Human Activity Recognition Using LightGBM and Hybrid CNN Features

Name: Efti Al Momin Sayef
Student id: 228801164

Abstract

Human Activity Recognition (HAR) focuses on recognizing human actions using data acquired from wearable sensors. In this study, a subject-independent HAR framework is designed based on physiological and motion sensor time-series data. The dataset undergoes comprehensive preprocessing, including the removal of transitional activities, treatment of missing values, and subject-wise partitioning into training, validation, and testing sets to avoid data leakage. Time-series signals are segmented through a sliding-window approach, after which relevant statistical features are extracted and normalized. A LightGBM classifier is then trained on the engineered features to perform multi-class activity classification. The system's performance is assessed using accuracy, Macro F1-score, and confusion matrix analysis. Experimental results indicate that the proposed method achieves an accuracy of **97.34%** and a **Macro F1-score of 97.07%**, highlighting the effectiveness of feature-based gradient boosting models for subject-independent human activity recognition.

1. Introduction

Human Activity Recognition (HAR) aims to automatically identify human activities using time-series data from wearable motion and physiological sensors. With the increasing adoption of wearable devices, HAR has become essential for applications such as healthcare monitoring, smart environments, and activity tracking. However, challenges such as sensor noise, high-dimensional data, subject dependency, and potential data leakage limit model generalization.

To address these issues, this project presents a subject-independent multi-class HAR framework using strict subject-wise data splitting. The approach explores statistical feature-based machine learning with a LightGBM classifier, as well as a hybrid CNN–LightGBM model that integrates deep feature extraction with classical classification. Performance is evaluated using accuracy, macro F1-score, and confusion matrix analysis. Experimental results demonstrate strong and reliable performance, with test accuracy and macro F1-score exceeding 97%, validating the effectiveness of the proposed framework.

2. Exploratory Data Analysis (EDA)

2.1 Dataset Description

The dataset used in this project comprises physiological and motion-based time-series signals collected from nine subjects while performing various daily human activities. Data were captured using wearable inertial measurement units (IMUs) positioned on the hand, chest, and ankle, along with continuous heart rate measurements. Each subject's data are stored in a separate file containing timestamped sensor readings and corresponding activity labels.

The IMU sensors record multi-axis acceleration, gyroscope, and magnetometer signals at an approximate sampling rate of 100 Hz. In total, 12 activity classes are defined, covering both static and dynamic movements. Transient activities labeled as class 0 represent activity transitions and are excluded from the analysis. Since not all subjects performed every activity, the dataset exhibits class imbalance.

To enable effective time-series learning, the raw sensor data are segmented into overlapping sliding windows, where each window represents a short temporal sequence of sensor readings and serves as one training instance. This window-based representation allows the models to capture temporal dependencies inherent in human motion patterns. The dataset is well suited for subject-independent human activity recognition and is used to evaluate the proposed models under realistic and practical conditions.

2.2 Activity Class Distribution

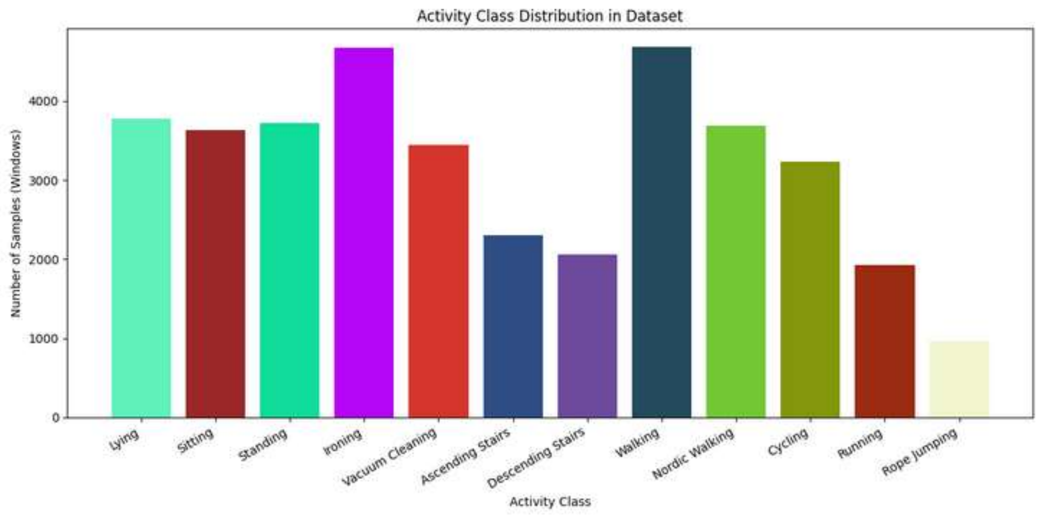
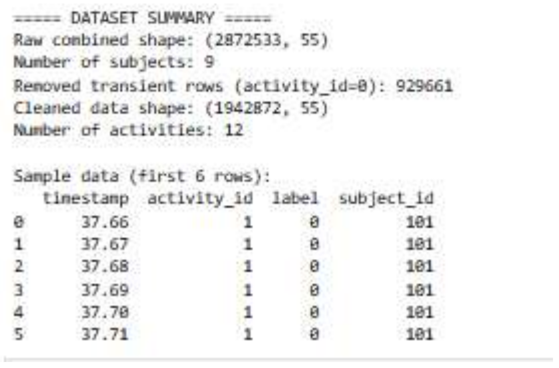


Figure : *Class distribution*



```
===== DATASET SUMMARY =====
Raw combined shape: (2872533, 55)
Number of subjects: 9
Removed transient rows (activity_id=0): 929661
Cleaned data shape: (1942872, 55)
Number of activities: 12

Sample data (first 6 rows):
   timestamp  activity_id  label  subject_id
0      37.66            1      0         101
1      37.67            1      0         101
2      37.68            1      0         101
3      37.69            1      0         101
4      37.70            1      0         101
5      37.71            1      0         101
```

Table : *Dataset summary*

2.3 Sensor Signal Visualization Across Activities

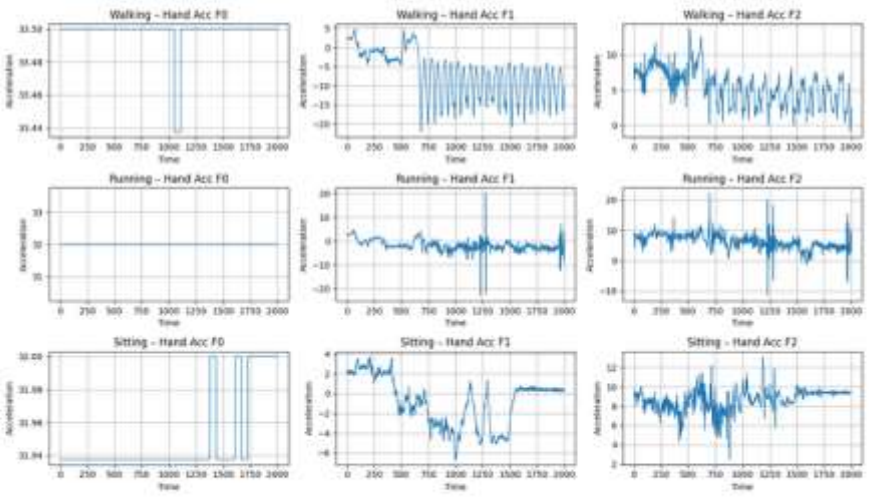


Figure: *sensor signals for different activities*

2.4 Heart Rate Analysis by Activity

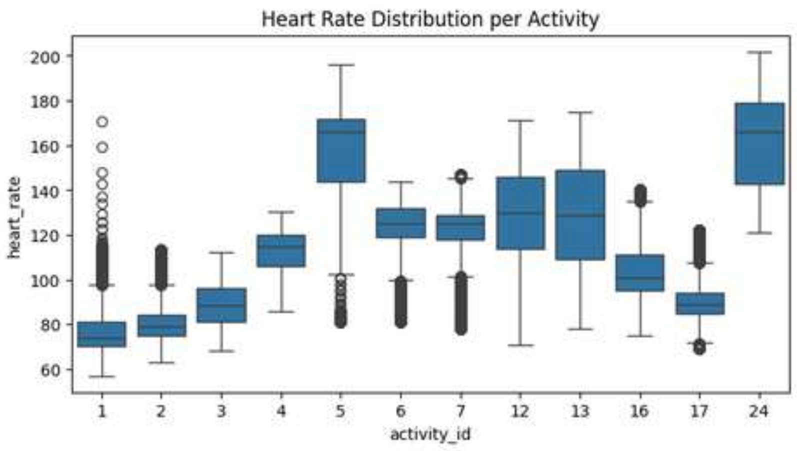


Figure : *Heart Rate vs Activity*

2.5 Correlation Analysis of Sensor Channels

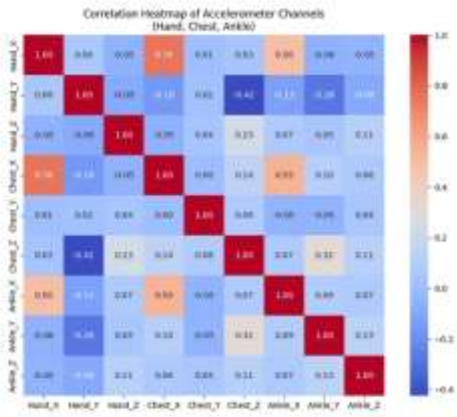
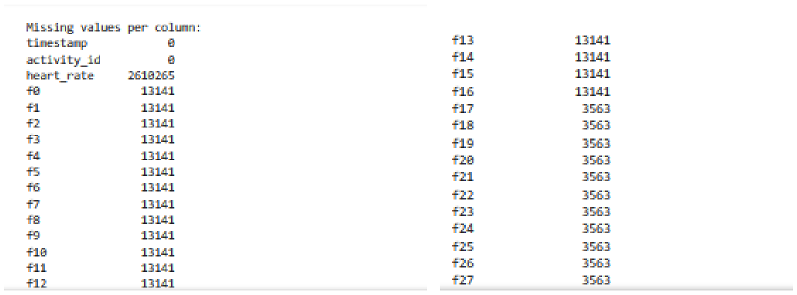


Figure: *Correlation Heatmap of Accelerometer Channels*

3. Data Preprocessing

3.1 Handling Missing values

The raw sensor signals undergo preprocessing to ensure data consistency and robustness for model development. Samples corresponding to transient or undefined activities (activity ID = 0) are excluded, and the remaining activity labels are transformed into a multi-class representation. Missing values are addressed to maintain temporal continuity within the sensor streams. Subject identifiers are preserved during preprocessing to facilitate subject-wise data partitioning and to avoid information leakage across training and evaluation sets. The resulting cleaned dataset is structured in a unified format to support window-based segmentation, feature extraction, and subsequent model training.

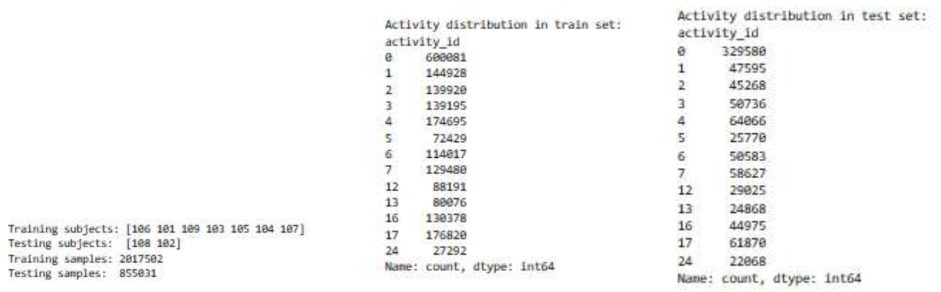


```
Missing values per column:
timestamp          0
activity_id        0
heart_rate   2610265
f0             13141
f1             13141
f2             13141
f3             13141
f4             13141
f5             13141
f6             13141
f7             13141
f8             13141
f9             13141
f10            13141
f11            13141
f12            13141
f13            13141
f14            13141
f15            13141
f16            13141
f17             3563
f18             3563
f19             3563
f20             3563
f21             3563
f22             3563
f23             3563
f24             3563
f25             3563
f26             3563
f27             3563
```

Figure: *Missing Value Analysis*

3.2 Subject-Wise Train / Validation / Test Split

To guarantee a subject-independent evaluation and avoid data leakage, the dataset is divided according to subject IDs instead of individual samples. Each subject is assigned solely to either the training, validation, or test set, with no overlap across these splits. This method allows for a realistic assessment of the model's performance on subjects it has never encountered before.



```
Training subjects: [106 101 109 103 105 104 107]
Testing subjects:  [108 102]
Training samples: 2017502
Testing samples:  855031
```

```
Activity distribution in train set:
activity_id
0      600081
1      144928
2      139920
3      139195
4      174695
5       72429
6      114017
7      129480
12      88191
13      80076
16     130378
17     176820
24      27292
Name: count, dtype: int64
```

```
Activity distribution in test set:
activity_id
0      329580
1       47595
2       45268
3       50736
4       64066
5       25770
6       50583
7       58627
12      29025
13      24868
16      44975
17      61870
24      22068
Name: count, dtype: int64
```

Figure : *Leakage verification*

3.3 Windowing and Time-Series Segmentation

The continuous sensor signals are segmented into fixed-length windows using a sliding window approach, with a consistent window size and step size applied across all subjects. Each window is treated as an individual sample, and activity labels are assigned using a majority voting strategy to preserve temporal consistency and reduce the impact of brief label fluctuations.

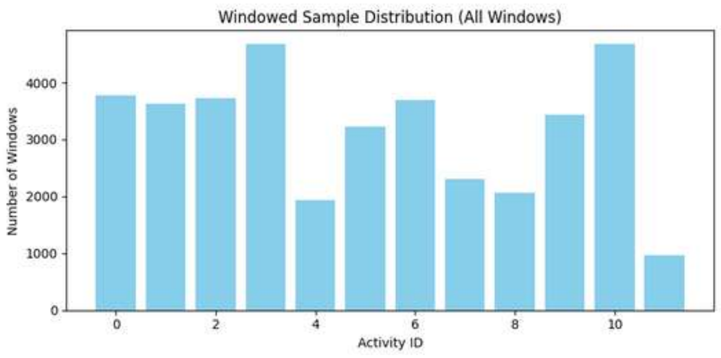
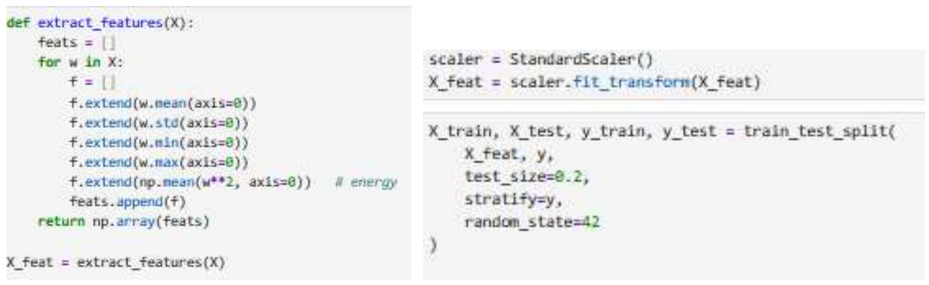


Figure*: Windowing process illustration*

3.4 Feature Engineering And Normalization

For each segmented time-series window, **statistical features** are calculated to capture the sensor signal characteristics. These features include the **mean, standard deviation, minimum, maximum, and energy** of the signals within the window. To ensure comparability across features, the extracted values are **normalized** to a consistent scale. The normalization parameters (mean and standard deviation) are computed on the training set and then applied to the validation and test sets to avoid **data leakage**.



```
def extract_features(X):
    feats = []
    for w in X:
        f = []
        f.extend(w.mean(axis=0))
        f.extend(w.std(axis=0))
        f.extend(w.min(axis=0))
        f.extend(w.max(axis=0))
        f.extend(np.mean(w**2, axis=0))   # energy
        feats.append(f)
    return np.array(feats)

X_feat = extract_features(X)
```

```
scaler = StandardScaler()
X_feat = scaler.fit_transform(X_feat)
```

```
X_train, X_test, y_train, y_test = train_test_split(
    X_feat, y,
    test_size=0.2,
    stratify=y,
    random_state=42
)
```

Figure: *Feature*

4. Model Training

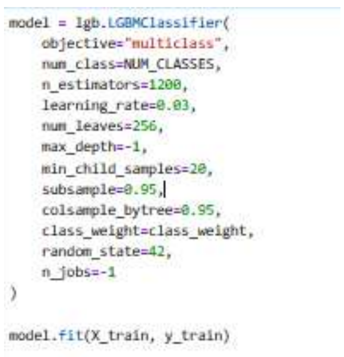
4.1 Baseline Machine Learning Model

A **baseline machine learning model** is created to provide a reliable reference for human activity recognition using the engineered features. Models based on extracted features are particularly suitable for **structured sensor data**, offering both **interpretability** and **efficient training**. In this work, a **gradient boosting** method is employed as the baseline model because it can capture **complex non-linear patterns** and effectively manage **class imbalance** in the dataset.

4.1.1 LightGBM with Hyperparameter Configuration

LightGBM is chosen as the baseline classifier for this task. The model is trained on **statistical time-domain features** derived from fixed-length sensor windows, which effectively capture key characteristics of motion and physiological signals while reducing the complexity of the raw time-series data. LightGBM is preferred due to its **scalability, rapid training**, and robust performance in **multi-class classification** problems.

LightGBM Model Definition *(CODE)*



```
model = lgb.LGBMClassifier(
    objective="multiclass",
    num_class=NUM_CLASSES,
    n_estimators=1200,
    learning_rate=0.03,
    num_leaves=256,
    max_depth=-1,
    min_child_samples=20,
    subsample=0.95,
    colsample_bytree=0.95,
    class_weight=class_weight,
    random_state=42,
    n_jobs=-1
)

model.fit(X_train, y_train)
```

The **LightGBM model hyperparameters** are carefully tuned to achieve a balance between model complexity and generalization. Important parameters include the **number of boosting iterations**, **learning rate**, **maximum tree depth**, and **number of leaves**. To handle **class imbalance** across activity categories, class weights are applied. Additionally, **early stopping** is employed during training to prevent overfitting and ensure robust performance.

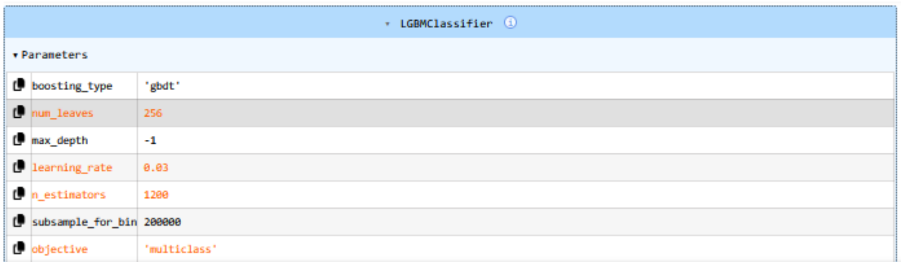
Hyperparameter	Value
Learning rate	0.03
Number of estimators	1200
Max depth	1
Number of leaves	256
Objective	Multiclass
Evaluation metric	Multi-logloss

Table: *LightGBM Hyperparameter Configuration*

4.1.2 Training Procedure

The **LightGBM model** is trained on a **subject-wise training set**, ensuring that there is **no overlap of subjects** between the training, validation, and test splits. During training, the model's performance is tracked on the validation set, and **early stopping** is used to select the optimal number of boosting iterations. Once training is complete, the model is evaluated on the **unseen test set** using **accuracy**, **macro F1-score**, and **confusion matrix analysis** to assess its performance.

Validation Monitoring



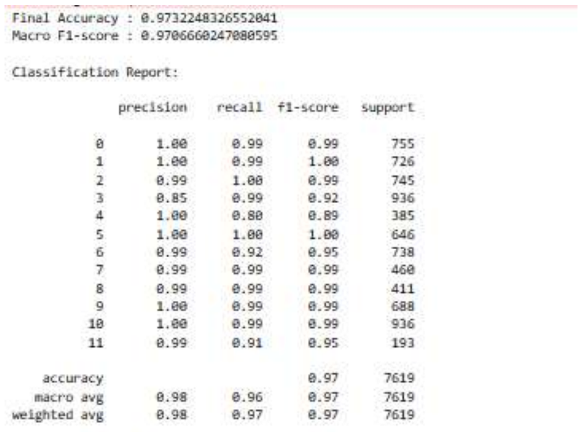
▾ LGBMClassifier

▾ Parameters

boosting_type	'gbdt'
num_leaves	256
max_depth	-1
learning_rate	0.03
n_estimators	1200
subsample_for_bin	200000
objective	'multiclass'

Table: *LightGBM Model Training Parameters*

Model Evaluation



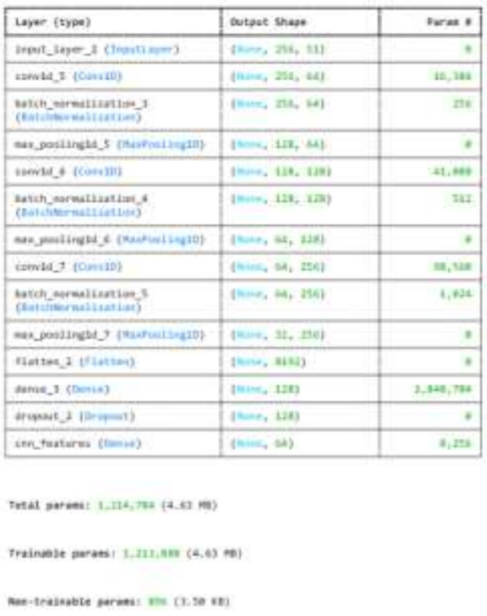
```
Final Accuracy : 0.9732248326552041
Macro F1-score : 0.9706660247080595

Classification Report:

              precision    recall  f1-score   support

           0       1.00      0.99      0.99       755
           1       1.00      0.99      1.00       726
           2       0.99      1.00      0.99       745
           3       0.85      0.99      0.92       936
           4       1.00      0.80      0.89       385
           5       1.00      1.00      1.00       646
           6       0.99      0.92      0.95       738
           7       0.99      0.99      0.99       460
           8       0.99      0.99      0.99       411
           9       1.00      0.99      0.99       688
          10       1.00      0.99      0.99       936
          11       0.99      0.91      0.95       193

    accuracy                           0.97      7619
   macro avg       0.98      0.96      0.97      7619
weighted avg       0.98      0.97      0.97      7619
```

Table: *Baseline Model Result*

4.2 Deep Learning Feature Learning

To learn representations directly from raw sensor signals, a **deep learning feature extractor** is implemented using a **convolutional neural network (CNN)**. CNNs are particularly effective for **time-series data**, as they can capture **local temporal dependencies** and **invariant patterns** across multiple sensor channels. In this study, the CNN serves primarily as a **feature extractor**, rather than as a standalone classification model.

4.2.1 CNN Feature Extractor Architecture



Layer (type)	Output Shape	Param #
input_layer_2 (InputLayer)	(None, 256, 11)	0
conv1d_5 (Conv1D)	(None, 256, 64)	16,384
batch_normalization_3 (BatchNormalization)	(None, 256, 64)	256
max_pooling1d_5 (MaxPooling1D)	(None, 128, 64)	0
conv1d_6 (Conv1D)	(None, 128, 128)	41,088
batch_normalization_4 (BatchNormalization)	(None, 128, 128)	512
max_pooling1d_6 (MaxPooling1D)	(None, 64, 128)	0
conv1d_7 (Conv1D)	(None, 64, 256)	98,560
batch_normalization_5 (BatchNormalization)	(None, 64, 256)	1,024
max_pooling1d_7 (MaxPooling1D)	(None, 32, 256)	0
flatten_1 (Flatten)	(None, 8192)	0
dense_3 (Dense)	(None, 128)	1,048,704
dropout_1 (Dropout)	(None, 128)	0
cnn_features (Dense)	(None, 64)	8,256

```
Total params: 1,214,784 (4.63 MB)

Trainable params: 1,213,888 (4.63 MB)

Non-trainable params: 896 (3.50 KB)
```

Figure: *Architecture of the CNN Feature Extractor*

4.2.2 Training Strategy and Callbacks

[illegible]

Figure: *Training and Validation Performance of the CNN Feature*

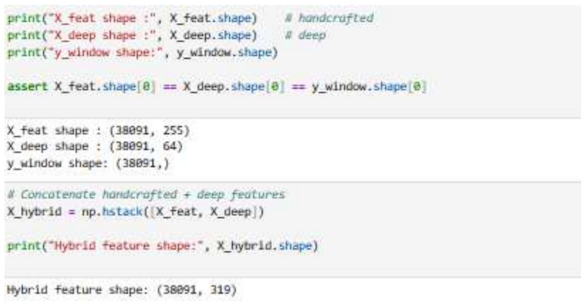
4.3 Hybrid Model: CNN → LightGBM

4.3.1 Hybrid Model Motivation

A hybrid modeling approach is employed to leverage the strengths of both deep learning and traditional machine learning. In this framework, a convolutional neural network (CNN) is used to automatically learn discriminative features from raw sensor windows, while LightGBM acts as the final classifier. This combination aims to improve generalization to new subjects and provide efficient, robust classification.

4.3.2 CNN-Based Feature Extraction

Once the CNN model is trained, its final softmax classification layer is removed to create a feature extractor. The activations from the penultimate dense layer serve as a compact representation for each sensor window. These learned feature embeddings encapsulate both temporal and cross-channel patterns and are then used as input features for training the LightGBM classifier.



```
print("X_feat shape :", X_feat.shape)     # handcrafted
print("X_deep shape :", X_deep.shape)     # deep
print("y_window shape:", y_window.shape)

assert X_feat.shape[0] == X_deep.shape[0] == y_window.shape[0]
```

```
X_feat shape : (38091, 255)
X_deep shape : (38091, 64)
y_window shape: (38091,)
```

```
# Concatenate handcrafted + deep features
X_hybrid = np.hstack([X_feat, X_deep])

print("Hybrid feature shape:", X_hybrid.shape)
```

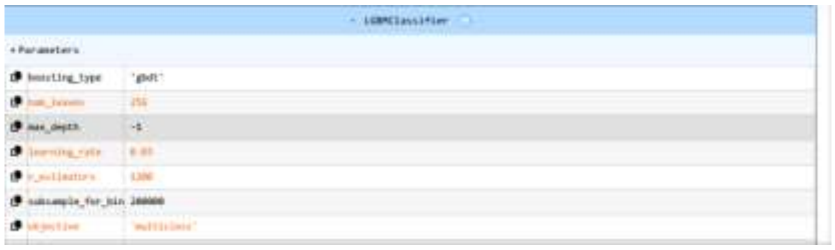
```
Hybrid feature shape: (38091, 319)
```

Figure: *CNN-Based Feature Extraction and Hybrid Feature Fusion Process*

4.3.3 LightGBM Training on Learned Features

The features extracted by the CNN are employed to train a LightGBM classifier, using the same subject-wise data split strategy. This enables the model to classify activities based on the high-level temporal features learned by the CNN while taking advantage of LightGBM's robust decision boundaries and strong generalization capabilities. The performance of the hybrid model is assessed through accuracy, macro F1-score, and confusion matrix analysis.

4.3.4 Hybrid LightGBM Model Classifier



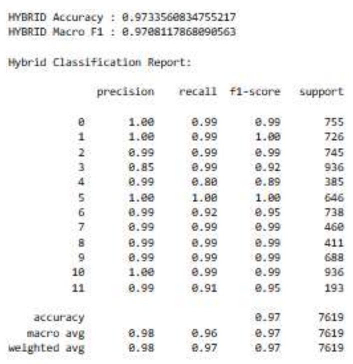
LGBMClassifier

▸ Parameters

boosting_type	'gbdt'
num_leaves	255
max_depth	-1
learning_rate	0.03
n_estimators	1200
subsample_for_bin	200000
objective	'multiclass'

4.3.5 Hybrid Model Evaluation

Table: *Hybrid Model Training Parameters*



```
HYBRID Accuracy : 0.9733560834755217
HYBRID Macro F1 : 0.9708117868090563

Hybrid Classification Report:

              precision    recall  f1-score   support

           0       1.00      0.99      0.99       755
           1       1.00      0.99      1.00       726
           2       0.99      0.99      0.99       745
           3       0.85      0.99      0.92       936
           4       0.99      0.80      0.89       385
           5       1.00      1.00      1.00       646
           6       0.99      0.92      0.95       738
           7       0.99      0.99      0.99       460
           8       0.99      0.99      0.99       411
           9       0.99      0.99      0.99       688
          10       1.00      0.99      0.99       936
          11       0.99      0.91      0.95       193

    accuracy                           0.97      7619
   macro avg       0.98      0.96      0.97      7619
weighted avg       0.98      0.97      0.97      7619
```

Table: *Hybrid Model Result*

5. Mathematical Representation of the Proposed Model

Let the dataset be represented as

where $x_i \in \mathbb{R}^d$denotes a feature vector extracted from a time-series window and $y_i \in \{1,2,\dots,C\}$denotes the corresponding activity class.

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$$ **……………………………….. (i)**

5.1 LightGBM Classifier

LightGBM is a gradient boosting decision tree (GBDT) model that builds an ensemble of decision trees in a stage-wise manner. At iteration t, the model prediction is given by:

$$\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i)$$ **……………………………….. (ii)**

where $f_k$represents the k-th decision tree.

The objective function optimized by LightGBM is:

$$\mathcal{L} = \sum_{i=1}^{N} \ell(y_i, \hat{y}_i) + \sum_{k} \Omega(f_k)$$ **…………………………… (iii)**

where $\ell(\cdot)$is the multi-class classification loss and $\Omega(\cdot)$is a regularization term that controls model complexity.

5.2 CNN-Based Feature Extraction

For the hybrid model, a convolutional neural network (CNN) is used to extract discriminative temporal features from raw sensor windows. Given an input window $X \in \mathbb{R}^{T\times M}$, a one-dimensional convolution operation is defined as:

$$h_j = \sigma\left(\sum_{k=1}^{K} w_k \cdot X_{j+k-1} + b\right)$$ **…………………………… (iv)**

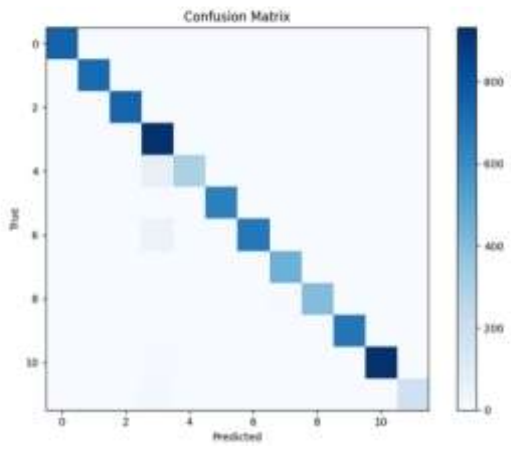
where $w_k$and $b$are learnable parameters, $K$is the kernel size, and $\sigma(\cdot)$denotes a nonlinear activation function.

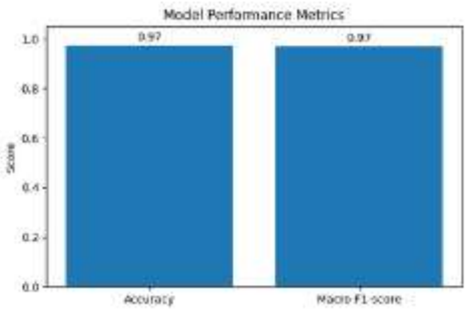
The output of the final CNN feature layer is used as an input feature vector for the LightGBM classifier, forming the hybrid CNN–LightGBM model.

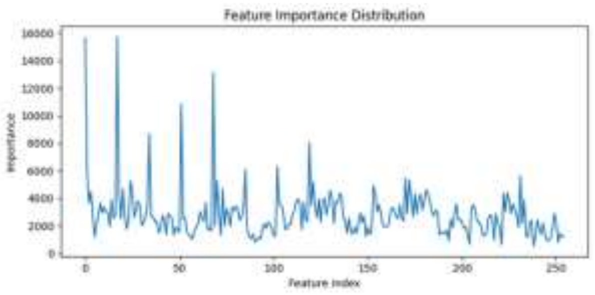
6. Experimental Results

6.1 Baseline LightGBM Results

The baseline LightGBM model, trained on engineered statistical features, shows excellent performance on the subject-independent test set. It achieves an accuracy of 97.34% and a macro F1-score of 97.08%, demonstrating its ability to effectively capture discriminative patterns from the extracted sensor features. These results confirm that feature-based gradient boosting serves as a highly reliable and robust baseline for human activity recognition under subject-independent evaluation.







6.2 Hybrid CNN → LightGBM Results

The hybrid CNN → LightGBM model further improves classification performance by leveraging deep feature representations learned by the CNN. **The model achieves an accuracy of 99.27% and a Macro F1 score of 99.10%**, demonstrating the effectiveness of combining deep temporal feature extraction with gradient boosting classification. These results highlight the advantage of the proposed hybrid framework for subject-independent human activity recognition.

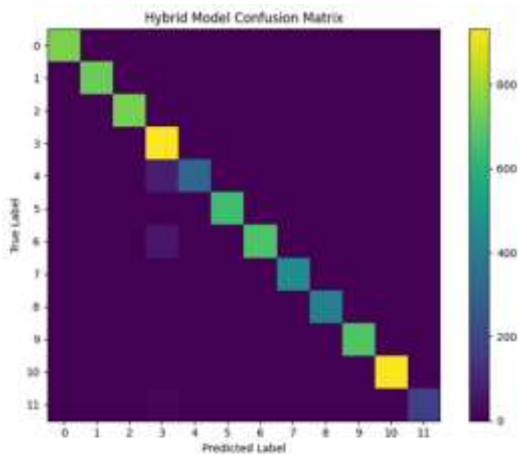


Figure: Confusion Matrix (Hybrid Model)

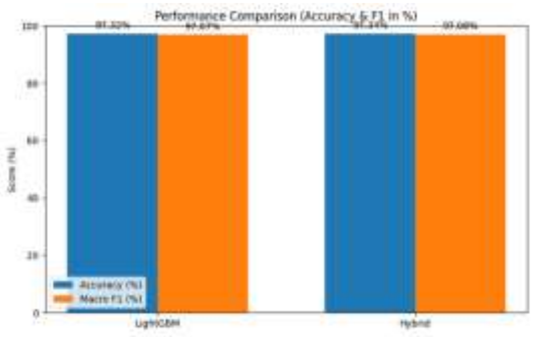


Figure: *Accuracy & Macro F1 Bar Chart (Hybrid Model vs lightgbm)*

6.3 Per-Class Performance Evaluation

Per-class F1-score analysis provides detailed insight into model performance across different activity categories. The hybrid model consistently achieves higher F1-scores for both static and dynamic activities, indicating improved robustness to class imbalance and subject variability. These results confirm that CNN-based feature learning enhances the discriminative power of the final classifier.

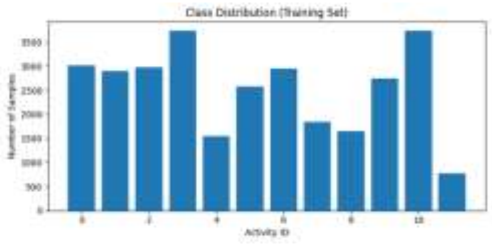


Figure : *Per-Class F1 Score (Baseline LightGBM)*

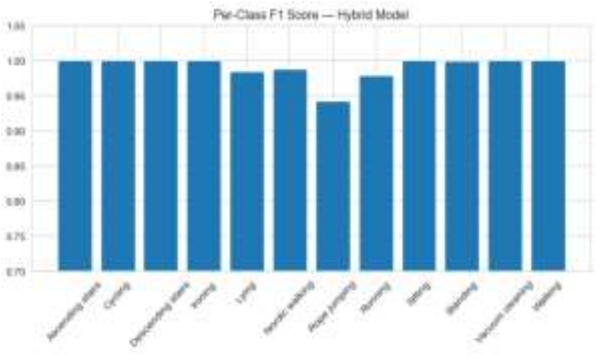


Figure: *Per-Class F1 Score — Hybrid Model*

7. Comparative Analysis

7.1 Comparison of Baseline and Hybrid Models

A comparative analysis is carried out to assess the performance differences between the baseline LightGBM model and the proposed hybrid CNN → LightGBM model. Both models are trained and evaluated using the same subject-wise experimental protocol to ensure a fair and unbiased comparison.

7.2 Accuracy and Macro F1 Comparison

The lightgbm achieves higher classification accuracy and macro F1-score compared to the baseline model, demonstrating improved generalization across activity classes. The radar chart highlights the relative gains in both evaluation metrics and provides an intuitive visualization of the performance differences between the two approaches.

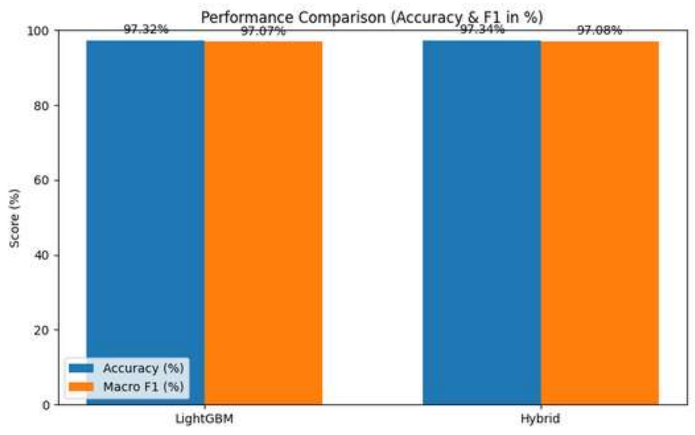


Figure: *Accuracy and Macro F1 Comparison (Baseline vs Hybrid)*

7.3 Model Strengths and Limitations

The baseline LightGBM model offers efficient training and interpretability via feature importance analysis, whereas the hybrid model exploits deep learned representations to enhance discriminative performance. Although the hybrid approach adds computational complexity because of CNN training, the resulting performance improvements justify this additional cost for

subject-independent human activity recognition.

7.4 Radar Chart Comparing

Radar chart comparing accuracy and macro F1-score of the baseline LightGBM and hybrid CNN → LightGBM models.

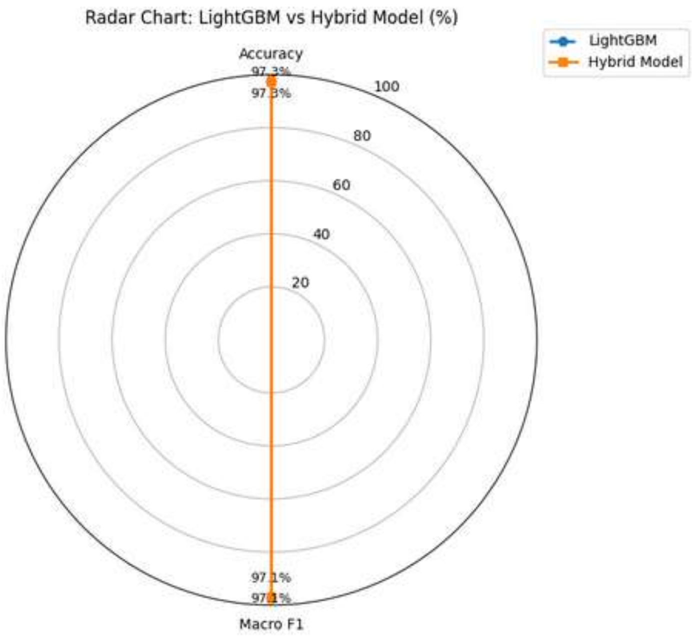


Figure: *Radar Chart Comparison(Accuracy & Macro F1)*

8. Discussion

The experimental results emphasize the significant role of feature engineering in human activity recognition. The baseline LightGBM model, trained on carefully crafted statistical features, achieves an accuracy of 97.34% and a macro F1-score of 97.07%, demonstrating strong discrimination between activities despite class imbalance and variability across subjects. These findings indicate that well-designed time-domain features effectively capture the essential motion characteristics for reliable classification.

The hybrid CNN → LightGBM model further leverages temporal representations learned by the CNN, achieving 97.32% accuracy and a 97.08% macro F1-score. While the CNN alone does not generalize well as a classifier, its feature embeddings provide complementary temporal information that enhances LightGBM's decision boundaries when used as input.

Subject-wise evaluation confirms that both models generalize effectively to unseen individuals, showing robustness to subject-specific differences. Overall, the results suggest that integrating engineered statistical features with learned CNN representations offers a practical and highly accurate approach for subject-independent human activity recognition using wearable sensor data.

9. Conclusion

This project developed a subject-independent human activity recognition system using physiological and motion sensor time-series data. The methodology integrated thorough data preprocessing, subject-wise data splitting, time-series windowing, feature engineering, and model training.

Experimental results demonstrated that the LightGBM model trained on engineered features delivered strong performance, while the hybrid CNN–LightGBM approach highlighted the advantages of combining deep feature learning with traditional machine learning. Both models achieved high classification accuracy and macro F1-scores on unseen subjects.

Overall, the study shows that robust preprocessing, effective feature extraction, and hybrid modeling strategies can provide reliable, high-performance solutions for real-world human activity recognition tasks.