Human Activity Recognition (HAR) using Machine Learning and Deep Learning

El Mehdi G'RIGUAA
Machine Learning Course

Supervised by:
Professor Fahd Kalloubi

February 14, 2026

Abstract

This project explores the development of Human Activity Recognition (HAR) systems using wearable Inertial Measurement Unit (IMU) sensor data. We implemented a complete end-to-end machine learning pipeline, starting from raw sensor data exploration and preprocessing to advanced feature engineering and dimensionality reduction. We evaluated both unsupervised clustering techniques (K-Means, DBSCAN) and supervised learning algorithms (Logistic Regression, Random Forest, SVM, Gradient Boosting). Furthermore, we explored Deep Learning approaches using Multi-Layer Perceptrons (MLP) and 1D Convolutional Neural Networks (CNNs). Our experiments, tracked via MLflow, demonstrate that supervised ensemble methods like Gradient Boosting and Random Forest achieve superior performance with an accuracy of over 97%, comparable to deep learning models, making them highly effective for activity classification tasks.

Contents

1 Introduction

Human Activity Recognition (HAR) is a crucial field in machine learning with applications in healthcare, elderly monitoring, sports analytics, and human-computer interaction. The objective is to classify human movements (e.g., walking, sitting, standing) based on time-series data collected from sensors like accelerometers and gyroscopes.

In this project, we utilize a dataset collected from 67 subjects wearing IMU sensors on their chest, hands, and knees. The dataset includes six core activities:

- Standing
- Walking
- Sitting
- Lying Down
- Going Upstairs
- Going Downstairs

Our goal is to build robust classification models that can accurately identify these activities from unseen sensor data.

2 Methodology

2.1 Data Exploration and Preprocessing

We began by analyzing the dataset structure, which consists of synchronized recordings of quaternions, acceleration, and angular velocity.

- **Data Cleaning:** We handled missing values and infinite numbers by replacing them with zeros or interpolating where appropriate.
- **Exploratory Data Analysis (EDA):** We visualized class distributions to check for imbalance and plotted raw sensor signals to understand the temporal nature of the activities. Correlation heatmaps were generated to identify redundant features.

2.2 Feature Engineering

Raw time-series data is high-dimensional and noisy. To improve model performance, we engaged in extensive feature engineering:

- **Statistical Features:** Mean, standard deviation, min, max, and percentiles were calculated for windowed signal segments.
- **Frequency Domain Features:** Fast Fourier Transform (FFT) was used to extract spectral energy and dominant frequencies, which are critical for distinguishing rhythmic activities like walking from static ones like sitting.

2.3 Dimensionality Reduction

To visualize the high-dimensional feature space and reduce computational cost, we applied:

- **PCA (Principal Component Analysis):** Reduced the feature space while retaining 95% of the variance.
- **t-SNE (t-Distributed Stochastic Neighbor Embedding):** Used for 2D visualization, showing clear separation between static activities (Sitting, Lying) and dynamic ones (Walking, Stairs).

3 Experiments

3.1 Unsupervised Learning

We first applied unsupervised methods to see if natural clusters corresponded to activity labels.

- **K-Means Clustering:** Tested with $K = 6$. While it separated static and dynamic groups well, it struggled to distinguish fine-grained classes like ”Going Upstairs” vs. ”Walking”.
- **DBSCAN:** Used for density-based clustering, primarily to detect outliers/anomalies in the dataset.

3.2 Supervised Learning

We trained four classical machine learning classifiers using a 5-fold stratified cross-validation strategy to ensure robustness.

1. **Logistic Regression:** A computed baseline model.
2. **Random Forest (RF):** An ensemble of decision trees, robust to overfitting.
3. **Support Vector Machine (SVM):** Effective for high-dimensional margins.
4. **Gradient Boosting Classifier (GBT):** A powerful boosting method that builds trees sequentially.

3.3 Neural Networks

We advanced to Deep Learning models using PyTorch:

- **MLP (Multi-Layer Perceptron):** Trained on the engineered feature set.
- **1D CNN (Convolutional Neural Network):** Trained directly on raw time-series data windows, allowing the model to learn spatial-temporal features automatically without manual feature engineering.

3.4 Experiment Tracking with MLflow

All experiments were tracked using **MLflow**. We logged:

- Hyperparameters (learning rates, tree depths, estimators).
- Metrics (Accuracy, F1-Score, Training Time).
- Artifacts (Confusion Matrices, Loss Curves).

This ensured full reproducibility and facilitated easy comparison of model performance.

4 Results and Discussion

The performance of the supervised models on the test set is summarized below:

Model	**Test Accuracy**	**F1-Score (Weighted)**	**Training Time (s)**
Logistic Regression	85.33%	0.8435	~19.0
SVM (RBF Kernel)	88.48%	0.8845	~31.7
Random Forest	**97.06%**	**0.9706**	**~22.7**
Gradient Boosting	**97.12%**	**0.9711**	**~23.6**

Table 1: Performance Comparison of Supervised Models

4.1 Key Findings

- **Ensemble Dominance:** Gradient Boosting and Random Forest significantly outperformed linear models and SVM, achieving over 97% accuracy. They effectively handled the non-linear relationships in sensor data.
- **Feature Engineering:** The custom features extracted (Statistical + FFT) played a vital role in the success of classical ML models.
- **Neural Networks:** The 1D CNN showed promising results on raw data, proving that deep learning can bypass manual feature extraction, though at the cost of higher training time and computational resources.

5 Conclusion

In this project, we successfully developed a high-performing Human Activity Recognition system. We verified that while deep learning (1D CNN) offers powerful automated feature extraction, classical ensemble methods like **Gradient Boosting**, when combined with expert feature engineering, provide state-of-the-art accuracy with significantly lower computational overhead.

The full experimental pipeline, tracked via MLflow, ensures that our results are reproducible and scalable for future research or real-time deployment on edge devices.