Capture-24 Human Activity Recognition

Project Summary

This project develops a deep learning pipeline to classify physical activities from the **Capture-24 dataset**. It forms part of the MOP Capstone – Health Behaviour Analysis (AI+IoT stream), contributing to **SDG 3: Good Health and Wellbeing** by demonstrating how wearable sensor data can support health monitoring.

The workflow included preprocessing smartphone accelerometer signals, segmenting them into fixed windows, encoding activity labels, and training a **CNN–BiLSTM hybrid model**. The final system outputs predicted activity labels with confidence scores, supported by a deployment-ready **Streamlit demo application**.

Pipeline Overview

Data Preprocessing

- **Signal Cleaning:** Removed invalid rows, corrected stray characters, and converted values to numeric format.
- **Segmentation:** Applied overlapping sliding windows (200 timesteps per window) to preserve temporal dynamics.
- **Normalisation:** Standardized sensor channels (x, y, z) to zero mean and unit variance.
- **Label Encoding:** Converted activity annotations (walking, jogging, sitting, standing, climbing stairs) into integers for model training.

Final Input Shapes:

- X_seq: (N, 200, 3) accelerometer windows.
- y: one-hot encoded activity labels.

Model Architecture

- **CNN Layers:** Extracted local motion features from accelerometer signals.
- **BiLSTM Layers:** Captured sequential dependencies across timesteps.
- **Dense Fusion:** Fully-connected layers integrated learned features.
- **Output:** Softmax layer for multi-class activity classification.
- **Loss Function:** Categorical Crossentropy.
- **Metrics:** Accuracy, Precision, Recall, Macro/Weighted F1.

Training Details

- **Split:** 70/15/15 train/validation/test.
- **Regularisation:** Dropout, batch normalization.
- **Callbacks:** Early stopping and model checkpointing.
- **Visualisation:** Training/validation accuracy and loss plots.
- **Sanity Checks:** Verified input shapes, ensured no NaNs in processed data.

Key Insights

- **Performance:** High accuracy on well-represented activities (e.g., walking, jogging); slightly lower for overlapping classes (e.g., sitting vs. standing).
- **Confusion Matrix:** Highlighted class-specific strengths and misclassifications.
- **Confidence Analysis:** Correct predictions showed higher probability scores than misclassifications.
- **Calibration:** Reliability diagrams showed overconfidence, improved with temperature scaling.

Contributors

Bhavithra Senthil Kumar (S224582135) – Data preprocessing, CNN–BiLSTM model development, evaluation, calibration, Streamlit demo.
Team: MOP Capstone "Chameleon" Health Behaviour Subgroup

- Rayudu, Vinitha, Natakkan, Alen, Ajay, Devin, Rishith.

Recommendations

- Apply augmentation to balance under-represented activities.
- Test Transformer-based sequential models for richer temporal understanding.
- Incorporate additional public datasets (e.g., UCI HAR, SHL) to strengthen generalisation.
- Deploy on mobile devices for **real-time activity recognition** to support digital health and behaviour monitoring.

Capstone Project — Human Activity Recognition (HAR)

This repository contains my final capstone project notebook on Physical Activity Recognition. The project applies machine learning techniques to identify and classify human activities from sensor data.

The workflow was designed to follow a structured approach — from data exploration and preprocessing to model building, evaluation, optimization, and interpretation. The aim was not just to achieve accuracy, but also to ensure fairness, reproducibility, and meaningful insights into which features drive activity recognition.

Repository Contents

Physical_Activity_Recognition_Vinitha.ipynb
The main notebook includes:

Step 1: Data Exploration & Preprocessing

- Handling missing values, encoding activity labels, and scaling features. Stratified train-test splitting to ensure balanced activity distribution.

Step 2: Baseline Model Development

- Implemented RandomForestClassifier as the primary model. Established baseline accuracy and interpretability.

Step 3: Hyperparameter Tuning

- Used GridSearchCV for parameter optimization and Improved performance and reduced overfitting.

Step 4: Model Evaluation & Insights

- Confusion matrix, precision, recall, F1-score. Feature importance ranking for interpretability.

Step 5: Finalization & Saving

- Exported trained model and reports for reuse. Documented findings and reflection for future improvements.
- **README.md** (this file) has Project overview, methodology, and repository guide.

How to Run

Open in Colab or locally.

Colab recommended. If local, ensure Python 3.9+ with pandas, numpy, scikit-learn, matplotlib, and seaborn.

Point to the dataset.

- In the load cell, update the CSV path (e.g., file_path = "/content/train.csv" or your local path).

Run cells top → bottom.

- Sections: Setup → EDA/cleaning → encoding/scaling → split → model train & GridSearch → evaluation → artifacts/report.

(Optional) Tweak config.

- Drop columns (e.g., subject), change test split size, or adjust the GridSearch parameter grid.
- A **quality gate** is applied with MIN_F1—raise/lower it to match your bar.

Find outputs.

- Trained model, metrics, confusion matrix, and feature-importance plots are saved under artifacts/week8 and artifacts/week9 (includes an HTML summary report).

Project Overview:

Goal: Build a practical Human Activity Recognition (HAR) classifier from wearable/sensor data that is accurate, interpretable, and easy to re-run.

Data & Prep:

- Label-encoded Activity, dropped subject to avoid leakage, scaled features, and used stratified train/test splits. Quick EDA for shape, types, and missing values.

Modelling:

- Strong baseline with RandomForestClassifier. GridSearchCV tuned depth, estimators, and split/leaf parameters. A "light" RF variant and simple latency check for efficiency.

Evaluation:

- Accuracy + macro-F1, classification report, and a confusion matrix to surface class-wise errors. Feature importance plot to see which signals drive predictions.

Artifacts & Reporting:

- Saves best_model.joblib, metrics_*.json, confusion-matrix and feature-importance PNGs, and a Week-9 HTML report. A quality gate (MIN_F1) fails the run if performance dips below the threshold.

What to try next:

- Subject-wise splits for realism, compact models or gradient boosting for speed/accuracy trade-offs, and robustness tests (noise/missing values).

Reflections

Through this project, I learned:

- **Random Forests are reliable baselines** – they capture patterns well without heavy tuning, but can be resource-intensive.
- **Feature scaling and stratified splits are key** – these steps ensured fairness across activity classes and prevented bias.

- **GridSearchCV improves robustness** – tuning hyperparameters systematically gave a noticeable boost in balanced accuracy.
- **Feature importance offers clarity** – seeing which signals contribute most improved interpretability and trust in results.
- **Saving models and reports ensures reproducibility** – good practice for teamwork and future deployment.

Next Steps

- **Explore hybrid models** – try combining tree-based methods with temporal models (e.g., RandomForest + LSTM).
- **Subject-wise testing** – check generalization by splitting data across individuals, not just random splits.
- **Optimize for efficiency** – use pruning, feature selection, or lightweight models (e.g., XGBoost, LightGBM) for faster deployment.
- **Robustness checks** – test the model on noisy, imbalanced, or real-world activity data.
- **Future scope** – extend beyond the current dataset to broader physical activity contexts for stronger generalization.