Efficient IMU Signal Classification for Edge Devices: A Lightweight Machine Learning Approach

Devin Setiawan[1], Maisoon Rahman[1], Liken Hananto[2]

[1] *The University of Kansas, Department of Electrical Engineering and Computer Science, 1415 Jayhawk Blvd. Lawrence, KS 66045*

[2] *The University of Kansas, Department of Mechanical Engineering, 1415 Jayhawk Blvd. Lawrence, KS 66045*

Abstract—This project investigates human activity recognition (HAR) using time-series data from Inertial Measurement Units (IMUs), focusing on three activity classes: No Activity, Walking, and Bending. While we initially planned to investigate deep learning models, the study pivoted to traditional machine learning methods, combining minimal feature extraction with models such as logistic regression, support vector machines, random forests, and XGBoost, after observing strong performance with significantly smaller model sizes. Our approach uses only mean and standard deviation features from sliding windows, enabling an efficient pipeline well-suited for real-time edge deployment. We found that the tradeoff between performance and model size overwhelmingly favored traditional methods, especially XGBoost, making further exploration into deep learning models unnecessary in this context. We showed the effectiveness of our approach for edge devices on both a public dataset and a custom lab-collected dataset

I. Introduction

Human Activity Recognition (HAR) using wearable sensors has gained increasing attention due to its applications in health monitoring, rehabilitation, fitness tracking, and human-computer interaction. In particular, Inertial Measurement Units (IMUs) provide a low-cost, non-invasive, and energy-efficient way to collect rich time-series motion data. However, deploying HAR systems on edge devices such as microcontrollers or smartphones presents unique challenges: models must be not only accurate but also lightweight and fast enough to run in real time with limited computational resources. This study is motivated by the need to bridge the gap between high-performance HAR models and their practical deployment on resource-constrained hardware. While recent work in HAR has often focused on deep learning approaches requiring significant computation and memory, we investigate whether traditional machine learning models can offer an alternative for real-time edge deployment.

The input to our framework consists of windowed IMU signals captured from wearable sensors. Each input is transformed into a low-dimensional feature vector using simple statistical descriptors: specifically, the mean and standard deviation of 3-axis accelerometer and gyroscope signals. The output is a predicted activity class label such as No Activity, Walking, and Bending, etc. To evaluate the performance-efficiency trade-off, we implemented four traditional machine learning models: Logistic Regression, Support Vector Machines (SVMs), Random Forests, and XGBoost. These models were selected to explore a range of complexity, interpretability, and decision boundary flexibility. While deep learning models were initially considered, preliminary experiments revealed that these simpler models could achieve sufficient accuracy with significantly reduced computational cost and model size, making them ideal for embedded systems. Our findings demonstrate that models like XGBoost offer both accuracy and efficiency, achieving high classification performance across both the public and custom datasets while maintaining a small memory footprint. This work not only highlights the effectiveness of minimal feature engineering in time-series classification but also provides a reproducible framework for future low-power activity recognition systems.

II. Related Work

A wide variety of classification methods have been investigated for HAR using wearable sensors, particularly IMUs. Early studies incorporated simple heuristic classifiers tailored to specific applications, while more recent work has adopted generic and automated methods from machine learning. These traditional approaches include decision trees, k-nearest neighbors, Bayesian networks, SVMs, artificial neural networks (ANNs), Gaussian mixture models (GMMs), and Markov models [1]. One notable example of a neural network-based system is presented by Khan et al. [2], who proposed a hierarchical architecture that utilizes a single triaxial accelerometer worn on the chest. The system recognizes 15 different activities with an average accuracy of 97.9%. This system excels in recognizing challenging posture transitions and demonstrates high accuracy, but its complexity and dependence on engineered features may hinder deployment on constrained platforms.

Another approach using traditional machine learning and ensemble methods is that of Minnen et al. [3], who apply boosting to select and combine 1D classifiers based on wearable sensor data. Their model achieves 90.3% accuracy in recognizing modeled activities and was validated using datasets collected both in controlled laboratory conditions. However, similar to the ANN approach, features are selected automatically and are not always interpretable. Our work aligns with these studies in focusing on HAR using wearable IMUs. Like many prior works, we leverage time-series data segmented through a sliding window approach and extract statistical features from these segments. We also utilize traditional machine learning models for activity classification. However, a key difference in our work is the explicit focus on deploying HAR systems on resource-constrained edge devices. This requires careful balancing of model accuracy with computational efficiency and memory footprint, making model size and runtime critical design considerations.

III. Dataset

This study utilizes three datasets, detailed in Table 1. The first is the UCI Human Activity Recognition Using Smartphones Dataset v1.0 by Reyes-Ortiz et al. [4], which includes data from 30 participants performing six daily activities. A waist-mounted smartphone recorded 3-axis accelerometer and gyroscope data at 50 Hz. Each sample is transformed into a 561-dimensional feature vector derived from time and frequency domain features, with activity labels annotated via synchronized video. To simulate a real-time, low-resource deployment scenario, we also employ a reduced version of the UCI dataset, referred to as the UCI Minimal Dataset. This version retains only 12 handpicked features: the mean and standard deviation of body acceleration and gyroscope signals across the X, Y, and Z axes. In addition, we introduce a custom Lab Dataset collected in-house using wearable IMUs placed on the upper and lower body of two subjects performing three types of activities: walking, bending, and other. Each IMU records triaxial acceleration and angular velocity data. From this raw sensor data, we extract the same 12 features per IMU as used in the UCI Minimal Dataset, resulting in a 24-dimensional feature vector per sample. The resulting input-output configurations are as follows: the full UCI dataset consists of 561-dimensional input vectors with six-class activity labels; the UCI Minimal Dataset contains 12-dimensional inputs with the same six classes; and the Lab Dataset yields 24-dimensional input vectors with three-class labels. Data splits are subject-based for the UCI datasets and stratified for the Lab Dataset. Table 1 summarizes the source, dimensionality, and label classes for each dataset.

Table 1: Overview of the Datasets Used in the Study

Dataset	Source	Subjects	Activities	Features	Feature Dimensions	Data Split	Additional Notes
UCI Dataset	Original Human Activity Recognition Dataset (v1.0) [1]	30 volunteers (19–48 yrs)	Walking, Walking Upstairs, Walking Downstairs, Sitting, Standing, Laying	561 time- and frequency-domain features	561	70% Training / 30% Testing	Features normalized to [-1, 1]
UCI Minimal	Reduced subset of UCI Dataset with selected features	30 volunteers (19–48 yrs)	Walking, Walking Upstairs, Walking Downstairs, Sitting, Standing, Laying	12 selected features (mean and std for body acceleration and gyroscope)	12	70% Training / 30% Testing	Same data split and normalization as UCI
Lab Dataset	Custom dataset collected using IMUs on upper and lower body	2 subjects	Walking, Bending, Other	24-dimensional feature vector (upper + lower body)	24	80% Training / 20% Testing	Data stratified by activity class

Fig. 1: Overview of the datasets used in the study, including the UCI dataset, UCI minimal dataset, and custom Lab dataset.

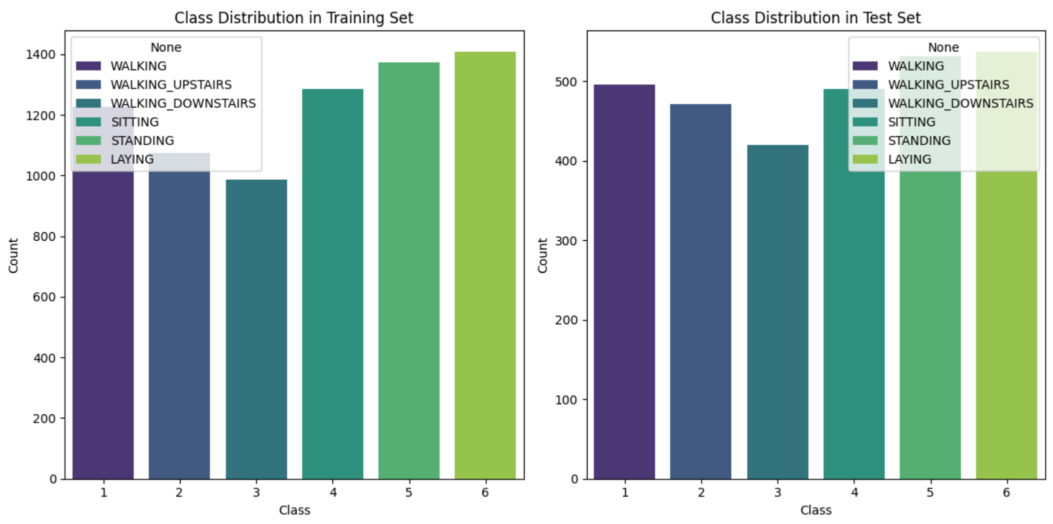

Fig. 2: Class distribution of the UCI Dataset

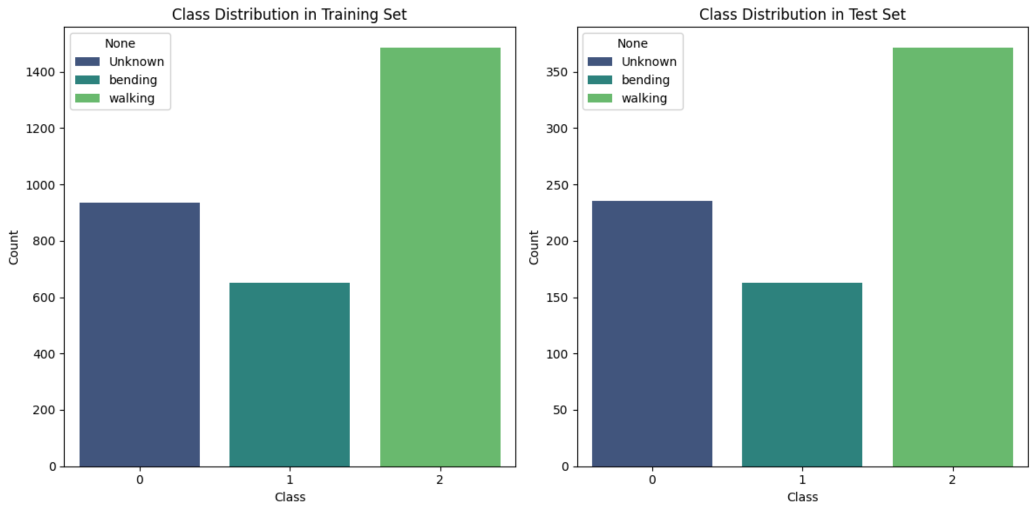

Fig. 3: Class distribution of the Lab Dataset

IV. Methodology

Problem Formulation

In this study, HAR is framed as a multi-class classification problem using a sliding window segmentation approach. Continuous sensor readings comprising 3-axis accelerometer and gyroscope data are segmented into fixed-size, overlapping windows, where each window is treated as a standalone data instance. This approach enables us to convert a time-series prediction task into a traditional supervised learning problem suitable for standard classification models. To ensure consistency across datasets, we apply a window size of 128 readings (corresponding to 2.56 seconds at a 50 Hz sampling rate) with 50% overlap to both the UCI and Lab datasets. This choice of parameters balances temporal resolution and data quantity, and aligns with conventions in prior HAR literature. Each segmented window is labeled according to the last activity observed within the window. This labeling strategy reflects the perspective of a real-time system, where timely detection of activity transitions is critical. By labeling based on the most recent activity, we aim to capture changes in behavior as soon as possible, allowing the system to respond promptly in edge deployment scenarios.

Let $X^{(i)} \in \mathbb{R}^{w \cdot d}$ denote the i-th windowed input, where w is the window size and d is the number of sensor channels (e.g., accelerometer and gyroscope axes). For each window $X^{(i)}$, we extract a handcrafted feature vector $z^{(i)} \in \mathbb{R}^{p}$, where p is the number of extracted features (e.g., statistical summaries such as mean and standard deviation). The corresponding class label $y^{(i)} \in Y$ is determined using the last activity label within the window to simulate a real-time system, capturing transitions as early as possible. We aim to learn a mapping function:

$$f_\theta : \mathbb{R}^{w \cdot d} \to Y$$

which maps each input window to a predicted activity label from the label set Y. This function is trained using labeled pairs $\left\{\left(z^{(i)}, y^{(i)}\right)\right\}_{i=1}^{N}$, where N is the number of training examples and θ denotes model parameters. An overview of the full data processing pipeline is provided in Figure 4, which illustrates the flow from raw IMU signals through windowing and feature extraction to classification. A closer look at the sliding window mechanism is shown in Figure 5, demonstrating how overlapping windows are created from continuous signals to preserve temporal continuity and enrich the training set.

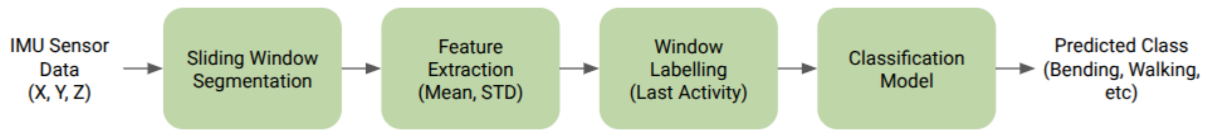
IMU Sensor Data (X, Y, Z)
Sliding Window Segmentation
Feature Extraction (Mean, STD)
Window Labelling (Last Activity)
Classification Model
Predicted Class (Bending, Walking, etc)

Fig. 4. Overview of the data processing pipeline used in this study, illustrating the transformation of raw IMU sensor signals into overlapping windows, extraction of minimal statistical features, and subsequent classification using traditional machine learning models.

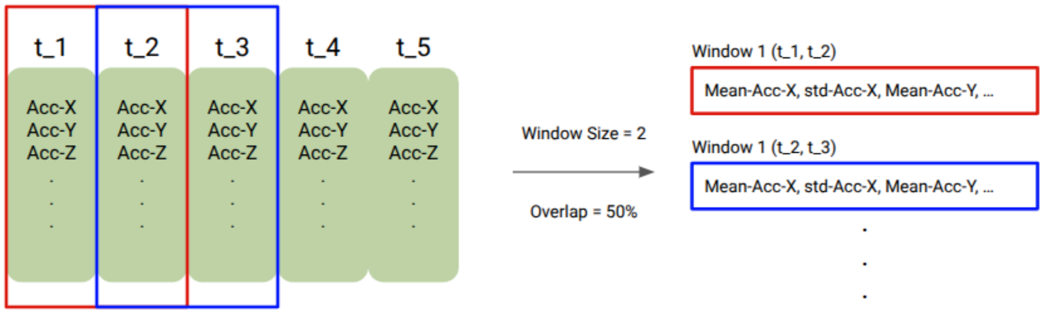

Fig. 5. Illustration of the sliding window segmentation process with a simplified window size of 2 and 50% overlap. The red box indicates the first window, and the blue box shows the second window, highlighting how overlapping windows are constructed from continuous sensor readings to preserve temporal context.

Machine Learning Models

To evaluate the effectiveness of traditional machine learning methods on IMU-based activity recognition, we employ four widely-used classifiers: Logistic Regression, SVM, Random Forest, and XGBoost. All models are implemented using Scikit-learn with default parameters. These models are trained on features extracted from sliding windows of IMU signals, as described in the Problem Formulation section.

(1) Logistic Regression:
Logistic Regression is a linear classification algorithm that models the probability that a given instance belongs to a specific class using a logistic (sigmoid) function. For multiclass classification, the softmax extension is used to compute the probability of each class as

$$P\left(y=k \mid z\right) = \frac{\exp\left(w_k^{\top} z\right)}{\sum_{j=1}^{K} \exp\left(w_j^{\top} z\right)}$$

where w_k is the weight vector associated with class k, and z is the input feature vector. The training objective is to minimize the categorical cross-entropy loss:

$$L_{LR} = -\sum_{i=1}^{C} y_i \log\left(\hat{y}_i\right)$$

In the context of activity recognition, Logistic Regression provides a fast and interpretable baseline that performs well when classes are linearly separable. However, it lacks the capacity to model complex, nonlinear decision boundaries and may struggle with overlapping or noisy data.

(2) Support Vector Machine:
Support Vector Machine (SVM) is a margin-based classifier that seeks the hyperplane which maximally separates different classes. In the linear case used in this study, the decision function takes the form

$$f(z) = \text{sign}\left(w^\top z + b\right)$$

where w and b define the separating hyperplane. The model is trained by minimizing the hinge loss with L2 regularization:

$$L_{svm} = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\max\left(0, 1 - y^{(1)}\left(w^\top z^{(i)} + b\right)\right)$$

SVMs are particularly effective in high-dimensional spaces and when the margin between classes is well defined. However, the model can be computationally expensive with large datasets and is sensitive to feature scaling.

(3) Random Forest:
Random Forest is an ensemble learning method based on decision trees. It constructs a collection of decision trees using bootstrapped samples of the training data and randomly selects a subset of features at each split. The final prediction is made by aggregating the outputs of all individual trees:

$$f(z) = mode\,(T_1(z), T_2(z), \ldots, T_M(z))$$

where $T_m(\cdot)$ denotes the prediction from the m-th tree. Internally, each decision tree minimizes impurity measures such as Gini index or entropy to determine the best splits. Random Forest is well-suited for our task because it naturally handles nonlinear relationships. The primary downside is that it is less interpretable than a single tree and can be computationally intensive in terms of memory and inference time.

(4) XGBoost:
XGBoost is a gradient-boosted tree method known for its speed and accuracy. It builds an ensemble of decision trees sequentially, where each new tree attempts to correct the prediction errors of the previous ensemble. The final prediction is the sum of the outputs of all trees:

$$f(z) = \sum_{m=1}^{M} T_m(z)$$

Training is performed by minimizing a regularized objective function that combines a differentiable loss with a complexity penalty:

$$L_{XGB} = \sum_{i=1}^{N} l\left(y^{(i)}, \hat{y}^{(i)}\right) + \sum_{m=1}^{M}\Omega(T_m)$$

where l is the loss function and Ω penalizes model complexity such as the number of leaves or the depth of each tree. In the activity recognition setting, XGBoost often outperforms other models due to its ability to capture complex patterns and interactions. However, it tends to be slower to train and less interpretable, and its performance can be sensitive to hyperparameter settings.

Evaluation Metrics

We evaluate models using both predictive performance and deployment feasibility. Accuracy is our primary metric, given the balanced class distributions in the UCI and Lab datasets. We supplement this with confusion matrices that provide insight into class-wise performance. To assess suitability for real-time embedded systems, we also report model size in kilobytes as a proxy for Flash memory usage. This hardware-aware metric is crucial for edge deployment, where resources are limited. By comparing accuracy alongside model size, we aim to identify models that offer the best trade-off between predictive performance and deployment efficiency.

V. Results

Heavy Preprocessing: Establishing a High-Performance Baseline

This phase aims to establish a strong baseline for IMU-based activity classification using the UCI dataset with heavy preprocessing. The dataset includes rich time- and frequency-domain features extracted via sliding windows and signal processing techniques like FFT. As shown in Figure 6, PCA and t-SNE projections reveal clear class clusters, demonstrating the effectiveness of these engineered features and setting an upper benchmark for later, lightweight approaches.

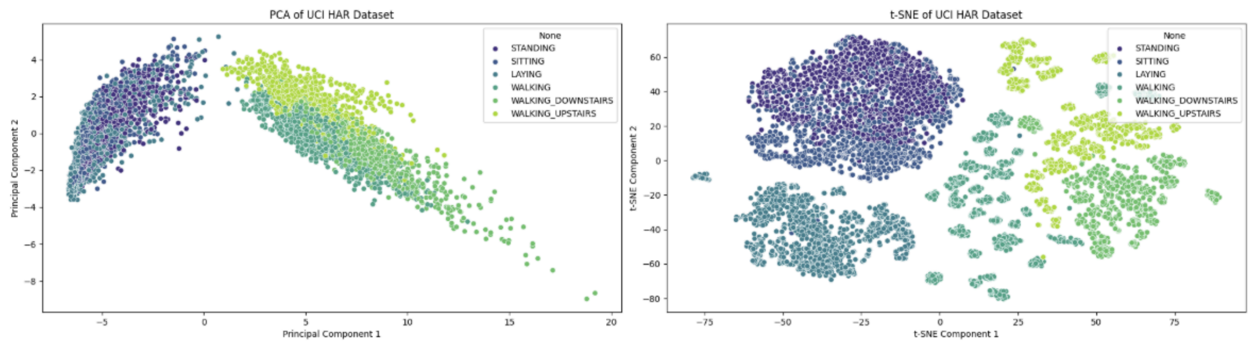

Fig. 6. Dimensionality reduction using PCA (left) and t-SNE (right) on the UCI dataset with heavy preprocessing. Each color represents a different activity class. The clear separation between clusters indicates that the provided features effectively capture class-specific patterns, making the data highly discriminative for classification.

We evaluated four classifiers: Logistic Regression, SVM, Random Forest, and XGBoost on the full UCI dataset. As shown in Table 2, Figure 7, Logistic Regression achieved the best results with 96% accuracy and a 27 KB model size, outperforming more complex models. However, despite its lightweight classifier, the heavy preprocessing pipeline remains a barrier to real-time edge deployment due to high computational demands. Future work should assess hardware requirements for implementing this full pipeline on embedded systems.

Table 2: Performance summary on UCI dataset with heavy preprocessing

Model Name	**Test Accuracy**	**Model Size**
Logistic Regression	96%	27.21 KB
Support Vector Machine	95%	9949.67 KB
Random Forest	93%	5809.88 KB

XGBoost	94%	675.05 KB

Fig. 7. Performance summary of traditional machine learning models on the UCI dataset with heavy preprocessing. The simplest model achieves the highest accuracy (96%) while maintaining the smallest model size (27 KB), highlighting the strong discriminative power of the preprocessed features.

Minimal Preprocessing: Toward Real-Time Feasibility

In this phase, we focus on a lightweight preprocessing pipeline suitable for low-power edge devices. To minimize computational load, we compute only the mean and standard deviation of each IMU signal within a sliding window, yielding 12 features. This minimal approach enables fast, low-memory execution but sacrifices some discriminative power. As shown in Figure 8, PCA and t-SNE plots reveal reduced class separability and greater overlap, indicating the simplified features are less expressive than those in Phase I.

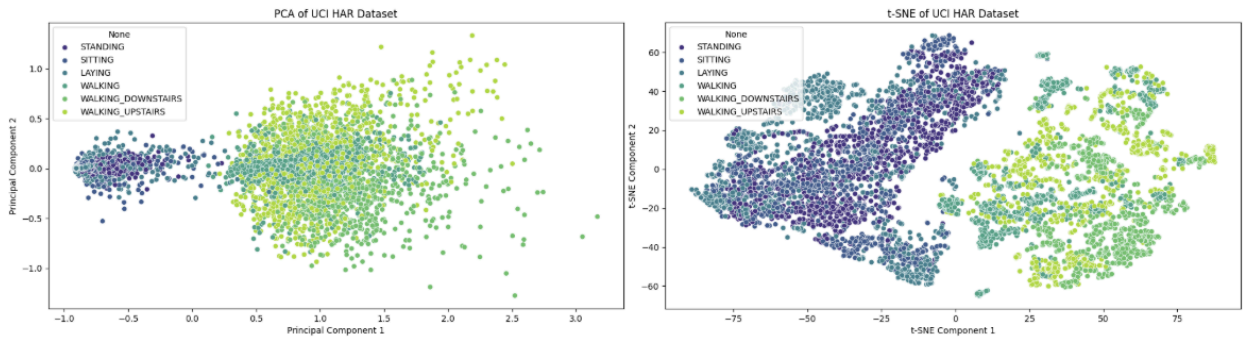

Fig. 8. Dimensionality reduction using PCA (left) and t-SNE (right) on the UCI dataset with minimal preprocessing. Each color represents a different activity class. The separation between clusters are now less clear, classification tasks at hand are more difficult.

Despite reduced feature quality, traditional models still perform reasonably well. As shown in Table 3 and Figure 9, XGBoost achieves 81% accuracy with a 1.4 MB model—down from 96% with heavy preprocessing but acceptable for real-time use. Tree-based models like XGBoost adapt by growing larger, while simpler models like Logistic Regression and SVM stay compact but fall below 65% accuracy. Overall, the minimal preprocessing pipeline offers a strong balance between performance and resource efficiency, making it well-suited for low-power, embedded applications. From an edge deployment perspective, the minimal preprocessing approach shows substantial promise. The preprocessing itself is computationally trivial, thus, while some accuracy is sacrificed, this configuration aligns closely with the practical constraints of real-time, low-power embedded systems.

Table 3: Performance summary on UCI dataset with minimal preprocessing

Model Name	**Test Accuracy**	**Model Size**
Logistic Regression	64%	1.48 KB
Support Vector Machine	61%	992.88 KB
Random Forest	81%	20383.43 KB
XGBoost	81%	1413.07 KB

Fig. 9. Performance summary of traditional machine learning models on the UCI dataset with minimal preprocessing. The XGBoost model achieves the highest accuracy (81%) while maintaining a small model size (1.4 MB).

Real-World Validation: Lab Dataset with Minimal Preprocessing

The final phase of our study validates the minimal preprocessing pipeline on a real-world lab dataset. This dataset involves only three activity classes, reducing classification complexity, and includes 24 features from upper and lower body IMUs. We apply the same mean and standard deviation per sliding window preprocessing. As shown in Figure 10, PCA and t-SNE visualizations reveal clearer class separability compared to the six-class UCI dataset, reflecting the simpler task and more distinct clustering.

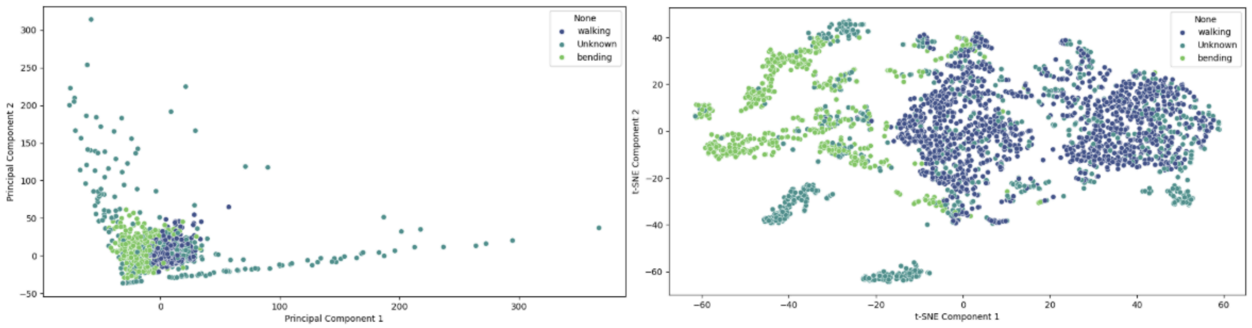

Fig. 10. Dimensionality reduction using PCA (left) and t-SNE (right) on the Lab Dataset with minimal preprocessing. Each color represents a different activity class. The classification tasks at hand are easier, with only three classes.

Figure 11 shows the confusion matrix for the best-performing model, XGBoost, which achieves strong accuracy across all classes but occasionally misclassifies activities as "walking", which is likely due to signal overlap during transitions. As summarized in Table 4, Figure 12, XGBoost reached 86% accuracy with a compact 650 KB model size. Simpler models like Logistic Regression and SVM also performed well, exceeding 75% accuracy, making them viable for real-time applications. Notably, despite doubling the feature count, model sizes decreased, suggesting that tree-based models efficiently adapt to more informative inputs. These results strongly support the feasibility of this pipeline on edge devices. In practice, this means achieving near state-of-the-art activity recognition with <1MB flash memory, making our approach highly suitable for real-world, resource-constrained deployments such as wearable health monitors or embedded motion tracking systems.

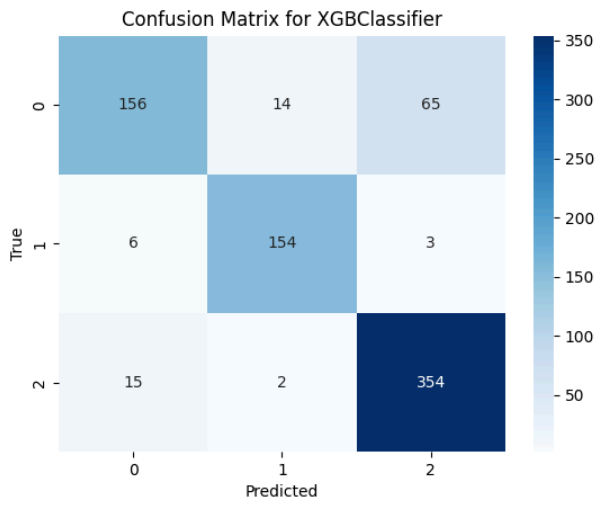

Fig. 11. Confusion matrix of the best-performing model (XGBoost) on the lab dataset. The model demonstrates strong classification performance across all three classes, though a noticeable portion of misclassifications are skewed toward predicting the "walking" class, suggesting potential overlap in sensor patterns for certain activities.

Table 4: Performance summary on Lab Dataset with minimal preprocessing

Model Name	**Test Accuracy**	**Model Size**
Logistic Regression	77%	1.44 KB
Support Vector Machine	82%	329.86 KB
Random Forest	85%	5340.00 KB
XGBoost	86%	650.00 KB

Fig. 12. Performance summary of traditional machine learning models on the Lab Dataset with minimal preprocessing. The XGBoost model achieves the highest accuracy (86%) while maintaining a small model size (650 KB).

VI. Conclusion

This study explored the feasibility of deploying traditional machine learning models for real-time human activity recognition using IMU data under various preprocessing scenarios. We demonstrated that while heavily preprocessed data yields high classification accuracy with simple models like logistic regression, the associated computational overhead limits practical deployment on edge devices. Conversely, our minimal preprocessing pipeline centered on lightweight statistical features offered a compelling trade-off between accuracy and efficiency. Validated with real-world lab data, the XGBoost model achieved 86% accuracy with a size under 1 MB, affirming the viability of this approach for low-power, resource-constrained embedded systems. Future work will focus on optimizing preprocessing on-device and benchmarking inference latency in live environments.

VII. References

[1] Majid Sepahvand, M. N. Meqdad, and Fardin Abdali-Mohammadi, “State-of-the-art in human activity recognition based on inertial measurement unit sensors: survey and applications,” *International Journal of Computers and Applications*, pp. 1–16, Nov. 2024, doi: https://doi.org/10.1080/1206212x.2024.2426501.

[2] A. M. Khan, Young-Koo Lee, S. Y. Lee, and Tae-Seong Kim, “A Triaxial Accelerometer-Based Physical-Activity Recognition via Augmented-Signal Features and a Hierarchical Recognizer,” *IEEE Transactions on Information Technology in Biomedicine*, vol. 14, no. 5, pp. 1166–1172, Sep. 2010, doi: https://doi.org/10.1109/titb.2010.2051955.

[3] D. Minnen, T. L. Westeyn, D. Ashbrook, P. Presti, and T. Starner, “Recognizing Soldier Activities in the Field,” pp. 236–241, Jan. 2007, doi: https://doi.org/10.1007/978-3-540-70994-7_40.

[4] J. Reyes-Ortiz, D. Anguita, A. Ghio, L. Oneto, and X. Parra. "Human Activity Recognition Using Smartphones," UCI Machine Learning Repository, 2013. [Online]. Available: https://doi.org/10.24432/C54S4K.