# Classification for Human Activity Based on time series data

WANG Xiaoxuan(58232446) *
*City University of Hong Kong*
Hong Kong S.A.R.
xwang2977-c@my.cityu.edu.hk

WANG Jun (57996922) *
*City University of Hong Kong*
Hong Kong S.A.R.
jwang2357-c@my.cityu.edu.hk

HUANG Kunlun (57878689) *
*City University of Hong Kong*
Hong Kong S.A.R.
kl.h@my.cityu.edu.hk

ZHENG Yinglin (58137098) *
*City University of Hong Kong*
Hong Kong S.A.R.
ylzheng23-c@my.cityu.edu.hk

***Abstract*—Human activity recognition (HAR) utilizing wearable inertial sensors has nowadays become a new research hotspot due to various advancements in sensor technology. Recently, methods based on deep learning (DL) have been successfully used to evaluate time series data recorded by wearable sensors and smartphone to predict a variety of human activity. In this project, the contribution is to utilize a variety of classification algorithms—KNN, Decision Tree, Random Forest, SVM, and CNN to classify the Human Activity Recognition (HAR) dataset (response to Q2: What is the contribution of this work?). During this process, the concept of feature selection is introduce to optimize performance (response to Q1: How does the performance compared with the existing work?). Our objective is to achieve the highest possible accuracy, aiming for over 95%, in a succinct timeframe. Finally, we use CNN algorithm to achieve 96.49% accuracy, and the whole process take 2min12.2s (response to Q1: How does the performance compared with the existing work? Q5: How long does it take to train the CNN?). Meanwhile, we try to apply clustering algorithms to this dataset. Through clustering analysis, it is possible to uncover unlabelled or undefined patterns of user behavior. For instance, clustering can reveal different types of walking (e.g., fast and slow walking) or unusual patterns of behavior which is particularly useful in applications such as health monitoring and elderly care (response to Q3: Why use clustering method considering it is a classification problem? ).**

***Index Terms*—HAR, Feature Selelction, KNN, Decision Tree, Random Forest, SVM, CNN**

* Same Contribution

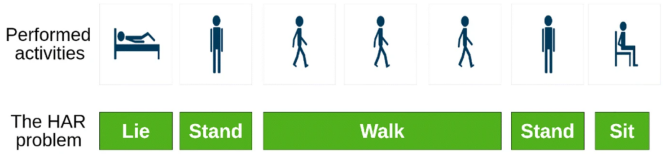



Fig. 1. Advanced Sensing and Human Activity Recognition in Early Intervention and Rehabilitation of Elderly People

## I. Introduction

Human Activity Recognition (HAR) is a broad field of research that involves identifying specific movements of people based on sensor data. Movements are usually typical activities such as walking, standing, sitting, lying down, going up and down stairs, etc. This area used to be challenging because sensor data for activity recognition was time-consuming and labor-intensive to collect and required hardware support. Fortunately, the widespread popularity of smartphones and wearable devices has made sensor data from these devices easy to obtain. Using sensor data from wearable devices, researchers can develop models to identify users' daily activities. The advancement of this technology is of great significance to improve the fields of personal health management, elderly care, and sports science. For example, for the elderly, HAR technology can be used to monitor their living patterns and promptly detect emergencies such as falls and potential health problems, thereby providing immediate assistance or medical intervention [1]. All in all, HAR has become an important research area for mobile health monitoring and daily activity tracking.

Faced with multi-modal and multi-location complex sensor data, how to use the characteristics of sensor data to extract features with good discriminability to improve the accuracy of HAR based on sensor data is a research issue with research value and practical significance.

In this study, we used the public dataset "A Public Domain Dataset for Human Activity Recognition Using Smartphones". The dataset includes data collected by 30 volunteers using built-in sensors, accelerometers and gyroscopes on smartphones while performing daily activities. 70% of the volunteers were selected to generate training data, and the remaining 30% were selected for generating test data. Because human activities are time series data with a high degree of time dependence, this data set also has a more special dimension, the time window sequence, which calculates 561 feature vectors in the time domain and frequency domain. Due to the high dimensionality of the data set, we also performed corresponding preprocessing. For example we use RFECV to remove redundant feature.

Our goal is to classify and cluster these activities through multiple machine-learning methods on this dataset and evaluate the effectiveness of different methods. Specifically, we explore several different classifiers, including traditional machine learning algorithms, such as K-Nearest Neighbors (KNN), decision trees, random forests, and support vector machines (SVM). There are also deep learning technologies like convolutional neural networks (CNN). Deep learning technology can automatically extract more complex and deep features directly from raw data. As for the intuitive and simple KNN algorithm, we determined the K value through result visualization. The test accuracy of this model is higher than the decision tree and random forest algorithms, but it has the risk of over-fitting. In the SVM model, we choose three different kernel train models. For RBF and sigmoid kernel, we use cross-validation to choose the best parameter gamma and cost. SVM is outstanding for effectively processing features in high-dimensional space. The RBF kernel is particularly effective at processing nonlinear features, which is very suitable for our data sets with complex nonlinear patterns. We actually found that using RBF as kernel indeedly can achieve the highest accuracy 96.56%. For CNN, we took inspiration from the paper titled "Human activity recognition from smartphone sensor data using CNN-LSTM". After using Keras tuner for hyperparameter tuning, the model can achieve 96.49% accuracy. We also tried using various methods such as K-means for clustering, but we did not get good classification results.

It's an important step to validate the performance of a machine learning model using multiple evaluation metrics as it can reveal the model's strengths and weaknesses from different perspectives. In this article, we verify the performance of these models through a variety of evaluation indicators, such as accuracy, recall, precision, etc., and also perform visualization. By visualizing these metrics, we can more easily understand and compare the performance of different models. Our charts and graphs (such as line graphs, bar charts, heat maps, etc.) clearly show the performance of the model on various categories, reveal possible bias or overfitting issues, which help us identify strategies to improve the model.

The subsequent sections of this report are as follows: the"Motivation" section II introduces our project inspiration, the "Related Work" section III surveys the historical and current methodologies in machine learning, the "Method" section IV articulates the HAR data processing approaches examined in this study, the "Experiment" section V describes the experimental setup and datasets used for validation, the "Evaluation Metrics" section V-E interprets the results and their evaluation,the "Future Work" section indicates the direction of further improvement of this project in the future and the "Conclusion" section VII summarizes the findings and suggests directions for future research.

## II. Motivation

### A. Maintaining the Integrity of the Specifications

Modern smartphones and wearable devices are equipped with various sensors that can continuously record users' movement data. This data contains rich information and has great potential for understanding and predicting human behavior patterns. However, to accurately identify specific activity types from these large-scale and complex data requires efficient and robust machine-learning methods.

Although traditional machine learning methods have been widely studied and applied in the field of HAR, these methods often require complex feature engineering and expert knowledge when processing high-dimensional, unstructured sensor data. In addition, as the types of activities increase and the environment changes, the generalization ability and adaptability of these methods are often challenged. Therefore, it has become a natural research trend to explore deep learning technologies that can automatically extract useful features from raw data.

The main motivation of this project is two-fold: first, to evaluate and compare the performance of various classic machine learning algorithms (including KNN, decision trees, random forests, and SVM) with advanced deep learning models (CNN) on HAR tasks. This comparison allows us to better understand the strengths and limitations of each approach. Second, we aim to mine and implement one or more machine learning models that not only have high accuracy but can also be deployed and executed efficiently across different devices and environments to achieve true ubiquity and accessibility.

Through empirical research, this project not only helps to promote the technological progress of HAR but may also reveal performance differences in the existing literature and provide new directions for future research. This has a profound impact on promoting the application of wearable devices in various fields such as health, safety and entertainment.

### B. Knowledge Discovery Philosophy

This is a question that has been asked for a long time. The answer is not simple. The human brain is a complex organ that has evolved over millions of years to process information in a way that is fundamentally different from the way computers do.

However, there are some similarities between the two systems, and by studying the brain, we can gain insights into knowledge discovery and how to build better machine learning algorithms.

There are two main viewpoints on this question:

*1) Should we focus on how the brains work:* In this genre, people are interested in understanding how the brain works and using that knowledge to build better machine learning algorithms. This approach is based on the idea that the brain is the most powerful information processing system we know of, and that by understanding how it works, we can build better algorithms that are more efficient, more robust, and more flexible than those currently available. Thus, people proposed the perception-action cycle, which is a model of how the brain processes information and makes decisions. According to this model, the brain takes in sensory information from the environment, processes it using a combination of bottom-up and top-down processes, and then generates motor commands to control the body's actions. This model has been used to build a variety of machine learning algorithms, including deep learning models, that are inspired by the brain's information processing mechanisms.

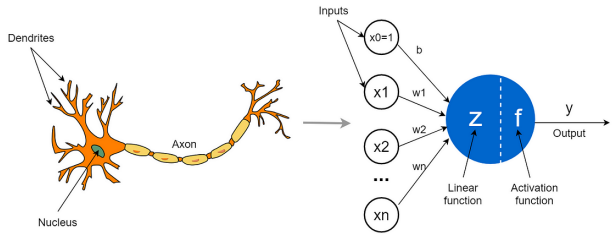



Fig. 2. Neuron v.s. Perceptron [2]

*2) Or how human being thinks, recognizes:* In this another genre, people are interested in understanding how humans think, recognize patterns, and make decisions, and using that knowledge to build better machine learning algorithms. This approach is based on the idea that humans are the most intelligent beings we know of, and that by understanding how they think, we can build better algorithms that are more intelligent, more creative, and more adaptable than those currently available. Thus, people proposed the cognitive architecture, which is a model of how humans think, recognize patterns, and make decisions. According to this model, humans use a combination of logical reasoning, intuition, and creativity to solve problems and make decisions. This model has been used to build a variety of machine learning algorithms, including cognitive computing systems, that are inspired by the human brain's cognitive processes.

So, we also want to explore more on the differences between the two approaches and how they can be combined to build better machine learning algorithms that are both powerful and flexible.

## III. Related Work

Human activity recognition has been extensively studied in the past decade, especially in the context of the popularity of smartphones and other portable devices. These studies mainly focus on utilizing device-built-in sensors, such as accelerometers and gyroscopes, to automatically detect and classify users' physical activities.

### A. Traditional machine learning methods

In early research, K-nearest neighbor (KNN) and support vector machine (SVM) were among the most commonly used classification techniques. For example, Bao and Intille (2004) [3] successfully identified 12 daily activities on subjects carrying sensors using these techniques. Their work demonstrates the effectiveness of using KNN and SVM in HAR.

### B. Decision trees and random forests

Decision trees and random forests are also widely used for activity recognition because of their good performance and ease of interpretation when dealing with high-dimensional data. Thakur et al. (2022) [4]used the random forest algorithm to classify smartphone data in their study, demonstrating its advantages in accuracy and computational efficiency.

### C. Deep learning methods

In recent years, with the development of deep learning technology, methods such as convolutional neural network (CNN) and recurrent neural network (RNN) have been applied to more complex HAR problems. For example, Ronao and Cho (2016) [5] achieved higher classification accuracy than traditional methods on a similar smartphone dataset by building a deep CNN model.

### D. Support vector machine and complex kernel function

Especially in the use of SVM, the use of complex kernel functions (such as RBF kernel) can process data in nonlinear feature space, thus showing better performance in a variety of studies [6] [7]. In addition, multiple studies have proven that after adding appropriate feature engineering, SVM with RBF kernel performs superiorly in HAR tasks. [8] [9]

## IV. Methodologies

### A. Datasets introduction

*1) Participants:* The dataset was built from the recordings of 30 participants, referred to as subjects. These individuals performed a variety of activities while carrying a waist-mounted smartphone that contained inertial sensors.

*2) Activities:* The activities recorded include six basic categories: three static postures (standing, sitting, lying) and three dynamic activities (walking, walking downstairs, and walking upstairs), which results in the 6 class for us to classify, as shown in Table I.

TABLE I
CLASS LABELS

| Class | Label |
|---|---|
| 0 | WALKING |
| 1 | WALKING_UPSTAIRS |
| 2 | WALKING_DOWNSTAIRS |
| 3 | SITTING |
| 4 | STANDING |
| 5 | LAYING |

*3) Data collection:* The embedded inertial sensors in the smartphone, namely the accelerometer and gyroscope, were used to capture 3-axial linear acceleration and 3-axial angular velocity at a constant rate. This resulted in time-series data that could be analyzed to distinguish different activities.

*4) Data processing:* The sensor signals were preprocessed by applying noise filters and then sampled in fixed-width sliding windows of $2.56\ seconds$ with $50\%$ overlap. The sensor acceleration signal, which has gravitational and body motion components, was separated using a Butter-worth low-pass filter.

*5) Features:* From these time windows, a vector of features was obtained by calculating variables from the time and frequency domain. This includes measures like mean, standard deviation, and maximum and minimum values.

*6) Dataset structure:* The resulting dataset has been divided into two sets:

- The training set, which includes about $70\%$ of the volunteers.
- The test set, which includes the remaining $30\%$.

*7) Applications:* This dataset is widely used for developing algorithms and models that can recognize and classify human activities using time-series data. It's particularly useful for researchers and developers working on mobile health monitoring systems, assistive technology, and fitness apps.

*8) Availability:* The HAR dataset is publicly available and can be found on platforms HCI for researchers and developers to download and use in their projects.

### B. Data preprocessing

*1) Feature standardization:* In machine learning, feature normalization is a common data preprocessing step that adjusts all features to the same scale to avoid adverse effects on model training due to large differences in the range of feature values.

We use **StandardScaler** in the sklearn.preprocessing module which is one of the tools used for this type of normalization to normalization data.

StandardScaler aims to adjust the distribution of features to have a mean of 0 and a standard deviation of 1, which corresponds to a standard normal distribution. Specifically, it transforms each feature using the following formula:

$$z = \frac{x - \mu}{\sigma}$$

where $\mu$ is the mean and $\sigma$ is the standard deviation of the feature. The result $z$ is the standardized value.

*2) Feature selection:* In this project, we address concerns regarding the efficacy of the 561 attributes in our dataset, which may include features that are irrelevant or detrimental to the classification accuracy. To identify and eliminate these redundant features, we have implemented Recursive Feature Elimination with Cross-Validation (RFECV). Following the feature selection process, we employ a Support Vector Machine (SVM) model to assess the impact of feature reduction on the classification accuracy. This approach ensures that only the most relevant features are retained, potentially improving the model's performance and efficiency.

### C. Classification

*1) K Nearest Neighbors (KNN):* K-Nearest Neighbors (KNN) is a classic machine learning algorithm used for classification tasks. It is a non-parametric algorithm that makes predictions based on the majority vote of its k nearest neighbors. In this study, we employed the KNN algorithm as one of the classifiers for human activity recognition.

The KNN algorithm follows a simple yet effective approach. Given a new input data point, it identifies the k nearest data points in the training dataset based on a distance metric (e.g., Euclidean distance). The class label of the majority of these k neighbors is assigned to the new data point.

While apply KNN as classification algorithm, it show some advantages as following:

- **No Training Phase**: Unlike many other machine learning algorithms, KNN does not require an explicit training phase. It stores the entire training dataset and makes predictions based on the similarity of new data points to the existing ones.
- **Non-parametric**:KNN is a non-parametric algorithm, meaning it does not make assumptions about the underlying distribution of the data. It can handle data with complex patterns and does not impose strict constraints on the data distribution.
- **Flexibility**: KNN can be applied to both classification and regression tasks. It can handle multiclass classification problems and works well with datasets that have nonlinear decision boundaries.
- **Adaptability to New Data**: KNN can easily incorporate new training data without the need for retraining the entire model. It can dynamically adjust to changes in the dataset, making it suitable for scenarios where new data is continuously added.

It's important to note that KNN also has some limitations. It can be computationally expensive, especially with large datasets, as it requires calculating distances between data points.

In this project, we use Euclidean distance to measure the distance between samples. Euclidean distance is a widely used measure to calculate the distance between two points in a Euclidean space. It represents the straight-line or shortest distance between the points. Euclidean distance measures the straight-line distance, assuming a continuous and obstacle-free space. Overall, Euclidean distance is a versatile and intuitive measure, commonly used in various applications. Its formula shows as following :

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots + (x_n - y_n)^2}$$

After comparing distance, the value of K plays an important role in classification, and even directly determines the accuracy of result. A small value of k may lead to overfitting, while a large value may result in underfitting.

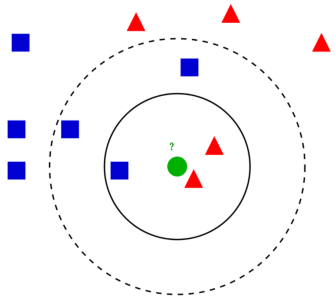


Fig. 3. Different K values

In the dataset like figure3, a new sample, labeled a green circle, calculates the distance from each training sample and performs the classification task. When the value of K is 3, the new sample is classified as the red triangle. However, when the value of k is 5, the new sample is classified as a blue square.

*2) Decision trees and random forests:* Decision tree is a commonly used machine learning algorithm for classification and regression tasks. It employs a tree-like structure to represent decision rules and partitions the dataset into different subsets based on the conditions of the features. Nodes in the decision tree represent features, edges represent feature values, and leaf nodes represent the final classification or regression outcomes. The construction of a decision tree involves selecting the best features and splitting criteria based on metrics like information gain and Gini impurity. Decision trees are known for their interpretability, ease of understanding,

and explanation. They are suitable for handling data with discrete features.

As a widely used classification algorithm, decision tree is applied in this project mainly because of its following advantages:

- **Handling both numerical and categorical data**: SVMs are effective in cases where the number of dimensions exceeds the number of samples.Decision trees can handle a variety of data types, including numerical and categorical features. They can handle mixed data without requiring extensive preprocessing.
- **Nonlinear relationships**: Decision trees can capture nonlinear relationships between features and the target variable by recursively dividing the data based on the selected features.
- **Feature importance**: Decision trees can provide a measure of feature importance, indicating which features are most influential in the decision-making process. This information can be valuable for feature selection and understanding the underlying data.
- **Scalability**: Decision trees can handle large datasets efficiently. The computational complexity of building a decision tree is generally linear with respect to the number of instances and features.

In this project, we choose CART (Classification and Regression Trees) with Gini impurity as the criterion.

The CART algorithm is a widely used approach for constructing decision trees that can handle both classification and regression tasks. It recursively partitions the data based on the selected features to create a binary tree. The splitting criterion in CART is determined by the Gini impurity, which measures the probability of misclassifying a randomly chosen element from the dataset. The goal of CART is to minimize the Gini impurity across all the resulting partitions.

The CART algorithm with Gini impurity is well-suited for this task because of several advantages. Firstly, the Gini impurity criterion is computationally efficient to calculate, making it suitable for large datasets. Secondly, Gini impurity tends to favor splits that result in balanced partitions, which can lead to more robust and less biased trees. Additionally, decision trees based on Gini impurity often produce simpler trees compared to other criteria like information gain, which can enhance interpretability.

The value of Gini impurity can be calculated by the following formula:

$$Gini(\mathbf{D}) = 1 - \sum_{i=1}^{k} p_i^2$$

where: $i$ is the number of classes, denoting the distinct categories present within the given node. $k$ is the index of a class. The range of i is from 1 to K, corresponding to each class. $p_i$ represents the probability of an instance belonging to class i.

*3) Support Vector Machine (SVM):* Support Vector Machine (SVM) is a powerful and versatile supervised machine learning algorithm used for both classification and regression tasks. However, it is most commonly used in classification problems. Developed in the 1960s and refined in the 1990s, SVMs are particularly well-suited for complex but small- or medium-sized datasets.

SVM has following advantages:

- **Effectiveness in High-dimensional Spaces**: SVMs are effective in cases where the number of dimensions exceeds the number of samples.
- **Memory Efficiency**: Uses a subset of training points (support vectors), so it is also memory efficient.
- **Versatility**: Different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

The goal of the SVM algorithm is to find the best separating hyperplane (also known as the decision boundary) that divides the data into classes with the maximum margin. The ”margin” refers to the distance between the hyperplane and the nearest data points from each class, which are known as support vectors, since these points support or define the hyperplane.

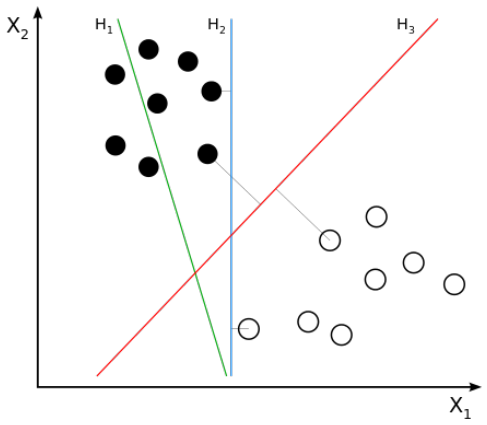



Fig. 4. SVM diagram

As shown in the figure4, H1 does not separate the classes. H2 does, but only with a small margin. H3 separates them with the maximal margin.

In this project, we use linear, RBF (Radial Basis Function), and sigmoid respectively as the kernels of SVM models:

- **Linear**: The linear kernel is one of the simplest and most commonly used kernel functions in Support Vector Machines (SVM). It is particularly useful when dealing with linearly separable data, where the goal is to find a hyperplane that separates the classes with a maximum margin.
  The equation for the linear kernel between two samples $x$ and $x'$ is given by:
  $$K(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{n} x_i x_i'$$
  where: $x_i$ and $x_i'$ are the components of the vectors respectively, and n is the dimensionality of the vectors.
  The decision function for a binary classification SVM model using a linear kernel is:
  $$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$$
  where: $w$ is the weight vector perpendicular to the hyperplane, $b$ is the bias term. $x$ is the input feature vector.
- **RBF**: The RBF kernel, often referred to as the Gaussian kernel, transforms the input data into a higher-dimensional space where a linear separator (hyperplane) might be found to effectively separate the classes. This kernel function is based on the distance between any two given points, providing a measure of similarity that decreases (exponentially) with distance.
  The equation for the RBF kernel between two samples $x$ and $x'$ is given by:
  $$K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$$
  where: $\|\mathbf{x} - \mathbf{x}'\|^2$ represents the squared Euclidean distance between the two feature vectors, $\gamma$ is a parameter that defines the spread of the kernel and must be set before training the model. A large $\gamma$ value leads to a narrower peak (i.e., the kernel's effect is limited to a closer range of its input space), while a smaller $\gamma$ value makes the kernel broader (i.e., it influences a larger region of the input space).
  The decision function for an SVM with an RBF kernel is not as straightforward to express in closed form due to the complexity introduced by the kernel transformation. However, conceptually, it is given by:
  $$f(x) = \sum_{i=1}^{N} \alpha_i y_i K(x_i, x) + b$$
  where: $\alpha_i$ are the Lagrange multipliers (non-zero for support vectors), $y_i$ are the class labels, $x_i$ are the support vectors, b is the bias term.
- **Sigmoid**: The sigmoid function, also known as the logistic function, maps any real-valued number into the interval (0, 1), making it suitable for probabilities. It is defined mathematically as:
  $$\sigma(z) = \frac{1}{1 + e^{-z}}$$
  When used in conjunction with SVM, the sigmoid function can be applied to the output of the SVM decision function to obtain a probability estimate. The decision function in SVM typically gives the raw output score $f(x)$ for a data point $x$, and you can convert this score to a probability $p$ using the sigmoid function:
  $$p = \sigma(f(x)) = \frac{1}{1 + e^{-f(x)}}$$
  Here, $f(x)$ is the output from the SVM model's decision function, which may depend on the kernel used and the parameters of the model.
  In practice, when using SVMs for probability estimation, the parameters of the sigmoid function $A$ and $B$ (where the sigmoid takes the form $\sigma(z)) = \frac{1}{1+e^{A+B_z}}$ are often calibrated using methods like Platt scaling. Platt scaling involves fitting the sigmoid function to the scores output by the SVM on a training set, typically using a maximum likelihood estimation.

SVM is primarily designed for binary classification. However, in this project, we extend its application to multiclass classification by employing the One-vs-All (OvA) and One-vs-One (OvO) strategies:

- One-vs-All, also known as One-vs-Rest, is a strategy for using binary classification algorithms for multi-class classification tasks. In OvA, a single classifier per class is trained with the samples of that class as positive samples and all other samples as negatives.
  For classifier Training, $N$ different SVM classifiers are trained. Each classifier $C_i$ is trained to distinguish the samples of a single class $i$ (positive class)

from the samples of all other classes (negative class).
In terms of decision rule, to classify a new sample, all $N$ classifiers are evaluated, and the classifier that has the highest output function value (distance from the decision boundary) assigns the class label to the sample.

- One-vs-One: One-vs-One is another strategy used for multi-class classification with binary classifiers. In this approach, a binary classifier is trained for every pair of classes. Thus, if there are $N$ classes, $\binom{N}{2} = \frac{N(N-1)}{2}$ classifiers are needed.
  In terms of classifier training, for each pair of classes $(i, j)$, a classifier $C_{ij}$ is trained using the data from these two classes only.
  In terms of decision rule, to classify a new sample, every classifier $C_{ij}$ votes for one of the two classes it was trained on. The class that receives the most votes overall is assigned to the sample.

*4) Convolutional Neural Network(CNN):* Convolutional neural network (CNN) is a regularized type of feed-forward neural network that learns feature engineering by itself via filters (or kernel) optimization. Vanishing gradients and exploding gradients, seen during backpropagation in earlier neural networks, are prevented by using regularized weights over fewer connections. [10] [11] For example, for each neuron in the fully-connected layer, 10,000 weights would be required for processing an image sized 100 × 100 pixels. However, applying cascaded convolution (or cross-correlation) kernels, [12] [13] only 25 neurons are required to process 5x5-sized tiles. [14] [15] Higher-layer features are extracted from wider context windows, compared to lower-layer features.

### D. Cluster

*1) Kmeans:* k-means clustering is a method of vector quantization, originally from signal processing, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster.

Given a set of observations $(x_1, x_2, ..., x_n)$, where each observation is a d-dimensional real vector, k-means clustering aims to partition the n observations into $k(\leq n)$ sets $S = S_1, S_2, ..., S_k$ so as to minimize the within-cluster sum of squares (WCSS) (i.e. variance). Formally, the objective is to find:

$$\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \mu_i\|^2$$

where $\mu_i$ is the mean (also called centroid) of points in $S_i$, i.e.

$$\mu_i^{(t+1)} = \frac{1}{\left|S_i^{(t)}\right|} \sum_{\mathbf{x}_j \in S_i^{(t)}} \mathbf{x}_j$$

.

The algorithm proceeds by alternating between two steps:

- **Assignment step**: Assign each observation to the cluster with the nearest mean: that with the least squared Euclidean distance.(Mathematically, this means partitioning the observations according to the Voronoi diagram generated by the means.)

$$S_i^{(t)} = \left\{ x_p : \left\| x_p - m_i^{(t)} \right\|^2 \leq \left\| x_p - m_j^{(t)} \right\|^2 \ \forall j, 1 \leq j \leq k \right\}$$

- **Update step**: Recalculate means (centroids) for observations assigned to each cluster.

$$m_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j$$

*2) OPTICS:* Ordering Points To Identify the Clustering Structure (OPTICS) is a data clustering algorithm developed by Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander in 1999. [16] It extends the concepts of DBSCAN, and is designed to overcome its limitations regarding varying density clusters. OPTICS is a density-based clustering non-parametric algorithm that creates an ordering of the database such that spatially closest points become neighbors in the ordering, which facilitates the identification of cluster structures at different scales. The OPTICS algorithm can be summarized by the following steps: [16]

- Construct a reachability plot for the dataset, which orders points so that spatially nearest neighbors become neighbors in the ordering.
- For each point, calculate the core distance, defined as the smallest distance such that the point is a core point and has minimum number of minPts within this distance.
- Compute the reachability distance for each point, which is defined as the maximum of the core distance of a point and the actual distance to the nearest core point.
- Process each point in the order, updating the reachability distances of neighboring points. This creates

a reachability plot capturing the data's density-related clustering structure.
- Extract clusters from the reachability plot by identifying valleys which correspond to regions of high density separated by peaks of low density.

Like DBSCAN, OPTICS deals with spatial data and identifies clusters based on density. However, it does not require a global density threshold (as DBSCAN's $\epsilon$) and can discover clusters of varying densities, making it more flexible and applicable to more complex datasets.

## V. Experiment

### A. Prerequisites

TABLE II
Experiment Environment

| ENV | Version/Specs |
|---|---|
| Machine | Apple M2 |
| Memory | Built-in 8GB |
| OS | OSX |
| Pytorch | 2.2.2 |
| Sklearn | 1.4.1.post1 |

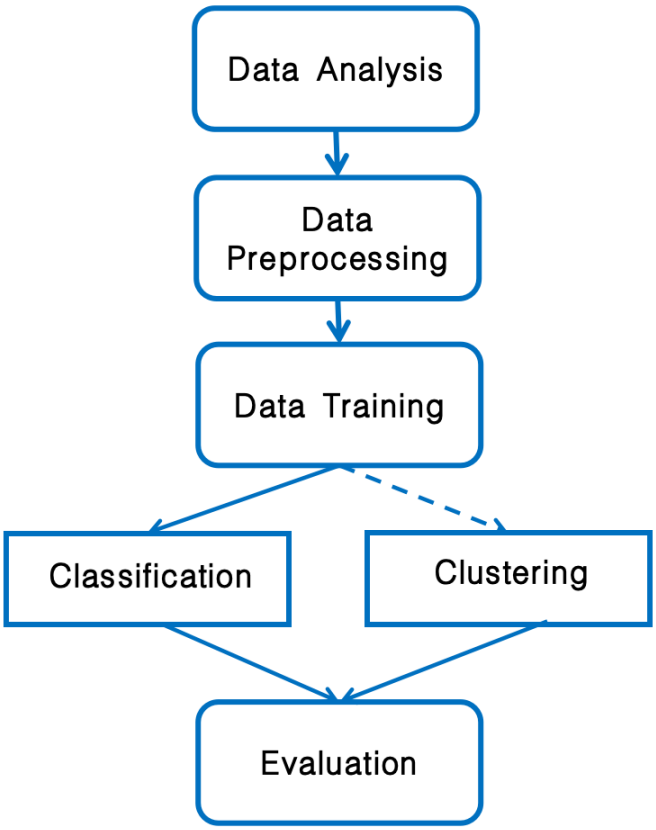



Fig. 5. Experiment step

### B. Experiment Setup

The whole setup of this experiment is shown in figure5.

*1) Data analysis:* By analyzing the data acquisition process, we can understand every feature of the data and its significance in real life, as well as how to effectively use time series data, which will help us to preprocess the data to achieve better training effect.

*2) Data preprocessing:* The experimental dataset features have 561 dimensions. Implementing effective dimensionality reduction and data standardization can significantly decrease the time required for data training while also enhancing accuracy.

*3) Data training:* The data training in this experiment comprises two components: data classification and data clustering. For classification, we employ several algorithms including KNN, Decision Tree, Random Forest, SVM, and CNN. For clustering, we utilize the K-means and OPTICS algorithms.

*4) Evaluation:* After training the data, we employ various methods to evaluate the effectiveness of the models.

### C. Datasets

*1) Dataset analysis:* Each sample contains the following information:

- Total three-axis acceleration and estimated body acceleration from the accelerometer;
- the Triaxial angular velocity from gyroscopes.
- 561 eigenvectors in the time and frequency domains.
- Labels.
- Volunteer identifiers.

*2) Raw Data Preprocess:* As shown in figure 6, the sensor signals are preferred to be preprocessed through a lowpass noise filter, and then the data are selected in a fixed time window of 2.56 $seconds$ per sample, that is, 128 sampling points of data. Whereas the sensor acceleration signal has gravity and body motion components, it is separated into body acceleration and gravity acceleration using a Butter-worth low-pass filter. It was assumed that gravity has only a low frequency component, so a filter with a cutoff frequency of 0.3 $Hz$ was used.

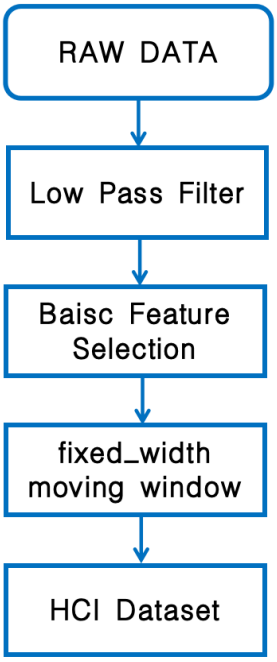



Fig. 6. Raw data preprocess procedures

In each time window, feature vectors were computed in both the time and frequency domains.

As shown in tables III, the dataset contains 7352 samples in the training set and 2947 samples in the test set. Each sample contains 561 features, which are the result of the HCI feature extraction process.

TABLE III
SHAPE OF DATASETS

| | X_train | X_test |
|---|---|---|
| Types of Sensors[a] | 9 | 9 |
| Sampling Rate | 50Hz | 50Hz |
| Fixed_Width Sample Windows | 2.56sec | 2.56sec |
| Cycles[b] | 128 | 128 |
| Counts | 7352 | 2947 |

[a] As mentioned in IV-A3 3 embedded sensors and each gets 3-axial.
[b] $50Hz * 2.56sec = 128cycle$.

When we breakdown the training dataset with different class as shown in table IV, we can see that the dataset is balanced.

As shown in Figure 7, We can say that this is close to Gaussian distribution and thus we cloud apply the standardization of the data.

As for time series distribution, we can see that the data is not evenly distributed, as shown in Figure 8. This is because the data is collected from different volunteers, and the activities performed by each volunteer are different.

TABLE IV
TRAIN DATASET CLASS BREAKDOWN

| Class | Counts | Percentage(%) |
|---|---|---|
| 0 | 1226 | 16.676 |
| 1 | 1073 | 14.595 |
| 2 | 986 | 13.411 |
| 3 | 1286 | 17.492 |
| 4 | 1374 | 18.689 |
| 5 | 1407 | 19.138 |

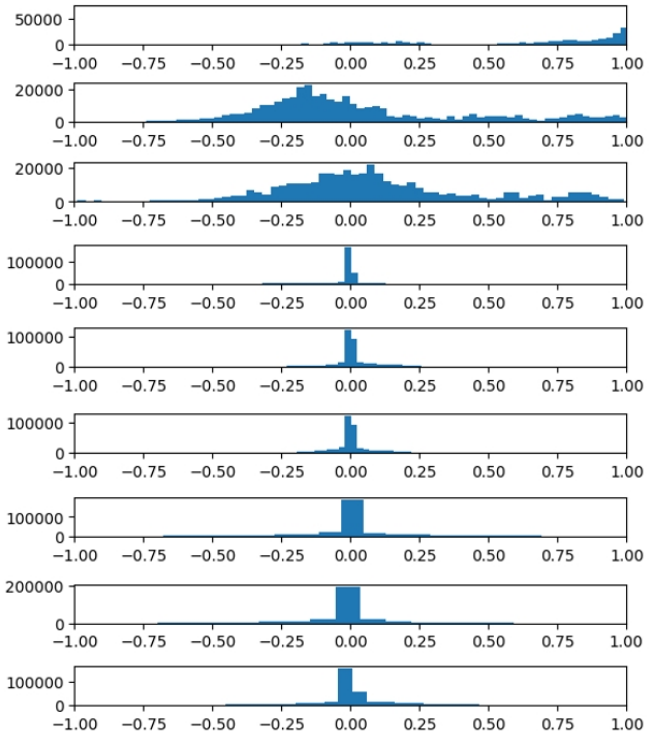



Fig. 7. Distribution for Different Sensor Collected Data

### D. Training Implementation

*1) Decision trees and random forests:* In this experiment, we employed the CART and Gini methods to construct the decision tree and initially trained it using the default parameters from the sklearn library. The accuracy evaluation on the test set for the initial training yielded only approximately 69.49%.

To enhance the accuracy, we considered adjusting the parameter settings of the decision tree, with particular focus on two parameters: min-samples-leaf and min-samples-split. These parameters respectively represent the minimum number of samples allowed in a leaf node

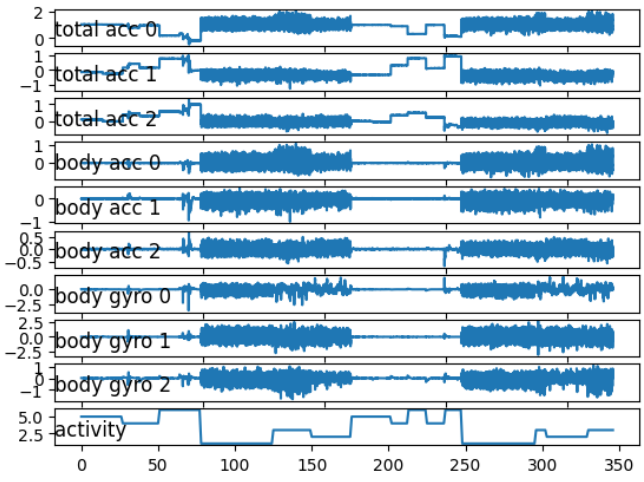



Fig. 8. Distribution for Time Series

during the tree construction and the minimum number of samples required for a node to be split.

During the parameter selection process, we varied the parameter values within the range of 2 to 15 with a step size of 2 and performed iterative training. Subsequently, we compared the accuracy obtained from each training iteration. As shown in figure9, the experimental results revealed that the decision tree model achieved the highest accuracy of approximately 73% when min-samples-leaf was set to 11 and min-samples-split was set to 4 respectively.

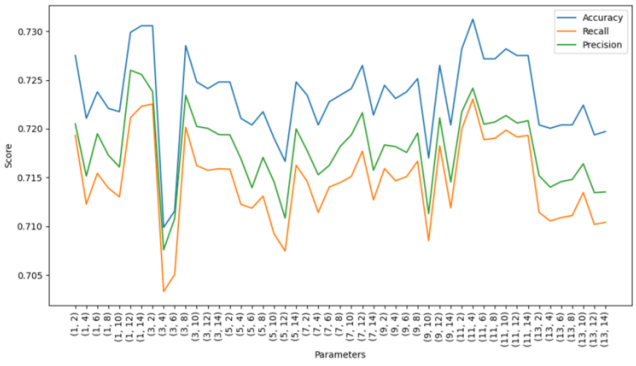



Fig. 9. Accuracy of different parameters of DT

When further experimenting with the Random Forest algorithm, an additional parameter needs to be considered. Since Random Forest consists of multiple decision trees, apart from the minimum number of samples in leaf nodes and the minimum number of samples required for node splitting, the number of trees in the forest also needs to be taken into account. Using the default parameters from the sklearn library for the initial training, the accuracy result already surpassed that of the decision tree method, reaching approximately 83.78%.

When setting the parameters for the Random Forest algorithm, we chose to set the number of trees in the forest as 100, 200, or 300. The range for the minimum number of samples in leaf nodes and the minimum number of samples required for node splitting was set from 1 to 20, with a step size of 3. We iteratively trained the model with different parameter combinations and ultimately compared the accuracy obtained from each training iteration. As shown in figure 10, the experimental results demonstrated that the highest accuracy of around 85.2% was achieved when the minimum number of samples in leaf nodes was set to 1, the minimum number of samples required for node splitting was set to 2, and the number of trees in the forest was set to 200 respectively.

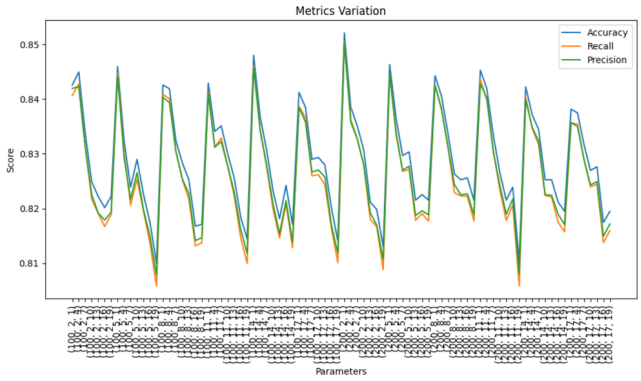



Fig. 10. Accuracy of different parameters of RF

*2) K Nearest Neighbors (KNN):* The dataset used in this experiment is relatively large, and the data dimensionality is complex. As mentioned earlier, when KNN is applied to complex and large-scale datasets, the computational time and overhead increase accordingly. The experimental results indicate that although training time is not required, KNN indeed took a significant amount of time to make predictions on the test set, resulting in relatively high accuracy. When K is set to 1, the accuracy can reach 91.41%.

However, when we conducted repeated experiments by varying the value of K and compared the accuracy, we observed that as the value of K increased, the accuracy decreased, as shown in figure 11.

*3) Feature selection results:* The figure12 illustrates how varying the number of features affects accuracy. Initially, as the number of features increases, there is a significant improvement in the accuracy of the SVM model. This trend continues until the feature count

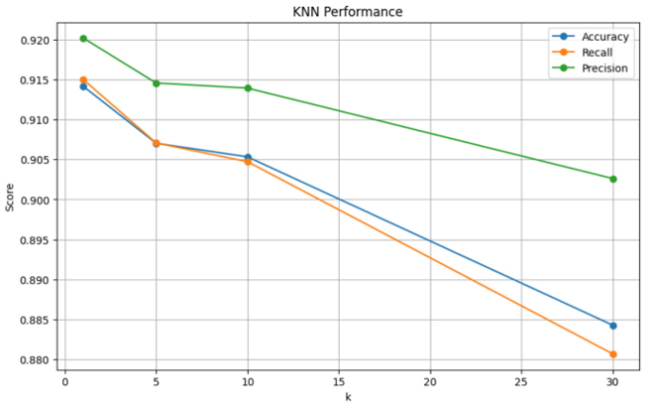



Fig. 11. Accuracy of different parameters of K

reaches approximately 160, at which point the accuracy peaks. Beyond this point, further increases in the number of features lead to a slight decline in accuracy.

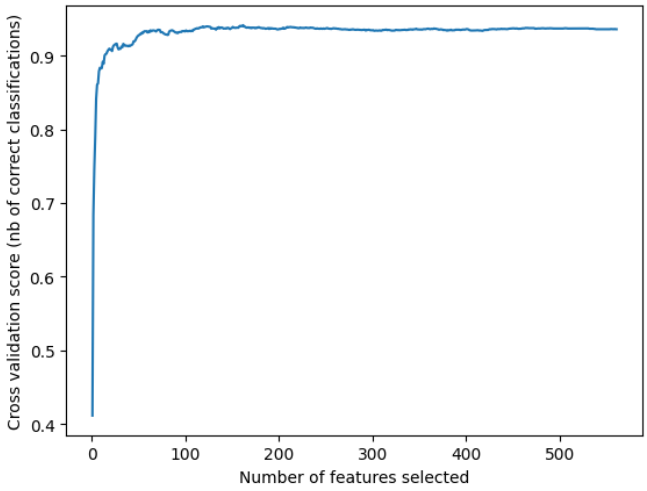



Fig. 12. Number of features selected vs accuracy

Ultimately, we reduced the number of features from 561 to 162. The pipeline of feature selection is shown in the figure13

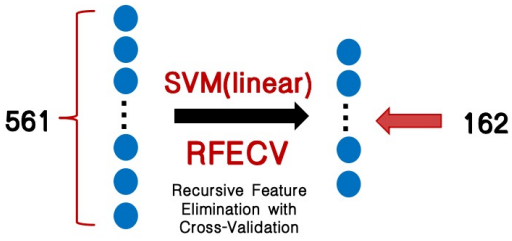



Fig. 13. The pipeline of feature selection

*4) SVM training:* In this experiment, we employ three distinct methods—Linear, KBF, and Sigmoid—as kernel functions for the SVM model to classify the data.

When using the KBF and Sigmoid kernels in an SVM model, it is necessary to establish the appropriate gamma and cost values before training the model. In this project, **cross-validation** is utilized to determine the optimal parameters for the KBF and Sigmoid kernels, ensuring the SVM model achieves the highest possible classification accuracy for this dataset.

The figure1415 depicts the accuracy obtained for various combinations of gamma and cost values when utilizing KBF and Sigmoid kernels, respectively.

For the KBF kernel, the optimal parameters identified are gamma equal to 0.001 and C equal to 100, yielding the highest cross-validation score of 93.89%.

For the sigmoid kernel, the optimal parameters identified are gamma equal to 0.001 and C equal to 10, yielding the highest cross-validation score of 94.61%.

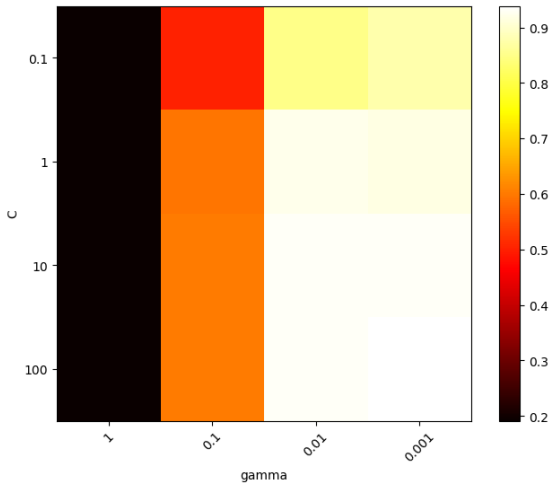



Fig. 14. Grid Search Mean Test Scores with RBF Kernel

*5) CNN training:* After numerous iterations and insights gained from research papers, our CNN model has been structured with two one-dimensional convolutional layers, both utilizing the ReLU activation function. It includes a one-dimensional pooling layer that reduces the spatial dimensions of the data and the number of parameters, which helps in controlling overfitting. Additionally, there is a global max pooling layer that extracts the most significant features from the feature map, further reducing the dimensionality of the data. To prevent overfitting, a Dropout layer is incorporated during training. The whole structure is shown in figure16. This CNN model use the Adam optimizer, minimize

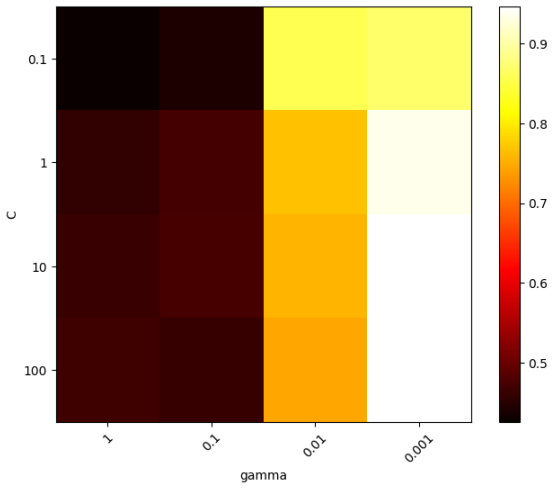



Fig. 15. Grid Search Mean Test Scores with Sigmoid Kernel

categorical crossentropy as its loss, and monitor the categorical accuracy during training.

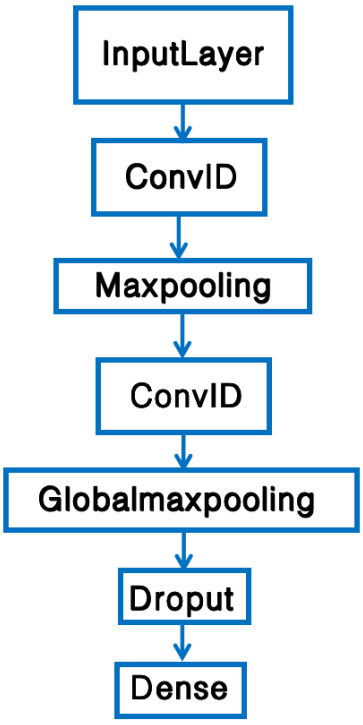



Fig. 16. CNN model

**Hyperparameter Tuning with Keras Tuner** is utilized to determine the optimal parameters for each layer of CNN model. Finally, the first convolutional layer is configured with 32 filters and a kernel size of 3, while the second convolutional layer also has 32 filters but with a larger kernel size of 5. The dropout layer is set to randomly discard 20% of the units during training to help mitigate overfittingV.

For the dataset, we allocate 19% of the training set for validation purposes.

TABLE V
PARAMETER OF CNN

| Layer | Parameter |
|---|---|
| Convolutional Layer1 | (32,3)) |
| Convolutional Layer2 | (32,5) |
| Droput | 0.2 |

*6) Kmeans clustering:* In this experiment, we employ two methods to determine the optimal number of clusters (k): the Elbow Method and the Silhouette Method.

*7) OPTICS clustering:* Through trial and error, we finally selected min_samples=40, xi=.000001, min_cluster_size=40 as the parameters of the OPTICS model.

### *E. Evaluation Metrics*

*1) Decision trees and random forests results:* Confusion matrix is a tool used to measure the performance of a classification model, which presents the relationship between actual and predicted categories in the form of a matrix.

In our experiment, it mainly has the following four indicators:

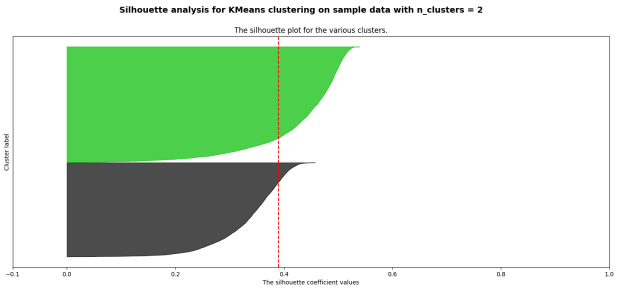
True Positives (**TP**): For the human activity prediction model, TP represents the correct prediction of a specific activity by the model. For example, if a person is walking and the model correctly predicts it as walking, then it would be considered a true positive.

False Positives (**FP**): FP represents the incorrect prediction of a non-activity sample as a specific activity. For instance, when a person is sitting, but the model incorrectly predicts it as walking, then it would be considered a false positive.

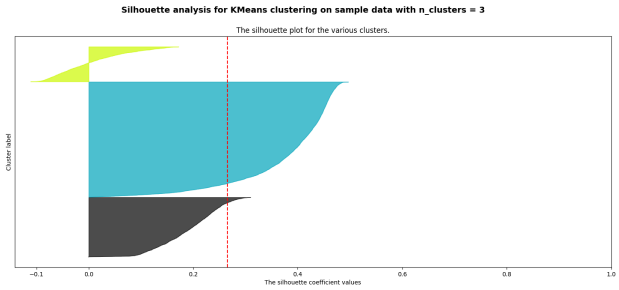
True Negatives (**TN**): TN represents the correct prediction of a non-activity sample as a non-activity. For example, when a person is lying down, and the model correctly predicts it as a non-walking activity, then it would be considered a true negative.

False Negatives (**FN**): FN represents the incorrect prediction of an activity sample as a non-activity. For example, when a person is walking, but the model incorrectly predicts it as sitting, then it would be considered a false negative.

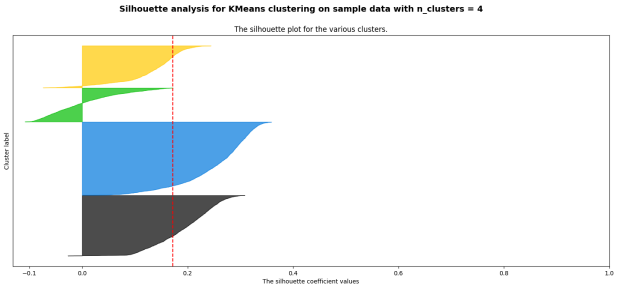
Using these metrics, we calculate the following evaluation metrics to measure the performance of the model:



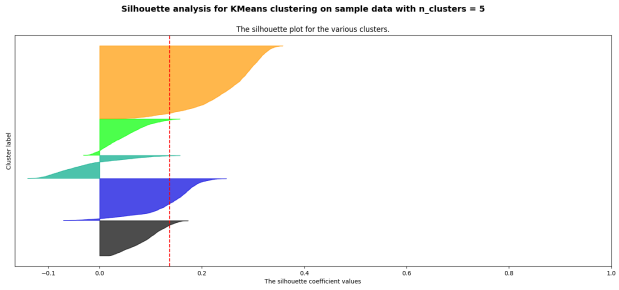

(a) Sihouette score: k=2



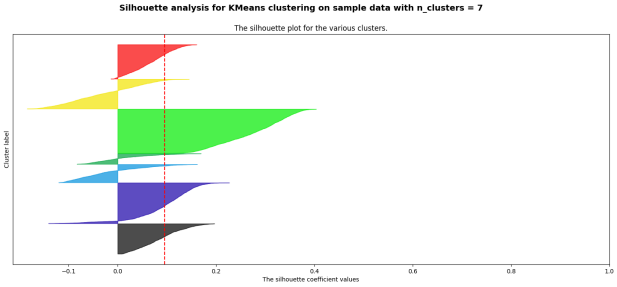

(b) Sihouette score: k=3



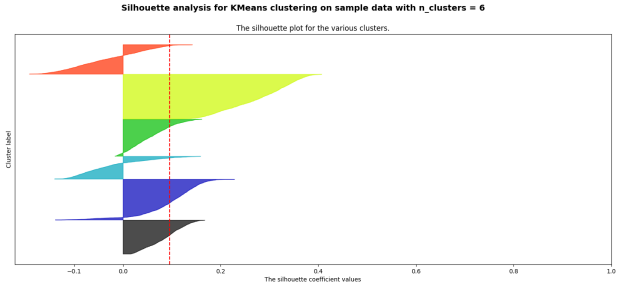

(c) Sihouette score: k=4




(d) Sihouette score: k=5




(e) Sihouette score: k=6




(f) Sihouette score: k=7

Fig. 17. Sihouette Result

**Accuracy**: Accuracy is the ratio of correctly predicted samples to the total number of samples. In the context of human activity prediction, accuracy measures the overall correctness of the model's classifications. It informs us about the model's accuracy in predicting all activities.

**Precision**: Precision is the ratio of correctly predicted samples of a particular activity to all samples predicted as that activity. In human activity prediction, precision measures the accuracy of the model in predicting a specific activity. It informs us about the model's confidence in labeling samples as a particular activity.

**Recall**: Recall is the ratio of correctly predicted samples of a particular activity to all actual samples of that activity. In human activity prediction, recall measures the model's ability to recognize the actual activities. It informs us about whether the model can effectively capture a specific activity.

By utilizing these metrics, it can be observed from figure9 that the decision tree model exhibits the highest prediction accuracy, albeit with a lower recall rate. This signifies that the decision tree model has a tendency to overlook certain actual activities, resulting in the erroneous labeling of some real activities as non-activities.

*2) KNN results:* In the evaluation of KNN, the confusion matrix is further employed. Based on previous findings, it has been established that the previous model exhibits a lower recall rate, indicating confusion among certain specific activities. To further elucidate this result, we visualize the predicted outcomes for each label using the confusion matrix.

As depicted in the figure18, the majority of labels in the KNN prediction results are accurately predicted, indicating a favorable performance in terms of accuracy. However, there exists confusion between label 3 and

label 4, whereby numerous samples of label 3 are erroneously predicted as label 4, or vice versa. This further confirms the presence of similarities and confusion among certain label samples within the dataset.

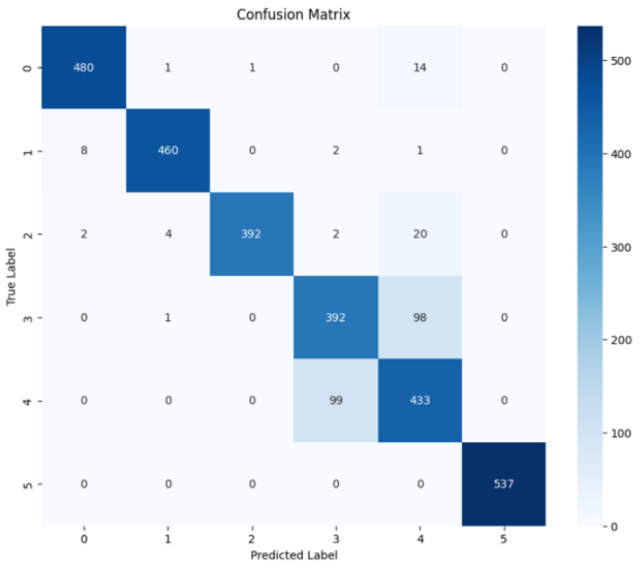



Fig. 18. Confusion matrix of KNN

*3) SVM results:* The experimental results of SVM are shown in tableVI. Among the three kernels, the highest accuracy(96.56%) is achieved when using the KBF kernel. For the same kernel(KBF), after feature selection, the accuracy increases from 95.32% to 96.56%, the running time significantly decreases from 18m56.4s to 2m30.7s.

TABLE VI
THE ACCURACY OF SVM

| Dataset | Algorithm | Accuracy | Running Time |
|---|---|---|---|
| original | SVM (RBF) | 95.32% | 18m 56.4s |
| feature selection | SVM (RBF) | 96.56% | 2m 30.7s |
| feature selection | SVM (linear) | 96.06% | 0.3s |
| feature selection | SVM (sigmoid) | 95.32% | 55.1s |

*4) CNN result:* The model evaluation was conducted with the configuration set to 100 epochs and a batch size of 64. Here are the observed performance metrics:

TABLE VII
MODEL PERFORMANCE ON DIFFERENT DATASETS.

| Data Set | Accuracy (%) | Loss |
|---|---|---|
| Training | 99.61 | 0.0155 |
| Validation | 98.35 | 0.0386 |
| Test | 96.49 | 0.0518 |

The results indicate a high level of accuracy in all datasets, with a slight decrease observed from training through to the test set. This trend may suggest mild overfitting to the training data.

Here are the key observations from the figure19:

- **Sharp Initial Decrease**: Both the training and validation loss experience a sharp decrease at the very beginning, suggesting that the model is learning rapidly from the initial state.
- **Convergence**: After the initial drop, both lines show a downward trend and tend to converge, indicating that the model is continuing to learn but at a slower rate as it begins to optimize and fit the data.
- **Stability**: As the epochs increase, particularly after around 20 epochs, the lines become relatively flat, showing minor fluctuations. This suggests that the model has reached a point where further training does not significantly reduce the loss, implying a stable convergence.
- **Close Proximity of Lines:** The training and validation loss lines are close to each other throughout the training process. This proximity indicates that the model is generalizing well and is not overfitting the training data. Overfitting would typically be indicated by a significant gap where the training loss continues to decrease while the validation loss starts to increase.
- **Final Values:** Towards the end of the training process (near 100 epochs), the loss values for both training and validation are low and close to each other, indicating a well-fitted model.

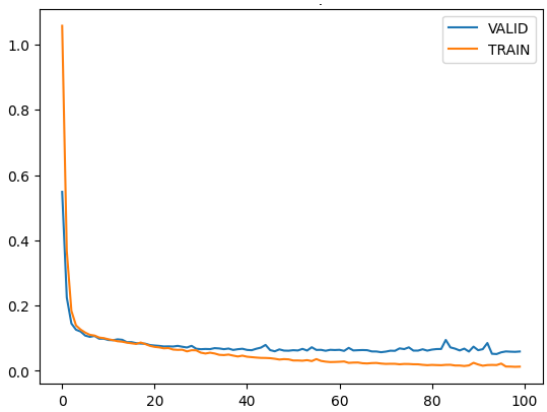



Fig. 19. Training and Validation Loss Trends across Epochs

*5) Clustering effect:* In this experiment, we attempted to apply the K-means and OPTICS algorithms to the dataset, but their performance was not very satisfactory.

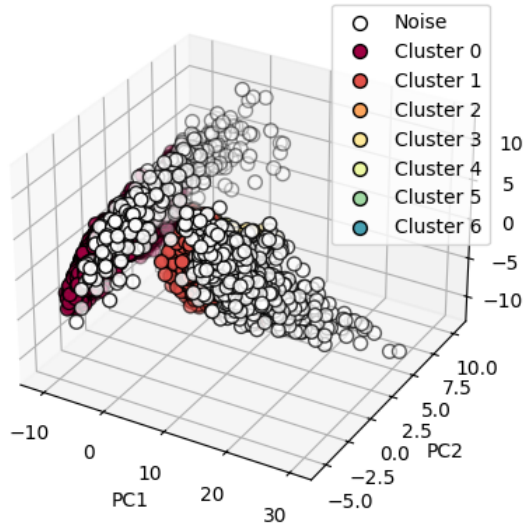



Fig. 20. OPTICS Clustering Result 3D

We evaluated the effectiveness of the K-means algorithm using the silhouette score, as illustrated in the figure. The results indicate that a silhouette score achieves favorable results when using $k = 2$. However, setting $k = 2$ implies that the dataset is divided into only two categories, whereas in reality, the dataset comprises six distinct human activities.

The performance of OPTICS is similarly disappointing. While OPTICS is capable of categorizing the dataset into six distinct groups, the clustering process generates a significant amount of noise(5002), as demonstrated in the figureIII. The reachability plot is shown in the figure21(response to Q4: Why doesn't OPTICS show a dendrogram? We did, however, because the clustering effect is not good, we did not show the relevant content too much in the presentation).

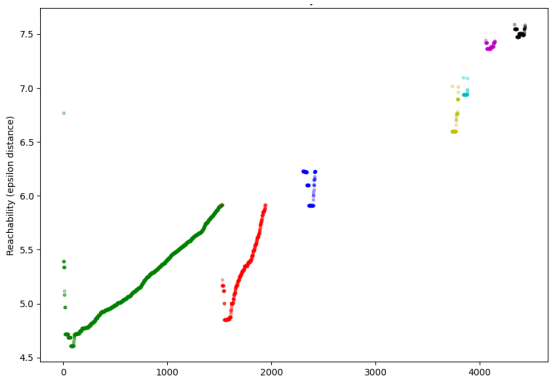



Fig. 21. OPTICS Clustering Result 3D

## VI. Future Work

To improve the clustering results for time-series data in our project, which did not meet our expectations, we are planning to adopt a more tailored approach in our methodology. Our future efforts will focus on integrating the distinctive characteristics inherent to time-series data more effectively.

One innovative strategy we will explore involves assigning different weights to data points depending on their position within the time series. This method acknowledges that not all data points contribute equally to the understanding of the sequence's dynamics. For instance, more recent observations might be given greater weight if they are more indicative of current trends, or specific periods known for volatility might be emphasized differently.

This approach is anticipated to improve the precision and relevance of our clustering results significantly. By modifying how we weigh different segments of the data, we aim to capture the true essence and underlying patterns more accurately. This adjustment will be central to our ongoing research and development efforts, aiming to refine and enhance the clustering techniques applied in our time-series analyses. Such advancements will not only bolster the robustness of our findings but also extend the applicability of our research to more complex and varied datasets.

## VII. Conclusion

In this project, we explored the application of various classification algorithms including KNN, Decision Tree, Random Forest, SVM, and CNN to classify the Human Activity Recognition (HAR) dataset, which utilizes wearable inertial sensors. This area of research has gained prominence due to significant advancements in sensor technology and the effective use of deep learning methods for evaluating time-series data from wearable and smartphone sensors.

Throughout the project, we emphasized the importance of feature selection to enhance the performance of our models. This strategic approach enabled us to meet our ambitious goal of achieving a classification accuracy higher than 95%. Notably, the CNN algorithm stood out by achieving an impressive accuracy of 96.49% in just 2 minutes and 12.2 seconds, demonstrating both efficiency and effectiveness.

Additionally, we experimented with clustering algorithms like K-means and OPTICS on the HAR dataset to uncover further insights and possibilities beyond traditional classification methods. This exploration into

clustering also aimed to complement our understanding of the dataset's structure and to identify novel patterns or groupings of activities.

In conclusion, the successful application of these advanced algorithms not only validated the robustness of the deep learning approach in handling complex human activity data but also opened new avenues for future research in the field. The integration of both classification and clustering algorithms provided a comprehensive approach to understanding and predicting human activity, paving the way for more personalized and context-aware applications in wearable technology.

## REFERENCES


[1] L. Schrader, A. Vargas Toro, S. Konietzny, S. Rüping, B. Schäpers, M. Steinböck, C. Krewer, F. Müller, J. Güttler, and T. Bock, "Advanced sensing and human activity recognition in early intervention and rehabilitation of elderly people," *Journal of Population Ageing*, vol. 13, pp. 139–165, 2020.

[2] R. Pramoditha, "The concept of artificial neurons (perceptrons) in neural networks," Dec. 2021. [Online]. Available: https://shorturl.at/yADL4

[3] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," in *International conference on pervasive computing*. Springer, 2004, pp. 1–17.

[4] D. Thakur and S. Biswas, "An integration of feature extraction and guided regularized random forest feature selection for smartphone based human activity recognition," *Journal of Network and Computer Applications*, vol. 204, p. 103417, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804522000741

[5] C. Ronao and S. Cho, "Human activity recognition with smartphone sensors using deep learning neural networks," *Expert Systems with Applications*, vol. 59, pp. 235–244, Oct. 2016, publisher Copyright: © 2016 Published by Elsevier Ltd.

[6] M. Alkhaleefah and C.-C. Wu, "A hybrid cnn and rbf-based svm approach for breast cancer classification in mammograms," in *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2018, pp. 894–899.

[7] M. S. Madhu and K. P. R, "Detection of liver disorder using quadratic support vector machine in comparison with rbf svm to measure the accuracy, precision, sensitivity and specificity," in *2022 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES)*, 2022, pp. 1–7.

[8] X. Ding, J. Liu, F. Yang, and J. Cao, "Random radial basis function kernel-based support vector machine," *Journal of the Franklin Institute*, vol. 358, no. 18, pp. 10 121–10 140, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0016003221006025

[9] M. Claesen, F. D. Smet, J. A. K. Suykens, and B. D. Moor, "Fast prediction with svm models containing rbf kernels," 2014.

[10] R. Venkatesan and B. Li, *Convolutional Neural Networks in Visual Computing: A Concise Guide*, 1st ed. CRC Press, 2017. [Online]. Available: https://doi.org/10.4324/9781315154282

[11] V. E. Balas, R. Kumar, and R. Srivastava, *Recent Trends and Advances in Artificial Intelligence and Internet of Things*. Springer Nature, 11 2019, archived from the original on 2023-10-16. Retrieved 2020-12-13.

[12] Y. Zhang, H. G. Soon, D. Ye, J. Y. H. Fuh, and K. Zhu, "Powder-bed fusion process monitoring by machine vision with hybrid convolutional neural networks," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, pp. 5769–5779, 9 2020, archived from the original on 2023-07-31. Retrieved 2023-08-12. S2CID 213010088.

[13] N. Chervyakov, P. Lyakhov, M. Deryabin, N. Nagornov, M. Valueva, and G. Valuev, "Residue number system-based solution for reducing the hardware cost of a convolutional neural network," *Neurocomputing*, vol. 407, pp. 439–453, 9 2020, archived from the original on 2023-06-29. Retrieved 2023-08-12. S2CID 219470398.

[14] H. H. Aghdam and E. J. Heravi, *Guide to Convolutional Neural Networks: A Practical Application to Traffic-Sign Detection and Classification*. Cham, Switzerland: Springer, 5 2017.

[15] Atlas, Homma, and Marks, "An artificial neural network for spatio-temporal bipolar patterns: Application to phoneme classification," in *Neural Information Processing Systems (NIPS 1987)*, 1987, archived from the original (PDF) on 2021-04-14.

[16] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: Ordering points to identify the clustering structure," in *ACM Sigmod Record*, vol. 28, no. 2. ACM, 1999, pp. 49–60.