Challenge 2025 – ADAMMA - Stefanos Konstantinou

1 Software Architecture and design

The ADAMMA app is an Android application for real-time activity recognition. It continuously classifies smartphone sensor data into four MET classes and displays both live status and cumulative history. The architecture combines background sensing, on-device AI inference, persistent storage, and a reactive UI.

1.1 System Components

1. **User Interface Layer**
   - `MainActivity`:
     - Displays current MET class and current distribution in a PieChart (can be manually re-initialized).
     - Provides start/stop tracking controls and navigation to history.
     - Manages runtime permissions.
   - `HistoryActivity` (& `HistoryAdapter`):
     - Show historical breakdown of daily activity by MET class.
     - Summarizes time spent per class (minutes/seconds) grouped by date.
2. **Background Sensing Layer**
   - `SensorService`:
     - A persistent foreground service ensures uninterrupted sensor access.
     - Collects accelerometer + gyroscope data at 50 Hz.
     - Buffers 150 samples (3s window), computes magnitudes, and prepares data for inference.
     - Extracts engineered features and invokes the model for classification.
     - Logs predictions into the datastore and database with elapsed duration.
3. **Preprocessing & Feature Engineering**
   - `Scaler`: Normalizes raw and feature vectors using means/standard deviations stored as *.npy* assets.
   - `FeatureExtractor`: Computes 36 features per window: mean, std, min, max for 8 channels (accel, gyro, magnitudes), plus jerk statistics.
4. **Model Inference**
   - `ModelInference`:
     - Runs a hybrid ONNX model with dual inputs: scaled raw windows $[1, 150, 8]$ and statistical features $[1, 36]$.
     - Produces probabilities for the four MET classes.
     - Applies temporal smoothing via weighted rolling average across recent windows to reduce jitter.
5. **Data Persistence & Repository Layer**
   - `ActivityEntity`: Defines Room database schema $(id, date, metClass, durationSec)$.
   - `ActivityDao`: SQL interface for insert, query by date, clear logs, and fetch all records.
   - `AppDatabase`: Singleton builder that initializes the Room database ($adamma\_db$).
   - `ActivityRepository`:

- Automatically tags entries with today's date.
- Exposes history retrieval grouped by date and MET class.

6. **Data Management Layer**

   - `DataStore`:
     - In-memory singleton holding the current MET class and cumulative counters.
     - Publishes live data streams for UI updates.
     - Interfaces with `ActivityRepository` for long-term persistence.

1.2 Design Rationale

- **Foreground Service**: Ensures continuous sensing while complying with Android's background execution limits.
- **Hybrid Modeling**: Combining raw sensor streams with statistical features yields both fine-grained temporal resolution and robust summary descriptors.
- **ONNX Runtime**: Lightweight, portable inference engine enabling cross-platform reproducibility.
- **Persistence with Room**: Guarantees long-term reproducibility of activity logs, allowing scientific usage (e.g., behavior tracking over weeks).
- **Reactive Live Data**: Decouples computation from presentation, ensuring UI always reflects the latest predictions.
- **User Transparency**: Real-time status + daily summaries align with the health context: quantifying sedentary vs. active time as a risk predictor.

2 Data

2.1 Sources

The model was trained and evaluated using a combination of publicly available human activity recognition datasets to maximize diversity of activities and device placements:

1. **WISDM v1.1** Kwapisz et al. (2010)
   - Smartphone accelerometer recordings of daily activities.
   - Only accelerometer channels are available.
   - Provides long-duration free-living data.
2. **MotionSense** Malekzadeh et al. (2019)
   - Accelerometer + gyroscope recordings from iPhones.
   - Contains scripted activities (walking, jogging, upstairs, downstairs).
   - Includes both linear acceleration and angular velocity.
3. **UCI Human Activity Recognition** Reyes-Ortiz et al. (2013)
   - Accelerometer + gyroscope recordings collected from waist-mounted smartphones.
   - Standard benchmark dataset for human activity recognition.

Each dataset was re-labeled into **four MET classes** via a mapping dictionary: **Sedentary** (Sitting, Standing, Lying), **Light** (Walking), **Moderate** (Upstairs, Downstairs), **Vigorous** (Jogging, Running).

2.2 Preprocessing

- **Resampling**: All datasets were resampled to a uniform 50 Hz sampling frequency.
- **Windowing**: Sliding windows of 3 seconds (150 samples) with 50
- **Axis Normalization**: Accelerometer readings normalized to units of g; gyroscope left in rad/s.
- **Feature Extraction**: In addition to raw sequences, 36 statistical features were computed (per-channel mean, std, min, max, and jerk stats).

2.3 Data Augmentation

To improve model robustness across different phone orientations and real-world usage conditions, augmentations during training were applied:

- Random rotation of sensor axes ($\pm 60°$).
- Time warping: Random stretching/compression of the time axis (0.9–1.1×).
- Subsampling of WISDM (40% of rows used) to avoid imbalance and memory overflow.

2.4 Dataset Statistics

After preprocessing, the combined dataset contained fixed-length windows of 3s (150 samples at 50 Hz), with both raw signals and 36-dimensional feature vectors extracted.

Dataset	Downstairs	Upstairs	Walking	Sitting	Standing	Jogging/Laying
MotionSense	59,294	71,048	158,622	234,633	224,793	104,303
UCI HAR	3,810	3,810	3,810	3,810	3,810	3,810 (Laying)
WISDM	40,132	49,140	167,261	23,870	19,463	134,720

After windowing and feature extraction, the dataset size was:

- Raw windows: 16,883 segments of shape $[150, 8]$
- Feature vectors: 16,883 segments of shape $[36,]$

With data augmentation (axis rotations, time warping), the final training set increased to **21,905** windows.

3 Features

3.1 Raw Sensor Features

The app directly processes smartphone **accelerometer** and **gyroscope** data. For each 3-second window (150 samples at 50 Hz), 6 base channels were collected:

- Linear acceleration: a_x, a_y, a_z
- Angular velocity: g_x, g_y, g_z

From these, two additional derived channels are computed:

- Acceleration magnitude: $\sqrt{a_x^2 + a_y^2 + a_z^2}$
- Gyroscope magnitude: $\sqrt{g_x^2 + g_y^2 + g_z^2}$

This yields an **8-channel raw input** sequence per window. The raw window is flattened and normalized using dataset-specific means and standard deviations (implemented in the `Scaler` module).

3.2 Engineered Statistical Features

To complement the raw temporal patterns, extract **36 statistical features** were extracted, per window (implemented in the `FeatureExtractor` module):

- For each of the 8 channels, compute: Mean, Standard deviation, Minimum, Maximum $\Rightarrow 8 \times 4 = 32$ features
- Jerk statistics (derivative of acceleration magnitude, approximating instantaneous force): Mean, Standard deviation, Minimum, Maximum $\Rightarrow +4$ features

Together, these form a **36-dimensional feature vector** that captures distributional properties of motion in each time window.

3.3 Normalization

All raw and statistical features are standardized via precomputed mean and standard deviation vectors (`raw_means.npy`, `raw_stds.npy`, `feat_means.npy`, `feat_stds.npy`). This ensures consistent scaling across datasets and subjects, improving stability during inference.

3.4 Hybrid Feature Design

Hybrid feature representation:

- Raw windows retain fine-grained temporal signatures, enabling the model to detect subtle activity transitions (e.g., distinguishing jogging vs. running).
- Engineered statistics provide compact descriptors that are robust to sensor noise, phone placement, and orientation.

By fusing these two representations, the model achieves higher generalization across datasets and subjects compared to either raw-only or feature-only approaches.

4 Model

4.1 Hybrid Architecture

The proposed model adopts a **hybrid design** that fuses deep temporal modeling of raw sensor signals with compact engineered features.

- **Raw stream encoder (1D CNN):** A stack of one-dimensional convolutional layers processes the accelerometer and gyroscope windows of shape $[150, 8]$ (time steps $\times$ channels). Convolutions slide along the temporal axis, learning filters that respond to characteristic motion patterns (e.g., periodic oscillations, bursts of angular velocity, or steady low-variance segments).
- **Feature encoder (MLP):** A lightweight multilayer perceptron (MLP) processes the 36-dimensional statistical feature vector. This pathway captures distributional characteristics such as overall variance, extremes, and jerk dynamics.
- **Fusion layer:** Outputs from both encoders are concatenated and passed through fully-connected layers with dropout regularization.
- **Output:** A softmax layer produces probabilities across the four MET classes: *Sedentary, Light, Moderate, Vigorous*.

This dual-pathway structure ensures that both fine-grained motion details and robust statistical summaries contribute to classification.

4.2 Why CNNs for Time-Series?

Although convolutional neural networks (CNNs) are most famous for image and text tasks, but they can also be used for a task such as **human activity recognition (HAR)**, from sensor time-series. The rationale is:

1. **Sequential nature of sensor data.** Accelerometer and gyroscope signals are streams of values over time. A fixed-length window (e.g. 150 samples $\times$ 8 channels) can be treated as a two-dimensional array: *time steps* $\times$ *sensor channels*. CNN filters scan across the temporal dimension, detecting local motifs such as oscillations in acceleration (walking), high-frequency bursts (running), or flat low-energy segments (sitting/lying).

2. **Automatic feature learning.** Handcrafted features (mean, std, min, max) are informative but discard much of the temporal structure. CNNs learn filters that may capture subtle frequency or shape signatures that manual features overlook (e.g. a stair-climbing rhythm detectable in vertical acceleration plus angular velocity).

3. **Efficiency.** Compared to recurrent models (LSTM/GRU), 1D CNNs are more computationally efficient, require fewer parameters, and are easier to deploy on smartphones. Convolution operations are highly optimized in ONNX Runtime and similar libraries.

4. **Empirical validation.** Numerous HAR studies on datasets such as WISDM, MotionSense, and UCI HAR report strong performance of CNNs on raw accelerometer and gyroscope windows, often outperforming purely feature-based or classical ML approaches.

Thus, CNNs are a natural fit for raw inertial sensor data: they capture short-term temporal dependencies (e.g. step cycles, arm swings) efficiently, while the statistical feature pathway provides additional robustness.

4.3 Baseline Alternatives

During model exploration, several candidate approaches were considered:

- **Classical ML on handcrafted features.** Models such as random forests, gradient boosting, or SVMs trained solely on statistical features (mean, std, min, max). These approaches are lightweight and interpretable but discard much of the temporal structure in the raw signals.

- **CNN-only on raw windows.** One-dimensional convolutional networks directly trained on $[150, 8]$ sensor windows. These models capture temporal patterns well but can be sensitive to orientation changes and noisy segments without handcrafted features.

- **Recurrent models (LSTM/GRU).** Recurrent networks are effective for capturing long-term dependencies across windows. However, they are computationally expensive, harder to optimize, as the computational resources (my 7 year old Intel-chip i5 based Macbook Pro) were very limited.

4.4 Scientific Motivation

The hybrid model also aligns with the scientific context since CNNs detect fine-grained kinematic patterns in accelerometer and gyroscope signals that correspond to movement intensity. Statistical features summarize variability and magnitude, connecting directly to physiological concepts of activity intensity (e.g. higher variance and jerk associated with vigorous exercise).

4.5 Training Setup

The model was trained on combined datasets (WISDM, MotionSense, UCI-HAR) resampled to 50 Hz and segmented into 3-second windows with 50% overlap. Details include:

- **Optimizer:** Adam with initial learning rate 10^{-3}.

- **Batch size:** 256 windows per update.
- **Loss function:** Sparse categorical cross-entropy.
- **Regularization:** Dropout layers (0.3) and early stopping based on validation loss (patience = 3).
- **Augmentations:** Random rotation of sensor axes (±60°), time warping (0.9–1.1× stretch), and balanced sampling across classes.
- **Training duration:** Up to 30 epochs per run, with early stopping typically halting earlier.

5 Evaluation

5.1 Experimental Setup

Evaluation was conducted using a combined dataset of WISDM, MotionSense, and UCI-HAR, with subject-wise partitioning to avoid overlap between training and test participants. Data were windowed into 3-second segments (150 samples at 50 Hz) with 50% overlap. A stratified split reserved **20% of subjects** for testing, ensuring representative coverage of all four MET classes.

The evaluation protocol trained the model **five times** with different random seeds. Metrics from each run were recorded, and final results are reported as the mean and standard deviation across runs.

5.2 Results

The hybrid CNN+MLP model achieved consistently strong performance across all runs. The averaged results (mean ± std) are summarized below:

Metric	Accuracy	Precision	Recall	F1-score
Mean ± Std	95.68% ± 0.90	95.77% ± 0.77	95.68% ± 0.90	95.68% ± 0.87

5.3 Interpretation

The evaluation demonstrates that the hybrid approach generalizes well across subjects with high accuracy and stability. Confusions were not often; most errors occurred between *Light* and *Moderate* activities, which is expected due to their physiological similarity (e.g., brisk walking vs. stair climbing). Sedentary and vigorous activities were most reliably detected.

6 Scientific Rationale

Metabolic Equivalent of Task (MET) classes provide a standardized measure of activity intensity. Classifying daily activity into *Sedentary, Light, Moderate,* and *Vigorous* is useful because:

- **Health relevance:** Time spent in each MET category is a strong predictor of chronic disease risk, physical fitness, and overall mortality.
- **Practical tracking:** Summarizing activity into MET classes offers a clear, interpretable overview of daily behavior, in contrast to raw step counts or uninterpreted motion signals.
- **Scientific validity:** MET categories align with established physiological benchmarks (e.g. < 1.5 METs for sedentary, > 6 METs for vigorous activity). This makes the app's outputs directly comparable to existing literature and public health guidelines.

By implementing a real-time, on-device classifier, the ADAMMA system provides both immediate feedback to the user and reproducible data for scientific and clinical applications.

References

Kwapisz, J. R., Weiss, G. M. & Moore, S. A. (2010), Activity recognition using cell phone accelerometers, *in* 'Proceedings of the Fourth International Workshop on Knowledge Discovery from Sensor Data (SensorKDD-10) at KDD-10', Washington, DC, USA.

Malekzadeh, M., Clegg, R. G., Cavallaro, A. & Haddadi, H. (2019), Mobile sensor data anonymization, *in* 'Proceedings of the International Conference on Internet of Things Design and Implementation', IoTDI '19, ACM, New York, NY, USA, pp. 49–58.
URL: *http://doi.acm.org/10.1145/3302505.3310068*

Reyes-Ortiz, J. L., Anguita, D., Ghio, A., Oneto, L. & Parra, X. (2013), 'Human activity recognition using smartphones', UCI Machine Learning Repository. `https://doi.org/10.24432/C54S4K`.