation (MAD) of the body acceleration signal along the X-axis. MAD is a robust measure of the variability of a dataset.
8. ***tBodyAcc-mad()-Y:*** This feature represents the median absolute deviation of the body acceleration signal along the Y-axis.
9. ***tBodyAcc-mad()-Z:*** This feature represents the median absolute deviation of the body acceleration signal along the Z-axis.

These features provide information about the central tendency (mean), variability (standard deviation), and distribution (median absolute deviation) of the body acceleration signals in different dimensions. They can be used as input variables to train machine learning models for human activity recognition tasks.

Counting the occurrences of each activity class in the training and test datasets using train_df['Activity'].value_counts() and test_df['Activity'].value_counts().

Train Test Data Visualization

Plotting the pie chart for Activity Classes Distribution in Train DataSet

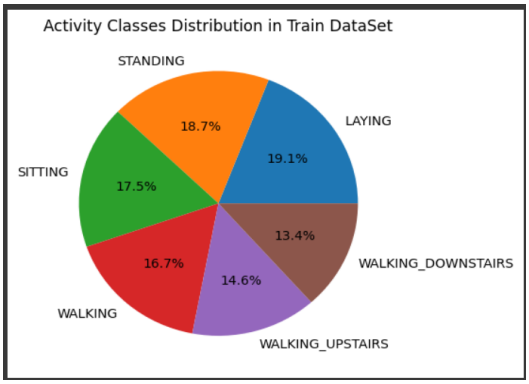

Figure 7: Activity Class Distribution in Train Dataset

Plotting the pie chart for Activity Classes Distribution in TestDataSet

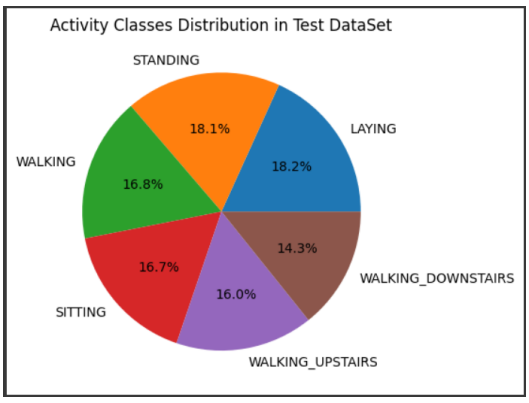

Figure 8: Activity Class Distribution in Test Dataset

- Separating the training data into input (x) and output (y) variables.
- Separating the test data into input (x_test) and output (y_test) variables.
- Split the Train Dataset into 2 Parts, Training and Validation set
- Training Data (70% of Dataset, already in a Separate csv File)
- Testing Data (30% of Dataset, already in a Separate csv File)
- Now Splitting Train Data Set in 2 Parts (80% Training, 20% Validation)
- Performing label encoding using preprocessing.LabelEncoder() to convert the output classes from text to integer values.

Model Building

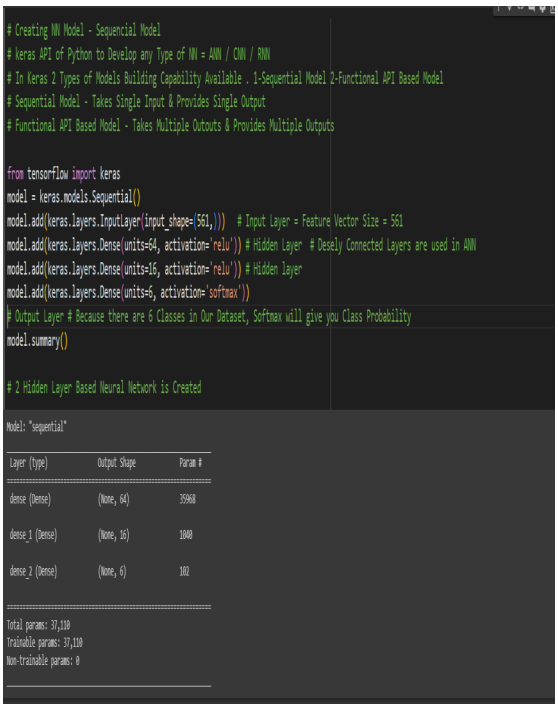

Figure 9: Creating ANN Moel

- Construction of a neural network model using Keras, This code imports the necessary modules from TensorFlow and creates a sequential model.

  from tensorflow import keras

  model = keras.models.Sequential()

- A sequential model is a linear stack of layers where the output of one layer becomes the input of the next. This line adds an input layer to the model with an input shape of (561,). This shape corresponds to the number of features in your dataset, which is 561 in this case.

  model.add(keras.layers.InputLayer(input_shape=(561,)))

- These lines add two hidden layers to the model. The first hidden layer has 64 units and uses the ReLU activation function. The second hidden layer has 16 units and also uses the ReLU activation function. Dense layers are fully connected layers in which each neuron is

connected to every neuron in the previous layer.

model.add(keras.layers.Dense (units=64, activation='relu'))

model.add(keras.layers.Dense (units=16, activation='relu'))

- This line adds the output layer to the model. Since there are six classes in the dataset, the output layer has six units. The activation function used is softmax, which produces a probability distribution over the classes.

  model.add(keras.layers.Dense (units=6, activation='softmax'))

Model Compilation.

Model Training

Epoch 1/30

368/368 [==========] - 1s 2ms/step - loss: 1.2507 - accuracy: 0.5846 - val_loss: 0.8615 - val_accuracy: 0.7763

Epoch 2/30

368/368 [==========] - 1s 2ms/step - loss: 0.6556 - accuracy: 0.8259 - val_loss: 0.5018 - val_accuracy: 0.8695

Epoch 3/30

368/368 [==========] - 1s 1ms/step - loss: 0.4032 - accuracy: 0.8859 - val_loss: 0.3310 - val_accuracy: 0.9069

Epoch 4/30

368/368 [==========] - 1s 2ms/step - loss: 0.2830 - accuracy: 0.9167 - val_loss: 0.2522 - val_accuracy: 0.9218

Epoch 5/30

368/368 [==========] - 1s 2ms/step - loss: 0.2193 - accuracy: 0.9316 - val_loss: 0.1989 - val_accuracy: 0.9388

Epoch 6/30

368/368 [==========] - 1s 2ms/step - loss: 0.1803 - accuracy: 0.9463 - val_loss: 0.1716 - val_accuracy: 0.9456

Epoch 7/30

368/368 [==========] - 1s 2ms/step - loss: 0.1568 - accuracy: 0.9522 - val_loss: 0.1503 - val_accuracy: 0.9545

Epoch 8/30

368/368 [==========] - 1s 1ms/step - loss: 0.1392 - accuracy: 0.9594 - val_loss: 0.1410 - val_accuracy: 0.9497

Epoch 9/30

368/368 [==========] - 1s 1ms/step - loss: 0.1267 - accuracy: 0.9619 - val_loss: 0.1280 - val_accuracy: 0.9599

Epoch 10/30

368/368 [==========] - 1s 2ms/step - loss: 0.1147 - accuracy: 0.9633 - val_loss: 0.1209 - val_accuracy: 0.9606

Epoch 11/30

368/368 [==========] - 1s 2ms/step - loss: 0.1046 - accuracy: 0.9689 - val_loss: 0.1056 - val_accuracy: 0.9653

Epoch 12/30

368/368 [==========] - 1s 1ms/step - loss: 0.0980 - accuracy: 0.9691 - val_loss: 0.0983 - val_accuracy: 0.9626

Epoch 13/30

368/368 [==========] - 1s 1ms/step - loss: 0.0915 - accuracy: 0.9709 - val_loss: 0.0959 - val_accuracy: 0.9680

Epoch 14/30

368/368 [==========] - 1s 2ms/step - loss: 0.0862 - accuracy: 0.9721 - val_loss: 0.1027 - val_accuracy: 0.9606

Epoch 15/30

368/368 [==========] - 1s 1ms/step - loss: 0.0812 - accuracy: 0.9745 - val_loss: 0.0860 - val_accuracy: 0.9660

Epoch 16/30

368/368 [==============================] - 1s 2ms/step - loss: 0.0773 - accuracy: 0.9760 - val_loss: 0.0818 - val_accuracy: 0.9687

Epoch 17/30

368/368 [==============================] - 1s 2ms/step - loss: 0.0725 - accuracy: 0.9772 - val_loss: 0.0764 - val_accuracy: 0.9728

Epoch 18/30

368/368 [==============================] - 1s 2ms/step - loss: 0.0696 - accuracy: 0.9767 - val_loss: 0.0771 - val_accuracy: 0.9694

Epoch 19/30

368/368 [==============================] - 1s 2ms/step - loss: 0.0653 - accuracy: 0.9787 - val_loss: 0.0707 - val_accuracy: 0.9769

Epoch 20/30

368/368 [==============================] - 1s 1ms/step - loss: 0.0643 - accuracy: 0.9774 - val_loss: 0.0689 - val_accuracy: 0.9742

Epoch 21/30

368/368 [==============================] - 1s 2ms/step - loss: 0.0635 - accuracy: 0.9779 - val_loss: 0.0657 - val_accuracy: 0.9776

Epoch 22/30

368/368 [==============================] - 1s 2ms/step - loss: 0.0599 - accuracy: 0.9804 - val_loss: 0.0624 - val_accuracy: 0.9776

Epoch 23/30

368/368 [==============================] - 1s 2ms/step - loss: 0.0580 - accuracy: 0.9798 - val_loss: 0.0719 - val_accuracy: 0.9701

Epoch 24/30

368/368 [==============================] - 1s 2ms/step - loss: 0.0581 - accuracy: 0.9808 - val_loss: 0.0672 - val_accuracy: 0.9721

Epoch 25/30

368/368 [==============================] - 1s 3ms/step - loss: 0.0548 - accuracy: 0.9813 - val_loss: 0.0593 - val_accuracy: 0.9789

Epoch 26/30

368/368 [==============================] - 1s 3ms/step - loss: 0.0534 - accuracy: 0.9828 - val_loss: 0.0715 - val_accuracy: 0.9735

Epoch 27/30

368/368 [==============================] - 1s 3ms/step - loss: 0.0558 - accuracy: 0.9815 - val_loss: 0.0652 - val_accuracy: 0.9721

Epoch 28/30

368/368 [==============================] - 1s 1ms/step - loss: 0.0495 - accuracy: 0.9840 - val_loss: 0.0575 - val_accuracy: 0.9789

Epoch 29/30

368/368 [==============================] - 1s 2ms/step - loss: 0.0486 - accuracy: 0.9850 - val_loss: 0.0628 - val_accuracy: 0.9769

Epoch 30/30

368/368 [==============================] - 1s 1ms/step - loss: 0.0491 - accuracy: 0.9828 - val_loss: 0.0515 - val_accuracy: 0.9837

Plot the Training and Validation Accuracy / Loss Curves to Visualize the Model's Performance during Training.

Model Evaluation

- Evaluating the trained model using the test data (x_test, y_test_encoder) and calculating the loss and accuracy metrics using model.evaluate.
- Predicting labels for the test data using model.predict.
- Calculating the Model Execution Time
- Computing the confusion matrix using confusion_matrix.
- Calculating additional evaluation metrics such as precision, recall, and F1-score using classification_report.

- Deploy the Model on Smart Phone Based Application to Identify the Human Activity / Positions using Test Data

5. RESULTS AND COMPARISION ANALYSIS

A. Training -Validation Accuracy and Loss

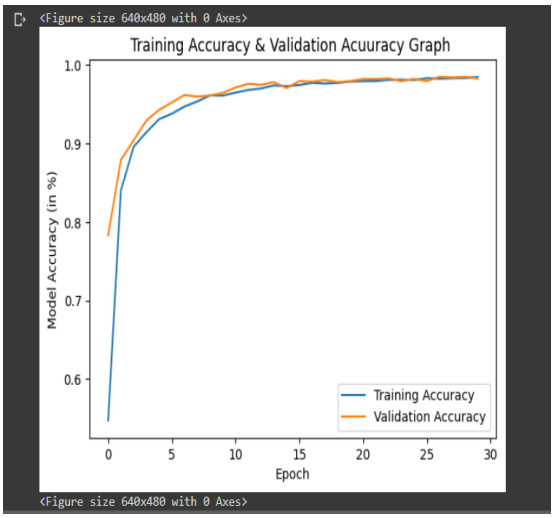

Figure 10: Training Accuracy and Validation Accuracy Graph

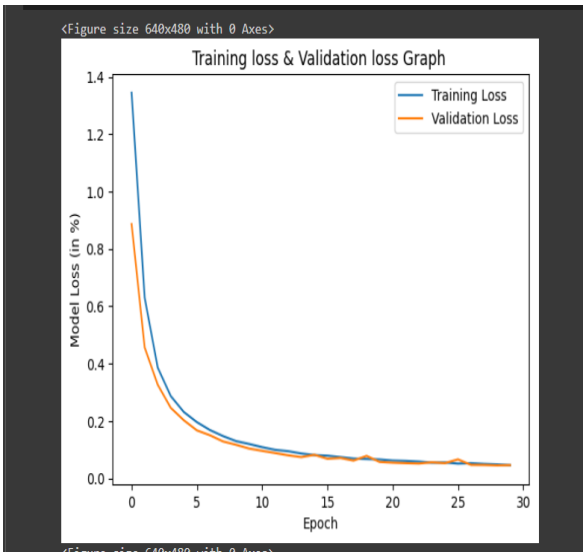

Figure 11: Training Loss and Validation Loss Graph

B. Model Performanceover Test Data

93/93 [==========] - 0s 2ms/step - loss: 0.1435 - accuracy: 0.9474

93/93 [==========] - 0s 2ms/step

Model Execution Time: 0.6850254535675049

Confusion Matrix

```
[[ 523   0  14   0   0   0 ]
 [   0 442  47   0   0   2 ]
 [   0  30 502   0   0   0 ]
 [   0   0   0 490   5   1 ]
 [   0   0   0   3 405  12 ]
 [   0   0   0  35   6 430 ]]
```

Classification Report

	precision	recall	f1-score	support
0	1.00	0.97	0.99	537
1	0.94	0.90	0.92	491
2	0.89	0.94	0.92	532
3	0.93	0.99	0.96	496
4	0.97	0.96	0.97	420
5	0.97	0.91	0.94	471
accuracy			0.95	2947
macro avg	0.95	0.95	0.95	2947
weighted avg	0.95	0.95	0.95	2947

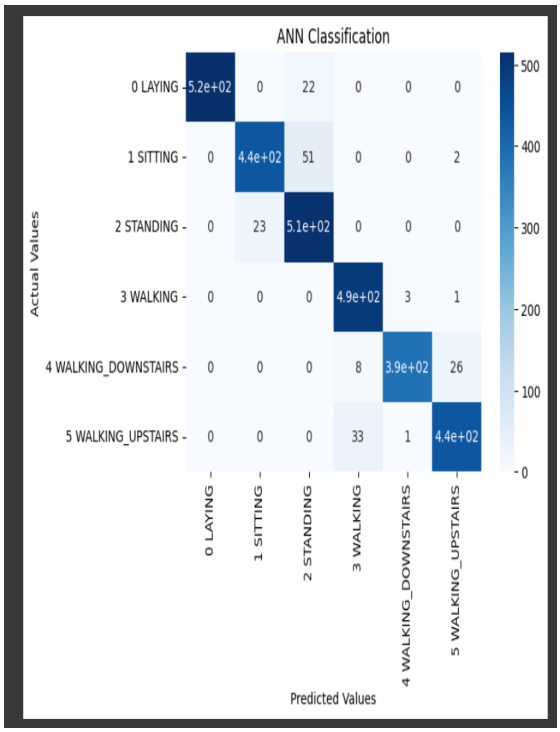

Figure 12: Confusion Matrix of ANN Classifier

6. MODEL ACCURACY & EXECUTION TIME COMPARISION

A. Existing Work (Base Paper)

Mst. Alema Khatuni, "Deep CNN-LSTM With Self-Attention Model for Human Activity Recognition Using Wearable Sensor", Received 24 January 2022; revised 15 May 2022; accepted 16 May 2022.Date of publication 25 May 2022; date of current version 16 June 2022.

Dataset	Type of sensor and location	Sampling Freq.	Samples	Volunteers	Placement of the sensor	Type of activities
H-Activity	Phone based accelerometer, gyroscope, and linear acceleration	10 Hz.	48920	10	Smartphone in trousers right pocket	4 activities. Sitting/standing, walking, jogging, and running
MHEALTH	accelerometer, gyroscope, and magnetometer	50 Hz	120	10	Sensors were implanted on the participant's chest, right wrist, and left ankle.	12 physical activities such as standing, sitting, lying, walking, climbing, cycling, jogging, running jumping.
UCI-HAR	Phone based accelerometer, and gyroscope	50 Hz	10299	30	Smartphone in Waist	6 activities such as: Walking, going upstairs/downstairs, sitting, standing, and lying

Figure 13: H-Activity, MHEALTH, and UCI-HAR datasets used with self-attention Model

Hyper Parameter	Setting
Input Size	10*9
Epochs	150
Batch Size	1000
Activation Function	Sigmoid
Batch Normalization Axis	-1
LSTM Dropout Rate	0.1
Number of Hidden Layer	2
Output Activation Function	Softmax
Optimizer	Adam
Loss	Categorical Cross Entropy

Figure 14: Hyper Parameter of the deep CNN-LSTM with self-attention model.

Dataset	Architecture	Accuracy	Year	Ref.
MHEALTH	LSTM-CNN	95.56	2017	Lyu et al. [53]
	CNN,LSTM	93.80	2016	Ordóñez et al. [54]
-	**CNN-LSTM with Self-Attention Model(M4)**	**98.76**	-	**- Proposed**
UCI-HAR	CNN-LSTM	92	2020	Mutegeki et al [55]
	CNN	91.98	2020	Cruciani et al [56]
	Stacked LSTM	93	2019	Ullah et al [57]
	Res-LSTM	91.6	2018	Yu Zhao et al [58]
-	**CNN-LSTM with Self-Attention Model(M4)**	**93.11**	-	**- Proposed**

Figure 15: Results of the Existing Work (Base Paper)

B. Existing Work (Supporting Paper)

Weiheng Kong, "Exploratory Data Analysis of Human Activity Recognition Based on Smart Phone", Received April 23, 2021, accepted May 8, 2021, date of publication May 12, 2021, date of current version May 24, 2021.

Table 1: Existing Work (Supporting Paper)

Sr. No.	Existing Work	Dataset	Algorithm	Techniques	Accuracy in %	Execution Time
1.	Exploratory Data Analysis of Human Activity Recognition Based on Smart Phone (Supporting Paper)	University of California, Irvin. The experiments were conducted by 30 people between the ages of 19 and 48, wearing smart phones (Samsung Galaxy Phones) on their waists. The dataset includes accelerometer and gyroscope data at a frequency of 50Hz. **Classes:** walk, walkup, down, sit, stand, lie down	Grid Search CV and Linear SVC	pre-processing sensor signals by applying a noise filter and sampling in a sliding window of fixed width for 2.56 seconds with a 50% overlap.	96.56%	NA

C. Proposed Work Dataset and Result

Proposed Work Dataset:

Table 2: Proposed Work-Dataset

Data Set	Type of Sensor	Sampling Frequency	Samples	Volunteers	Placement of the Sensor	Type of Activities
UC Irvine Machine Learning Repository https:// archive.ics.uci.edu/ ml/datasets/ Human+Activity+R ecognition+Using+ Smartphones train.csv (45.9 MB) (7352, 563) test.csv (18.4 MB) (2947, 563)	Embedded Accelerometer and Gyroscope	50Hz	10299	30 volunteers within an age bracket of 19-48 years	Volunteers wearing a smartphone (Samsung Galaxy S II) on the Waist	Laying, Sitting, Standing, Walking, Walking_Downstairs, Walking_Upstairs

D. Proposed Work -ANN Model:

Table 3: Proposed Work – ANN Model

Hyperparameter	Setting
Input Size	561
Epochs	30
Batch Size	16
Activation Function	relu
Batch Normanization Axis	-1
ANN Dropout Rate	No Explicit Dropout
Number of Hidden Layers	2
Output Activation Function	softmax
Optimizer	Adam
Loss	SparseCategoricalCros sentropy

Proposed Work Results:

Training and Validation Dataset Results:

In the given results, it appears that a neural network model is being trained for a classification task over the course of 30 epochs. Let's break down the information provided for each epoch:

- Epoch 1/30: The model is trained for the first epoch
- Epoch 2/30: The model is trained for the second epoch,
- Epoch 3/30 to Epoch 30/30: The training continues for the remaining epochs.

After 30 Epochs -

- Training Loss: 0.0491
- Training Accuracy: 0.9828
- Validation Loss: 0.0515
- Validation Accuracy: 0.9837

Testing Dataset Results:

Table 4: Proposed Work Results-Accuracy, Loss and Execution Time

Dataset	Ar-chitecture	Ac-curacy in %	Loss in %	Model Execu-tion Time (in Sec)
UC Irvine Machine Learning Reposi-tory	ANN	0.94 74	0.1435	0.68502545356 75049

7. CONCLUSIONS

Human activity recognition has become a significant research area in the field of machine learning and data science. The ability to accurately classify human activities based on smartphone sensor data has the potential to revolutionize various domains, including healthcare, fitness tracking, and behavior analysis. In this study, our aim was to develop an Artificial Neural Network (ANN) model capable of classifying human activities using smartphone sensor data.

The dataset utilized for this study consisted of recordings from a group of 30 volunteers who wore a smartphone on their waist while performing six different activities. The smartphone captured 3-axial linear acceleration and 3-axial angular velocity through its embedded accelerometer and gyroscope. By leveraging this dataset, we sought to train an ANN model that could effectively recognize and classify the activities performed by individuals. To achieve this goal, the dataset was divided into a training set, which encompassed 70% of the volunteer data, and a test set, which contained the remaining 30% of the data. The ANN model was trained using the training dataset, employing 30 epochs and a batch size of 16. The model's performance was evaluated on the test dataset, measuring its accuracy and loss as evaluation metrics.

The results of the model training and evaluation revealed the efficacy of the developed **ANN model** in accurately classifying human activities based on smartphone sensor data. The model achieved an impressive **Accuracy of 0.9474%**, **Loss of 0.1435%**& Model **Execution Time is 0.685025453in Sec**, on the test dataset, showcasing its ability to generalize well and predict activities with high precision. The implications of this study extend beyond the realm of research, offering potential applications in various real-world scenarios. Accurate activity recognition using smartphone sensor data can provide valuable insights into individuals' physical movements and behaviors. These insights can be harnessed to develop personalized applications and interventions in areas such as healthcare monitoring, fitness tracking, and behavior analysis.

8. FUTURE WORK

To mitigate overfitting, First and foremost, hyperparameter tuning should be conducted to optimize the model's configuration, including adjusting the learning rate, number of layers, and neurons in each layer. Additionally, careful data preprocessing is crucial, involving handling missing values, feature normalization, and potential data augmentation to expand the dataset's effective size. Experimenting with different neural network architectures, such as CNNs or RNNs, and considering ensemble methods could yield improved results. Regularization techniques should also be applied to prevent overfitting, and learning rate scheduling might help speed up convergence. Furthermore, investigating model compression techniques could lead to reduced execution times without sacrificing accuracy. Utilizing transfer learning and implementing cross-validation can aid in leveraging pre-trained models and obtaining better performance estimates. Regular monitoring and periodic retraining will ensure the model stays up-to-date with new data. Finally, for certain applications, interpreting the model's decisions can provide valuable insights into the most influential features in the predictions. By addressing these aspects, the ANN model's overall performance can be significantly enhanced, making it more robust and effective for real-world applications.

REFERENCES:

[1] Mst. Alema Khatun, Mohammad Abu Yousuf, Sabbir Ahmed,Md. Zia Uddin, Salem A. Alyami, Samer Al-Ashhab, Hanan F. Akhdar, Asaduzzaman Khan, Akm Azad, Mohammad Ali Moni, "Deep CNN-LSTM With Self-Attention Model for Human Activity Recognition Using

Wearable Sensor", Received 24 January 2022; revised 15 May 2022; accepted 16 May 2022, Date of publication 25 May 2022; date of current version 16 June 2022.

[2] Dipanwita Thakur, Suparna Biswas, Edmond S.L. Ho, Samiran Chattopadhyay, "ConvAE-LSTM: Convolutional Autoencoder Long Short-Term Memory Network for Smartphone-Based Human Activity Recognition", Received December 12, 2021, accepted December 23, 2021, date of publication January 4, 2022, date of current version January 13, 2022.

[3] Weiheng Kong, Lili He, Hailong Wang "Exploratory Data Analysis of Human Activity Recognition Based on Smart Phone", Received April 23, 2021, accepted May 8, 2021, date of publication May 12, 2021, date of current version May 24, 2021.

[4] Mutegeki Ronald, Alwin Poulose, Dong Seog Han "iSPLInception: An Inception-ResNetDeep Learning Architecture for Human Activity Recognition", Received April 21, 2021, accepted May 3, 2021, date of publication May 7, 2021, date of current version May 14, 2021.

[5] Sylvain Iloga, Alexandre Bordat, Julien Le Kernec, Olivier Romain, "Human Activity Recognition Based on Acceleration Data From Smartphones Using HMMs", Received September 9, 2021, accepted September 28, 2021, date of publication October 4, 2021, date of current version October 18, 2021.

[6] Anna Ferrari, Daniela Micucci, Marco Mobilio, Paolo Napoletano "On the Personalization of Classification Models for Human Activity Recognition", Received January 16, 2020, accepted February 2, 2020, date of publication February 12, 2020, date of current version February 24, 2020.

[7] Isah A. Lawal, Sophia Bano, "Deep Human Activity Recognition With Localisation of Wearable Sensors", Received July 24, 2020, accepted August 5, 2020, date of publication August 18, 2020, date of current version September 3, 2020.

[8] Tahmina Zebin, Patricia J. Scully, Niels Peek, Alexander J. Casson, And Krikor B. Ozanyan, "Design and Implementation of a Convolutional Neural Network on an Edge Computing Smartphone for Human Activity Recognition", Received August 29, 2019, accepted September 10, 2019, date of publication September 16, 2019, date of current version September 27, 2019.

[9] Nikhil B. Gaikwad, Varun Tiwari, Avinash Keskar, N. C. Shivaprakash, "Efficient FPGA Implementation of Multilayer Perceptron for Real-Time Human Activity Classification", Received November 27, 2018, accepted February 8, 2019, date of publication February 25, 2019, date of current version March 12, 2019.

[10] Meo Vincent C. Caya; Analyn N. Yumang; Jhayvee V. Arai; John Daryll A. Niñofranco; Kenneth Aaron S. Yap, "Human Activity Recognition Based on Accelerometer Vibrations Using Artificial Neural Network", 978-1-7281-3044-6/19/$31.00 ©2019 IEEE

[11] Erhan BÜLBÜL, Aydın ÇETİN, İbrahim Alper DOĞRU, "Human Activity Recognition Using Smartphones", 978-1-5386-4184-2/18/$31.00 ©2018 IEEE

[12] Abdul Kadar Muhammad Masum,

Arnab Barua, Erfanul Hoque Bahadur, "Human Activity Recognition Using Multiple Smartphone Sensors", 978-1-5386-8524-2/18/$31.00 ©2018 IEEE

[13] Hongkai Chen, Sazia Mahfuz, Farhana Zulkernine, "Smart Phone Based Human Activity Recognition", 978-1-7281-1867-3/19/$31.00 ©2019 IEEE.

[14] Zhenghua Chen, Chaoyang Jiang∗, and Lihua Xie Fellow, IEEE, "A Novel Ensemble ELM for Human Activity Recognition Using Smartphone Sensors", IEEE, Transactions On Industrial Informatics (Volume: 15, Issue: 5, May 2019)

[15] Zhenghua Chen, Qingchang Zhu, Yeng Chai Soh, and Le Zhang, "Robust Human Activity Recognition Using Smartphone Sensors via CT-PCA and Online SVM", IEEE Transactions on Industrial Informatics (Volume: 13, Issue: 6, December 2017)

[16] Andreea Valeria Vesa, Simion Vlad, Renato Rus, Marcel Antal, Claudia Pop, Ionut Anghel,Tudor Cioara, Ioan Salomie, "Human Activity Recognition using Smartphone Sensors and Beacon-based Indoor Localization for Ambient Assisted Living Systems", 978-1-7281-9080-8/20/31.00c 2020IEEE.

[17] Ashim Saha, Tulika Sharma, Harshika Batra, Anupreksha Jain, Vabna Pal, "Human Action Recognition Using Smartphone Sensors", 2020 International Conference on Computational Performance Evaluation (ComPE), North-Eastern Hill University, Shillong, Meghalaya, India. July 2–4, 2020.

[18] Mst. Alema Khatun, Mohammad Abu Yousuf†, "Human Activity Recognition Using Smartphone Sensor Based on Selective Classifiers", 2020 2nd International Conference on Sustainable Technologies for Industry 4.0 (STI) | /20/$31.00 ©2020 IEEE | DOI: 10.1109/STI50764.2020.9350486.

[19] Yanjia Zhang, Kandethody M. Ramachandran, "Offline Machine Learning for Human Activity Recognition with Smartphone", ©2020 IEEE

[20] Tuan Dinh Le, Chung Van Nguyen, "Human Activity Recognition by smartphone", 978-1-4673-6640-3/15/ $31.00 ©2015 IEEE

[21] Florenc Demrozi, Graziano Pravadelli, Azra Bihorac, Parisa Rashidi "Human Activity Recognition Using Inertial, Physiological and Environmental Sensors: A Comprehensive Survey", Received September 22, 2020, accepted October 15, 2020, date of publication November 16, 2020, date of current version December 7, 2020.

[22] Muhammad Ehatisham-Ul-Haq, Muhammad Awais Azam, Yasar Amin, Usman Naeem "C2FHAR: Coarse-to-Fine Human Activity Recognition With Behavioral Context Modeling Using Smart Inertial Sensors", Received December 23, 2019, accepted December 30, 2019, date of publication January 6, 2020, date of current version January 15, 2020.

[23] Zhengjie Wang, Yushan Hou, Kangkang Jiang, Chengming Zhang, Wenwen Dou, Zehua Huang, Yinjing Guo, "A Survey on Human Behavior Recognition Using Smartphone-Based Ultrasonic Signal", Received

June 24, 2019, accepted July 15, 2019, date of publication July 25, 2019, date of current version August 9, 2019.

[24] Akram Bayat, Marc Pomplun, Duc A. Tran, "A Study on Human Activity Recognition Using Accelerometer Data from Smartphones", Procedia Computer Science 34 (2014) 450 – 457, DOI: 10.1016/ j.procs.2014.07.009.

[25] Daniel Garcia-Gonzalez,Daniel Rivero, Enrique Fernandez-Blanco, Miguel R. Luaces, "New machine learning approaches for real-life human activity recognition using smartphone sensor-based data", https://doi.org/10.1016/ j.knosys.2023.110260.

[26] Usharani J, Dr. Usha Sakthivel, "Human Activity Recognition using Android Smartphone", International Journal of Advanced Networking and Applications (IJANA), ISSN: 0975-0282.

[27] Neetish Singh, Rajat Yadav, Harshit Kumar Singh, Shivani Agarwal, "Human Activity Recognition Using Smartphone sensors", International Journal of Information Sciences and Application (IJISA). ISSN 0974-2255, Vol.11, No.1, 2019, (Special Issue) ©International Research Publication House. http:// www.irphouse.com.