

Challenge 2025 – ADAMMA: On-Device MET Class Prediction from Smartphone Accelerometers

Arda Baris Basaran¹

¹ École Polytechnique Fédérale de Lausanne (EPFL)

ardabarisbasaran@hotmail.com

Abstract

We present a mobile system that continuously estimates a user’s Metabolic Equivalent of Task (MET) *class* (Sedentary, Light, Moderate, Vigorous) from smartphone accelerometer data and displays the cumulative time spent in each class throughout the day. The model is trained and evaluated on the WISDM “Activity Prediction” dataset, which contains six activities recorded at 20.00 Hz from phones carried in users’ front pockets [1, 2]. We map activities to MET classes using the Compendium of Physical Activities’ conventional intensity thresholds (sedentary 1.0–1.5, light 1.6–2.9, moderate 3.0–5.9, vigorous ≥ 6.0) [3, 4]. Our pipeline uses 5.00 s windows with a 1.00 s stride, a 43-dimensional hand-crafted feature set implemented in PyTorch for fast on-device computation, and both classic ML classifiers (logistic regression, kNN, RBF-SVM, random forest, Gaussian NB, and a small MLP) and a compact 1D CNN suitable for on-device deployment. We report standard metrics (accuracy, precision, recall, F1) and per-sample inference latency, and describe an Android foreground-service implementation that maintains near real-time predictions and a clean UI with live cumulative time bars. We release source code (Android app, training code), trained models (including ONNX exports), and scripts to reproduce our results.

1 Introduction

Daily time spent across intensity classes (sedentary, light, moderate, vigorous) is a strong predictor of cardio-metabolic risk and all-cause mortality. METs (Metabolic Equivalents of Task) provide a simple, widely used scale to relate activity energy cost to resting metabolism. In the Compendium, intensities are typically categorized as sedentary 1.0–1.5 METs, light 1.6–2.9, moderate 3.0–5.9, and vigorous ≥ 6.0 [3, 4].

Modern smartphones ship with tri-axial accelerometers and sufficient compute to run lightweight ML models continuously. Our goal is a fully on-device app that (i) infers the current MET *class* in near real time from acceleration alone and (ii) maintains an always-

updating tally of cumulative minutes per class for “today”, with a small, readable UI and low battery overhead.

We build on the WISDM “Activity Prediction” dataset [1, 2]. While WISDM contains six activity labels—Walking, Jogging, Upstairs, Downstairs, Sitting, Standing—health guidance and the ADAMMA challenge operate in terms of *MET classes*. We therefore adopt a principled mapping from activities to intensity bins, train models to predict the four-class intensity label, and deploy a compact classifier on Android.

Contributions.

- A complete training & evaluation pipeline for four-class MET intensity recognition using WISDM accelerometer data.
- A 43-feature, PyTorch-implemented extractor adapted from the original WISDM feature set [1].
- Classic ML baselines and a compact 1D CNN, with accuracy/F1 and per-sample latency suitable for on-device use.
- An Android architecture (foreground service + live UI) for continuous sensing and cumulative time tracking.
- Reproducibility: scripts, configuration, and model artefacts.

2 Methodology

2.1 Dataset

We use the WISDM Activity Prediction dataset [2], collected with Android phones carried in the *front pants pocket* at 20.00 Hz (one sample every 50.00 ms). The canonical WISDM processing derives 43 features on 10 s windows (≈ 200 samples) [1], while our on-device pipeline (below) uses shorter windows for lower latency. The dataset provides six activities: *Walking, Jogging, Upstairs, Downstairs, Sitting, Standing*. According to the dataset metadata there are 1,098,207 labeled samples across 36 users (IDs 1 .. 36) with the following distribution:

Activity	Count	Percent
Walking	424,400	38.6%
Jogging	342,177	31.2%
Upstairs	122,869	11.2%
Downstairs	100,427	9.1%
Sitting	59,939	5.5%
Standing	48,395	4.4%

Each line in the WISDM dataset has the form:

```
[user], [activity], [timestamp],
[x-acceleration], [y-accel], [z-accel];
```

Fields:

- `user` — nominal, integers 1..36.
- `activity` — nominal in {Walking, Jogging, Sitting, Standing, Upstairs, Downstairs}.
- `timestamp` — numeric; generally phone uptime in nanoseconds (future datasets may use milliseconds since Unix epoch).
- `x-acceleration`, `y-accel`, `z-accel` — floating-point values typically in $[-20, 20]$. A value of 10 corresponds to $1g \approx 9.81 \text{ m/s}^2$; values include gravitational acceleration, so a resting vertical axis registers around ± 10 .

Dataset choice. Before committing to WISDM, we considered several other widely used activity recognition datasets. The UCI-HAR dataset [5] provides accelerometer and gyroscope signals from a waist-mounted smartphone, pre-segmented into 2.56 s windows; however, it lacks vigorous-intensity classes (no jogging or running). The MHEALTH dataset [6] includes rich multi-sensor streams (ECG, accelerometer, gyroscope, magnetometer) and diverse activities such as cycling and running, but its setup with multiple body-worn devices makes it less representative of a single-phone scenario. PAMAP2 [7] similarly offers detailed multi-sensor data and a broad range of activities mapped to METs, yet requires three body-mounted IMUs. More recently, WEEE [8] combines accelerometry with indirect calorimetry for precise energy expenditure, but is not phone-based. Compared to these, WISDM [1, 2] is uniquely well aligned with our goals: it uses only the smartphone’s internal accelerometer, sampled at 20 Hz, in the front pocket carry position. Since our aim is a deployable Android app using just the phone’s own sensors, WISDM provides the most realistic match in terms of modality and placement, while still allowing a principled mapping from activities to MET intensity classes (Table 1).

Why classification (not regression). We formulate intensity recognition as a *four-class* classification task rather than direct MET regression for several reasons.

First, the selected dataset (WISDM) does not provide continuous, ground-truth energy expenditure labels; it contains activity categories only, so any MET value would have to be *imputed* from external tables (e.g., the Compendium) and then treated as truth, introducing label noise and cascading bias [3, 4, 2]. Second, mapping a regressed MET *value* to intensity requires a second thresholding step (e.g., 1.5/3/6 METs); this two-stage pipeline compounds errors (regression + thresholding), whereas a single classifier can optimize the decision boundaries for the end objective directly. Third, absolute MET varies substantially with person-specific physiology (mass, fitness), biomechanics, and phone placement/orientation—factors that an accelerometer-only model cannot reliably disambiguate without per-user calibration. Treating the problem as classification avoids conveying spurious numerical precision and aligns the model’s outputs with clinical/public-health guidance and our UI.

Our windowing. For low-latency on-device prediction, we use **5 s** windows with a **1 s** stride at 20.00 Hz ($5 \times 20 = 100$ samples per window). This choice, aligned with typical mobile phone sensing setups, balances decision latency with sufficient temporal context to capture periodic structure in gait and stair-climbing, while providing more frequent predictions.

Preprocessing. We drop malformed or all-zero rows, cast types, and sort by `user`, `timestamp`. Windows are generated sequentially per user with a fixed stride (no overlap unless otherwise noted). Each window’s label is the *majority* activity over its samples. (Implementation in `datasets/dataset.py`.)

2.2 Mapping Activities to MET Classes

We map WISDM activities to intensity classes using Compendium thresholds [3, 4] and typical MET values:

Activity	MET Class (rationale)
Sitting, Standing	Sedentary (< 1.5 MET)
Walking (casual)	Light (1.6–2.9 MET)
Upstairs, Downstairs	Moderate (3.0–5.9 MET; stair climbing ~ 4 –8)
Jogging/Running	Vigorous (≥ 6.0 MET)

Table 1: Activity categories and their MET classes.

This yields four balanced-by-design intensity labels for the challenge: *Sedentary*, *Light*, *Moderate*, *Vigorous*.

2.3 Feature Extraction (43 dims)

We adopt the classic WISDM feature family [1], implemented in PyTorch (`models/model.py`) so the same code can run on CPU/GPU and be ported to mobile if

desired. For each window (shape $T \times 3$; $T=60$ with our 3 s windows) and each axis x, y, z :

- Mean, standard deviation, and average absolute deviation ($3 \times 3 = 9$).
- Average resultant acceleration $\mathbb{E}[\sqrt{x^2 + y^2 + z^2}]$ (1).
- Time-between-peaks per axis (3): local-peak detection; average spacing converted to ms via sampling rate.
- Binned distributions per axis (10 bins $\times 3 = 30$).

Total $9 + 1 + 3 + 30 = 43$ features per window. The extractor is window-length agnostic.

2.4 Models

We evaluate classic scikit-learn classifiers and a compact neural network suitable for mobile deployment:

- **Logistic Regression** (max_iter=1000).
- **kNN** ($k = 5$).
- **RBF-SVM**.
- **Random Forest** (100 trees).
- **Gaussian Naive Bayes**.
- **MLP** (hidden layers $128 \rightarrow 64$, ReLU, Adam, max_iter=200).
- **SimpleCNN1D** (ours): three depthwise-separable 1D convolution blocks with dilations 1/2/4 and ReLU6, global average pooling, and a small dense head ($64 \rightarrow 32 \rightarrow 4$). Trained with cross-entropy using Adam (lr 10^{-3}), dropout 0.1, for 10 epochs.

Training protocol. By default we use stratified 5-fold cross-validation *over windows* (Scikit’s StratifiedKFold, random_state=42). For fold k , we train on 4 folds and test on the held-out fold. The same protocol is applied to both ML models and the CNN. *Note:* this is not subject-exclusive; we discuss the implications in Section 6.

Metrics and latency. We report accuracy, weighted precision/recall/F1, and *per-sample* inference time (mean over the test set: batch predict \Rightarrow wall time / #samples, expressed in ms/sample). Code in `main.py`. For the CNN, we measure forward-pass wall time in evaluation mode.

2.5 Device Specifications

Build environment. The Android app is developed in Android Studio Narwhal 3 Feature Drop 2025.1.3, Android Gradle Plugin (AGP) 8.5.2, Gradle 8.7, with `compileSdk=34`, `targetSdk=34`, and `minSdk=24`.

Test device. We tested end-to-end sensing and the UI on a Samsung Galaxy A15. The app requests a 20.00 Hz accelerometer sampling rate; on this hardware, the sensor reports the closest supported rate, approximately

19.00 Hz. The window size and stride are set to **5 s** and **1 s**, respectively, matching the training setup.

3 App Design

The Android app follows a simple, readable flow:

- **Splash Screen:** Lightweight splash while the app initializes services and loads model assets.
- **Starter Screen:** Shows the app logo and a prominent *Start* button to begin sensing.
- **Activity Monitor:** The main screen presents the *current state* (Sedentary/Light/Moderate/Vigorous) with an icon and label. A compact *Debug Information* panel displays live accelerometer values (x, y, z), the measured sampling rate in Hz, the model’s output probability distribution over the four classes, and the window/stride configuration in seconds. These values update in real time as new sensor data arrives.
- **History Screen:** A four-tab view (*Day/Week/Month/Year*) summarizing time spent per intensity. The Day tab shows, for each class, a 24-segment bar (one per hour) and a *pie chart* of today’s distribution, plus a *calories today (net)* estimate computed from per-second labels and user profile (age, weight, height, sex). Week/Month/Year aggregate into 7/30/12 segments respectively. A *Reset Data* action in Settings clears all history.

3.1 Calorie Estimation

What a MET means. A Metabolic Equivalent of Task (MET) is a unit that scales energy cost relative to quiet rest. By convention,

$$1 \text{ MET} \triangleq 3.5 \text{ ml O}_2 \text{ kg}^{-1} \text{ min}^{-1}$$

and gross caloric expenditure (including resting) can be estimated from METs and body mass.¹ Importantly, MET \rightarrow kcal depends primarily on *body mass*; height/age/sex mainly matter through basal metabolic rate (BMR), which is separate from activity energy.

From class posteriors to a per-second MET. Our on-device model outputs, each second t , a posterior over four MET classes

$$\mathbf{p}_t = [p_t^{(\text{sed})}, p_t^{(\text{light})}, p_t^{(\text{mod})}, p_t^{(\text{vig})}], \quad \sum_c p_t^{(c)} = 1.$$

We assign each class a representative MET intensity m_c based on the Compendium ranges and our activity mapping (*sedentary*: sitting/standing; *light*: walking; *moderate*: stairs; *vigorous*: jogging). We use mid-range values for stability:

$$m_{\text{sed}} = 1.2, \quad m_{\text{light}} = 2.5, \quad m_{\text{mod}} = 4.5, \quad m_{\text{vig}} = 8.0.$$

¹The factor 3.5 ml/kg/min and the caloric equivalent 1 L O₂ \approx 5 kcal lead to the 200 constant below.

The per-second *expected* MET is the posterior mean:

$$\widehat{M}_t = \sum_{c \in \{\text{sed}, \text{light}, \text{mod}, \text{vig}\}} p_t^{(c)} m_c.$$

To reduce jitter while remaining responsive, we apply an exponential moving average (EMA) with smoothing parameter $\alpha \in (0, 1]$:

$$\widetilde{M}_t = \alpha \widehat{M}_t + (1 - \alpha) \widetilde{M}_{t-1}, \quad \widetilde{M}_0 = \widehat{M}_0.$$

Gross vs. net calories (personalized). Let W be body mass (kg), H be height (cm), A age (years), and S a sex indicator (+5 for male, −161 for female). We first compute resting metabolic rate (RMR) using the Mifflin–St Jeor equation:

$$\text{RMR} = 10W + 6.25H - 5A + S \quad [\text{kcal/day}].$$

Converting to seconds:

$$\dot{E}_{\text{rest}} = \frac{\text{RMR}}{24 \cdot 3600} \quad [\text{kcal/s}].$$

This defines a personalized baseline: $1 \text{ MET} = \dot{E}_{\text{rest}}$.

Thus, the instantaneous gross and net caloric rates are

$$\dot{E}_t^{\text{gross}} = \widetilde{M}_t \cdot \dot{E}_{\text{rest}}, \quad \dot{E}_t^{\text{net}} = \max(\widetilde{M}_t - 1, 0) \cdot \dot{E}_{\text{rest}}.$$

Daily totals are obtained by summation:

$$E_{\text{day}}^{(\cdot)} = \sum_{t \in \text{day}} \dot{E}_t^{(\cdot)}.$$

Units sanity check (worked example). Example: $W = 70$ kg, $H = 175$ cm, $A = 25$ years, male ($S = +5$). Then

$$\text{RMR} = 10 \cdot 70 + 6.25 \cdot 175 - 5 \cdot 25 + 5 = 1673.75 \text{ kcal/day},$$

$$\dot{E}_{\text{rest}} = \frac{1673.75}{86400} \approx 0.0194 \text{ kcal/s}.$$

For $\widetilde{M}_t = 8$ (vigorous jog):

$$\dot{E}_t^{\text{gross}} = 8 \cdot 0.0194 \approx 0.155 \text{ kcal/s} \quad (\approx 9.3 \text{ kcal/min}),$$

which closely matches standard tables for jogging intensity.

4 Results

We evaluate the six classic classifiers *and* the compact 1D CNN under identical 5-fold CV. Unless noted, handcrafted features (43-D) with window length 5.00 s at 20.00 Hz are used. We report *mean* over folds.

To enable a fair comparison across modeling approaches, we report performance using a consistent set of evaluation metrics. Each table entry is formatted as (Acc, Prec, Rec, F1, Inf. time).

Model	Acc (%)	Prec (%)	Rec (%)	F1 (%)	Inf. time (ms)
Logistic Regression	85.4 ± 0.2	84.9 ± 0.2	85.4 ± 0.2	84.9 ± 0.2	0.0003
kNN (k=5)	76.2 ± 0.4	75.8 ± 0.3	76.2 ± 0.4	75.4 ± 0.4	0.0119
RBF-SVM	75.1 ± 0.2	75.4 ± 0.4	75.1 ± 0.2	69.2 ± 0.3	1.8195
Random Forest	98.0 ± 0.1	98.0 ± 0.1	98.0 ± 0.1	98.0 ± 0.1	0.0096
Gaussian NB	84.0 ± 0.3	84.0 ± 0.3	84.0 ± 0.3	83.9 ± 0.3	0.0006
MLP (128,64)	96.8 ± 0.5	96.9 ± 0.5	96.8 ± 0.5	96.8 ± 0.5	0.0015
SimpleCNN1D	90.6 ± 1.2	91.5 ± 1.0	90.6 ± 1.2	90.8 ± 1.1	1.8067

Table 2: 5-fold CV performance using handcrafted 43-D features (5 s windows, 1 s stride). Metrics are mean ± std across folds (in %), inference time reported as mean ± std (ms/sample).

Model	Acc (%)	Prec (%)	Rec (%)	F1 (%)	Inf. time (ms)
Logistic Regression	56.7 ± 0.1	60.2 ± 0.5	56.7 ± 0.1	53.5 ± 0.2	0.001
kNN (k=5)	72.2 ± 0.3	81.0 ± 0.3	72.2 ± 0.3	72.3 ± 0.3	0.057
RBF-SVM	93.5 ± 0.4	93.5 ± 0.4	93.5 ± 0.4	93.4 ± 0.4	5.223
Random Forest	88.5 ± 0.2	89.6 ± 0.2	88.5 ± 0.2	87.3 ± 0.3	0.015
Gaussian NB	62.3 ± 0.3	68.2 ± 0.3	62.3 ± 0.3	63.0 ± 0.2	0.003
MLP (128,64)	94.7 ± 0.3	94.7 ± 0.3	94.7 ± 0.3	94.7 ± 0.3	0.002
SimpleCNN1D	99.0 ± 0.2	99.0 ± 0.2	99.0 ± 0.2	99.0 ± 0.2	0.1702

Table 3: 5-fold CV performance using raw features (5 s windows, 1 s stride). Metrics are mean ± std across folds (in %), inference time reported as mean ± std (ms/sample).

Latency. For each model, we measure test-time mean inference latency (ms/sample) using batch prediction, then divide wall time by the number of test windows (see `main.py`). For the CNN, latency is measured with the network in evaluation mode (BatchNorm/Dropout frozen). Classic scikit-learn classifiers are timed on CPU, whereas the CNN is timed on GPU; these are offline desktop measurements and *not* on-device estimates.

5 Future Work

Pre-filtering (orientation + gravity robustness). Maintain an IIR low-pass estimate of gravity per axis (cutoff ≈ 0.25 Hz), then compute linear acceleration: $\mathbf{g}_t = \alpha \mathbf{g}_{t-1} + (1 - \alpha) \mathbf{a}_t$, $\mathbf{a}_{\text{lin}} = \mathbf{a}_t - \mathbf{g}_t$, with $\alpha = \exp(-2\pi f_c / f_s)$. Also compute the magnitude $m = \|\mathbf{a}_{\text{lin}}\| = \sqrt{x^2 + y^2 + z^2}$ for orientation-invariant cues.

Feature set extensions (per window, from \mathbf{a}_{lin}). Time-domain, fast, orientation-robust features that can complement or replace raw inputs:

- Per axis (x,y,z) and magnitude m : mean, std, MAD $\Rightarrow 4 \text{ signals} \times 3 \text{ stats} = 16$.
- Interquartile range (IQR, $p_{75} - p_{25}$) for x,y,z,m $\Rightarrow 4$.
- Zero-crossing rate (after mean removal) for x,y,z,m $\Rightarrow 4$.
- Signal Magnitude Area (SMA) = $\frac{1}{N} \sum_t (|x| + |y| + |z|) \Rightarrow 1$.
- Energy = $\text{mean}(x^2)$, $\text{mean}(y^2)$, $\text{mean}(z^2)$, $\text{mean}(m^2) \Rightarrow 4$.
- Jerk stats on first difference Δa : std for x,y,z,m $\Rightarrow 4$.

Temporal smoothing \Rightarrow stable MET classes. Per-window predictions may wiggle. Apply lightweight smoothing directly to the class probabilities with an exponential moving average (EMA): $\hat{\mathbf{p}}_t = \alpha \mathbf{p}_t + (1 - \alpha) \hat{\mathbf{p}}_{t-1}$, with $\alpha \approx 0.7$. Combine with simple hysteresis on class switches: require k consecutive frames (e.g., $k = 2-3$) before committing a new class to the UI/history. This yields steadier labels without masking offline errors.

Cross-dataset evaluation. To assess generalization beyond WISDM and reduce dataset-specific bias, we can train/test on additional public benchmarks: *UCI-HAR*, *MHEALTH*, *PAMAP2*, and *WEEE*. We can harmonize sampling rates and windowing, map activities to MET classes, and report subject-exclusive results where possible. Models (43-D features and raw 1D CNN) can be re-tuned per dataset and evaluated for cross-dataset transfer.

Model architectures beyond a simple 1D CNN. Due to the limited time available in this challenge, we restricted ourselves to basic classifiers and a simple 1D CNN. With more time, we would conduct a focused literature review on human activity recognition and efficient on-device time-series modeling, and evaluate additional architectures that better capture long-range temporal structure and multi-scale patterns while remaining lightweight:

- **Recurrent models:** LSTM/GRU with sequence-to-label pooling or temporal attention.
- **Temporal Convolutional Networks (TCN):** Dilated causal convolutions (WaveNet-style) for long receptive fields.
- **CNN-RNN hybrids:** DeepConvLSTM and variants that combine convolutional feature extractors with recurrent back-ends.
- **Multi-scale CNNs:** InceptionTime/ResNet1D/FCN-style architectures with parallel kernels and residual connections.
- **Transformer-based encoders:** Lightweight or linear-attention transformers (e.g., Performer/Informer) and other efficient time-series transformers with positional encodings suitable for on-device inference.
- **Self-supervised pretraining:** Contrastive style objectives (e.g., SimCLR variants) on unlabeled motion data followed by a small classifier head.

We would also explore model compression for deployment (post-training quantization, pruning, etc) and architecture scaling under strict latency/energy budgets on typical smartphones.

6 Discussion

Handcrafted vs. raw windows. Tables 2 and 3 show a consistent pattern: apart from RBF-SVM and

the 1D CNN, most classic models perform better with the *handcrafted* 43-D features than with *raw* windows. This is expected: engineered statistics reduce dimensionality, normalize scale, and summarize periodic structure, which benefits linear models (LR), tree ensembles (RF), Naive Bayes, and small MLPs. In contrast, training these models directly on raw, high-dimensional sequences suffers from the curse of dimensionality and a lack of built-in temporal invariances.

Why SVM and CNN behave differently. The RBF-SVM improves on raw windows, likely because the kernel can capture complex, nonlinear similarities in the original sequence space without relying on human-designed summaries. The 1D CNN performs *best* on raw inputs: convolution and pooling layers learn local, shift-tolerant patterns (e.g., gait cycles) and aggregate them over time, which aligns naturally with accelerometer time series. When forced onto handcrafted features, CNNs lose their main inductive bias and bring less value.

On-device choice: compact 1D CNN on raw inputs. We deploy an ONNX-exported 1D CNN trained on raw windows in the Android app because it offers the best trade-off for our goals:

- **Accuracy:** Highest cross-validated accuracy/F1 among evaluated models on raw data.
- **Latency/size:** Small parameter count, fast forward pass; packaged once as a single `.onnx` file.
- **Simplicity:** Avoids replicating an exact, numerically identical handcrafted feature pipeline on-device; fewer moving parts and less drift risk.
- **Robustness:** Convolutions provide translation tolerance in time; the model generalizes across minor sampling-rate variation (we feed $[1, N \times 3]$ with measured N).

For flexibility, our app can also accept a 43-D feature model (detected via the input shape of the ONNX model) if future experiments favor that path.

7 Software & Reproducibility

Repository layout.

- `app/` and `gradle/wrapper/`: Kotlin app with foreground service, `SensorManager`, feature computation, and UI.
- `MET_Tracker_Trainer/`: contains the scripts to train and evaluate different ML/DL models `main.py`, `datasets/`, `models/`.
- `video_onnx_and_apk_report/`: contains the APK, a video demonstration of the app, the ONNX file `cnn.onnx`, and the report in PDF.

Running training. Example (handcrafted features, classic ML):

```
python main.py \
  --dataset WISDM \
  --dataset_path /path/to/WISDM/ \
  --feature_extraction handcrafted \
  --model ml \
  --n_splits 5 \
```

Run the compact CNN by switching `--model cnn`; use either handcrafted features or raw windows:

```
python main.py \
  --dataset WISDM \
  --dataset_path /path/to/WISDM/ \
  --feature_extraction raw \
  --model cnn \
  --n_splits 5 \
  --batch_size 32
```

8 Conclusion

We implement a compact, on-device MET-class recognizer trained on WISDM accelerometer data, using 43 handcrafted features with light ML models *and* a compact 1D CNN to balance accuracy, latency, and battery. A foreground service drives continuous sensing and a clear UI for cumulative daily minutes by intensity. Future work includes subject-exclusive evaluation, stronger temporal models (e.g., temporal CNNs/TCN, lightweight transformers), and robustness to diverse phone placements.

References

- [1] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, “Activity recognition using cell phone accelerometers,” in *Proceedings of the Fourth International Workshop on Knowledge Discovery from Sensor Data (SensorKDD) at KDD-10*, (Washington, DC), 2010.
- [2] “Wisdm lab: Dataset (activity prediction).” <https://www.cis.fordham.edu/wisdm/dataset.php>. Accessed 2025-09-04.
- [3] S. D. Herrmann *et al.*, “2024 adult compendium of physical activities: A third update of activity codes and met values,” *Journal of Sport and Health Science*, 2024. Defines sedentary 1.0–1.5 METs, light 1.6–2.9, moderate 3.0–5.9, vigorous ≥ 6.0 .
- [4] B. E. Ainsworth, W. L. Haskell, S. D. Herrmann, *et al.*, “2011 compendium of physical activities: a second update of codes and met values,” *Medicine & Science in Sports & Exercise*, vol. 43, no. 8, pp. 1575–1581, 2011.
- [5] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, “Smartphone-based recognition of human activities and postural transitions,” *Pattern Recognition Letters*, vol. 34, no. 16, pp. 2183–2193, 2013.
- [6] O. Banos, R. Garcia, J. A. Holgado-Terriza, M. Damas, H. Pomares, and I. Rojas, “mhealth-droid: a novel framework for agile development of mobile health applications,” in *International Workshop on Ambient Assisted Living*, pp. 91–98, Springer, 2014.
- [7] A. Reiss and D. Stricker, “Benchmarking real-world activity recognition for mobile devices,” in *2012 16th International Symposium on Wearable Computers*, pp. 1–8, IEEE, 2012.
- [8] K. Ailneni and G. Beltrame, “The wearable energy expenditure (weee) dataset: benchmarking energy expenditure estimation from wearable sensors,” *Scientific Data*, vol. 10, no. 1, p. 95, 2023.