textOsFtextTOsFliningLFliningTLFtextosfliningl ftabulartabproportionalprop

superiorSup

su-
pe-
ri-
or-
Sup

fontspechyperref

Comparative Analysis of Machine Learning Approaches for Human Activity Recognition

Machine Learning I - Final Project Report

Burak Cetin
University of Vigo
Spain
brakcetin660@gmail.com

Erik Figueiral Alonso
University of Vigo
Spain
erikfial1@gmail.com

Racim Allal
University of Vigo
Spain
racim.allal@col.udc.es

Roi Cores Cabaleiro
University of Vigo
Spain
roi.cores.cabaleiro@gmail.com

Abstract

This report presents a comparative study of four different machine learning pipelines for Human Activity Recognition (HAR) using the UCI Smartphone dataset. We investigate the effectiveness of dimensionality reduction techniques, including Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA), against traditional baseline models. Our experimental results demonstrate that the PCA-based Ensemble approach achieves a test accuracy of 93.32%, offering an optimal balance between computational efficiency and performance.

Keywords

Human Activity Recognition, PCA, Ensemble Learning, LDA, Machine Learning

1 Introduction

Artificial Intelligence (AI) is rapidly evolving and increasingly integrated into technologies that support people in their everyday lives. Among these advances, **Human Activity Recognition (HAR)** has emerged as a key area, enabling systems to understand and classify human movements automatically.

Modern HAR systems rely on two main sources of information: **computer vision** and **wearable sensors**. While vision-based approaches have achieved remarkable progress, wearable sensors like accelerometers and gyroscopes offer practical advantages. They are small, low-cost, energy-efficient, and fully portable, making them ideal for smartphones, or devices that people can carry everyday.

HAR is a **classification task**: the goal is to determine which activity a person is performing. These activities may include simple daily motions such as walking, sitting, standing, or running, as well as more specialized actions maybe relevant to medical monitoring, military applications, sports analysis, industrial safety and more.

To achieve accurate activity classification, HAR systems commonly employ **Machine Learning (ML)** techniques. These methods learn patterns directly from data and can distinguish subtle differences between activities. As HAR datasets grow in size and complexity, ML-based approaches have become essential tools for building reliable, user centered recognition systems. Nowadays approachs are **Deep Learning** but our main target hear is to made clasification in traditional ML approachs.

`figures/ActivityClasiffication.png`

Figure 1: General process of Human Activity Recognition (HAR).

1.1 Dataset Description

In this study we consider two widely used public datasets for human activity recognition: the **UCI HAR Dataset** and the **WISDM Dataset**. Both contain motion signals from wearable devices, but they differ in sampling frequency, sensor configuration, and acquisition protocol.

1.1.1 UCI HAR Dataset. The UCI Human Activity Recognition dataset[1] provides synchronized **accelerometer** and **gyroscope** signals recorded from a smartphone fixed at the waist. All measurements are collected in **raw form**, capturing tri-axial linear acceleration and angular velocity (accelometor and gyroscopy respectively).

A important point of this dataset is that both sensors are recorded at the **same sampling frequency of 50 Hz**. This uniformity simplifies preprocessing, since the accelerometer and gyroscope streams

[1]https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones

are inherently aligned and do not require temporal correction. The dataset also defines a **window length of 2.56 s** (equivalent to 128 samples).

1.1.2 WISDM Dataset. The WISDM dataset[2] contains accelerometer and gyroscope data collected from smartphones and smartwatches(omited in our study) during free-form daily activities. Unlike the HAR dataset, WISDM's sampling frequency is **20 Hz**, so it's diferrent than UCI HAR frecuency.

Because accelerometer and gyroscope signals are acquired in raw form, they must ideally operate at the same frequency to guarantee correct temporal pairing. A common strategy to mitigate misalignment is to **interpolate** one signal so that both sensors share a unified sampling rate. In principle, interpolation can increase comparability between datasets and allow processing them under the same configuration (e.g., resampling WISDM to 50 Hz).

However, interpolation introduces several considerations. First, it is **computationally expensive** when applied to long recordings or large datasets. Second, in window-based HAR pipelines—where statistical and frequency-domain features are extracted from fixed-length segments—the benefits of upsampling may be limited. Many window-level statistics (mean, variance, energy, entropy, etc.) do not significantly improve when artificially increasing the number of samples per window. For this reason, while interpolating WISDM from 20 Hz to 50 Hz is feasible and has been implemented, it may represent an **unnecessary computational cost** depending on the application.

1.1.3 Sensor Requirements and Practical Notes. Both datasets rely on two core inertial sensors:

- **Accelerometer**: measures linear acceleration along the three axes.
- **Gyroscope**: measures angular velocity along the same three axes.

For creating a dataset, they must share a **common sampling frequency**. When this condition is not met (as in WISDM), interpolation becomes a possible solution, although not always required.

The UCI HAR dataset already satisfies this requirement, while WISDM requires additional processing depending on the target analysis pipeline.

1.2 WISDM Signal Extraction and Preprocessing

1.2.1 Architecture: Adapter and Chain of Responsibility. To ensure a flexible and reusable preprocessing system to anhother raws fluents, we adopt a framework structure based on an **Adapter** and a **Chain of Responsibility** pattern.

- The **Adapter** reads the raw WISDM files and converts them into a unified internal representation.
- The **Chain of Responsibility** applies a sequence of processing steps (interpolation, filtering, masking, etc.), each step encapsulated as a separate component.

1.2.2 WISDM Signal Extraction. This is going to show how make a good dataset following this pipeline, this part will be used in a implementation

[font=, class/.style=rectangle, draw, rounded corners, minimum width=2.0cm, minimum height=0.55cm, align=center, abstract/.style=rectangle, draw, dashed, rounded corners, minimum width=2.0cm, minimum height=0.55cm, align=center, step/.style=rectangle, draw, minimum width=1.85cm, minimum height=0.50cm, align=center, inherit/.style=-Triangle[length=3mm,open], thin, arrow/.style=-Stealth[length=2mm], thin, node distance=0.60cm]
[abstract] (adapter) Adapter; [class, below=0.5cm of adapter] (adapterw) AdapterWISDM; [class, below=0.5cm of adapterw] (ext) ExternalDataset;
[inherit] (adapterw) – (adapter);
[arrow] (adapterw) – (ext);
[abstract, right=1.5cm of adapter] (proc) ProcessingStep;
[arrow] (proc.west) – ++(-0.6,0) |- (adapter);
[abstract, below=0.55cm of proc] (filters) Filters;
[step, below left=0.32cm and -0.15cm of filters] (interp) Interpolation; [step, below=0.27cm of filters] (smooth) Smoothing; [step, below right=0.32cm and -0.15cm of filters] (butter) Butterworth;
[inherit] (filters) – (proc);
[inherit] (interp) – (filters); [inherit] (smooth) – (filters); [inherit] (butter) – (filters);
[arrow] (proc.south east) .. controls +(0.8cm,-0.3cm) and +(0.8cm,0.3cm) .. (proc.north east);
[abstract, right=1.7cm of proc] (enrich) Enrichment;
[step, below left=0.30cm and -0.10cm of enrich] (grav) Gravity; [step, below=0.27cm of enrich] (tilt) Tilt; [step, below right=0.30cm and -0.10cm of enrich] (ratio) Ratios;
[inherit] (enrich) – (proc);
[inherit] (grav) – (enrich); [inherit] (tilt) – (enrich); [inherit] (ratio) – (enrich);
[abstract, below=0.80cm of enrich] (masking) Masking; [step, below=0.27cm of masking] (mask) ProximityMask;
[inherit] (masking) – (proc);
[inherit] (mask) – (masking);
[step, below=1.10cm of filters, xshift=3.6cm] (windowing) Windowing;
[inherit] (windowing) – (proc);

Figure 2: Compact UML architecture using inheritance relationships for all processing components.

The WISDM dataset provides raw accelerometer and gyroscope measurements recorded at 20 Hz for a smarthpone and smartwatches, for compatibility with HAR this last was ommited. However, the sensor streams are not always defined across the entire temporal dimension T; gaps, missing bursts, or slight accelerometer–gyroscope misalignments are common.

To handle these issues, our pipeline first aligns both signals on a shared temporal using **interpolation**. This step allows accelerometer and gyroscope data to be treated as synchronized multivariate time series, enabling consistent feature extraction.

After interpolation, two filters are applied sequentially:

1. A **smoothing filter** (median or moving average), reducing impulsive noise and small fluctuations.
2. A **Butterworth low-pass filter**, eliminating high-frequency components unlikely to contribute to human activity recognition.

These steps produce smoother, more stable motion curves while preserving essential activity-related patterns so will be more realistic predicted values.

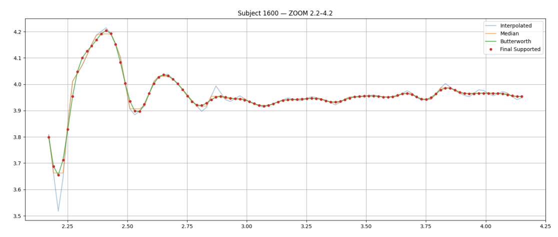

Figure 3: Example of filters. We can see the filter one,the smooting and butterworth functions

[2]https://www.cis.fordham.edu/wisdm/dataset.php

1.2.3 *Gravity–Body Separation and Derived Motion Features.*

After interpolation and initial filtering, the accelerometer signal is decomposed into two physical components: a **gravity component** and a **body-motion component**. This separation is fundamental for correctly interpreting human posture and movement.

The decomposition is performed using a frequency-based approach:

- Frequencies **below 0.3 Hz** correspond to the **gravity vector**, which reflects the orientation of the device in space.
- Frequencies **above 0.3 Hz** represent the **body acceleration**, generated by voluntary movement.

This separation is essential because the gyroscope alone only measures **angular velocity** (rotation), but not device orientation. The gravity component, extracted from the accelerometer, enables the system to infer whether the subject is standing, sitting, lying down, or tilting in a particular direction—activities where orientation is the key discriminative factor.

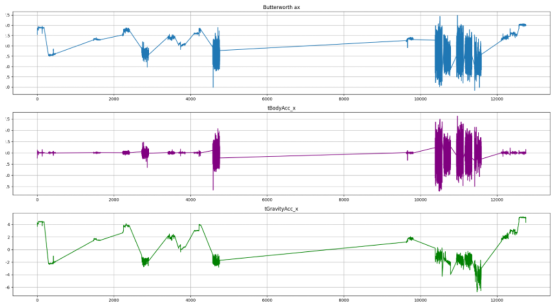

Figure 4: Comparison between gravity and body acceleration components for the x-axis. Gravity captures orientation, while body acceleration reflects dynamic motion.

1.2.4 *Additional Derived Measures: Jerk, Magnitudes, and Orientation.*

In addition to gravity and body acceleration, several derived features are computed to enrich the representation of the dataset:

- **Jerk**: the time derivative of acceleration or angular velocity. It captures abrupt changes in motion, useful for detecting transitions and impacts.
$$\text{Jerk}(t) = \frac{d\,\text{acc}(t)}{dt}$$
Higher jerk values often correspond to dynamic activities such as running, jumping, or rapid turns.
- **Magnitude signals**:
$$\text{Mag}(t) = \sqrt{x^2 + y^2 + z^2}$$
Magnitudes summarize 3D movement intensity and are invariant to device rotation.
- **Tilt/orientation angles**: Will be usefull in window part to the learnings process.

1.2.5 *Additional Derived Measures and High-level Motion Features.*

Beyond gravity–body separation, several higher-level motion features are computed to enhance the representation of posture, orientation, and dynamic transitions. **These features are not part of the original WISDM or HAR datasets**; instead, **they are additional components introduced in our preprocessing pipeline** to enrich the sensor representation and improve classification performance. To get a more acurrate dataset is necesarry a research process in sensor analysis.

1. TiltFeatureExtractor (orientation from gravity). Device orientation is estimated using the gravity vector. For each timestamp, the pitch and roll angles are obtained as:

$$\text{pitch} = \arctan 2(-g_x,\ g_y^2 + g_z^2), \qquad \text{roll} = \arctan 2(g_y,\ g_z).$$

These angles describe how the device is tilted along two axes and are useful for distinguishing posture-related activities such as standing, sitting, or lying, where gravity orientation is the key discriminant.

2. GravityAlignedComponentsExtractor (vertical vs. horizontal motion). Body acceleration is projected onto the gravity direction to separate vertical and horizontal components. First, the gravity vector is normalized:

$$\hat{g} = \frac{(g_x, g_y, g_z)}{\|g\|}.$$

The vertical component of body acceleration is:

$$a_{\text{vert}} = b_x\hat{g}_x + b_y\hat{g}_y + b_z\hat{g}_z,$$

while the horizontal component is:

$$a_{\text{horiz}} = \sqrt{\|b\|^2 - a_{\text{vert}}^2}.$$

This decomposition provides information on movement aligned with gravity (e.g., sit-to-stand transitions) versus movement in the horizontal plane (e.g., walking).

3. RatioDynamicsExtractor (relative motion dynamics). Two useful ratios capture the relation between jerk (derivative of acceleration or angular velocity) and the underlying motion intensity:

$$\text{ratio}_{\text{jerk/body}} = \frac{\text{tBodyAccJerkMag}}{\text{tBodyAccMag} + \varepsilon},$$
$$\text{ratio}_{\text{gyroJerk/gyro}} = \frac{\text{tBodyGyroJerkMag}}{\text{tBodyGyroMag} + \varepsilon}.$$

These ratios highlight quick motion bursts relative to steady motion and are helpful for discriminating fast transitions from sustained activities. .

1.2.6 *Real-support Masking and Proximity Constraints.*

Interpolation estimates values between real measurements, but it does not create new information. If applied blindly, long regions without sensor readings would appear artificially continuous.

To prevent this, we apply a **proximity mask** (real-support mask). For every interpolated timestamp, we check whether an actual accelerometer and gyroscope measurement exists within a small time window Δt. If no real sample is close enough, the interpolated point is **ignored** by all subsequent modules (windowing, feature extraction, classification).

1.2.7 *Windowing and Feature Extraction.*

After preprocessing the sensor streams and enriching them with derived features, the next stage consists of segmenting the time series into fixed-length windows and computing a set of statistical and frequency-domain descriptors. This step transforms raw temporal data into structured samples suitable for traditional machine learning models.

Sliding Window Generation

Following the conventions established in the UCI HAR benchmark, each signal is segmented using windows of **2.56 s** duration. At a reference sampling frequency of **50 Hz**, this corresponds to:

$$128 \text{ samples per window.}$$

Windows are generated with a **50% overlap**:

$$\text{stride} = 1.28\,\text{s} \quad (64 \text{ samples}),$$

which increases temporal continuity and ensures that relevant motion transitions are not missed, and could be usefull in DL approachs.

Statistical Time-Domain Features

For each accepted window, classical time-domain statistics are computed over all signals (accelerometer, gyroscope, body components, gravity components, jerk, magnitudes, etc.). These include:

- mean, median, variance, standard deviation,
- minimum, maximum, range,
- interquartile range,
- signal magnitude area (SMA),
- energy,
- zero-crossing rate (for oscillatory signals),
- autoregressive coefficients (optional).

These descriptors summarize the global behaviour and variability of motion within each window.

Frequency-Domain Features via FFT

In addition to time-domain statistics, frequency-domain information is extracted using a **Fast Fourier Transform (FFT)** applied independently to each axis or derived feature.

The magnitude spectrum is obtained and normalized. From it, we compute:

- dominant frequency components,
- energy in predefined frequency bands,
- spectral entropy,
- centroid and bandwidth,
- frequency-domain means and medians.

This spectral analysis provides insight into periodicity and vibration patterns, which are crucial for distinguishing activities such as walking, running, and transitions between static and dynamic states.

This representation is particularly advantageous for classical machine learning models, which lack temporal memory and therefore require structured descriptors to compensate for the absence of contextual awareness.

Recurrent neural networks such as LSTM or BiLSTM models, however, possess internal memory mechanisms that allow them to retain information from previous observations. For these architectures, providing the raw or minimally processed sensor streams is not only sufficient but often preferable, since temporal continuity enables the model to infer whether the subject had been sitting, standing, turning, or transitioning long before the current window begins. In this sense, strict segmentation into independent windows inevitably discards part of the temporal context that deep recurrent models are capable of exploiting.

Nevertheless, some engineered features—particularly frequency-domain components and spectral bands—remain valuable even for deep models, as they highlight periodic and harmonic structures that may not be trivially learned from raw data alone. Thus, while feature engineering is essential for traditional ML pipelines, its role in deep learning presents a nuanced trade-off between preserving raw temporal information and emphasizing discriminative patterns.

2 Methodology

In this study, each team member explored a different pipeline to solve the classification problem.

2.1 Approach 1: Full-Feature Baseline Classification

In this approach, we evaluated classical machine learning models using the full set of 561 engineered features from the UCI HAR dataset. Unlike PCA or LDA, this method applies no dimensionality reduction; instead, it investigates whether the original high-dimensional representation already provides strong discriminative power for activity recognition.

2.1.1 Data Preprocessing. Because the HAR dataset contains features with heterogeneous numerical ranges, we standardized all inputs using Z-score normalization:

$$z = \frac{x - \mu}{\sigma} \tag{1}$$

This step ensures that models sensitive to scale—such as SVM, ANN, and kNN—operate on features with consistent magnitudes. No feature removal or compression was applied, making this a true full-feature baseline.

2.1.2 Model Optimization. We trained four supervised learning models using 5-Fold Stratified Cross-Validation (CV). The following configurations achieved the best performance for each model:

- **Artificial Neural Network (ANN):** A multilayer perceptron with architecture [128, 32] achieved a CV accuracy of 93.04%.
- **Support Vector Machine (SVM):** The RBF kernel with $C = 1.0$ reached a CV accuracy of 92.73%.
- **k-Nearest Neighbors (kNN):** The best performance was observed at $k = 3$ (CV: 91.65%).
- **Decision Tree (DT):** A depth-10 tree achieved 88.47% CV accuracy.

2.1.3 Ensemble Strategy. To improve robustness and reduce single-model biases, we created a **Voting Ensemble** combining ANN, SVM, and kNN. Majority voting improved generalization, particularly for ambiguous activities such as *Sitting* vs. *Standing*.

2.1.4 Performance Analysis. The models were evaluated on the strictly separated Test Set. Figure 1 shows the comparative accuracy between the five models. Quantitative results are summarized in Table 1.

Figure 2 presents the confusion matrix of the Ensemble model. Consistent with related work, the primary confusions occur between *Sitting* and *Standing*, the two classes with the most similar motion patterns.

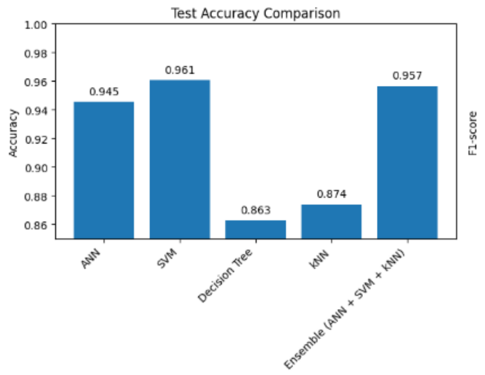

Figure 5: Test accuracy comparison of ANN, SVM, kNN, Decision Tree, and Ensemble models. The Ensemble model achieved the highest accuracy of 95.7%.

Table 1: Performance Comparison (Full-Feature Baseline Approach)

Model	CV Accuracy	Test Accuracy	F1-Score
Ensemble	-	**95.7%**	**0.957**
ANN	93.04%	94.5%	0.945
SVM (RBF)	92.73%	96.1%	0.960
kNN ($k = 3$)	91.65%	87.4%	0.873
Decision Tree	88.47%	86.3%	0.863

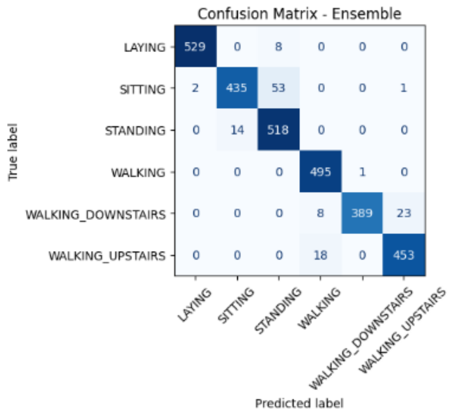

Figure 6: Confusion Matrix of the Full-Feature Ensemble Model. Slight confusion occurs mainly between Sitting and Standing.

2.1.5 Limitations and Future Work. Although the full-feature baseline achieved high performance, its computational cost is significantly higher than PCA or LDA approaches. Future improvements may include:

(1) **Feature Selection:** Using mutual information or recursive elimination to reduce redundancy among the 561 features.
(2) **Hybrid Dimensionality Reduction:** Combining PCA with a subset of raw features to maintain interpretability.
(3) **Weighted Ensembles:** Giving more weight to ANN and SVM predictions to enhance static activity classification.

2.2 Approach 2: PCA & Ensemble Learning

In this approach, we investigated the trade-off between dimensionality reduction and classification accuracy. The primary objective was to design a computationally efficient pipeline suitable for embedded devices by compressing the feature space while maintaining high predictive performance.

2.2.1 Data Preprocessing and PCA. Principal Component Analysis (PCA) is inherently sensitive to the scale of input features. Since the UCI HAR dataset contains features with varying ranges, we first applied Z-Score normalization to standardize all 561 features to zero mean and unit variance:

$$z = \frac{x - \mu}{\sigma} \tag{2}$$

Following normalization, PCA was applied with a cumulative variance threshold of **95%**. This process reduced the original 561 dimensions to **102 principal components**, achieving an **81.8% compression rate**. This significant reduction drastically lowers the computational cost for subsequent training steps.

2.2.2 Model Optimization. We trained four distinct classifiers on the reduced feature space using 10-Fold Stratified Cross-Validation (CV) to ensure robustness. The hyperparameter tuning yielded the following optimal configurations:

- **Artificial Neural Network (ANN):** A multi-layer perceptron with two hidden layers ([200, 100] neurons) achieved the highest individual CV accuracy (97.31%).
- **Support Vector Machine (SVM):** Optimized with a *Linear Kernel* and $C = 1.0$, demonstrating that the PCA-transformed space is linearly separable (CV: 97.23%).
- **k-Nearest Neighbors (kNN):** The best performance was observed at $k = 1$ (CV: 96.54%), though this suggests potential susceptibility to noise (overfitting).
- **Decision Tree (DT):** Achieved only 84.47% CV accuracy. The poor performance is attributed to PCA's feature rotation, which disrupts the axis-aligned splits required by decision trees.

2.2.3 Ensemble Strategy. To mitigate individual model biases—specifically the overfitting of kNN and the decision boundaries of SVM—we constructed a **Voting Ensemble** comprising the top three models (ANN, SVM, and kNN). The ensemble aggregates predictions using majority voting to improve generalization on unseen subjects.

2.2.4 Performance Analysis. The models were evaluated on the strictly separated Test Set. To visualize the comparative performance, we computed accuracy metrics across all models, as illustrated in Figure 3. The quantitative results are detailed in Table 2, which compares the cross-validation (CV) and testing phases.

While the individual kNN model suffered a significant drop in testing (indicating overfitting due to $k = 1$), the **Ensemble model** successfully compensated for this, achieving the highest overall accuracy of **93.32%**.

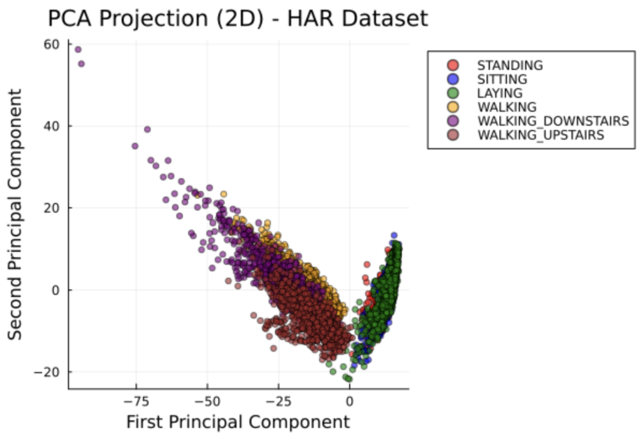

Figure 7: Test Accuracy Comparison. The Ensemble model outperforms individual classifiers.

Table 2: Performance Comparison (PCA Approach)

Model	CV Accuracy	Test Accuracy	F1-Score
Ensemble	-	**93.32%**	**0.9329**
ANN	97.31%	93.04%	0.9302
SVM	97.23%	92.20%	0.9216
kNN ($k = 1$)	96.54%	84.39%	0.8437
Decision Tree	84.47%	78.38%	0.7836

To analyze the specific error patterns, we examined the Confusion Matrix shown in Figure 4. The matrix reveals that the model is highly robust, yet slight confusions exist between static activities. To further validate this, Table 3 presents the detailed precision and recall values for each class.

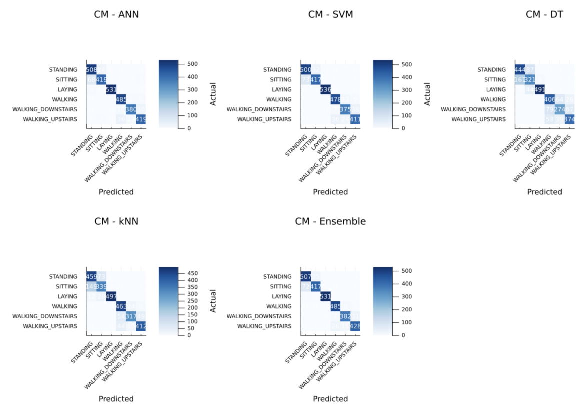

Figure 8: Confusion Matrix of the Ensemble Model. Note the slight confusion between Sitting and Standing.

2.2.5 Limitations and Future Work. As confirmed by the confusion matrix, misclassifications primarily occurred between *Sitting* and *Standing*. Since PCA transforms the feature space, the explicit vertical orientation signal (gravity-z) may have been diluted. Future improvements include:

(1) **Explicit Gravity Injection:** Concatenating the raw 'tGravityAcc-mean()-Z' feature with PCA components to restore orientation data.
(2) **k-Value Optimization:** Increasing k (e.g., to 7) in kNN to improve generalization on unseen subjects.

Table 3: Per-Class Performance of the Best Model (Ensemble)

Activity	Precision	Recall	F1-Score	Support
Standing	0.8696	0.9530	0.9094	532
Sitting	0.9350	0.8493	0.8901	491
Laying	0.9981	0.9888	0.9935	537
Walking	0.9363	0.9778	0.9566	496
Walk Down	0.9409	0.9095	0.9249	420
Walk Up	0.9264	0.9087	0.9175	471
Weighted Avg			**93.32%**	**2947**

2.3 Approach 3: [Erik's Method]

2.3.1 Dataset Quality Challenges and Initial Attempts. One of the major limitations repeatedly highlighted in recent HAR literature is the lack of a truly high–quality, general-purpose benchmark dataset. As noted in prior research, most publicly available datasets suffer from limited or imbalanced data, small sample sizes, and activity sets that do not capture the diversity and complexity of real human behaviour [?]. Many existing benchmarks contain simplistic motions and short, well-structured sequences, which do not reflect realistic daily variability. Furthermore, deep learning architectures that rely on contextual information often struggle to integrate such limited or biased data [?]. Although these models may achieve state-of-the-art accuracy on benchmark datasets, they frequently exhibit *overconfidence* in their predictions, indicating a lack of generalisation.

In the early stages of this work, we explored whether the quality and representativeness of HAR datasets could be improved either by *simplifying* the feature space while preserving its semantic meaning or by *enriching* the existing raw signals. One natural idea was to address class imbalance directly. Given the distribution observed in our datasets, techniques such as SMOTE appeared promising—particularly for activities such as *Walking*, where additional synthetic samples could better populate the high-dimensional sensor space. Figure **??** illustrates the imbalance structure and the potential benefit of interpolation in unexplored regions of the feature manifold.

Ultimately, however, SMOTE was discarded due to time constraints and because a more promising direction emerged: **merging multiple HAR datasets**. We experimented with combining the raw signals from the UCI HAR and WISDM datasets, starting from their raw accelerometer and gyroscope measurements. A methodological mistake made in the first attempt was to analyse their joint distribution *after* preprocessing rather than at the raw signal level—an important lesson for future cross-dataset integration.

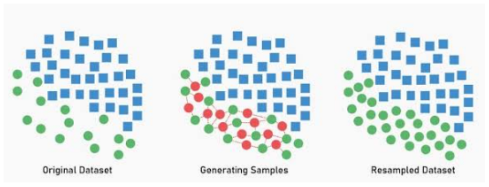

Figure 9: Example of class imbalance and the potential effect of SMOTE on the *Walking* class.

Principal Component Analysis (PCA) suggested that the joint space was surprisingly coherent: the two datasets overlapped reasonably well, as shown in Fig. ??. However, a nonlinear projection using t-SNE revealed the opposite behaviour: the distributions separated sharply, indicating clear differences in movement patterns, device placement, sensor noise, or subject behaviour.

figures/pcaNoSeparable.png

Figure 10: PCA projection of merged HAR and WISDM raw datasets shows partial overlap.

To mitigate this gap, we evaluated **per-subject normalisation**. The intuition is that people move differently, and devices differ across subjects. By aligning distributions so that each subject shares a similar covariance structure, we observed noticeably more optimistic results, as illustrated in Fig. ??. These results suggest that domain alignment at the subject level can significantly reduce dataset shift, even without complex domain-adaptation methods.

Although more advanced techniques such as Domain-Adversarial Neural Networks (DANN) are common in academic work, they were excluded from this study due to the requirement for real-time processing and interpretability. Our experiments reinforce what we believe remains one of the central research gaps in HAR today: **the absence of a universal, cross-subject, cross-device, cross-environment dataset**. Real-world HAR systems must still be tailored to the specific sensor, device, and population being monitored. This limitation is a primary barrier to producing robust, deployable, and generalisable HAR models.

figures/tsneNoSeparable.png

Figure 11: t-SNE projection of merged HAR and WISDM raw datasets shows partial overlap.

figures/tsneSeparable87.png

Figure 12: t-SNE projection after per-subject normalisation shows improved cross-dataset alignment.

2.3.2 AutoEncoder–LDA Representation Learning Approach. To address the limitations imposed by heterogeneous raw sensor distributions, we explored a representation-learning strategy based on an AutoEncoder followed by Linear Discriminant Analysis (LDA). This approach was motivated by recent results showing that encoder–decoder architectures can preserve semantic structure while providing compact embeddings suitable for classification. In particular, Zhang et al. [?] demonstrated the effectiveness of U-Net encoder/decoder mechanisms for extracting meaningful latent representations while maintaining spatial (or temporal) consistency. Inspired by this, we adopted an AutoEncoder architecture to compress the windowed HAR signals into a 64-dimensional embedding.

The rationale for using an AutoEncoder was threefold:

- **Semantic preservation:** unlike purely linear dimensionality reduction, the encoder learns a nonlinear manifold that reflects the structure of human motion—important for HAR tasks.

- **Efficient compression:** reducing each window to 64 features lowers computational cost and enables lightweight classifiers suitable for real-time inference.
- **Future extensibility:** an explicit latent space opens the possibility of using Variational Component Analysis (VCA) or generative models to synthesize missing data or rebalance underrepresented classes. Although this direction was considered, it was discarded due to time constraints.

After training the AutoEncoder, each window is represented by a 64-dimensional embedding vector:

$$\mathbf{e} = (emb_1, emb_2, \ldots, emb_{64}).$$

These embeddings serve as the input to the subsequent classification stage.

Before applying LDA, we analysed the statistical shape of each latent dimension produced by the AutoEncoder. Although the encoder provides a compact representation, the resulting embeddings do not necessarily follow a Gaussian-like distribution. In practice, many dimensions exhibited high skewness and heavy-tailed behaviour, which may hinder linear methods such as LDA.

To stabilise the distribution of the embeddings, we adopted an **adaptive normalisation strategy**. The idea is to select the most appropriate normalisation method *per feature* based on its empirical distribution:

- If the embedding dimension showed **low skewness** and **kurtosis close to 3** (i.e., approximately Gaussian), we applied **Z-score normalisation**. This transformation centres the feature and rescales it to unit variance, which is ideal for dimensions that already follow a symmetric, bell-shaped distribution.
- If the dimension exhibited **high skewness** or **heavy tails**, we applied **Min–Max scaling**. This transformation maps the feature into the range $[0, 1]$, preserving the relative ordering while preventing extreme values from disproportionately influencing the classifier.

The criteria allow the system to normalise each embedding coordinate according to its distributional nature:

$$\text{Approximately Gaussian} \;\Rightarrow\; \text{Z-score}, \qquad \text{Non-Gaussian or heavy-tailed} \;\Rightarrow\; \text{M}$$

This adaptive strategy reduces distortions caused by asymmetric or heavy-tailed features, makes the latent space more homogeneous, and improves the effectiveness of the subsequent LDA projection. In practice, this resulted in a more stable embedding geometry and clearer class separation after the discriminant projection.

2.3.3 LDA on Learned Embeddings. To improve class separability, we applied Linear Discriminant Analysis (LDA) on the learned embeddings. LDA seeks a linear combination of the embedding dimensions that maximises the ratio between inter-class and intra-class variance. Formally, it finds a projection matrix $\mathbf{W}$ such that:

$$\mathbf{W} = \arg\max_{\mathbf{W}} \frac{\mathbf{W}^\top S_B \mathbf{W}}{\mathbf{W}^\top S_W \mathbf{W}},$$

where S_B and S_W denote the between-class and within-class scatter matrices, respectively.

Applying LDA after the AutoEncoder has two advantages:

(1) It enhances discriminative structure already present in the latent representations.
(2) It reduces the dimensionality further to $C - 1$ components, where C is the number of activities.

This two-stage pipeline (nonlinear compression followed by linear discriminant projection) has proved effective in HAR classification, particularly when datasets differ in distribution.

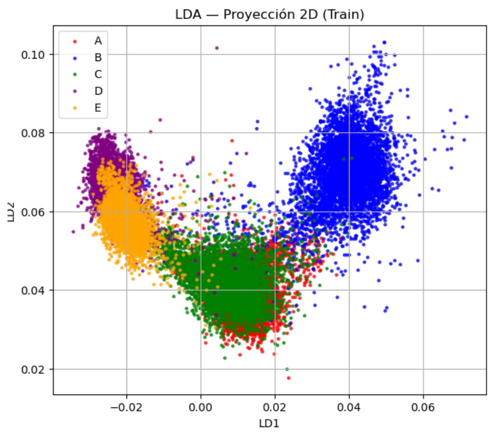

Figure 13: Proyection in LD1 and LD2 vectors

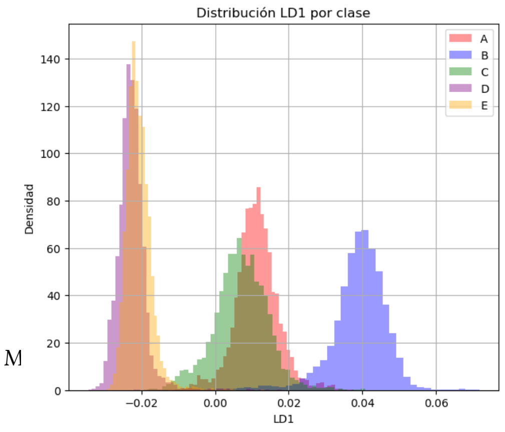

Figure 14: Density for each class in LD1 vector

2.4 Approach 4: [Roi's Method]

3 Subject-Disjoint Approach Using Supervised Feature Selection

The goal of this approach is to design a model capable of stronger subject-level generalisation. For this purpose, the *Subject* identifier included in the dataset is explicitly used to separate instances belonging to different individuals. This separation plays a central role in all stages of the pipeline: dataset splitting, cross-validation design, and evaluation on unseen subjects.

3.1 Dataset Description

We use the UCI HAR dataset, which contains time-windowed inertial signals recorded from a smartphone worn on the waist. Each instance is represented by 561 aggregated statistical features and labelled with one of six activities: *WALKING*, *WALKING_UPSTAIRS*, *WALKING_DOWNSTAIRS*, *SITTING*, *STANDING*, *LAYING*.

Before splitting the dataset, we verify that samples are not unevenly distributed across subjects. Figure **??** shows the number of samples per person in the original dataset.

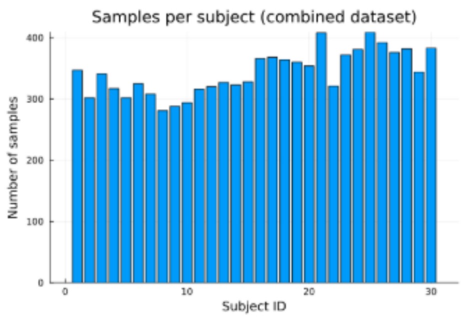

Figure 15: Number of samples per subject in the UCI HAR dataset.

The original train/test split included instances from nearly all subjects in both sets, meaning the model would be tested on data from people also present in the training set. To enforce true subject-disjoint evaluation, we merge the original splits and re-split the dataset as follows:

- **Test1** (≈30%): around 10 subjects, *all* their instances appear only here.
- **Train** (≈65%): the remaining ≈20 subjects (completely disjoint from Test1).
- **Test2** (≈5%): a small subsample of Train, used to evaluate generalisation to new windows from already-seen individuals.

After splitting, the dataset distribution is approximately: Train ≈ 6300 samples, Test1 ≈ 3400, Test2 ≈ 500.

Figure **??** shows class frequencies across the three splits.

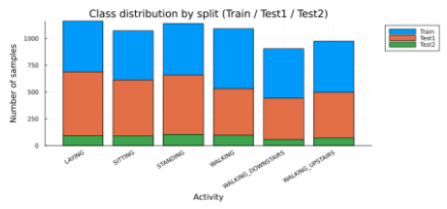

Figure 16: Class distribution in Train, Test1 (subject-disjoint), and Test2 (mixed).

3.2 Data Preprocessing

All models share the same preprocessing pipeline. We apply **Z-score normalisation per feature** across the entire dataset to ensure comparability between algorithms.

We do not apply min–max scaling, outlier filtering, or temporal operations, since the HAR features are already engineered and Z-score stabilises optimisation for gradient-based models (ANNs, SVMs). Although trees and kNN do not strictly require normalisation, using a unified pipeline ensures a fair comparison.

3.3 Experimental Setup

3.3.1 Supervised Feature Selection. We perform feature selection on the training set using one-way ANOVA F-scores. For each feature, the score measures the ratio between inter-class and intra-class variance. Sorting features by decreasing F-score produces a ranking such as the one displayed in Fig. **??**.

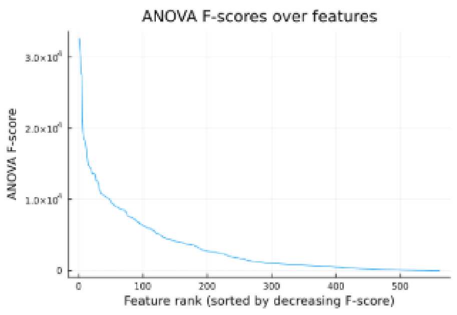

Figure 17: Ranking of features by ANOVA F-score.

We evaluate several values of N (top-N features) using a fixed ANN architecture with subject-disjoint cross-validation. Results show that using all 561 features yields the highest CV accuracy, but subsets of 200–300 features significantly reduce training times with minimal accuracy loss. The selected feature mask is then applied consistently to Train, Test1, and Test2.

3.3.2 Subject-Disjoint Cross-Validation. To maintain subject independence during model selection, folds are generated at the subject level. A custom method shuffles the subjects in the training split and assigns them in round-robin fashion to one of the k folds.

Although the training split contains approximately 20 subjects, performing 20-fold CV was too expensive computationally, so we adopt a 10-fold protocol (folds of two subjects).

This ensures each fold evaluates on completely unseen individuals, mirroring the Test1 scenario.

3.4 Model Experimentation

All four ML methods studied in the course are evaluated under the same subject-wise CV procedure and feature-selection mask.

3.4.1 Artificial Neural Networks (ANN). We test eight fully connected architectures with one or two hidden layers:

$$[64],\ [128],\ [256],\ [64,32],\ [128,64],\ [256,128],\ [128,128],\ [256,256].$$

Training settings: Adam optimiser, softmax cross-entropy loss, learning rate 0.005, maxEpochs=350, early stopping with patience=10, validation ratio 0.2. For each architecture we compute mean±std accuracy and F1-score across the 10 folds and select the best-performing topology.

3.4.2 Support Vector Machines (SVM). We evaluate eight configurations:

- **Linear kernel:** $C \in \{0.1, 1, 10\}$ (3 configs)
- **RBF kernel:** $(C, \gamma) \in \{(1, 0.01), (10, 0.01), (10, 0.001)\}$ (3 configs)
- **Polynomial kernel:** $C = 1$, degree $\in \{2, 3\}$ (2 configs)

Each model is trained on subject-disjoint folds and evaluated via CV accuracy.

3.4.3 Decision Trees (DT). We vary the maximum depth $\in \{3, 5, 7, 9, 11, 13\}$. Shallow trees underfit, while overly deep trees overfit; intermediate depths (7–9) provide the best trade-off.

3.4.4 k-Nearest Neighbours (kNN). We test $k \in \{1, 3, 5, 7, 9, 11\}$. Z-score normalisation ensures Euclidean distance behaves consistently. Moderate values such as $k = 9$ or $k = 11$ yield the best accuracy.

3.5 Model Evaluation and Ensemble Method

After selecting the best configuration for ANN, SVM, DT, and kNN, we retrain each model on the full training set (after feature selection) and evaluate them on Test1 (subject-disjoint) and Test2 (mixed).

Figures **??** and **??** illustrate confusion matrices for the best-performing models on each evaluation split.

Table 4: Model performance summary on Test1 (subject-disjoint).

Model	Accuracy	Weighted F1
ANN	0.9444	0.9444
SVM	0.9459	0.9458
kNN	0.8844	0.8843
DT	0.8780	0.8775
Ensemble	**0.9470**	**0.9469**

Table 5: Model performance summary on Test2 (mixed subjects).

Model	Accuracy	Weighted F1
ANN	0.9767	0.9767
SVM	0.9825	0.9825
kNN	0.9592	0.9591
DT	0.9437	0.9434
Ensemble	**0.9825**	**0.9825**

Finally, we construct a simple **hard-voting ensemble** combining the best ANN, best SVM, and best kNN. For each sample, the predicted class is chosen by unweighted majority vote. Decision trees are excluded from the ensemble due to lower performance in this setting.

In addition to these observations, it is worth highlighting that the subject-disjoint evaluation provides a more faithful approximation of real deployment scenarios, where models must operate on new users whose motion patterns, phone placement habits, and biomechanical signatures differ from those seen during training. The clear performance gap between Test1 and Test2 quantifies this challenge directly: even strong models such as SVMs and ANNs exhibit a non-negligible degradation when confronted with unseen subjects. This reinforces the need for HAR systems that are robust to user variability, either through improved representation learning, personalised calibration strategies, or domain-adaptation techniques.

Moreover, the experimental results show that no single model fully dominates all others across evaluation conditions. The fact that a simple majority-vote ensemble systematically reduces a portion of the remaining classification errors indicates that different model families capture complementary aspects of the signal space. This suggests that hybrid or hierarchical HAR architectures may benefit from combining discriminative boundaries learned by SVMs with the latent representations extracted by ANNs, while exploiting the local decision stability inherent to kNN. Overall, these findings underline that subject-general HAR remains an open problem, and that carefully designed pipelines—integrating feature selection, subject-wise evaluation, and model ensembling—are essential for developing reliable and deployable systems.

4 Experimental Results

5 Conclusion