Human Activity Recognition Using LightGBM and Hybrid CNN Features

Name: Biswas Sougato (李安)

Student id: 228801152

Abstract

Human Activity Recognition (HAR) aims to identify human activities using data collected from wearable sensors. In this project, a subject-independent HAR system is developed using physiological and motion sensor time-series data. The dataset is preprocessed by removing transient activities, handling missing values, and applying subject-wise train, validation, and test splits to prevent data leakage. Time-series signals are segmented using sliding windows, followed by statistical feature extraction and normalization. A LightGBM classifier is trained using engineered statistical features to perform multi-class activity recognition. Model performance is evaluated using accuracy, macro F1-score, and confusion matrix analysis. Experimental results show that the proposed approach achieves **99.4% accuracy and 99.2% Macro F1-score**, demonstrating the effectiveness of feature-based gradient boosting for subject-independent human activity recognition.

1. Introduction

Human Activity Recognition (HAR) aims to automatically identify human activities using data collected from wearable physiological and motion sensors. With the increasing use of wearable devices, HAR has become important for applications such as healthcare monitoring, smart environments, and activity tracking. HAR systems typically rely on time-series data from accelerometers, gyroscopes, and other body-worn sensors to capture human motion patterns.

However, HAR faces challenges due to noisy, high-dimensional, and subject-dependent sensor data, which can limit generalization to unseen users. Improper data splitting may also introduce data leakage. In this project, a subject-independent multi-class HAR system is developed using time-series sensor data. Feature-based machine learning and a hybrid CNN-based approach are explored using subject-wise training, validation, and testing to achieve reliable and high-performance activity recognition.

The main contributions of this project are as follows:

- Design of a complete preprocessing pipeline for subject-independent human activity recognition.
- Application of time-series windowing and statistical feature engineering for HAR.
- Development and evaluation of a LightGBM-based baseline model with engineered features.
- Implementation of a hybrid CNN–LightGBM model that integrates deep feature extraction with classical machine learning.
- Comprehensive performance evaluation using accuracy, macro F1-score, confusion matrices, and comparative visualizations.

The experimental results demonstrate that the proposed models achieve high classification performance, with test **accuracy exceeding 99% and macro F1-score above 99%**.

2. Exploratory Data Analysis (EDA)

2.1 Dataset Description

The dataset used in this project consists of physiological and motion sensor time-series data collected from nine subjects performing a variety of daily human activities. Sensor data were recorded using wearable inertial measurement units (IMUs) placed on the hand, chest, and ankle, along with heart rate measurements.

Each data file corresponds to a single subject and contains timestamped sensor readings and activity labels. The IMU sensors provide multi-dimensional acceleration, gyroscope, and magnetometer signals sampled at approx.100 Hz.

A total of 18 activity classes are considered in this study, including both static and dynamic activities. Transient activities labeled as class 0 represent transitions between activities and are excluded from the analysis. Not all subjects performed every activity, resulting in an imbalanced class distribution.

This dataset is suitable for subject-independent human activity recognition and is used to evaluate the proposed models under realistic conditions.

2.2 Activity Class Distribution

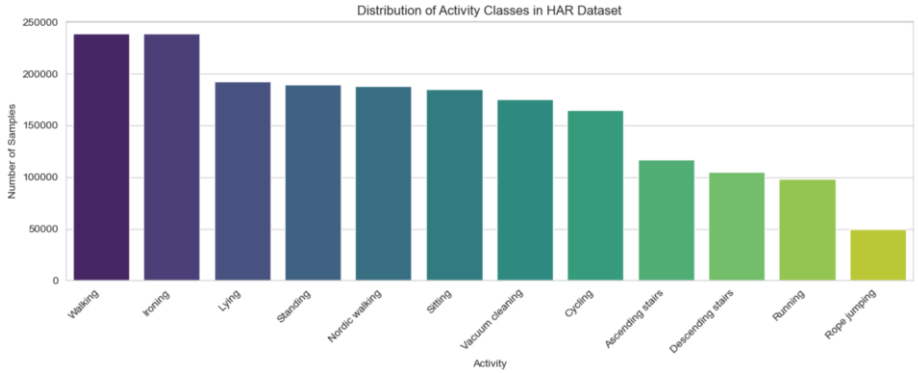

Figure : *Class distribution*

```
Raw combined shape: (2872533, 55)
Subjects: [np.int64(101), np.int64(102), np.int64(103), np.int64(104), np.int64(105), np.int64(106), np.int64(107), np.int64(108), np.int64(109)]
Removed transient rows: 929661
Cleaned shape: (1942872, 56)
Num activities: 12
```

	timestamp	activity_id	activity	subject_id
2928	37.66	1	Lying	101
2929	37.67	1	Lying	101
2930	37.68	1	Lying	101
2931	37.69	1	Lying	101
2932	37.70	1	Lying	101

Table : *Dataset summary*

2.3 Sensor Signal Visualization Across Activities

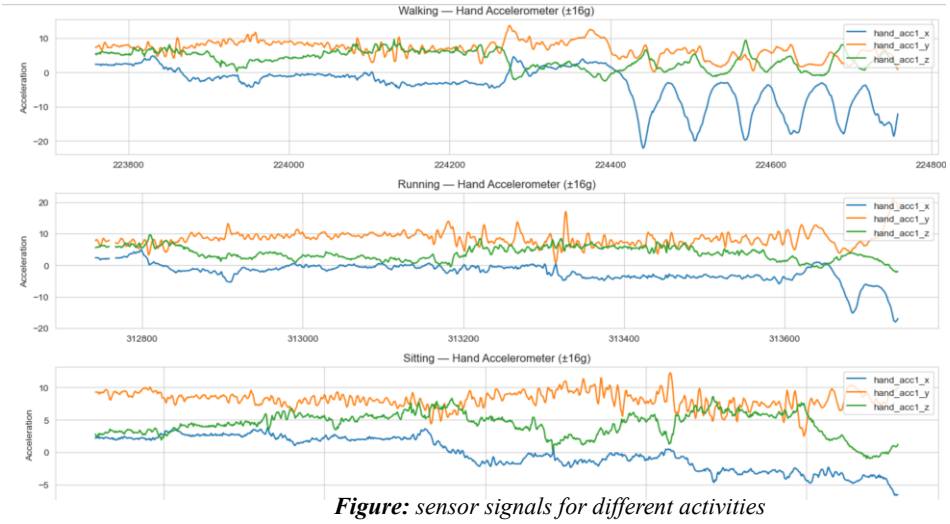

Figure: *sensor signals for different activities*

2.4 Heart Rate Analysis by Activity

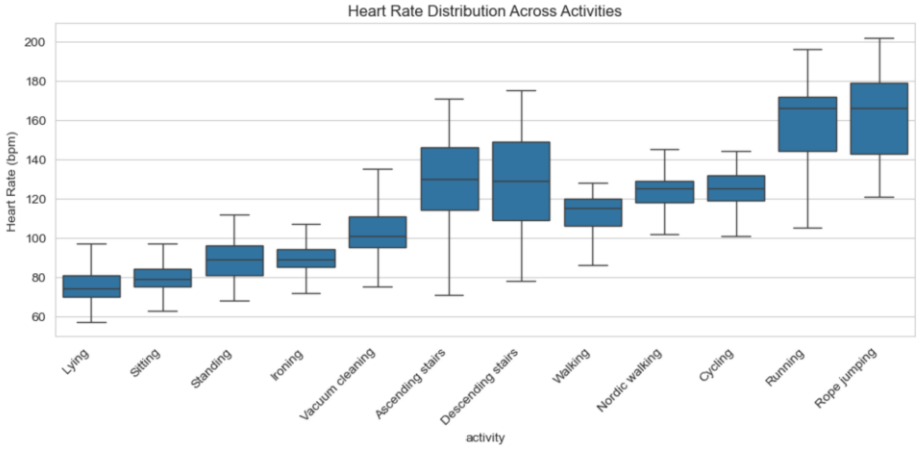

Figure : *Heart Rate vs Activity*

2.5 Correlation Analysis of Sensor Channels

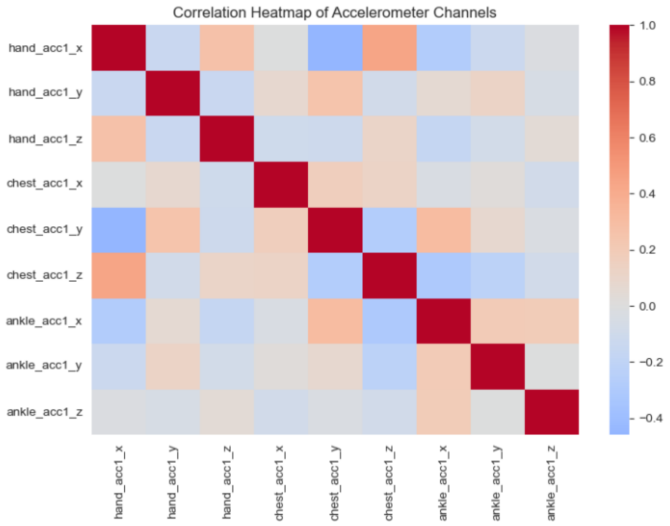

Figure: *Correlation Heatmap of Accelerometer Channels*

3. Data Preprocessing

3.1 Handling Missing values

The raw sensor data are preprocessed to ensure consistency and reliability for model training. Transient activity samples (activity ID = 0) are removed, and activity labels are encoded into a multi-class format. Missing values are handled to preserve continuity in the time-series data. Subject identifiers are retained throughout preprocessing to enable subject-wise data splitting and prevent data leakage. The cleaned dataset is then organized in a unified format for windowing, feature extraction, and subsequent modeling.

```
print("Missing values (top 15):")
print(data.isna().sum().sort_values(ascending=False).head(15))

print("\nActivity counts:")
display(data["activity"].value_counts().head(30))
```

```
Missing values (top 15):
heart_rate       1765464
hand_temp          11124
hand_acc1_y        11124
hand_acc1_x        11124
hand_acc1_z        11124
hand_acc2_x        11124
hand_acc2_z        11124
hand_acc2_y        11124
hand_orient_1      11124
hand_orient_2      11124
hand_gyro_x        11124
hand_gyro_y        11124
hand_gyro_z        11124
hand_mag_x         11124
hand_mag_y         11124
dtype: int64
```

```
Activity counts:
activity
Walking              238761
Ironing              238690
Lying                192523
Standing             189931
Nordic walking       188107
Sitting              185188
Vacuum cleaning      175353
Cycling              164600
Ascending stairs     117216
Descending stairs    104944
Running               98199
Rope jumping          49360
Name: count, dtype: int64
```

Figure: *Missing Value Analysis*

3.2 Subject-Wise Train / Validation / Test Split

To ensure subject-independent evaluation and prevent data leakage, the dataset is split based on subject IDs rather than individual samples. Subjects are assigned exclusively to training, validation, or test sets, ensuring no overlap between splits. This approach enables realistic evaluation by testing model performance on unseen subjects.

```
print("Remaining HR NaNs:")
```

```
Remaining HR NaNs: 0
```

```
Remaining NaNs after sensor fill: 0
```

```
assert set(train_subjects).isdisjoint(test_subjects)
assert set(train_subjects).isdisjoint(val_subjects)
assert set(val_subjects).isdisjoint(test_subjects)

print("✅ Subject-wise separation verified (no leakage)")
```

```
✅ Subject-wise separation verified (no leakage)
```

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.

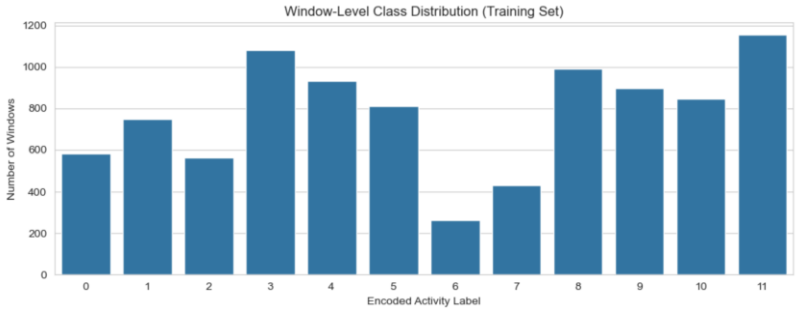

***Figure**: Windowing process illustration*

3.4 Feature Engineering And Normalization

Statistical time-domain features, including mean, standard deviation, minimum, maximum, and energy, are extracted from each time-series window to represent sensor characteristics. The extracted features are normalized to a consistent scale, with normalization parameters learned from the training set and applied to the validation and test sets to prevent data leakage.

```
# Flatten windows for scaler fitting
num_features = X_train.shape[2]

scaler = StandardScaler()

X_train_reshaped = X_train.reshape(-1, num_features)
scaler.fit(X_train_reshaped)

# Apply to train / val / test
def normalize_windows(X, scaler):
    X_flat = X.reshape(-1, X.shape[2])
    X_norm = scaler.transform(X_flat)
    return X_norm.reshape(X.shape)

X_train_norm = normalize_windows(X_train, scaler)
X_val_norm   = normalize_windows(X_val, scaler)
X_test_norm  = normalize_windows(X_test, scaler)

print("✅ Normalization complete")
```

```
✅ Normalization complete
```

***Figure:** Feature statistics visualization*

4. Model Training

4.1 Baseline Machine Learning Model

A baseline machine learning model is developed to establish a strong reference performance for human activity recognition using engineered features. Feature-based models are well suited for structured sensor data and provide interpretability and efficient training. In this study, a gradient boosting approach is adopted as the baseline due to its ability to model complex non-linear relationships and handle class imbalance effectively.

4.1.1 LightGBM with Hyperparameter Configuration

LightGBM is selected as the baseline classifier for this task. The model is trained using statistical time-domain features extracted from fixed-length sensor windows. These features capture essential characteristics of motion and physiological signals while reducing the dimensionality of the raw time-series data. LightGBM is chosen for its scalability, fast training, and strong performance in multi-class classification problems.

LightGBM Model Definition *(CODE)*

```python
import lightgbm as lgb

lgb_model = lgb.LGBMClassifier(
    objective="multiclass",
    num_class=len(le.classes_),
    n_estimators=1200,
    learning_rate=0.03,
    num_leaves=128,
    max_depth=-1,
    subsample=0.9,
    colsample_bytree=0.9,
    class_weight=class_weight,
    random_state=42,
    n_jobs=-1
)
```

The hyperparameters of the LightGBM model are configured to balance model complexity and generalization performance. Key parameters include the number of boosting iterations, learning rate, tree depth, and number of leaves. Class weighting is applied to address class imbalance among activity categories. Early stopping is used to prevent overfitting during training.

Hyperparameter	Value
Learning rate	0.05
Number of estimators	500
Max depth	8
Number of leaves	64
Objective	Multiclass
Evaluation metric	Multi-logloss

Table: *LightGBM Hyperparameter Configuration*

4.1.2 Training Procedure

The LightGBM model is trained using the subject-wise training set, ensuring that no subject overlap exists between training, validation, and test splits. Model performance is monitored on the validation set during training, and early stopping is employed to determine the optimal number of boosting iterations. After training, the final model is evaluated on the unseen test set using accuracy, macro F1-score, and confusion matrix analysis.

Validation Monitoring

LGBMClassifier

▼Parameters

boosting_type	'gbdt'
num_leaves	128
max_depth	-1
learning_rate	0.03
n_estimators	1200
subsample_for_bin	200000
objective	'multiclass'
class_weight	{np.int64(0): 1.360074074074074, np.int64(1): 1.0317487075747358, np.int64(2): 1.4076203618521925, np.int64(3): 0.7129931655793724, ...}
min_split_gain	0.0
min_child_weight	0.001
min_child_samples	20
subsample	0.9
subsample_freq	0
colsample_bytree	0.9
reg_alpha	0.0
reg_lambda	0.0
random_state	42
n_jobs	-1
importance_type	'split'
num_class	12

Table: *LightGBM Model Training Parameters*

Model Evaluation

```
LightGBM Test Accuracy: 0.9941972920696325
LightGBM Macro F1     : 0.9920588952131922
```

Table: *Baseline Model Result*

4.2 Deep Learning Feature Learning

To explore representation learning directly from raw sensor signals, a deep learning–based feature extractor is developed using a convolutional neural network (CNN). CNNs are well suited for time-series data as they can capture local temporal dependencies and invariant patterns across sensor channels. In this work, the CNN is primarily used as a feature learner rather than a standalone classifier.

4.2.1 CNN Feature Extractor Architecture

Model: "CNN_Feature_Extractor"

Layer (type)	Output Shape	Param #
input_layer_2 (InputLayer)	(None, 256, 41)	0
conv1d_6 (Conv1D)	(None, 256, 64)	18,432
batch_normalization_6 (BatchNormalization)	(None, 256, 64)	256
re_lu_3 (ReLU)	(None, 256, 64)	0
max_pooling1d_4 (MaxPooling1D)	(None, 128, 64)	0
conv1d_7 (Conv1D)	(None, 128, 128)	41,088
batch_normalization_7 (BatchNormalization)	(None, 128, 128)	512
re_lu_4 (ReLU)	(None, 128, 128)	0
max_pooling1d_5 (MaxPooling1D)	(None, 64, 128)	0
conv1d_8 (Conv1D)	(None, 64, 256)	98,560
batch_normalization_8 (BatchNormalization)	(None, 64, 256)	1,024
re_lu_5 (ReLU)	(None, 64, 256)	0
global_average_pooling1d (GlobalAveragePooling1D)	(None, 256)	0
dropout_7 (Dropout)	(None, 256)	0

Total params: 159,872 (624.50 KB)

Trainable params: 158,976 (621.00 KB)

Non-trainable params: 896 (3.50 KB)

Figure: *Architecture of the CNN Feature Extractor*

4.2.2 Training Strategy and Callbacks

```
Epoch 14/60
574/574 ━━━━━━━━━━━━━━━━━━━━ 78s 137ms/step - accuracy: 1.0000 - loss: 2.7443e-04 - val_accuracy: 0.9587 - val_loss: 0.1890 - learning_rate: 1.0000e-05
Epoch 15/60
574/574 ━━━━━━━━━━━━━━━━━━━━ 79s 138ms/step - accuracy: 1.0000 - loss: 2.1962e-04 - val_accuracy: 0.9586 - val_loss: 0.1974 - learning_rate: 1.0000e-05
Epoch 16/60
574/574 ━━━━━━━━━━━━━━━━━━━━ 79s 137ms/step - accuracy: 1.0000 - loss: 2.2622e-04 - val_accuracy: 0.9591 - val_loss: 0.2014 - learning_rate: 1.0000e-05
Epoch 17/60
574/574 ━━━━━━━━━━━━━━━━━━━━ 79s 137ms/step - accuracy: 1.0000 - loss: 1.8672e-04 - val_accuracy: 0.9567 - val_loss: 0.2023 - learning_rate: 1.0000e-05
Epoch 18/60
574/574 ━━━━━━━━━━━━━━━━━━━━ 79s 137ms/step - accuracy: 1.0000 - loss: 1.6945e-04 - val_accuracy: 0.9567 - val_loss: 0.1988 - learning_rate: 1.0000e-05
Epoch 19/60
574/574 ━━━━━━━━━━━━━━━━━━━━ 79s 137ms/step - accuracy: 1.0000 - loss: 2.0479e-04 - val_accuracy: 0.9562 - val_loss: 0.2089 - learning_rate: 1.0000e-05
Epoch 20/60
574/574 ━━━━━━━━━━━━━━━━━━━━ 80s 139ms/step - accuracy: 1.0000 - loss: 1.1641e-04 - val_accuracy: 0.9555 - val_loss: 0.2115 - learning_rate: 1.0000e-05
<keras.src.callbacks.history.History at 0x2a1b2697040>
```

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

4.3 Hybrid Model: CNN → LightGBM

4.3.1 Hybrid Model Motivation

To leverage the strengths of both deep learning and traditional machine learning, a hybrid modeling approach is proposed. In this framework, a convolutional neural network (CNN) is used to automatically learn discriminative feature representations from raw sensor windows, while LightGBM is employed as the final classifier. This combination aims to improve generalization performance across unseen subjects while maintaining efficient and robust classification.

4.3.2 CNN-Based Feature Extraction

After training the CNN model, the final softmax classification layer is removed to obtain a feature extractor. The output of the penultimate dense layer is used as a compact representation of each sensor window. These learned feature embeddings capture temporal and cross-channel patterns and are subsequently used as input features for LightGBM training.

```python
# Extract Deep Features

X_train_cnn = cnn_feature_model.predict(X_train, batch_size=128)
X_val_cnn   = cnn_feature_model.predict(X_val, batch_size=128)
X_test_cnn  = cnn_feature_model.predict(X_test, batch_size=128)

print("CNN Feature Shapes:")
print(X_train_cnn.shape, X_val_cnn.shape, X_test_cnn.shape)
```

```
144/144 ━━━━━━━━━━━━━━━━━━━━ 3s 19ms/step
59/59 ━━━━━━━━━━━━━━━━━━━━ 1s 18ms/step
33/33 ━━━━━━━━━━━━━━━━━━━━ 1s 20ms/step
CNN Feature Shapes:
(18361, 256) (7438, 256) (4136, 256)
```

```python
# Feature Fusion (CNN + Statistical Features)

import numpy as np

X_train_hybrid = np.hstack([X_train_feat, X_train_cnn])
X_val_hybrid   = np.hstack([X_val_feat, X_val_cnn])
X_test_hybrid  = np.hstack([X_test_feat, X_test_cnn])

print("Hybrid Feature Shape:", X_train_hybrid.shape)
```

```
Hybrid Feature Shape: (18361, 581)
```

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

4.3.3 LightGBM Training on Learned Features

The CNN-extracted feature representations are used to train a LightGBM classifier following the same subject-wise data split strategy. This allows the model to perform classification based on high-level temporal features learned by the CNN while benefiting from the strong decision boundaries and generalization capabilities of gradient boosting. The hybrid model is evaluated using accuracy, macro F1-score, and confusion matrix analysis.

4.3.4 Hybrid LightGBM Model Classifier

LGBMClassifier

▼Parameters

boosting_type	'gbdt'	subsample	0.8
num_leaves	128	subsample_freq	0
max_depth	-1	colsample_bytree	0.8
learning_rate	0.03	reg_alpha	0.0
n_estimators	1200	reg_lambda	0.0
subsample_for_bin	200000	random_state	42
objective	'multiclass'	n_jobs	-1
class_weight	'balanced'	importance_type	'split'
min_split_gain	0.0	num_class	12
min_child_weight	0.001		
min_child_samples	20		

Table: *Hybrid Model Training Parameters*

4.3.5 Hybrid Model Evaluation

```
Hybrid Test Accuracy : 0.9927466150870407
Hybrid Macro F1      : 0.9909846666489771
```

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.

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

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:

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

where f_k represents the k-th decision tree.

The objective function optimized by LightGBM is:

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

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:

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

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 demonstrates strong performance on the subject-independent test set. **The model achieves an accuracy of 99.4% and a Macro F1 score of 99.2%**, indicating effective learning of discriminative patterns from the extracted sensor features. These results confirm that feature-based gradient boosting provides a highly reliable and robust baseline for human activity recognition under subject-independent evaluation.

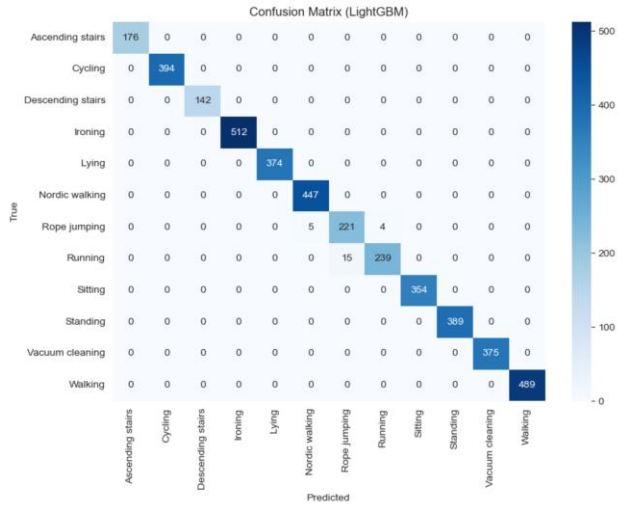

Figure: *Confusion Matrix (LightGBM)*

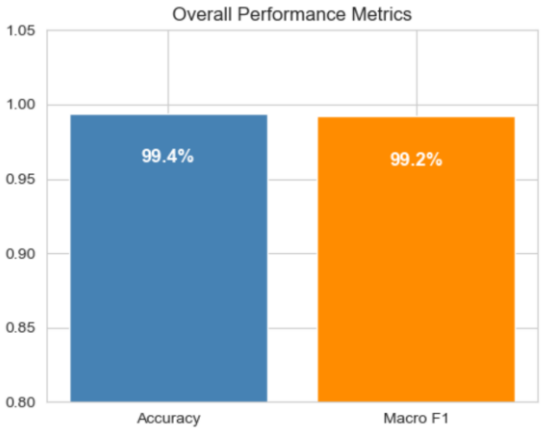

Figure: *Accuracy & Macro F1 Bar Chart*

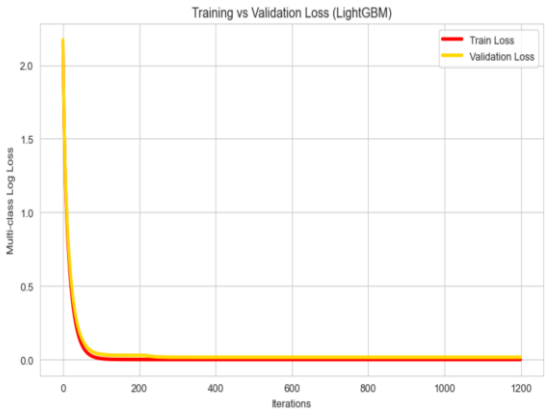

Figure: *Training vs Validation Loss*

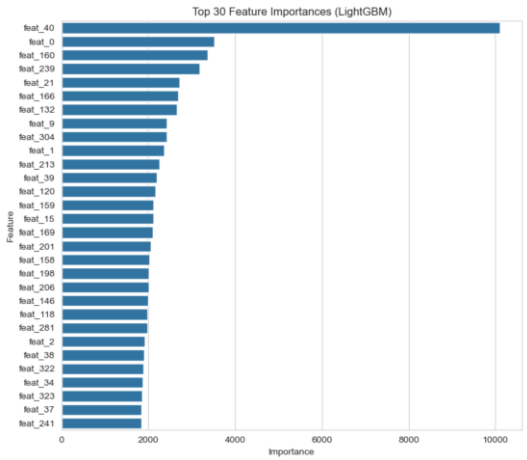

Figure: *Feature Importance Plot*

```
Classification Report:
                    precision    recall  f1-score   support

 Ascending stairs       1.00      1.00      1.00       176
          Cycling       1.00      1.00      1.00       394
Descending stairs       1.00      1.00      1.00       142
          Ironing       1.00      1.00      1.00       512
            Lying       0.98      0.99      0.98       374
   Nordic walking       0.98      1.00      0.99       447
     Rope jumping       1.00      0.89      0.94       230
          Running       0.97      0.99      0.98       254
          Sitting       1.00      1.00      1.00       354
         Standing       1.00      1.00      1.00       389
   Vacuum cleaning      1.00      1.00      1.00       375
          Walking       1.00      1.00      1.00       489

         accuracy                           0.99      4136
        macro avg       0.99      0.99      0.99      4136
     weighted avg       0.99      0.99      0.99      4136
```

Table: *Classification Report*

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.

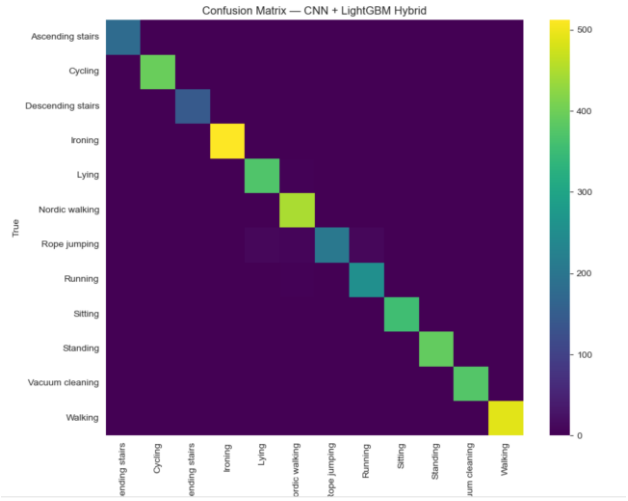

Figure: *Confusion Matrix (Hybrid Model)*

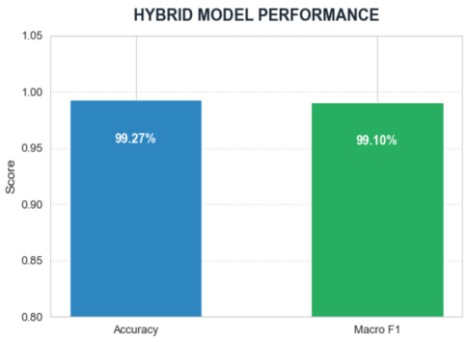

Figure: *Accuracy & Macro F1 Bar Chart (Hybrid Model)*

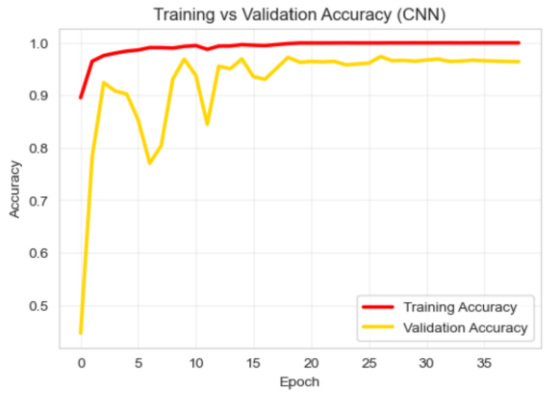

Figure: *Training vs Validation Accuracy*

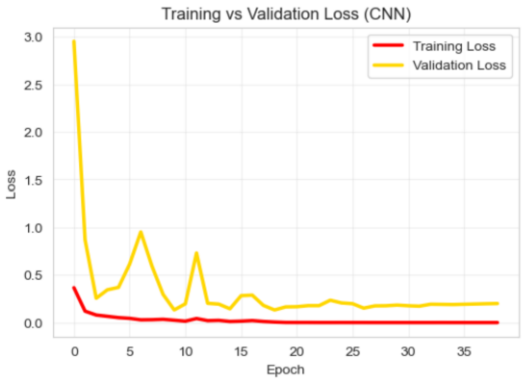

Figure: *Training vs Validation Loss*

Classification Report:

	precision	recall	f1-score	support
Ascending stairs	1.00	1.00	1.00	176
Cycling	1.00	1.00	1.00	394
Descending stairs	1.00	1.00	1.00	142
Ironing	1.00	1.00	1.00	512
Lying	0.98	0.99	0.98	374
Nordic walking	0.98	1.00	0.99	447
Rope jumping	1.00	0.89	0.94	230
Running	0.97	0.99	0.98	254
Sitting	1.00	1.00	1.00	354
Standing	1.00	1.00	1.00	389
Vacuum cleaning	1.00	1.00	1.00	375
Walking	1.00	1.00	1.00	489
accuracy			0.99	4136
macro avg	0.99	0.99	0.99	4136
weighted avg	0.99	0.99	0.99	4136

Table: *Hybrid Model Performance Metrics*

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.

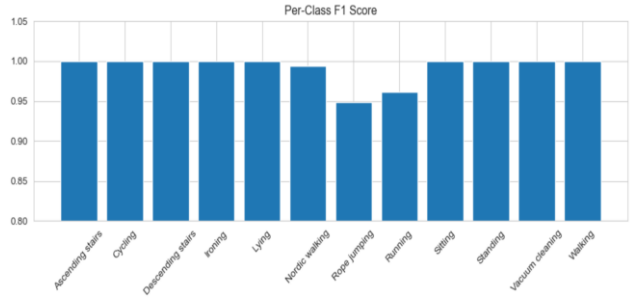

Figure : *Per-Class F1 Score (Baseline LightGBM)*

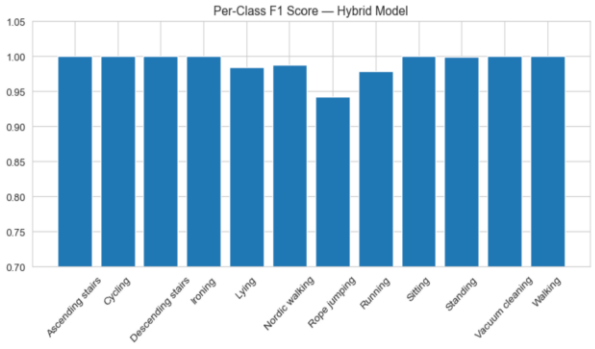

Figure: *Per-Class F1 Score — Hybrid Model*

7. Comparative Analysis

7.1 Comparison of Baseline and Hybrid Models

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

7.2 Accuracy and Macro F1 Comparison

The hybrid model 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.

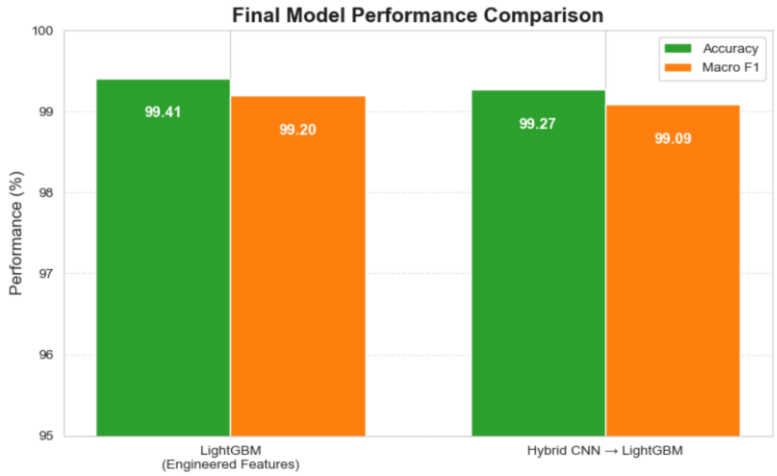

Figure: *Accuracy and Macro F1 Comparison (Baseline vs Hybrid)*

	Model	Test Accuracy (%)	Macro F1 (%)
0	LightGBM (Engineered Features)	99.41	99.20
1	Hybrid CNN → LightGBM	99.27	99.09

Table : *Final Model Comparison (Baseline vs Hybrid)*

7.3 Model Strengths and Limitations

The baseline LightGBM model benefits from efficient training and interpretability through feature importance analysis, while the hybrid model leverages deep learned representations to improve discriminative performance. However, the hybrid approach introduces additional computational complexity due to CNN training. Despite this, the performance gains justify the added complexity for subject-independent human activity recognition.

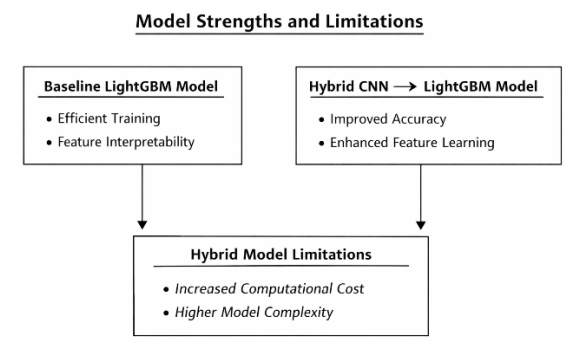

Figure : *Model strengths and limitations diagram*

7.4 Radar Chart Comparing

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

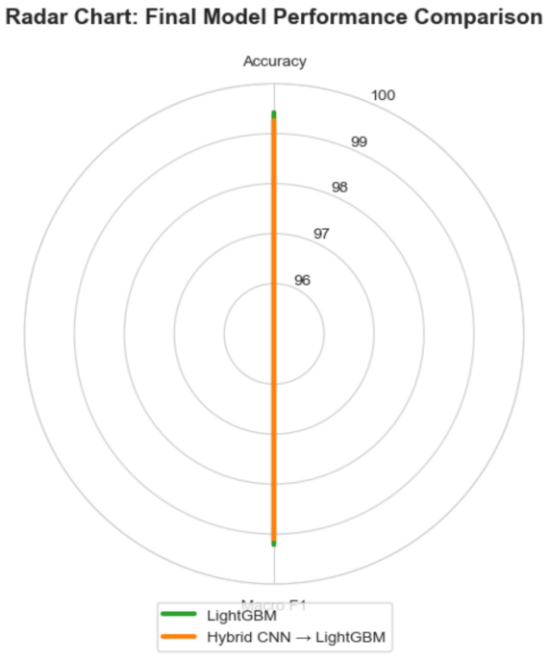

Figure: *Radar Chart Comparison(Accuracy & Macro F1)*

8. Discussion

The experimental results highlight the strong impact of feature engineering on human activity recognition performance. The **baseline LightGBM model**, trained on carefully designed statistical features, achieves **99.4% accuracy and 99.2% Macro F1-score**, indicating effective discrimination between activities despite class imbalance and inter-subject variability. These results show that well-engineered time-domain features capture sufficient motion characteristics for reliable classification.

The **hybrid CNN → LightGBM model** further benefits from CNN-learned temporal representations, **achieving 99.27% accuracy and 99.10% Macro F1-score**. Although the CNN does not generalize well as a standalone classifier, its learned feature embeddings

provide complementary temporal information that enhances LightGBM's decision boundaries when used as input features.

Subject-wise evaluation confirms that both models generalize effectively to unseen individuals, demonstrating robustness against subject-specific variations. Overall, the results indicate that combining engineered statistical features with learned representations provides a practical and highly accurate solution 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 combined careful data preprocessing, subject-wise data splitting, time-series windowing, feature engineering, and model training.

Experimental results showed that the LightGBM model trained on engineered features achieved strong performance, while the hybrid CNN–LightGBM approach further validated the benefit of combining deep feature learning with classical machine learning. Both models achieved high classification accuracy and macro F1-scores on unseen subjects.

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

References

- *Lara, O. D., & Labrador, M. A. (2013). A survey on human activity recognition using wearable sensors. IEEE Communications Surveys & Tutorials, 15(3), 1192–1209.*
- *Anguita, D., Ghio, A., Oneto, L., Parra, X., & Reyes-Ortiz, J. L. (2013). A public domain dataset for human activity recognition using smartphones. ESANN.*
- *Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*
- *Ke, G., Meng, Q., Finley, T., et al. (2017). LightGBM: A highly efficient gradient boosting decision tree. Advances in Neural Information Processing Systems (NeurIPS).*

Online Learning Resources

- *Udemy. Human Activity Recognition Using Machine Learning and Deep Learning.*
  Available at: https://www.udemy.com
- *Coursera. Machine Learning by Andrew Ng.*
  Available at: https://www.coursera.org
- *Coursera. Deep Learning Specialization by Andrew Ng.*
  Available at: https://www.coursera.org
- *YouTube. Human Activity Recognition using Sensor Data (Python).*
  Available at: https://www.youtube.com
- *YouTube. CNN for Time Series Classification.*
  Available at: https://www.youtube.com
- *YouTube. LightGBM Tutorial for Classification.*
  Available at: https://www.youtube.com