STACKING/ENSEMBLE MODEL FOR SMARTPHONE-BASED HUMAN ACTIVITY RECOGNITION USING FEATURE ENGINEERING

Pratik Pal*, **Priyanshu Kumar Rai***
Department of Electrical Engineering and Computer Science
Indian Institute of Science Education and Research, Bhopal
{pratik20, priyanshu20}@iiserb.ac.in

ABSTRACT

Human activity recognition, commonly known as HAR, involves identifying numerous physical activities individuals conduct across diverse situations. Examples of such actions include walking, jogging, running, ascending stairs, descending stairs, and more. Our work attempts to construct a model that accurately recognizes human activity based on data obtained from cellphones. To achieve optimal performance, we plan to use an ensemble model in conjunction with feature engineering.

1 INTRODUCTION

In recent decades, a growing focus has been on leveraging wearable sensor technology to recognize body activities. The intricacies and diversity inherent in body movements challenge the prompt, precise, and automated identification of such activities. Our work reframes the issue of body activity recognition as a classification problem, utilizing data acquired through wearable sensors. Recent research suggests many approaches to getting effective classification results for the problem. Chadha et al. (2023) used T-SNE for dimensionality reduction, then trained the model using SVM, Logistic Regression, CNN, and other models. Dhanraj et al. (2019) developed EHARS, a simple novel CNN architecture for classification. San-Segundo et al. (2016) used Hidden Markov Models (HMMs), and Hernández et al. (2019) designed a bidirectional LSTM model for classification. The limitations of the existing work are that they struggle to generalize in terms of different contexts, environments, and agents, and that some advanced models demand higher computational resources when trained on large datasets. In our work, we have used the feature engineering technique of maximum relevance - minimum redundancy (mRMR) (Peng et al. (2005)) followed by an ensemble model to perform the task.

2 DATASET

We obtained the dataset used for this task from the UCI ML repository Anguita et al. (2013), which includes data from 30 individuals with ages ranging from 19 to 48 years. The dataset contains 562 different features and 10299 data points. There are no missing values in the dataset. For our problem, we use the mRMR feature engineering technique and different classification models to accurately classify human activity into one of the six Activities of Daily Living (ADL), namely walking, walking upstairs, walking downstairs, laying, sitting, and standing.

3 PROPOSED ALGORITHM

The study employs an ensemble/stack classifier to predict class labels from test data, stacking the output of individual classifiers and using an estimator to compute the final prediction. The stack includes k-NN, SVM, Random forest, and MLP as individual models, with Logistic Regression as the final estimator. The proposed algorithm consists of two steps: (i) We use the feature engineering technique known as Maximum Relevance - Minimum Redundancy (mRMR) Doewes et al. (2017). mRMR is a feature selection method that assesses mutual information to determine both the

*These authors contributed equally to this work

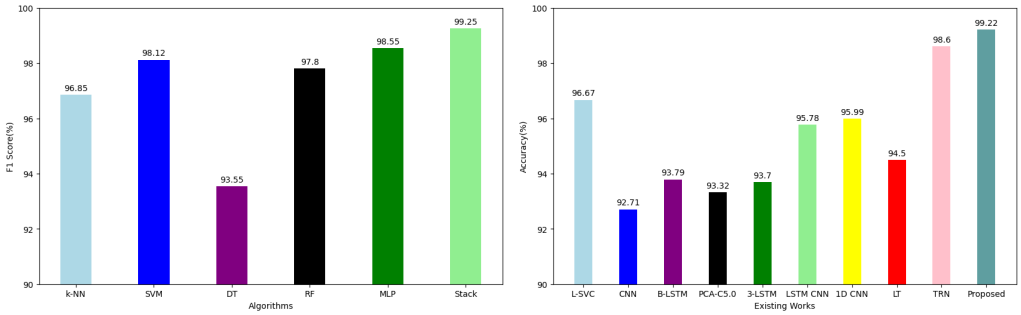

Figure 1: Comparison of the ensemble classifier with the benchmark SOTAs and individual models.

Table 1: Confusion matrix of the proposed algorithm reflecting the potential in detecting different defect classes.

True ↓ Predicted →	Laying	Sitting	Standing	Walking	Walking downstairs	Walking upstairs
Laying	**389**	0	0	0	0	0
Sitting	0	**349**	6	0	0	1
Standing	0	8	**373**	0	0	0
Walking	0	0	0	**344**	0	0
Walking downstairs	0	0	0	0	**280**	1
Walking upstairs	0	0	0	0	0	**309**

relevance and redundancy of features. To implement mRMR, we evaluate the untuned classifiers including the stack model for different values of K, ranging from 1 to 562. The goal was to identify an optimal subset of features that maximized the F1-score without sacrificing model performance. Using this methodolgy we find an optimum value of K that maintains a high F1-score while keeping relevant and non-redundant features. We plot the macro-averaged F1 score against the number of selected features (K), as illustrated in Figure 2 in appendix A.2. Based on the recorded data, the optimal F1-score is achieved when selecting 422 features for the ensemble classifier. Consequently, for the proposed framework, this subset of 422 features is chosen. (ii) After dimensionality reduction using mRMR, we tune the hyperparameters for the classifiers before stacking them to get the most optimized parameters for the individual classifiers using GridSearchCV employing macro-averaged F1-score as the metric. The tuned parameters we obtained are shown in Appendix Table 3. We split the dataset into training and validation sets using 80% as the training data and 20% as the test data in a stratified fashion and trained the model using the best K features obtained from mRMR using the best parameters we got by performing hyperparameter tuning.

4 Experimental Analysis

We have summarized the performance of the classifiers used along with the ensemble model in Figure 1 (left). Furthermore, we compare the performance of the proposed framework in terms of accuracy with several existing state-of-the-arts such as Linear SVC (L-SVC) by Chadha et al. (2023), CNN by Wan et al. (2020), Bidirectional LSTM (B-LSTM) by Yu & Qin (2018), PCA-C5.0 by Elmoudden et al. (2016), 3-Layer LSTM model by Tufek et al. (2020), LSTM CNN by Xia et al. (2020), 1D CNN by Yen et al. (2020), Lightweight Transformers (LT) by EK et al. and Transformer (TRN) by Saidani et al. (2023). We have summarized the comparison in Figure 1 (right). The comparison in figure 1 (left) reflects that the proposed ensemble classifier is successful in increasing the performance of the standalone state-of-the-art machine learning models significantly. The comparison in figure 1 (right) suggests that our proposed ensemble classifier surpasses state-of-the-art deep learning models.

5 Conclusion

The identification of ADLs is very essential for applications such as health tracking, disaster management, and sports. Using this model, we are able to save a significant amount of computational time and resources using feature engineering, thereby reducing overfitting and boosting the model's performance. Our findings suggest the promising application of this approach to low-power edge computing devices for the efficient classification of human activity.

URM STATEMENT

The authors acknowledge that at least one key author of this work meets the URM criteria of ICLR 2024 Tiny Papers Track.

REFERENCES

Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, Jorge Luis Reyes-Ortiz, et al. A public domain dataset for human activity recognition using smartphones. In *Esann*, volume 3, pp. 3, 2013.

Soumiya Chadha, Ishita Raj, and D. Saisanthiya. Human activity recognition for analysing fitness dataset using a fitness tracker. In *2023 International Conference on Computer Communication and Informatics (ICCCI)*, pp. 1–5, 2023. doi: 10.1109/ICCCI56745.2023.10128242.

Surya Dhanraj, Suddhasil De, and Dinesh Dash. Efficient smartphone-based human activity recognition using convolutional neural network. In *2019 International Conference on Information Technology (ICIT)*, pp. 307–312, 2019. doi: 10.1109/ICIT48102.2019.00061.

Afrizal Doewes, Sri Edi Swasono, and Bambang Harjito. Feature selection on human activity recognition dataset using minimum redundancy maximum relevance. In *2017 IEEE International Conference on Consumer Electronics - Taiwan (ICCE-TW)*, pp. 171–172, 2017. doi: 10.1109/ICCE-China.2017.7991050.

Sannara EK, François Portet, and Philippe Lalanda. Lightweight transformers for human activity recognition on mobile devices. URL `http://arxiv.org/abs/2209.11750`.

Ismail Elmoudden, Badreddine Benyacoub, Souad Elbernoussi, and Mounir Ouzir. Modeling human behavior using feature extraction and class prediction. In *2016 IEEE Intl Conference on Computational Science and Engineering (CSE) and IEEE Intl Conference on Embedded and Ubiquitous Computing (EUC) and 15th Intl Symposium on Distributed Computing and Applications for Business Engineering (DCABES)*, pp. 476–479, 2016. doi: 10.1109/CSE-EUC-DCABES.2016.227.

Fabio Hernández, Luis F. Suárez, Javier Villamizar, and Miguel Altuve. Human activity recognition on smartphones using a bidirectional lstm network. In *2019 XXII Symposium on Image, Signal Processing and Artificial Vision (STSIVA)*, pp. 1–5, 2019. doi: 10.1109/STSIVA.2019.8730249.

Hanchuan Peng, Fuhui Long, and C. Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1226–1238, aug 2005. doi: 10.1109/tpami.2005.159.

Oumaima Saidani, Majed Alsafyani, Roobaea Alroobaea, Nazik Alturki, Rashid Jahangir, and Leila Jamel. An efficient human activity recognition using hybrid features and transformer model. *IEEE Access*, 11:101373–101386, 2023. doi: 10.1109/ACCESS.2023.3314492.

Rubén San-Segundo, Julián David Echeverry-Correa, Christian Salamea, and José Manuel Pardo. Human activity monitoring based on hidden markov models using a smartphone. *IEEE Instrumentation Measurement Magazine*, 19(6):27–31, 2016. doi: 10.1109/MIM.2016.7777649.

Nilay Tufek, Murat Yalcin, Mucahit Altintas, Fatma Kalaoglu, Yi Li, and Senem Kursun Bahadir. Human action recognition using deep learning methods on limited sensory data. *IEEE Sensors Journal*, 20(6):3101–3112, 2020. doi: 10.1109/JSEN.2019.2956901.

Shaohua Wan, Lianyong Qi, Xiaolong Xu, Chao Tong, and Zonghua Gu. Deep learning models for real-time human activity recognition with smartphones. *Mobile Networks and Applications*, 25: 743–755, 2020.

Kun Xia, Jianguang Huang, and Hanyu Wang. Lstm-cnn architecture for human activity recognition. *IEEE Access*, 8:56855–56866, 2020. doi: 10.1109/ACCESS.2020.2982225.

Chih-Ta Yen, Jia-Xian Liao, and Yi-Kai Huang. Human daily activity recognition performed using wearable inertial sensors combined with deep learning algorithms. *IEEE Access*, 8:174105–174114, 2020. doi: 10.1109/ACCESS.2020.3025938.

Shilong Yu and Long Qin. Human activity recognition with smartphone inertial sensors using bidir-lstm networks. In *2018 3rd International Conference on Mechanical, Control and Computer Engineering (ICMCCE)*, pp. 219–224, 2018. doi: 10.1109/ICMCCE.2018.00052.

A APPENDIX

A.1 HYPERPARAMETERS

The ranges for the parameters used for hyperparameter tuning for different classifers are shown in Table 2 and the tuned parameters received after running Grid Search on the parameter ranges is shown in Table 5.

Table 2: Hyperparameters of various individual classifiers signifying their respective ranges

Model	**Hyperparameters**
Decision Tree	max_depth: $[\text{None}, 5, 10, 20]$ max_features: $[100, \text{sqrt}, \text{log2}]$ ccp_alpha: $[0.1, 0.01, 0.001]$ criterion: $[\text{gini}, \text{entropy}, \text{log_loss}]$
Random Forest	max_depth: $[\text{None}, 5, 10, 20]$ max_features: $[100, \text{sqrt}, \text{log2}]$ ccp_alpha: $[0.1, .01, .001]$ criterion: $[\text{gini}, \text{entropy}, \text{log_loss}]$
SVM	C: $[0.1, 0.5, 1, 5, 10, 50, 100]$ gamma: $[1, 0.1, 0.01, 0.001, 0.0001, \text{scale}, \text{auto}]$ kernel: $[\text{linear}, \text{poly}, \text{rbf}, \text{sigmoid}]$
KNN	n_neighbors: $n \in (1, 200)$ weights: $[\text{uniform}, \text{distance}]$
MLP	max_iter: $[500]$ hidden_layer_sizes: $[(50,), (100,), (50, 50), (100, 100)]$ alpha: $[0.0001, 0.001, 0.01]$ activation: $[\text{identity}, \text{logistic}, \text{tanh}, \text{relu}]$ solver: $[\text{lbfgs}, \text{sgd}, \text{adam}]$

Table 3: Tuned Parameters for the Classifiers

Model	**Best Parameters after Hyperparameter Tuning**
k-NN	n_neighbors: 4, weights: distance
SVM	C: 100, gamma: 0.1, kernel: rbf
MLP	activation: tanh, alpha: 0.01, max _iter: 500, solver: lbfgs, hidden_layer_sizes: $[100]$
DT	ccp_alpha: 0.001, criterion: entropy, max_depth: 20, max_features: 100
RF	ccp_alpha: 0.001, criterion: entropy, max_depth: none, max_features: sqrt

- **k-NN (k-Nearest Neighbors):**
  - *n_neighbors:* Number of neighbors to be be considered for queries.
  - *weights:* Metric on which weights are assigned to neighbors.
- **SVM (Support Vector Machine):**

  - *C:* Regularization parameter.
  - *gamma:* Kernel coefficient for 'rbf' kernel.
  - *kernel:* Specifies the kernel to be used.
- **MLP (Multi-Layer Perceptron):**
  - *activation:* Activation function for the hidden layer.
  - *alpha:* L2 penalty (regularization term) parameter.
  - *max_iter:* Maximum number of iterations. The solver iterates until this number of iterations.
  - *solver:* The solver for weight optimization.
  - *hidden_layer_size:* The number of neurons in the hidden layers in form of a tuple.
- **Decision Tree:**
  - *ccp_alpha:* The complexity parameter for Minimal Cost-Complexity Pruning.
  - *criterion:* The function to measure the quality of a split.
  - *max_depth:* The maximum depth of the tree.
  - *max_features:* The number of features to consider when looking for the best split.
- **Random Forest:** Same as for Decision Tree.

For MLP, we have used L-BFGS as the solver which requires learning rate to be set to auto as it automatically decides the rate for the next iteration depending on the Jacobian matrix which computes the direction of steepest descent and the Hessian matrix which computes the descent step which repeats until convergence. Momentum is set to 500, max functions is 15000 and maximum number of iterations are set at 500. Initial learning rate is set at 0.0001. L2 penalty parameter is set at 0.01 and we have used Nesterov momentum for accelerated convergence.

A.2 PLOTS

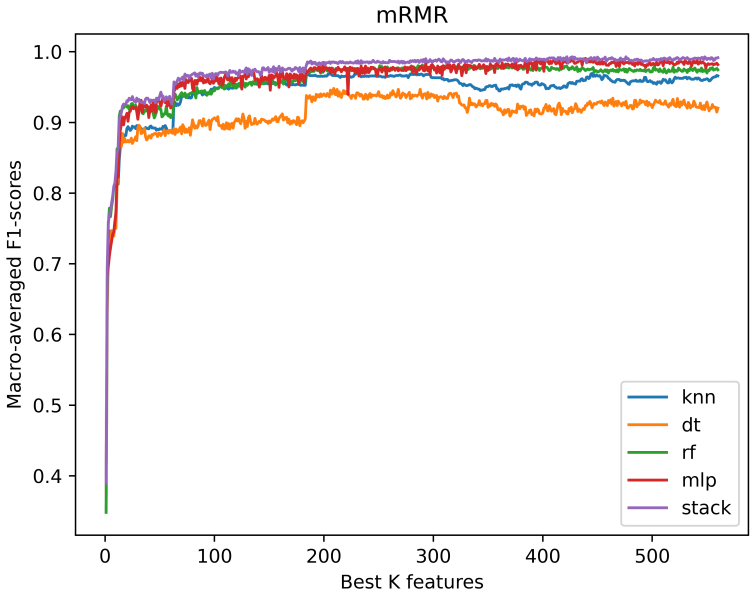

Figure 2: Plot for F1 score vs the best K features for mRMR

The mRMR implementation involves a systematic evaluation of various K values, ranging from 1 to 562, and the F1-scores are computed for each iteration. A plot is generated to show the relationship between the top K characteristics and their corresponding F1-scores. According to the recorded data, the ensemble classifier achieves the highest F1-score when 422 characteristics are selected. Therefore, the subset of 422 features is selected for the proposed framework.

A.3 OTHER RESULTS

A.3.1 MODEL TIMING ANALYSIS

Table 4: Comparison of runtime of proposed model with existing works

Models	Training Time (in minutes)
3-Layer LSTM	20
LSTM-CNN	31.38
Our Work	113 (for Hyperparameter Tuning) + 3 (for training) on the tuned parameters

Although the existing works have not quantitatively reported timing analysis, we have found few mentions of timing analysis, such as in 3-Layer LSTM, which takes 20 minutes for training, and LSTM-CNN, which takes 31.38 minutes judging from the individual epoch timings. The hyperparameter tuning process of our proposed model requires 113 minutes, followed by an additional 3 minutes for training with validation and subsequent testing using the tuned parameters. The inference time of our proposed model is **1.03 ms** per test data point. The ensemble model used in this work has a memory footprint of **27.1 MB**.

Due to the hyperparameter tuning involved, with each classifier being tuned for a combination of around 108 parameters, the running time for the hyperparameter is large. Nevertheless, parameter adjustment is necessary only once for the dataset. After the initial tuning, further runs just need to be performed using the optimized parameters. The reason for this is that the addition of additional data points to the test set does not necessitate parameter adjustments.

A.3.2 EXPERIMENT ON DISJOINT DATASETS

As part of our experiment, we adjusted the criteria for dividing participants into disjoint sets for training and validation and subsequently for testing as well, avoiding any overlap across sets. 26 subjects were included in the training set, 2 in the validation set, and the remaining 2 in the testing set. Using this method, we achieve an accuracy of **97.2%** and a macro-averaged F1-score of **97.7%**, which is still better than the state-of-the-arts in the literature considering the limited number of subjects in the dataset. However, it suffers heavily due to model bias, and therefore it can be improved for real-world applications by including more subjects in the training set so as to provide more diversity in the real-world dataset. The tuned parameters for this experiment are summarized in table 5.

Table 5: Tuned Parameters for the Classifiers for the disjoint set experiment

Model	Best Parameters after Hyperparameter Tuning
k-NN	n_neighbors: 10, weights: distance
SVM	C: 10, gamma: 1, kernel: linear
MLP	activation: identity, alpha: 0.0001, max _iter: 500, solver: lbfgs, hidden_layer_sizes: [100]
DT	ccp_alpha: 0.001, criterion: entropy, max_depth: 20, max_features: 100
RF	ccp_alpha: 0.001, criterion: entropy, max_depth: 20, max_features: log2

A.4 METRICS USED FOR EVALUATION

We summarize the metrics used for the evaluation of the models at different phases of our work. For each possible value of K (the number of features), we ran every model and the ensemble model, and for every iteration, we recorded its F1-score. Our goal was to identify an optimal subset of features that would maximize the F1-score, a robust metric that considers both precision and recall. While reducing the dimensionality of the data can simplify the model and improve computational

efficiency, it can also lead to a loss of performance if important features are excluded. To overcome this, we adopted a similar approach to decrease the feature subset without losing the F1-score metric. Through this rigorous process, we were able to identify an optimum value of K that maintains a high F1-score while keeping features that were not only relevant but also non-redundant. For parameter tuning, the F1-score metric was used for evaluating the performance of both the individual models and the stack model. The performance of the final model using the classifiers with the tuned parameters was measured using both accuracy and the macro-averaged F1-score **on the test dataset**.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \times 100 \tag{1}$$

$$\text{Precision}_i = \frac{\text{True Positives}_i}{\text{True Positives}_i + \text{False Positives}_i} \tag{2}$$

$$\text{Recall}_i = \frac{\text{True Positives}_i}{\text{True Positives}_i + \text{False Negatives}_i} \tag{3}$$

$$\text{F1-score}_i = 2 \times \frac{\text{Precision}_i \times \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i} \tag{4}$$

$$\text{Macro-averaged F1-score} = \frac{1}{N} \sum_{i=1}^{N} \text{F1-score}_i \tag{5}$$

where i stands for the i-th class in the data.

A.5 Source Code and Dataset

The source code for the above implementation has been made available by the authors and can be accessed at https://anonymous.4open.science/r/stacking-model-for-har-268D/. The dataset in the UCI ML repository can be accessed at https://archive.ics.uci.edu/dataset/240/human+activity+recognition+using+smartphones.

A.6 Abbreviations

HAR	Human Activity Recognition
CNN	Convolutional Neural Network
SVM	Support Vector Machine
MLP	Multi-Layer Perceptron
SVC	Support Vector Classifier
PCA	Principal Component Analysis
EHARS	Efficient Human Activity Recognition System
mRMR	Maximum Relevance - Minimum Redundancy
T-SNE	T-distributed Stochastic Neighbor Embedding
FCN-LSTM	LSTM Fully Convolutional Networks