HUMAN ACTIVITY RECOGNITION USING DEEP LEARNING

A PREPRINT

Utsav Panchal
Information Technology
University of Stuttgart
st184584@stud.uni-stuttgart.de

Gautham Mohan
Electrical Engineering
University of Stuttgart
st184914@stud.uni-stuttgart.de

February 13, 2024

ABSTRACT

Human Activity Recognition (HAR) is a problem that is an active research field in pervasive computing. An HAR system has the main goal of analyzing human activities by observing and interpreting ongoing events successfully. This paper presents a Deep Learning-based approach for Human Activity Recognition (HAR) using accelerometer and gyroscope sensor data. It explores various neural network architectures including LSTM, bidirectional LSTM, GRU, and 1D CNN, and evaluates their performance on two datasets: HAPT and Real World HAR. Additionally, ensemble learning methods are employed for improved accuracy and reducing the biases of individual models. Furthermore, the study demonstrates the deployment of the model in an Android application, showcasing real-time activity detection.

1 Introduction

Human Activity Recognition (HAR) is a problem aimed at analyzing activities and classifying them. This can be done by visual methods and non-visual sensor-based methods. The sensor-based methods can be further divided into external and body-worn sensors. While visual methods and external sensor-based methods are used in an external environment for surveillance and behavioural analysis body-worn sensors concentrate on areas such as healthcare and human-machine interactions in industrial applications [1].

The goal of the project is to use Deep learning techniques to identify various human activities. The project uses time-series data which is acquired from accelerometer and gyroscope sensors in smartphones. The first section of the paper discusses the datasets and preprocessing methods. Then we present different architectures used in the project which is followed by various ensemble techniques to reduce the generalization errors. In the end, we integrated the model into an Android application to detect real-time activities performed by the user.

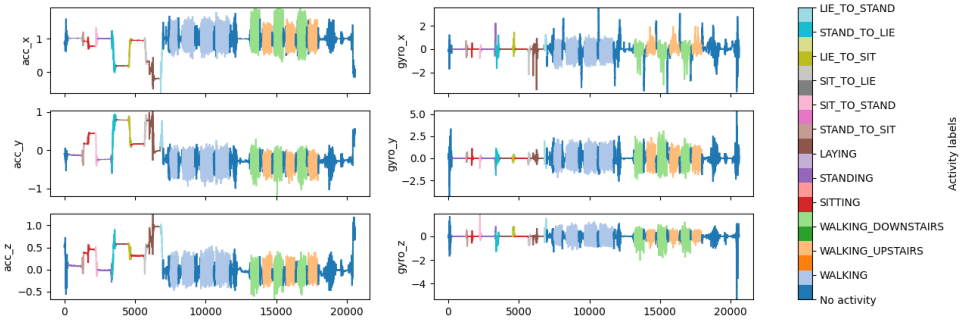

Figure 1: Example of the accelerometer and gyroscope signals and their labels

2 Datasets

We have trained our model on two datasets.

2.1 Human Activity and Postural Transitions Dataset (HAPT)

The HAPT dataset contains data from the tri-axial accelerometer and gyroscope of a smartphone, both captured at a frequency of 50 Hz. The dataset consists of 12 activities which includes standing, sitting, lying. walking, walking downstairs, walking upstairs, stand-to-sit, sit-to-stand, sit-to-lie, lie-to-sit, stand-to-lie, and lie-to-stand.

2.2 Real World dataset 2016 (RWHAR) ^{up}

The RWHAR dataset contains signals from various smartphone embedded sensors like the accelerometer, Global-Positioning-System (GPS), gyroscope, light, magnetic field and sound. The dataset consists of 8 activities which includes climbing downstairs, climbing upstairs, jumping, lying, standing, sitting, running/jogging, and walking. Smartphones were placed at different body positions: chest, head, shin, thigh, upper arm, waist, and a smartwatch at the forearm that were recorded simultaneously.

3 Data Preprocessing ^{up}

The **HAPT** data contains accelerometer and gyroscope data from 30 users. We read the data of each user and create a data-frame which consists of combined data from both sensors. Then we normalize the data by z-score normalization so that large offsets do not affect training performance. Further, we remove 5 seconds of data from the beginning and the end. We use the Sequence to Label method (S2Q), hence we convert the dataset into separate windows with 50% overlap and of length 100 samples or 2 seconds, then these windows were assigned to the maximum occurring activities in that window. We separate the data into train, test and validation. Furthermore, we also resample the data to remove the imbalance between the classes [2]. At the end of the pipeline, we convert our data into TFRecord format.

The **RWHAR** dataset contains 15 proband data belonging to 15 different subjects. Each proband data covers acceleration, GPS, gyroscope, light, magnetic field, and sound level data of different activities while the probands were placed at different body parts of the subject. We precisely separated the accelerometer and gyroscope data for individual users. Then we procedded with similar procedures as described above.

4 Architectures

In this project we employed four different architectures. Given the inherent sequential dependencies within the data, our primary emphasis was on variations of recurrent neural networks (RNNs), including Long Short-Term Memory (LSTM), bidirectional LSTM, and Gated Recurrent Unit (GRU) models [3]. Additionally, we explored the efficacy of a one-dimensional Convolutional Neural Network (CNN) model. Intriguingly, our experimentation revealed that the CNN model yielded comparable, and sometimes, superior results compared to the recurrent neural network counterparts.

4.1 LSTM Model ^{up}

LSTM models are variants of recurrent neural networks (RNNs) with an additional cell state over the hidden state. It also has input, output and forget gates that can help the model to be selective about the data it chooses to retain. These mechanisms make the model less vulnerable to vanishing gradients during backpropagation and help learn larger sequences when compared to regular RNNs. The LSTM model used within the project has 3 LSTM layers and 2 final dense layers.

4.1.1 Bi-directional LSTM Model ^{gm}

Bi-directional LSTMs process the same input sequence in two direction one in forward direction through time and the other in backward direction through time. This helps the model consider both past and future context in the data and hence, extract features which are more representative from the data. In the project we are using 4 bi-directional LSTM layers combined with 2 dense layers for final classification.

---

up: utsav panchal
gm: gautham mohan

4.1.2 GRU Model ^{gm}

They are a variant of RNN which uses 2 gates, update and reset to select the information retained by the model during training. Unlike LSTMs, GRU models do not have a separate cell state and only have the hidden state which reduces its efficiency when the length of the sequence increases, however, they require fewer parameters and are generally faster than LSTMs. The GRU model used in the project has 4 GRU layers and 2 dense layers for final classification.

4.1.3 One dimensional CNN ^{gm}

One-dimensional kernels can extract information from time series data in the same way as 2-dimensional kernels extract information from image data. This method is computationally more efficient as the operations can be easily parallelised and if the patterns in the data are localised then convolutional layers can successfully capture this. However, it cannot be used to capture long-term dependencies and will not perform well in cases where the length of the input sequence is very large. The CNN model used in the project has 2 convolutional layers, 1 global average pooling layer and finally 2 dense layers.

5 Training and Evaluation $^{up-gm}$

The dataset when divided into windows, might be mixed i.e. it might contain more than one activity. This is because the data is recorded in a continuous format where it moves from one activity to another without any particular distinction. The model was trained with both the mixed windows and also by avoiding such mixed windows while evaluation was conducted by strictly avoiding such windows. It was seen that incorporating the mixed windows into training yielded higher accuracy when compared to avoiding it, this could be attributed to the loss of training data that occurs when dropping such mixed windows.

To counter the effects of unbalanced dataset we resampled the dataset and tried class weights. However, class-weights are assigned based on the number of data points for different classes and after resampling the minority classes to the size of the majority class the weights becomes equally distributed [2].

6 Ensemble Learning ^{gm}

Ensemble learning was conducted through hard voting and soft voting methodologies, utilizing all four models to compose the ensemble model. These models were loaded from their respective checkpoints that yielded the highest accuracies before being combined in the ensemble. They were then evaluated on the test dataset.

7 Results HAPT and HAR

- Table 1 illustrates the best results of the four models after training on the HAPT dataset.
- The accuracies were consistent across the runs and did not deviate much from the best result.

Model	Balanced Accuracy	Unbalanced Accuracy	F1-Score
LSTM	87.18	94.51	85.14
Bidirectional LSTM	86.51	94.57	84.43
GRU	84.35	94.74	79.86
1D Convolutional	88.81	96.39	86.03
Ensemble-Soft voting	89.65	96.06	88.37
Ensemble- Hard Voting	89.37	95.73	87.52

Table 1: HAPT Results

- Table 2 shows the sparse categorical accuracies for the Real World (HAR) dataset. We have separate results for each body part.
- We have seen variations by 15% in the result and they were particularly hard to reproduce.

up: utsav panchal
gm: gautham mohan

	LSTM	Bi-directional LSTM	GRU	1D-Convolutional
Chest	66.74	92.03	90.98	75.33
Forearm	73.29	69.46	57.84	72.09
Head	68.91	66.97	65.10	68.68
Shin	82.90	70.16	71.71	72.70
Thigh	65.88	62.61	64.89	66.45
Upperarm	68.16	63.10	65.99	71.60
Waist	65.33	72.08	73.27	72.45

Table 2: RWHAR Results

- Figure 2 illustrates the results of the LSTM model trained on the HAPT dataset as a heatmap, the red box indicates the transition activities. This heatmap should be read along with table 3 showing the recall values for the model.
- It can be seen that the recall value for some tasks such as "stand-to-lie" is particularly low and is confused with "sit-to-lie".
- This could be attributed to the very low train samples for the transition activities compared to the activities outside the box and the similarity between the activities.

Activity	Recall
walking	100.00%
walking_upstairs	100.00%
walking_downstairs	85.79%
sitting	82.86%
standing	99.29%
laying	95.58%
stand_to_sit	88.24%
sit_to_stand	100.00%
sit_to_lie	73.08%
lie_to_sit	90.91%
stand_to_lie	36.67%
lie_to_stand	76.00%
Balanced Accuracy	85.70%
Sparse Categorical Accuracy	93.05%

Table 3: Recall for each activity

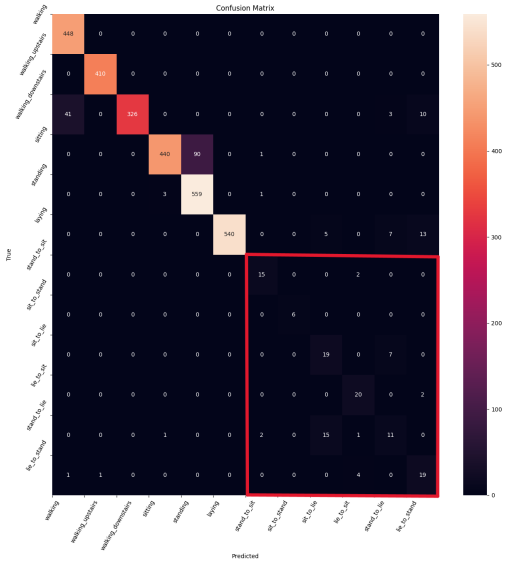

Figure 2: Heatmap, true labels on Y axis and predictions on X axis.

8 Android Application $^{up-gm}$

To test our model on real time data from the user we have implemented an android application. We used TensorflowLite (tflite) to convert our model into a smaller and efficient format such that it can be used in low level platforms like android.

We used the model trained on Real world har dataset (RWHAR) specifically the chest and upperarm bodyparts.The android application uses real time data from accelerometer and gyroscope sensors. As the sampling rate is very high in

up: utsav panchal
gm: gautham mohan

android version 11+, we need to provide special permissions to access sensor hardware at a high sampling rate. We read the data in separate lengths which is equal to window length used in the input pipeline while training the model.

The android application is able to detect 8 positions:climbing down, climbing up, jumping, lying, standing, sitting, running, walking. We tested the application in real time and it is successfully able to detects activities like standing, jumping and running.

NOTE: The GitHub folder har-app contains the source code. The code is uploaded on a single commit. Please note that while developing the application both the team members have done equal work.

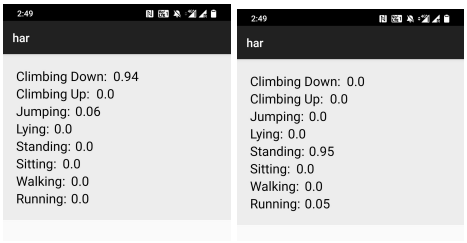

Figure 3: Screenshots of Android application

References

[1] *Deep Learning Lab*. Institut für Signalverarbeitung und Systemtheorie, 2022.

[2] A. Thakur, "Simple ways to tackle class imbalance," WB, 07 2020. [Online]. Available: https://wandb.ai/authors/class-imbalance/reports/Simple-Ways-to-Tackle-Class-Imbalance--VmlldzoxODA3NTk

[3] Y. Tang, "Three types of recurrent neural networks," Medium, 02 2022. [Online]. Available: https://pub.towardsai.net/three-types-of-recurrent-neural-networks-567b4e9c4261