Ambient Intelligence and Domotics

Domotics Exam Project

ANASS NASSIRI

Domotics Exam Project Report

Project Overview

The project focuses on **Supervised Human Activity Recognition** to classify Activities. The following Pipeline of the Project:

1. Exploratory Data Analysis (EDA)
2. Data Preprocessing: Denoising, Normalization, Segmentation, Balancing the dataset.
3. Feature Engineering: Handcrafted feature extraction.
4. Training the Models:
    a. Baseline Model (MLP)
    b. Conv1D model
    c. LSTM model
    d. Combined Conv1dLSTM

HARTH Dataset

The Human Activity Recognition Trondheim (HARTH) dataset is a professionally annotated dataset containing 22 subjects wearing two 3-axial accelerometers for around 2 hours in a free-living setting. The sensors were attached to the right thigh and lower back. The professional recordings and annotations provide a promising benchmark dataset for researchers to develop innovative machine learning approaches for precise HAR in free living.

Dataset Characteristics Multivariate, Time-Series

Subject Area Computer Science

Associated Tasks Classification

Feature Type Real

Instances 6461328

Features 8

Dataset Information For what purpose was the dataset created?

The dataset was created to train machine learning classifiers for human activity recognition based on professional annotations of activities in a free-living setting.

Who funded the creation of the dataset?

NTNU Helse

Additional Information

The HARTH dataset contains recordings of 22 participants wearing two 3-axial Axivity AX3 accelerometers for around 2 hours in a free-living setting. One sensor was attached to the right front thigh and the other to the lower back. The provided sampling rate is 50Hz. Video recordings of a chest-mounted camera were used to annotate the performed activities frame-by-frame.

Each subject's recordings are provided in a separate .csv file. One such .csv file contains the following columns:

1. timestamp: date and time of recorded sample
2. back_x: acceleration of back sensor in x-direction (down) in the unit g
3. back_y: acceleration of back sensor in y-direction (left) in the unit g
4. back_z: acceleration of back sensor in z-direction (forward) in the unit g
5. thigh_x: acceleration of thigh sensor in x-direction (down) in the unit g
6. thigh_y: acceleration of thigh sensor in y-direction (right) in the unit g
7. thigh_z: acceleration of thigh sensor in z-direction (backward) in the unit g
8. label: annotated activity code

The dataset contains the following annotated activities with the corresponding coding: 1: walking 2: running 3: shuffling 4: stairs (ascending) 5: stairs (descending) 6: standing 7: sitting 8: lying 13: cycling (sit) 14: cycling (stand) 130: cycling (sit, inactive) 140: cycling (stand, inactive)

Has Missing Values?: No

Data Loading and Preprocessing

For the data loading you i use two piece of code one for google colab Environment and one for Kaggle. because Kaggle give more computation power and time.

1. **Preprocessing:** Denoising, Normalization, Segmentation, Balancing the dataset.

Segmentation:

I performs data segmentation by dividing the dataset into fixed-size windows of 50 samples each. For each window, it extracts six features (back_x, back_y, back_z, thigh_x, thigh_y, thigh_z), determines the most frequent label and subject, and appends these to separate lists. The segmented data, labels, and subjects are then converted into NumPy arrays, with X_segments shaped as (num_samples, window_size, num_features).

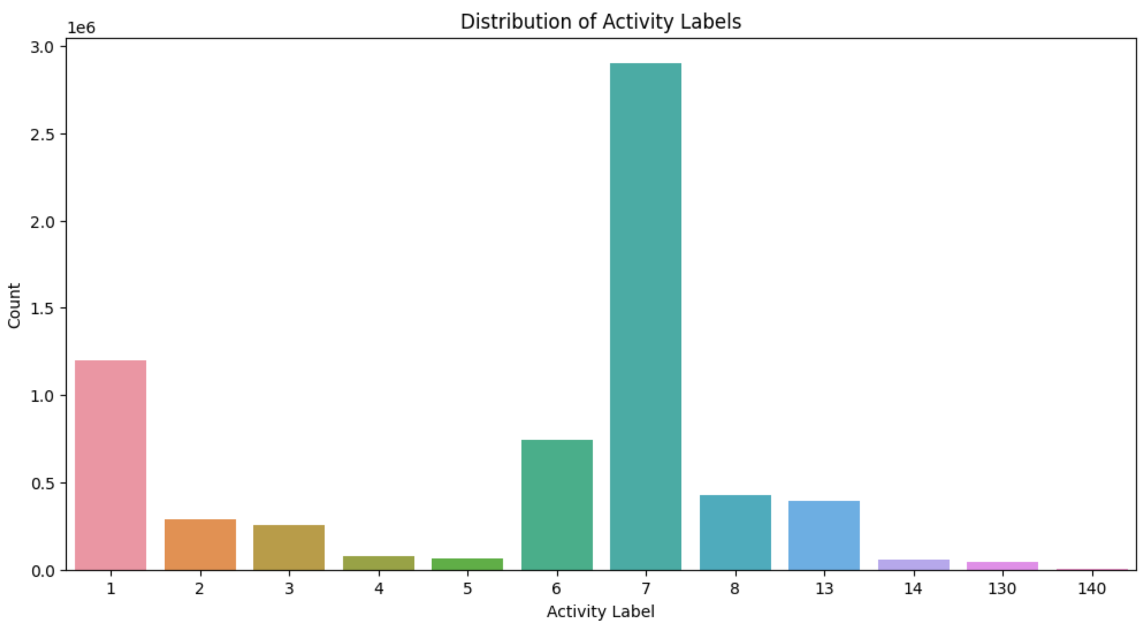
Distribution of dataset:

Activity map:

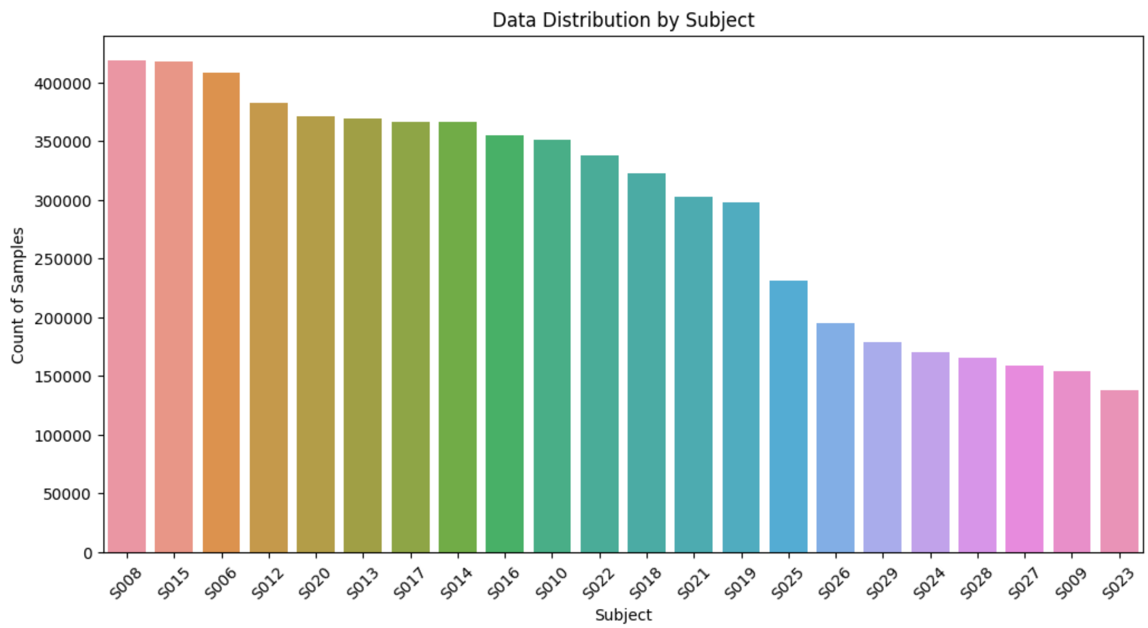
```
activities_map={
1: 'walking',
2: 'running',
3: 'shuffling',
4: 'stairs (ascending)',
5: 'stairs (descending)',
6: 'standing',
7: 'sitting',
8: 'lying',
13: 'cycling (sit)',
14: 'cycling (stand)',
130: 'cycling (sit, inactive)',
140: 'cycling (stand, inactive)'
}
```

The distribution of activities in dataset:

The distribution of activities is showing clear that the dataset is unbalanced



Data Distribution by Subject

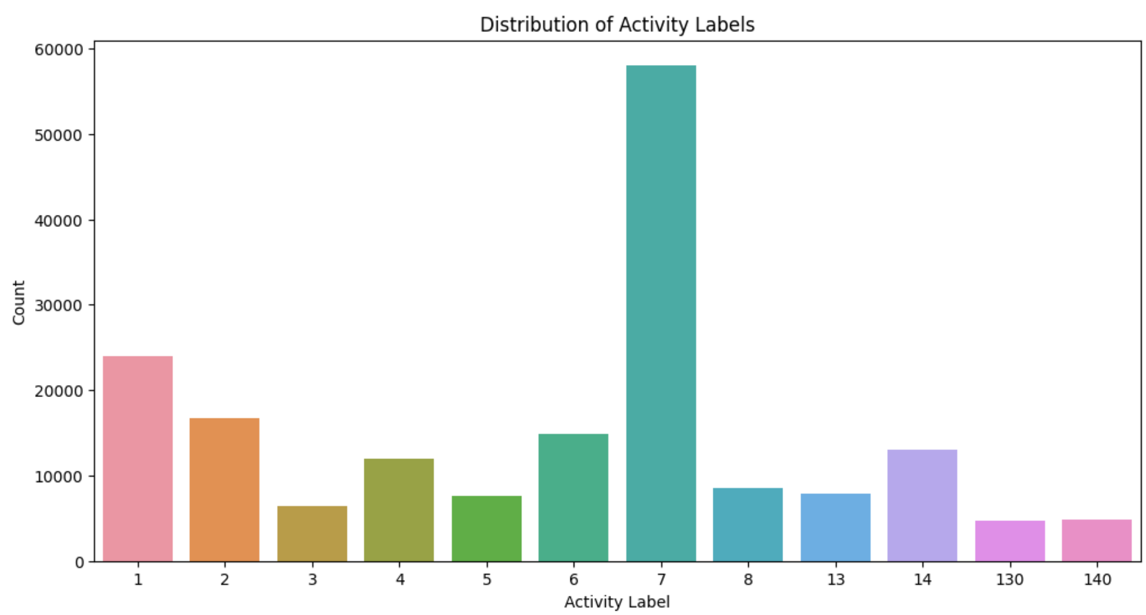


Data Balancing:

Data balancing is a crucial step in machine learning, particularly when dealing with imbalanced datasets. An imbalanced dataset can lead to biased models that perform poorly on minority classes. I use this method: Synthetic Minority Over-sampling Technique (SMOTE).

Let's Plot the distribution again after balancing.

We can notice that the distribution become better but not perfect.



Feature Engineering

In The project employs automatic feature extraction methods with conv1d and LSTM models and combined model Conv1D-LSTM. is compared against a baseline model that uses handcrafted feature extraction.

Model Training and Evaluation

Models Used:

1. Baseline MLP (Multi-Layer Perceptron): Uses handcrafted features.

2. Conv1D: Conv1D is highly effective in detecting local patterns and features in time-series data due to its convolutional filters

3. LSTM (Long Short-Term Memory LSTMs are designed to capture long-term dependencies in sequential data.

Combined Conv1D and LSTM: Combining Conv1D and LSTM leverages the strengths of both architectures. Conv1D layers can efficiently extract local features and patterns from the input sequences, while LSTM layers can capture long-term dependencies and temporal dynamics

Training Process:

for the epochs I use few numbers of epochs because the data set is large, and I have computational environments very limited. Even that I got a good performance.

The baseline model trained with handcrafted feature extraction data.

The advanced model trained Conv1D and LSTM and Conv1D-LSTM trained balanced data

Evaluation Metrics:

The project includes code to plot confusion matrices and model architectures to visualize the results. It is observed that the automatic feature extraction models (Conv1D, LSTM, and their combination) outperform the baseline MLP model.

Results Table

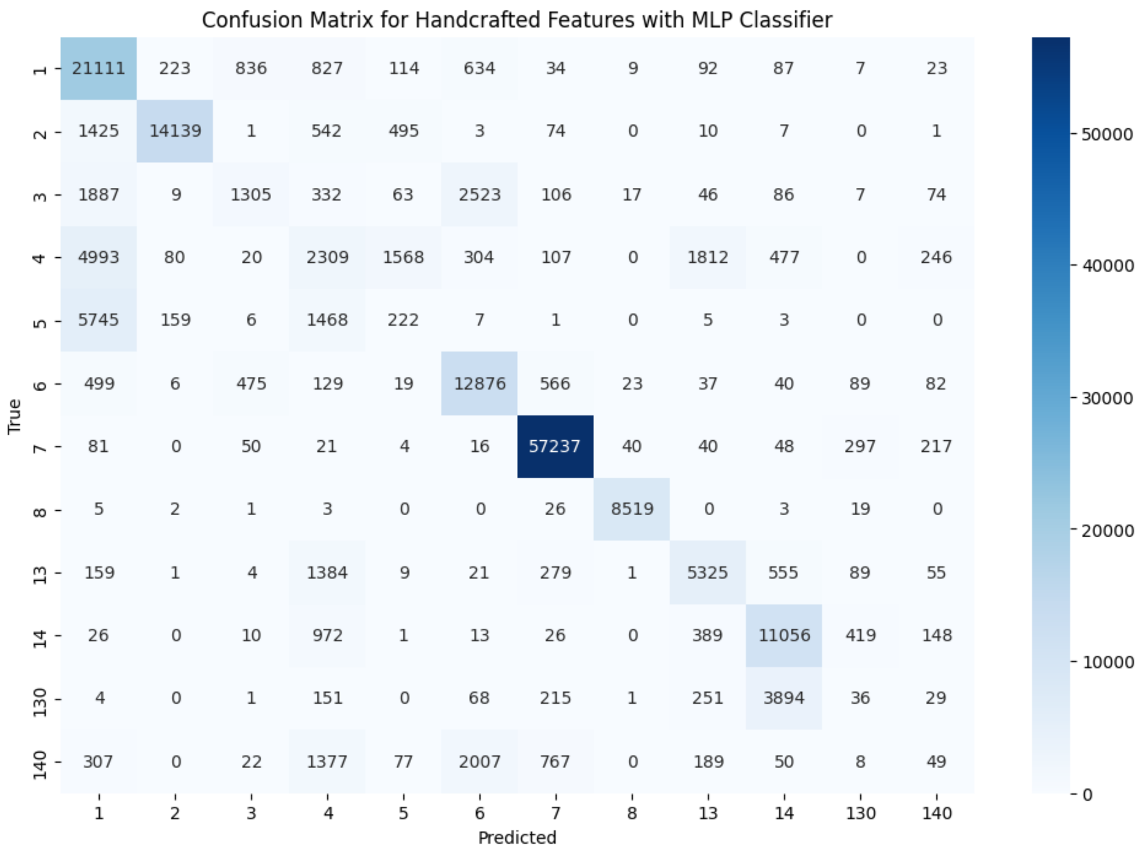
Model	Overall Accuracy	Overall Recall	Overall F1 Score
Baseline MLP	0.7391	0.5127	0.4542
Conv1D	0.8219	0.6041	0.5557
LSTM	0.7975	0.5578	0.5056

Combined Conv1D and LSTM	0.8106	0.5780	0.5249

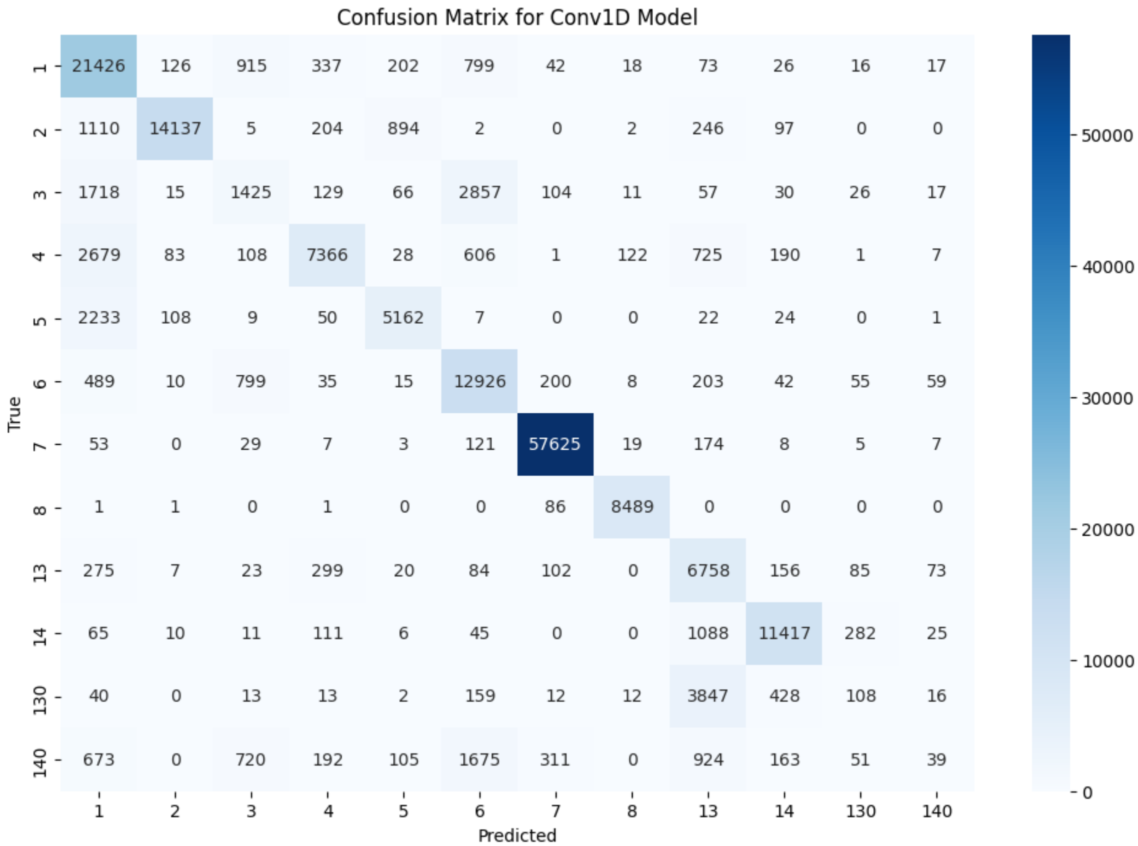
Confusion Matrix Comparisons

Below are the placeholders for confusion matrix images comparing the performance of different models. Please insert the respective images in these placeholders.

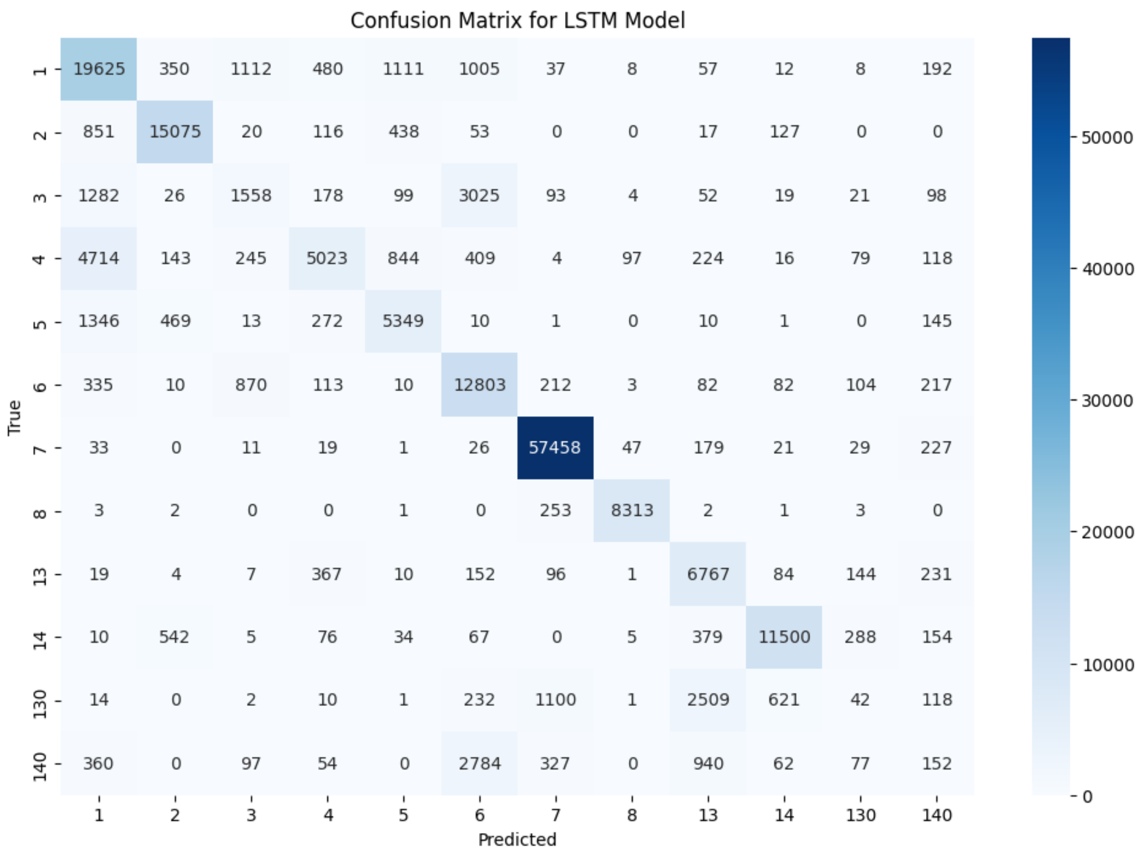
Baseline MLP Confusion Matrix



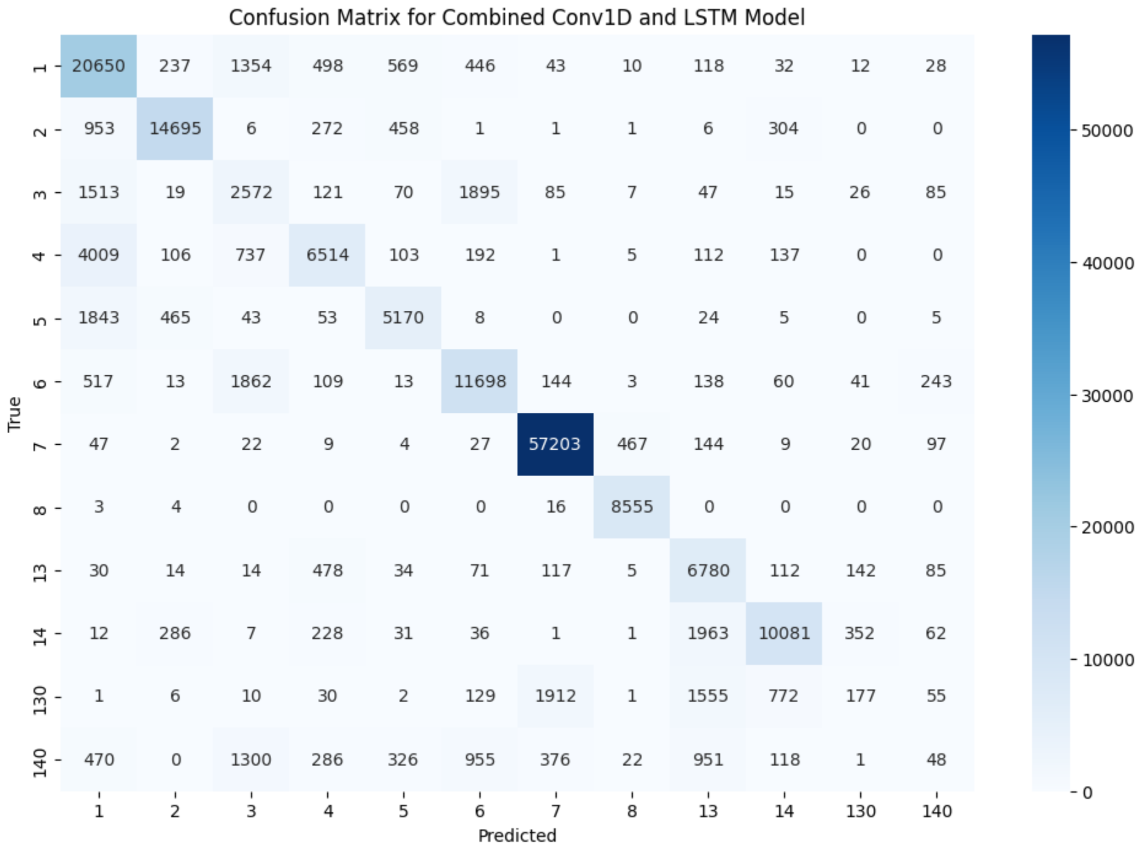
Conv1D Confusion Matrix



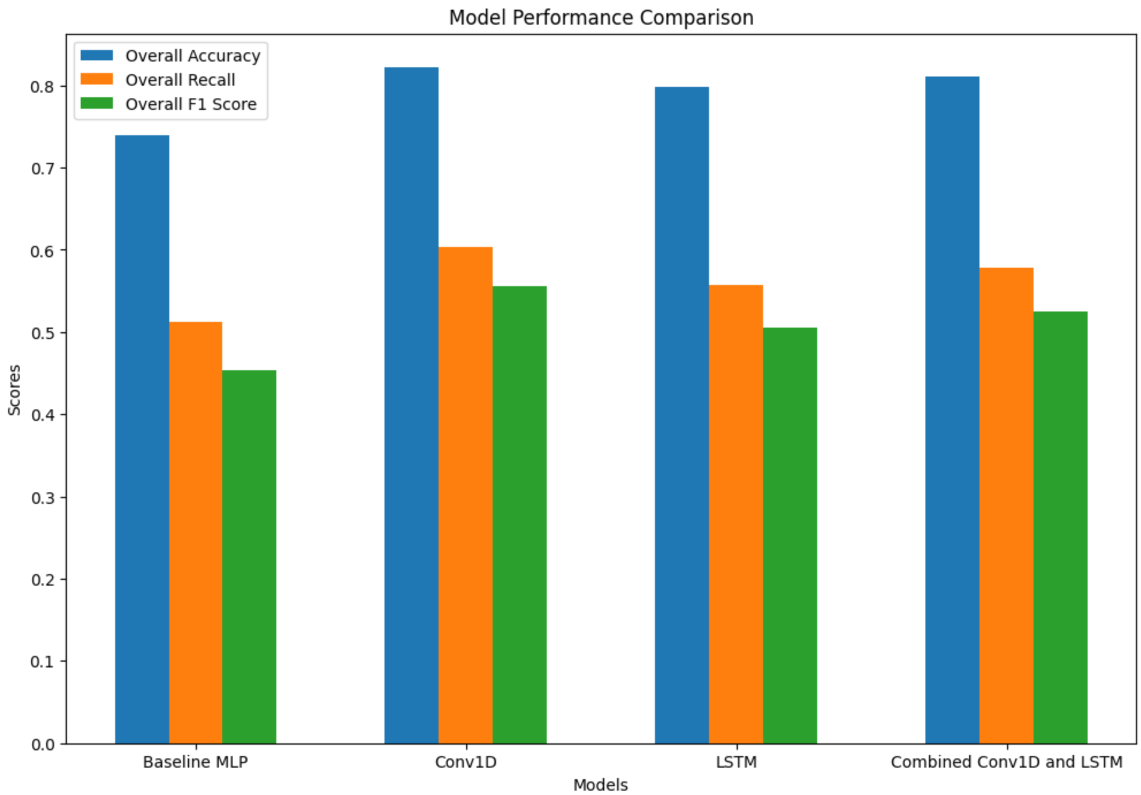
LSTM Confusion Matrix



Combined Conv1D and LSTM Confusion Matrix



Model Comparison



Conclusion

The project concludes that automatic feature extraction models, particularly those combining Conv1D and LSTM layers, perform better than the baseline model that relies on handcrafted features. This highlights the effectiveness of deep learning techniques in extracting and leveraging features from raw data for classification tasks in home automation.

Final Remarks

I enjoy work on this project I learned a lot. But I faced a lot of problems and difficulties because the dataset Too large and computation environments I have is not suitable for big projects and large dataset. I trued my best to manage the situation. I kept switching between the platform Kaggle and google colab.

Future work and improvement.

With a powerful environment we can use advanced model and method and techniques. also perform a better balancing for the data set also more complex preprocessing.

Table des matières

Project Overview....2

HARTH Dataset....2

Data Loading and Preprocessing....3

Segmentation:....3

Distribution of dataset:....4

Data Balancing:....5

Feature Engineering....6

Model Training and Evaluation....6

Results Table....6

Confusion Matrix Comparisons....7

Baseline MLP Confusion Matrix....7

Conv1D Confusion Matrix....8

LSTM Confusion Matrix....9

Combined Conv1D and LSTM Confusion Matrix....10

Model Comparison....11

Conclusion....11

Final Remarks....11

Future work and improvement.....11