Electromyography Signal Analysis and Classification

A Comprehensive Report on Signal Processing and Machine Learning

Harshit Patel
Department of ECE
Indian Institute of Information technology, Sricity
Email: `harshit.p22@iiits.in`

***Abstract*—This report presents a systematic approach to processing and classifying Electromyography (EMG) data. The methodology includes signal segmentation, feature extraction, and classification using machine learning models such as Random Forest (RF) and LightGBM. The report also evaluates model performance across varying numbers of activity classes. The results highlight the effectiveness of the proposed techniques for real-time EMG signal classification tasks.**

I. Introduction

Electromyography (EMG) signals are widely used in biomedical and motion analysis applications, including prosthetics, gesture recognition, and physical therapy. Processing EMG data involves several steps, including data cleaning, segmentation, feature extraction, and classification. This study implements a pipeline for EMG signal analysis and evaluates the performance of classification models in distinguishing between multiple activity types.

II. System Overview

A. Libraries and Tools Used

The analysis was conducted using Python, with the following libraries:

- `pandas`: Data manipulation and preprocessing.
- `numpy`: Numerical computations.
- `matplotlib` and `seaborn`: Visualization of data and results.
- `scikit-learn`: Machine learning utilities.
- `xgboost` and `lightgbm`: Advanced gradient boosting models.
- `tensorflow.keras`: Neural network implementation.

III. Methodology

The pipeline for EMG data analysis involves the following stages:

A. Data Collection and Organization

EMG data was organized in a hierarchical directory structure, with folders for each subject and subfolders for different activity types (*Normal* and *Aggressive*). The data files were in `.txt` format, containing time-series signals recorded from multiple channels.

B. Data Loading and Cleaning

The script iterates through the folder structure, reads each data file into a `pandas` DataFrame, and ensures uniformity by padding signals with zeros to a fixed length of 10,000 rows. Additional metadata columns, including `Activity`, `Subject`, and `ActionType`, were appended for identification.

C. Segmentation and Feature Extraction

Each trial is divided into N_{seg} segments. For each segment, the following time-domain features are extracted:

- Mean, Median, Variance
- Skewness, Kurtosis
- Root Mean Square (RMS)
- Mean Absolute Value (MAV)
- Signal Energy (SE)
- Zero-Crossing Rate (ZC)

This results in a comprehensive feature vector for each trial.

D. Classification and Evaluation

Machine learning models, including Random Forest (RF) and LightGBM, were trained on the extracted features. A Monte Carlo approach was used to evaluate model performance on subsets of activity classes, varying the number of classes from 2 to 19.

IV. Results and Discussion

A. Feature Distribution and Visualization

Visualizations of the EMG signals and extracted features demonstrated clear differences between activity types, as shown in Fig. 1.

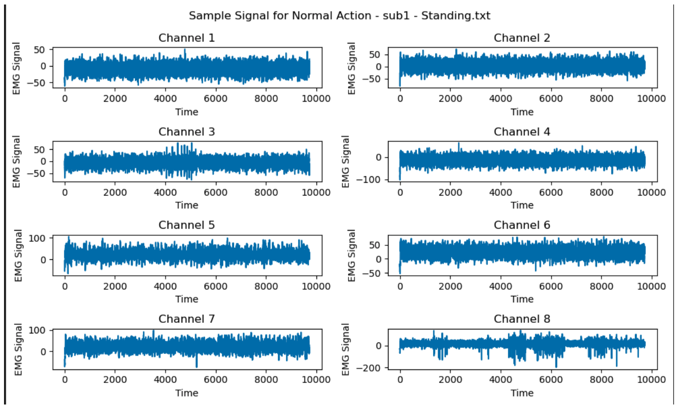

Fig. 1: Sample EMG Signal from Channel 1 for Normal and Aggressive Actions.

B. Model Performance

The mean classification accuracy of RF, LightGBM and Knn models was plotted against the number of activity classes. As expected, classification accuracy decreases with an increasing number of classes, as shown in Fig. 2.

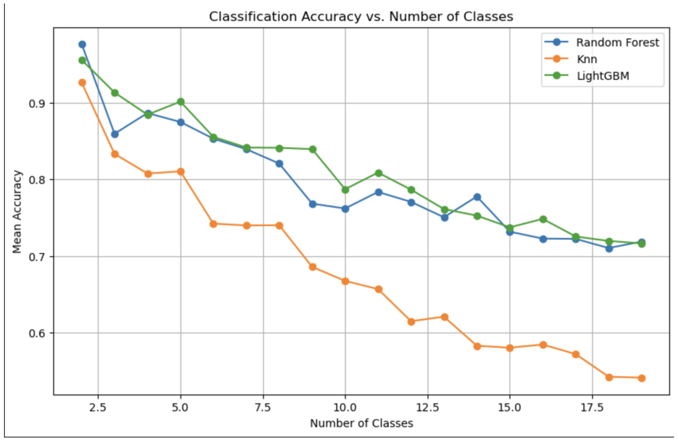

Fig. 2: Classification Accuracy vs. Number of Activity Classes.

C. Comparison of Models

The LightGBM model showed consistently higher accuracy compared to RF for most class subsets, highlighting its efficiency in handling complex feature spaces.

V. Conclusion

The proposed pipeline successfully demonstrates the utility of feature-based approaches for EMG signal classification. Future work includes extending the feature set and exploring deep learning models for end-to-end analysis.

References

[1] Wes McKinney, *pandas: A Python Data Analysis Library*. Available: https://pandas.pydata.org/.

[2] Pedregosa et al., *Scikit-learn: Machine Learning in Python*. JMLR 12, pp. 2825–2830, 2011.

[3] Martín Abadi et al., *TensorFlow: A System for Large-Scale Machine Learning*. Available: https://www.tensorflow.org/.

[4] My code for the following https://github.com/Awesomeoverflow/Pr_ assignment/blob/main/S20220020278_assignment.ipynb