Multimodal HAR Classification Model

Melis E. Durgut
School of Engineering
Rutgers University
Piscataway, New Jersey
med284@rutgers.edu

Gunjan Adya
School of Engineering
Rutgers University
Piscataway, New Jersey
ga402@rutgers.edu

***Abstract*— This research seeks to develop a predictive system for human-activity-based accelerometer and GPS data. Inspired by the "Location-based Daily Human Activity Recognition using Hybrid Deep Learning Network"2 article, our roadmap involves utilizing a CNN for location-based classification and employing an LSTM encoder-decoder to predict human activities. The combination of spatial insights from CNN and temporal dependencies captured by LSTM aims to enhance the accuracy of real-world human activity recognition.**

Keywords—CNN, LSTM, multimodal, sensor data, human activity recognition

I. Introduction

In the era of the Internet of Things (IoT), the fusion of sensor technologies and machine learning has paved the way for innovative applications across various domains. One intriguing challenge within this intersection is the development of machine learning models capable of processing multimodal data from diverse sensors. In this context, our research focuses on creating a robust model for action classification using data from accelerometers and GPS devices. We aim to harness the potential of multimodal sensor data, particularly from accelerometers and GPS devices, to construct a machine learning model proficient in precisely categorizing human actions.

The significance of this problem becomes evident in its direct applicability to real-world scenarios. With the proliferation of wearable devices and smart technologies, the need for accurate and context-aware action classification has become increasingly pertinent. Understanding human actions through multimodal sensor data can enhance applications ranging from health monitoring and fitness tracking to context-aware services in smart cities.

Moreover, the integration of accelerometer and GPS data allows for a more comprehensive understanding of human activities. Accelerometer data provides insights into physical movements, while GPS data adds a spatial and contextual dimension, enabling a more nuanced interpretation of actions in different environmental settings. Achieving accurate classification in this context contributes to the advancement of human-centric IoT applications.

II. Related Work

In "Location-based Daily Human Activity Recognition using Hybrid Deep Learning Network", S. Mekruksavanich et al. address the challenging task of Human Activity Recognition (HAR) through the introduction of a novel deep learning framework. The proposed model, termed the location-based CNN-LSTM hybrid, combines Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) architectures, leveraging both time-series data from smartwatch sensors and additional location information. The study validates the framework using the DHA dataset, a wristwatch accelerometer dataset and demonstrates superior accuracy (96.75%) compared to alternative deep learning approaches. The authors highlight the increasing relevance of HAR in applications such as wellness tracking and biometric identification systems, emphasizing the potential of their location-based CNN-LSTM model in addressing the challenges associated with traditional machine learning approaches. [2]

We intend to use this research as a framework to build our own HAR model using accelerometer and GPS data to classify human activity into one of 8 categories.

III. Dataset Description

A. *The Dataset*

For this human activity recognition project, we used the Real World (2016) dataset from the University of Mannheim. This dataset collected data across 6 modalities using 6 sensors per subject for 8 human body activities:

- Climbing up
- Climbing down
- Jumping
- Lying
- Standing
- Sitting
- Running/Jogging
- Walking

The dataset was collected using fifteen subjects of ages 31.9 ± 12.4, a height of 173.1 ± 6.9, and a weight of 74.1 ± 13.8, eight males and seven females. For each activity, data was simultaneously collected across body positions chest, forearm, head, shin, thigh, upper arm, and waist. Each subject performed each activity for roughly 10 minutes, except for jumping due to physical exertion (Jumping was performed for around 1.7 minutes). [1]

For our purposes, we used two modalities, accelerometer, and GPS, from the upper arm sensor of subject 1. This choice was made due to challenges with processing the entire dataset due to its large size.

B. *Challenges*

We predominantly faced challenges in preprocessing our data. Our first issue came in the form of the dataset's size. We initially intended to build a model that processed a more significant part of the entire dataset. However, many issues quickly arose with this approach: Firstly, the uncompressed dataset takes up a significant amount of disk space. Furthermore, the inclusion of many modalities across many sensors led to the creation of an unwieldy data frame with dozens of features. Finally, the size of the dataset made preprocessing very difficult.

As the data was organized into a series of many CSV files, they first had to be extracted and combined into Pandas dataframes. Because the timestamps in the raw data were not aligned, we also had to group data by timestamps within a certain tolerance to remove NaN values. Overall, the bulk of our time was spent on this first step.

IV. Methodology

A. Preprocessing

Due to the challenges described, we decided to use a very small portion of the dataset: the upper arm accelerometer and GPS data from subject 1. Initially, accelerometer data was combined into a single dataframe, and GPS data was combined into another dataframe. These dataframes were then sorted by timestamp and combined along the timestamp column with a tolerance of 5ms. This tolerance was added to eliminate NaN values caused by misaligned time series.

Once our modalities were combined into a single dataframe, this dataframe was further split 80/20 into training and testing data. The X values were normalized to ensure that all features contribute proportionally to the model training process and prevent any one feature from dominating the learning process due to its larger scale. The Y values were encoded using one-hot encoding, because they initially were strings denoting the activity associated with each datapoint. Finally, both X and Y values were reshaped to fit into a sequential Keras model.

B. CNN Architecture

A CNN (Convolutional Neural Network) is a class of deep neural networks designed for processing structured grid data, such as multidimensional arrays. When applying CNN to sensor data, the architecture needs to be adapted to handle one-dimensional data since sensor data is often represented as time series or sequences. For the adaptation process, we used 1D convolutional layers instead of 2D convolutional layers. The convolutional layers are also helped with capturing local patterns and features in the sensor data over time.

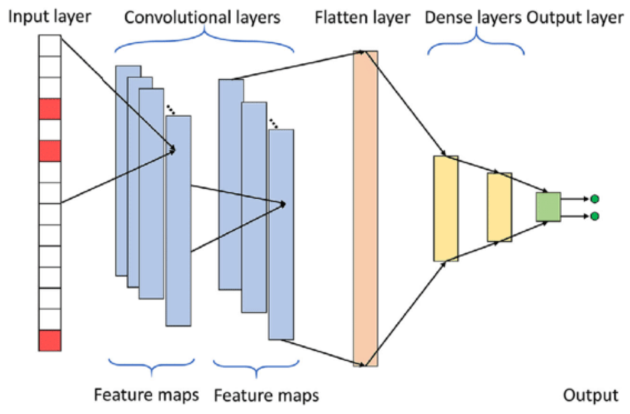


Fig. 1. A diagram of the CNN architecture.

After the convolutional layers, we applied a non-linear activation function called ReLU (Rectified Linear Unit), allowing the neural network to learn complex patterns and representations.

The ReLU activation is defined as:

$$f(x) = \max(0, x)$$

If the input x is greater than zero, then the output is equal to the input. If the input x is less than or equal to zero, then the output x is equal to zero.

Between the convolutional layers and the pooling layers, we included a dropout. This is a regularization technique that helps prevent overfitting. The dropout technique involves randomly setting a fraction of neurons (input units) to zero during each update of the training. The key idea behind dropout is to improve the generalization of the model by making it more robust and less sensitive to the exact configuration of the training data.

After the convolutional layers, there's a pooling layer that down-samples the temporal dimensions of the data. Specifically, we used the max pooling layer that takes the maximum value in a region, which helps reduce the length of the sequence while preserving important information.

We then added a flatten layer to flatten the output from the last pooling layer into a one-dimensional vector, preparing the data for input into the dense layers. Fully connected dense layers capture global independencies and relationships in the sensor data, helping the algorithm to learn.

Finally, after the dense layers, we had a SoftMax layer to help the output layer with classification. Then, the output layer produces the final predictions based on the learned features.

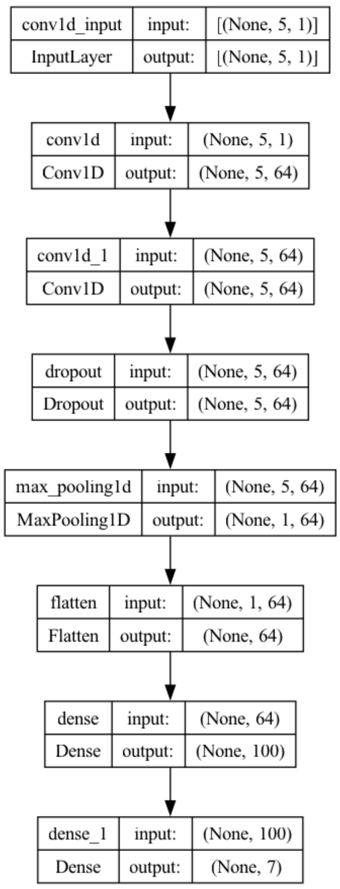

conv1d_input | input: | [(None, 5, 1)]
InputLayer | output: | [(None, 5, 1)]
conv1d | input: | (None, 5, 1)
Conv1D | output: | (None, 5, 64)
conv1d_1 | input: | (None, 5, 64)
Conv1D | output: | (None, 5, 64)
dropout | input: | (None, 5, 64)
Dropout | output: | (None, 5, 64)
max_pooling1d | input: | (None, 5, 64)
MaxPooling1D | output: | (None, 1, 64)
flatten | input: | (None, 1, 64)
Flatten | output: | (None, 64)
dense | input: | (None, 64)
Dense | output: | (None, 100)
dense_1 | input: | (None, 100)
Dense | output: | (None, 7)

Fig. 2. The CNN model diagram

For the convolutional layers, we added padding of `stride=1` to get more accurate results. We used a dropout value of 0.5, and we picked `kernel_size=4` and `pool size=4` to get better results.

Early stopping was implemented with `patience=3` to stop training when the validation performance plateaus. This is useful to avoid overfitting when training a machine learning algorithm with an iterative method.

C. *LSTM Architecture*

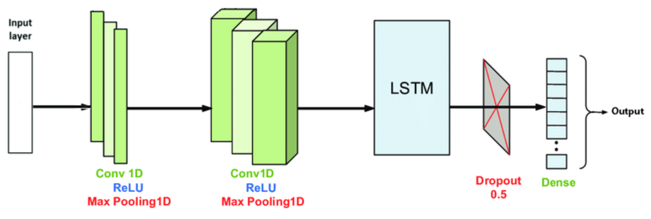


Fig. 3. A diagram of the LSTM architecture.

Long Short-Term Memory (LSTM) is a type of recurrent neural network architecture that is especially effective in handling time-series data or sequences, making it a match for sensor data applications. LSTMs are designed to capture and learn long-term dependencies in sequential data, addressing the vanishing gradient problem. LSTMs include memory cells that can store information over long periods of time. These memory cells are equipped with gates to control the flow of information, allowing them to decide when to read, write, and erase information.

LSTMs use built-in forget gates to erase information. The forget gate determines how much of the information from the previous state should be discarded. It applies a sigmoid activation function to the combination of the current input and the previous hidden state, outputting values between 0 and 1.

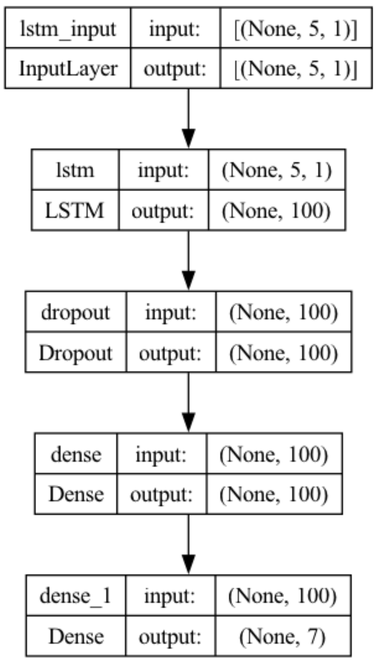
The hidden state in LSTMs addresses the limitations of the traditional RNNs by introducing a more sophisticated memory mechanism. The hidden state is split into two components: the cell state and the output (also known as hidden) state. The cell state represents the long-term memory of the network and can carry information across long sequences, which fixes the problem of long-term dependency. It is updated through the forget gates, input gates, and output gates. The output state represents the short-term memory or the relevant information for the current time step. It is a filtered cell state and is passed to the next step.

The input gate in LSTMs determines what new information should be stored in the memory cell. It also consists of a sigmoid activation function that decides which values to update, and in addition to that, it consists of a tanh activation function that creates a vector of new candidate values. Based on the input and the forget gates, LSTMs update their memory cells.

The output gate the LSTMs have decides what information should be the output as the hidden state. It also applies a sigmoid activation function to the combination of the current input and the updated memory cell, and the result is multiplied by the tanh of the new memory cell.

We also used the dropout technique in our LSTM model to encourage the network to learn more robust and generalized representations by preventing overfitting. The dropout technique in LSTMs can be applied to the input and output connections of the LSTM units as well as the connections between the memory cells. The dropout technique randomly “drops out” a fraction of the neurons during training, meaning that their contributions are temporarily removed from the equation.

We added dense layers on top of the LSTM layers to introduce non-linearity to the network and for sequence classification. The dense layers follow the LSTM layer, with ReLU activation in the first dense layer and SoftMax activation in the output layer for a classification task.


lstm_input | input: | [(None, 5, 1)]
InputLayer | output: | [(None, 5, 1)]
lstm | input: | (None, 5, 1)
LSTM | output: | (None, 100)
dropout | input: | (None, 100)
Dropout | output: | (None, 100)
dense | input: | (None, 100)
Dense | output: | (None, 100)
dense_1 | input: | (None, 100)
Dense | output: | (None, 7)
Fig. 4. The LSTM model diagram

For the optimizer, we used `learning_rate= 0.001`. We used a dropout value of 0.5.

Early stopping was implemented with `patience=3` to stop training when the validation performance plateaus.

D. CNN-LSTM Architecture

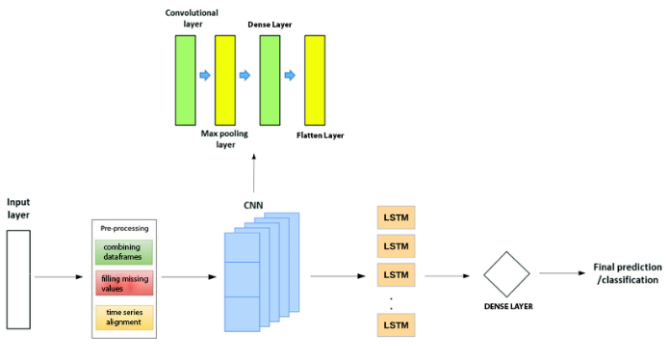
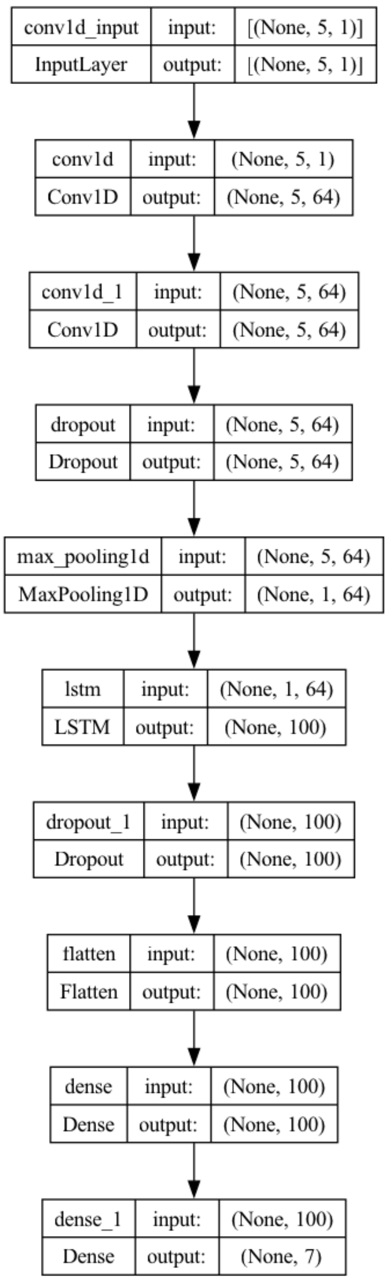


Fig. 5. A diagram of the LSTM architecture

The decision to integrate a Convolutional Neural Network (CNN) with a Long Short-Term Memory (LSTM) network in our proposed model stems from the complementary strengths of these two architectures in handling sequential and spatial data.

CNNs excel in extracting hierarchical spatial features from input data, making them adept at recognizing patterns and spatial relationships in time-series data. This capability is particularly valuable when dealing with sensor data, where identifying intricate patterns of human movements is crucial. On the other hand, LSTMs are well-suited for capturing temporal dependencies and long-range dependencies within sequences.

By combining the spatial feature extraction capabilities of CNNs with the temporal modeling prowess of LSTMs, our hybrid architecture aims to leverage the synergies between these networks, providing a robust framework for Human Activity Recognition. This fusion allows our model to effectively capture both spatial and temporal aspects of the data, enhancing its ability to discern and classify complex human activities accurately.


conv1d_input | input: | [(None, 5, 1)]
InputLayer | output: | [(None, 5, 1)]
conv1d | input: | (None, 5, 1)
Conv1D | output: | (None, 5, 64)
conv1d_1 | input: | (None, 5, 64)
Conv1D | output: | (None, 5, 64)
dropout | input: | (None, 5, 64)
Dropout | output: | (None, 5, 64)
max_pooling1d | input: | (None, 5, 64)
MaxPooling1D | output: | (None, 1, 64)
lstm | input: | (None, 1, 64)
LSTM | output: | (None, 100)
dropout_1 | input: | (None, 100)
Dropout | output: | (None, 100)
flatten | input: | (None, 100)
Flatten | output: | (None, 100)
dense | input: | (None, 100)
Dense | output: | (None, 100)
dense_1 | input: | (None, 100)
Dense | output: | (None, 7)
Fig. 6. The CNN-LSTM model diagram

We didn't change the parameters and used the same parameters of CNN and LSTM for our CNN-LSTM model.

V. Results

The results revealed intriguing insights into the strengths of each architecture. Surprisingly, the LSTM model exhibited a 97.33% accuracy. This was the highest accuracy among the three, outperforming both the CNN and the combined CNN-LSTM approach. This unexpected outcome underscores the significance of temporal dependencies and the effectiveness of LSTMs in capturing intricate patterns within sequential data. While the individual accuracy of the CNN and CNN-LSTM models yielded high accuracy as well, the superior performance of the standalone LSTM demonstrates the importance of maintaining temporal relationships in HAR data.

One aspect of the models to note as well is the difference in the amount of epochs necessary before the validation accuracy begins to plateau. While all three models returned similar accuracy figures, the LSTM + CNN model was able to achieve this result much more quickly than the LSTM alone, requiring only 14 epochs. Therefore, the combination of CNN and LSTM is very useful in creating a more efficient model.

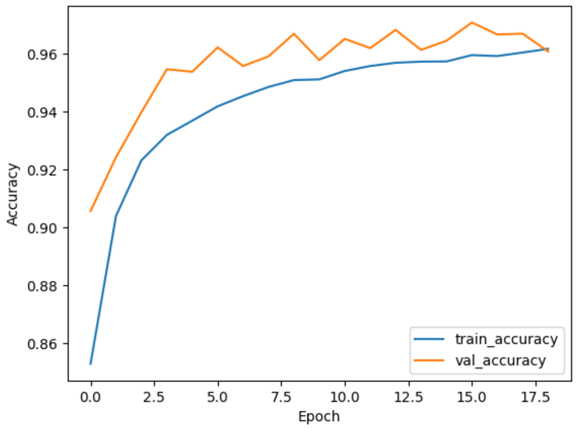


Fig. 7. The CNN accuracy graph

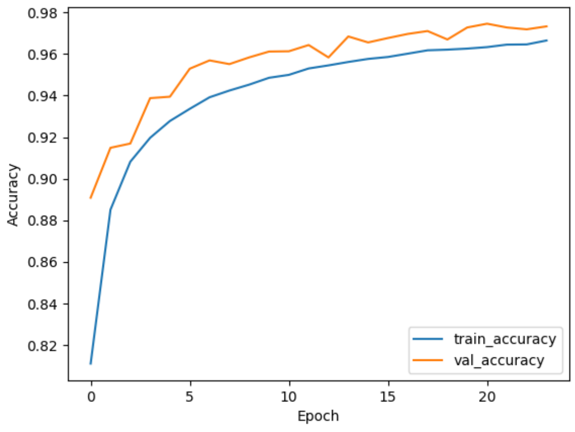


Fig. 8. The LSTM accuracy graph

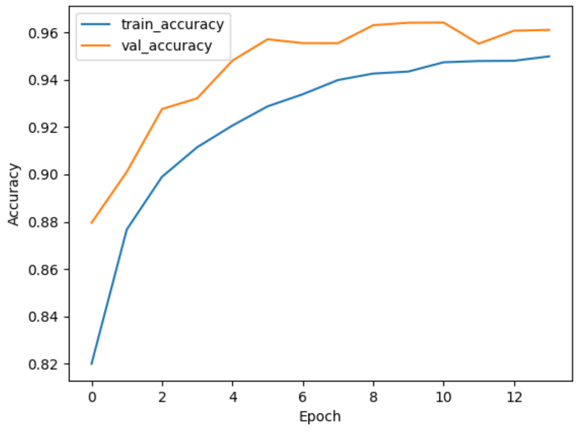


Fig. 9. The CNN-LSTM accuracy graph

Model Architecture	Accuracy	Epochs
CNN	0.9608	19
LSTM	0.9733	24
CNN + LSTM	0.9611	14

Fig. 10. Validation Accuracy data for different model architectures

VI. Future Work

Looking ahead, our future research endeavors aim to broaden the scope of our proposed model by incorporating data from a more extensive range of subjects. This expansion will not only enhance the diversity and representativeness of our dataset but will also contribute to a more comprehensive understanding of human activity patterns.

Moreover, we envision the development of a real-time classification system that can process sensor data as it is generated, enabling instantaneous recognition and categorization of ongoing human actions. This advancement would have significant implications for applications requiring timely and dynamic responses, such as in healthcare monitoring or smart environments. By evolving our model to operate in real-time scenarios and accommodating data from a larger pool of subjects, we aspire to contribute to the practical implementation of Human Activity Recognition systems in diverse and dynamic contexts.

Acknowledgment

We would like to thank Professor Jorge Ortiz for this opportunity.

References

[1] University of Mannheim: Data and Web Science Group, DataSet – RealWorld (HAR) (2016)

[2] S. Mekruksavanich et al, "Location-based Daily Human Activity Recognition using Hybrid Deep Learning Network," 2021 18th International Joint Conference on Computer Science and Software

Engineering (JCSSE), Lampang, Thailand, 2021, pp. 1-5, doi: 10.1109/JCSSE53117.2021.9493807.