Bioengineering

https://www.aimspress.com/journal/Bioengineering

AIMS Bioengineering, 11(4): 527–560.
DOI: 10.3934/bioeng.2024024
Received: 27 May 2024
Revised: 28 October 2024
Accepted: 15 November 2024
Published: 27 November 2024

Research article

Human activity recognition: an approach 2D CNN-LSTM to sequential image representation and processing of inertial sensor data

Wallace Camacho Carlos[1*], **Alessandro Copetti**[1], **Luciano Bertini**[2], **Leonard Barreto Moreira**[1], **Otávio de Souza Martins Gomes**[2]

[1] Science and Technology Institute, Fluminense Federal University (ICT-UFF), Rio das Ostras, RJ, Brazil

[2] Institute of Systems Engineering and Information Technology, Federal University of Itajubá (IESTI-UNIFEI), Itajubá, MG, Brazil

[*] **Correspondence:** Email: wallacecamacho@id.uff.br; Tel: +5521983352503.

Abstract: The field of human activity recognition, abbreviated as HAR, benefits significantly from deep learning by addressing the complexity of human behavior and the vast volume of data produced by sensors. This work adopted the strategy of converting inertial data, such as accelerometer and gyroscope signals, into 2D images through recurrence plots. This approach facilitated the effective exploration of data input and neural network architectures. By utilizing the recent history of movements as input for the models, this study evaluated the impact of this methodology on HAR using two adapted architectures: 2D convolutional neural networks combined with long short-term memory layers (2D CNN-LSTM) and standalone 2D convolutional neural networks (2D CNN). Their performances were compared with other state-of-the-art deep learning models. The contributions of this study were threefold: the handling of input data, the development of the two network architectures for HAR, and the high accuracy achieved, ranging from 97% to 98%, on the public University of California, Irvine human activity recognition dataset (UCI-HAR). These results highlighted the benefit of incorporating temporal data to enhance accuracy in activity classification.

Keywords: activity recognition; recurrence plot; 2D-DL; CNN; LSTM

1. Introduction

The methods employed for human activity recognition (HAR) are often framed within the literature as a pattern recognition problem, where a set of actions performed repetitively can be classified as an activity. This includes actions such as walking, running, and jumping, among others. A well-established method for *HAR*, as documented in the literature, involves using data from inertial sensors as input for machine learning and deep learning techniques [1]. Deep learning approaches, in particular, have shown promise for *HAR* through wearable devices, offering performance enhancements over existing methodologies and possessing the potential to uncover features associated with the dynamics of human movement. This capability is advantageous for extending *HAR* applications to more complex tasks [2].

Recently, the 2D-DL (two-dimensional deep learning) technique, where the input for DL is inertial data converted into 2D images, has emerged as a promising candidate for HAR [3]. These new approaches leverage image generation and computer vision resources for activity recognition, representing a more complex context compared to classical machine learning approaches, which rely on feature extraction from raw sensor data. Despite this added complexity, the use of 2D-DL networks presents significant advantages by allowing the application of powerful architectures and tools developed in the field of computer vision, such as CNNs, which are highly effective for detecting complex patterns in images. This approach has shown promise in recent studies that explore the potential of 2D-DL for HAR, health monitoring, and gesture detection, capturing both the temporal characteristics and spatial relationships of inertial sensor data. In parallel, object detection and segmentation in 2D and 3D spaces have also significantly evolved with models like YOLO (You Only Look Once) and its more recent variants, enabling real-time object identification with an efficient balance between precision and speed [4]. This progress is especially relevant to HAR, where detecting and segmenting moving subjects and their interactions with the environment can enhance activity recognition even in complex and dynamic environments. Additionally, the incorporation of attention mechanisms in convolutional networks has been a critical advancement, allowing models to focus on specific regions of the image, which enhances segmentation and detection of more complex patterns, particularly in challenging scenarios. As explored in [5], this approach has been fundamental in the growing sophistication of techniques in computer vision.

A relatively unexplored challenge in applying deep learning techniques to *HAR* is the complex task of integrating specific activity data into the neural network. Traditionally, the model input comprises a single image derived from inertial data. However, as highlighted by [6], employing a CNN to classify time-series data necessitates the direct input of the original time-series data into the network. This direct approach may overlook subtle yet critical information pertaining to various phases in a sequential activity, thereby not fully leveraging the rich spatiotemporal features that multiple images can provide. According to [7], initially, researchers primarily focused on single frames for HAR in simple and controlled settings. However, current research is centered on complex and realistic human activities captured in challenging environments, employing multiple frames of motion.

The integration of spatiotemporal features at the frame level in activity recognition offers a promising avenue for discerning patterns and temporal structures, applicable not only to activities exhibiting short-term repetitive patterns but also to those requiring recognition of long-term patterns. In realistic environments where actions display diverse characteristics and exhibit complex temporal structures, the effective utilization of temporal dynamics becomes a significant challenge [8]. This underscores the need for more sophisticated strategies to incorporate temporal information into *HAR* models, particularly when addressing the complexity of various human activities in real-world scenarios.

To date, few studies have explored the use of image sequences as inputs for networks in activity recognition comprehensively. This work aims to bridge this gap by developing a DL architecture for

HAR that represents inertial sensor data as images and accounts for a sequence of temporal motion features.

The methodology for generating images from inertial sensor data utilizes the recurrence plot method, which provides a visual representation of repetitive patterns in a time series. It maps the behavior of a system over time, emphasizing instances of similar behaviors and thus unveiling insights into its dynamics. Data clustering via recurrence plots, combined with DL, has found applications in sensitive biomedical fields [9–11] and in engineering disciplines characterized by high levels of unpredictability [12]. Moreover, the adoption of time-distributed layers, particularly *LSTM* (Long short-term memory) networks, offers substantial benefits for processing complex temporal sequences, as typically encountered in *HAR* tasks. These layers facilitate the handling of long-term temporal dependencies and the capture of nonlinear patterns, often resulting in enhanced model performance.

In this work, the experiments leveraged a public dataset, facilitating the comparison of four distinct architectures and achieving metrics exceeding 97% with our proposed solution. Beyond the high classification accuracy observed, the contributions of this study are delineated as follows:

- Development of a *HAR* solution employing the strategy of representing numerical data as images, utilizing the recurrence plot method to transform multi-sensor inertial data into images.
- Creation of a model that effectively incorporates motion transition variations, demonstrating high efficacy with performance metrics surpassing 97% accuracy.
- Achievement in designing a model for activity recognition that classifies multi-sensor inertial data through the analysis of sequential images over time.

2. Background

This section introduces the basic concepts of the main DL architectures employed in this work, as well as the time-distributed layer and the technique of recurrence plot.

2.1. DL bidimensional - 2D DL

2D DL applied to inertial sensor data has substantially benefited from methods that transform time series into two-dimensional visual representations, such as recurrence plots. These plots convert inertial signals, like accelerometer and gyroscope data, into images that capture dynamic and recurrent temporal patterns, enabling the visualization of temporal relationships in a graphical format. This transformation is crucial for highlighting repetitive events, trends, and irregularities in the signals, providing a rich basis for learning and analysis in CNNs.

In the context of *2D DL*, a central issue is how to transform the 1D inertial signal into a 2D image that preserves the relevant features of the data. A common approach is the use of raw plots, where sensor data is plotted over time and organized into an image used as input for CNNs. Alternatively, the *multichannel* technique treats the different sensor axes as overlapping color channels RGB (red, green, and blue), creating a robust visual representation from the normalization and scaling of the sensors' real values.

By applying CNNs to visual representations of inertial data, the networks can automatically extract complex spatiotemporal features without the need for manual feature extraction. This results in higher accuracy in classifying human activities, gestures, and other movement-based events, such as in *HAR*.

Thus, 2D DL techniques have proven highly effective in tasks such as health monitoring, gesture detection, and tracking movement patterns over time.

2.2. *CNNs*

These advanced neural networks are designed to process data structured in grids, such as images and videos. Unlike traditional neural networks, *CNNs* consider spatial relationships, using convolutions to apply filters over input data, which generates feature maps that identify patterns like edges and textures. These feature maps evolve through the network layers, capturing hierarchies from simple to complex features. *CNNs* excel in computer vision tasks, including object recognition and image segmentation. During training, loss functions like cross-entropy are combined with optimizers such as the *Adam* (adaptive moment estimation) or the *SGD* (stochastic gradient descent) to minimize errors and enhance model accuracy [13–15]. A key feature of *CNNs* is weight sharing, which reduces the number of trainable parameters and computational complexity, making them efficient for large datasets.

The architecture of a *CNN* is illustrated in Figure 1, highlighting the difference between convolutional and FC (fully connected) layers. The convolutional layer is not fully connected, which enhances convergence by reducing input parameters. *CNN* kernels act as receptors for various input features. Activation functions allow only signals above a threshold to pass to the next layer, enabling the *CNN* to filter relevant information and generate feature maps efficiently. In addition to convolutions, *CNNs* utilize pooling to reduce the dimensionality of feature maps, consolidating important data and enhancing robustness to minor variations like translations and rotations. Pooling methods include max-pooling, which captures the maximum value, and average-pooling, which computes the average.

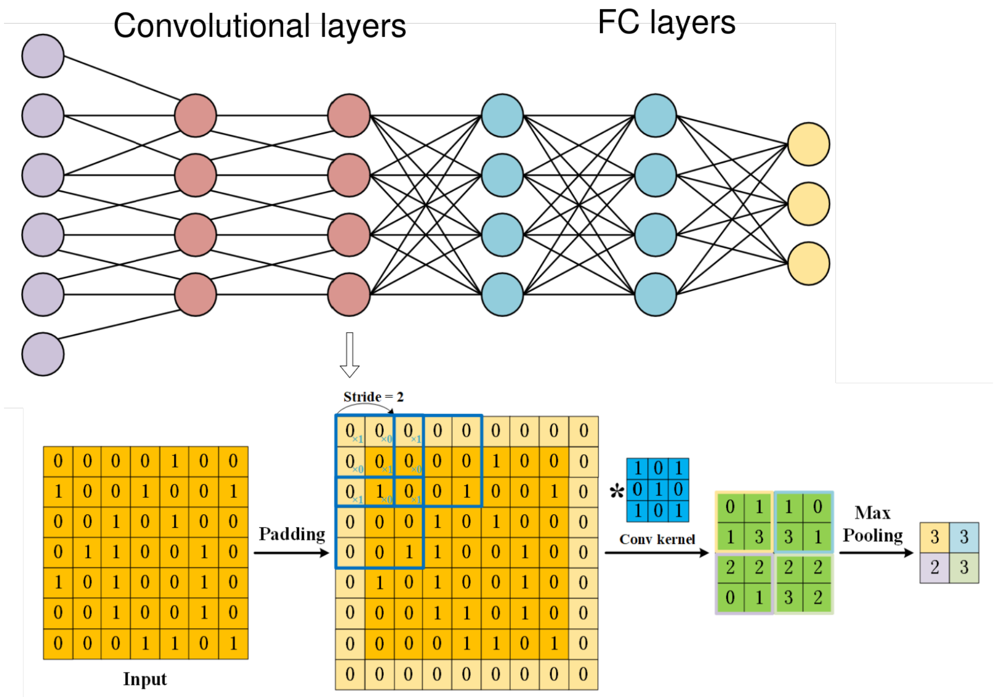


Figure 1. Diagram of a *CNN* architecture, showcasing a convolutional layer [17].

A typical *CNN* structure includes several convolutional layers followed by pooling layers, forming a deep network capable of learning increasingly sophisticated representations of the data. After the convolutional layers, the CNN usually includes one or more FC layers, which interpret the extracted features to perform the classification or regression task. In addition to traditional image applications, CNNs have been adapted for other types of data, such as spectrograms, which represent audio signals in a visual format, and even text processing, where they can be used to extract features from sequences of words. In videos, three-dimensional CNNs (3D-CNNs) are used to capture both spatial and temporal information, enabling the recognition of complex actions and events [16].

2.3. LSTM

LSTMs are an advanced type of RNN (recurrent neural network) designed to address the vanishing gradient problem, which hinders conventional RNN training on tasks involving long-term dependencies [18, 19]. This issue arises when gradient magnitudes during back propagation become too small, causing a loss of memory for past information.

As a variant of RNNs, LSTMs were introduced to capture global sequence dependencies from input data by identifying hidden patterns frame by frame [20]. The basic features of CNNs are passed to the LSTM layer to check temporal dependencies. LSTMs, by their architecture, are especially effective at modeling long-term temporal dependencies. As a variant of RNNs, they were introduced to capture global dependencies in temporal data sequences by identifying contours and hidden patterns across a sequence, frame by frame [20]. This makes them ideal for tasks such as speech recognition, machine translation, and time series modeling, where capturing and remembering important information at distant points in the sequence is crucial.

This architecture can also be combined with CNNs to enhance performance in tasks involving spatiotemporal data, such as video recognition. Features extracted by a CNN can be passed to an LSTM layer, which then verifies and accumulates the temporal dependencies of these features. This combination is particularly effective in learning long-range dependencies, allowing the model to capture not only spatial patterns in each frame but also how these patterns evolve over time [21]. The LSTM layer accumulates temporal dependencies from CNN outputs, highlighting its effectiveness in learning long-range dependencies.

The *LSTM* operates in three stages consisting of the following steps:

Input Gate: Decides which new information to add to the memory cell based on the previous cell state and the current sequence input. Output Gate: Determines the output from the memory cell, considering the current cell state and the current sequence input. Forget Gate: Evaluates which old information to forget, using the previous cell state and the current sequence input.

These gates perform tasks of updating, maintaining, and deleting information, allowing the representation of input sequences $x = (x_1, x_2, x_3, ..., x_n)$. The inputs control which information should be stored, forgotten, or used to generate the network output at each time step.

The update operations for the *LSTM* gates are mathematically defined below, as illustrated in (Equation 2.1):

$$\begin{cases} f_t & = \sigma\left(W_{x_f} x_t + W_{h_f} h_{t-1} + b_f\right), \\ i_t & = \sigma\left(W_{x_i} x_t + W_{h_i} h_{t-1} + b_i\right), \\ o_t & = \sigma\left(W_{x_o} x_t + W_{h_o} h_{t-1} + b_o\right), \\ c_t & = f_t \times C_{t-1} + i_t \times \tanh\left(W_{x_c} x_t + W_{h_c} h_{t-1} + b_c\right), \\ h_t & = o_t \times \tanh\left(c_t\right). \end{cases} \tag{2.1}$$

In these equations, σ is the sigmoid function, tanh denotes the hyperbolic tangent function, x_t represents the current input at time step t, h_{t-1} is the output from the previous time step, and C_{t-1} is the previous cell state. The W and b symbols denote the weights and biases associated with the *LSTM* gates, respectively. The operation $\cdot$ indicates element-wise multiplication, enabling the *LSTM* to selectively retain, discard, or update information across its memory cells, thereby addressing long-term dependencies in time-sequenced data effectively.

2.4. ConvLSTM

According to [22], although LSTM networks are highly effective at capturing temporal correlations in sequential data, they tend to overlook spatial information, which is crucial in many applications such as video recognition and weather forecasting. This occurs because traditional LSTM works with 1D data, treating each input sequence as a series of vectors without considering the spatial structure that may be present in the data.

To overcome this limitation, the ConvLSTM model, introduced by [23], was specifically designed to handle problems involving both spatial and temporal dimensions. ConvLSTM extends the traditional LSTM architecture by integrating convolutional operations into the state update process, which enables it to process data with two-dimensional or three-dimensional spatial structure, such as videos, heat maps, or any other form of spatiotemporal data.

State transitions in the ConvLSTM cell are performed through convolutional operations, which produce 3D data, in contrast to traditional LSTM, which operates with 1D data. This approach allows ConvLSTM to capture spatial features through convolutions applied to multidimensional data, enriching temporal modeling with spatial information. The first two dimensions represent the spatial dimension (e.g., height and width of an image or video frame) and the third dimension represents the temporal dimension (e.g., time). This structure allows ConvLSTM to capture complex spatial patterns that evolve over time. Unlike traditional LSTM, which uses matrix multiplication for state transitions, ConvLSTM relies on convolutions, providing more effective capture of spatial dependencies in the data [24], as illustrated in Figure 2.

The `ConvLSTM` model adheres to equations analogous to those of the traditional *LSTM* cell, with its mathematical representation detailed in (Equation 2.2):

$$\begin{cases} i_t & = \sigma\left(W_{x_f} * X_t + W_{h_f} * h_{t-1} + b_f\right), \\ f_t & = \sigma\left(W_{x_i} * X_t + W_{h_i} * h_{t-1} + b_i\right), \\ o_t & = \sigma\left(W_{x_o} * X_t + W_{h_o} * h_{t-1} + b_o\right), \\ C_t & = f_t \circ C_{t-1} + i_t \circ \tanh\left(W_{x_c} * X_t + W_{h_c} * h_{t-1} + b_c\right),\ h_t \\ C_t & = o_t \circ \tanh\left(C_t\right) \end{cases} \tag{2.2}$$

The basic processing architecture of a *ConvLSTM* is as follows:

1. The input to a ConvLSTM layer consists of sequences of two-dimensional or three-dimensional maps, such as a series of video frames treated as a three-dimensional tensor.
2. During each time step, the *ConvLSTM* performs convolution operations on the input data and previous states, capturing spatial structures and local features, unlike traditional *LSTMs* which rely on matrix multiplication.
3. The *ConvLSTM* maintains an internal memory of cell and hidden states, updating them using convolutions to preserve spatial information instead of traditional multiplication.
4. The *LSTM* operations in the *ConvLSTM* capture long-term temporal dependencies while integrating spatial features over time, which is essential for applications like object tracking in videos or weather prediction.
5. The output of the *ConvLSTM* can be either a sequence of feature maps or a dense layer, often producing a probability map in video segmentation tasks or a predicted sequence of frames in prediction tasks.

In this architecture, the symbol $*$ represents the convolution operation, while $\circ$ denotes the Hadamard product. The weight matrices W_{c_f}, W_{c_i}, W_{c_o} and the bias vectors are updated in each training process.

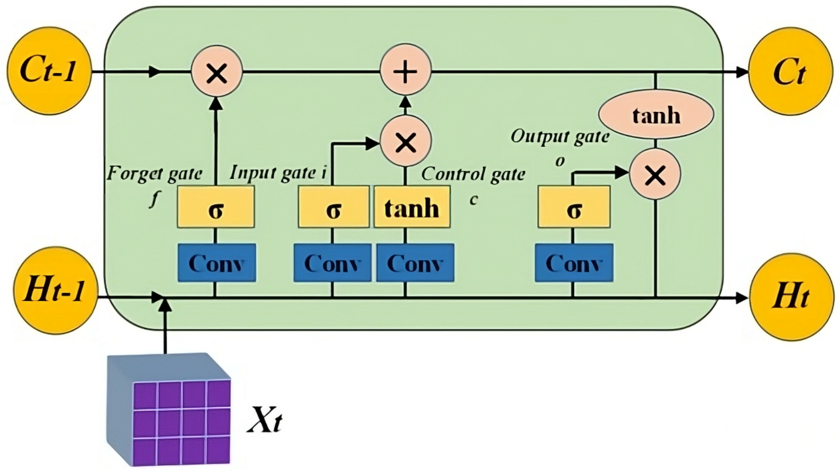


Figure 2. Architecture of a ConvLSTM network [22].

2.5. *Time-distributed layer*

In [35], local feature extractors distributed over time are simultaneously applied to each temporal subsequence, allowing for the extraction of local spatiotemporal features. This approach is fundamental in applications that require understanding dynamic patterns in temporal sequences, such as videos, time series, and other types of data where relationships over time are crucial.

A particularly important layer used in RNN, such as *LSTM* and *GRU (gated recurrent unit)*, is the “TimeDistributed” layer. As discussed in [25], this layer functions to apply other layers at each time step of a temporal sequence. In other words, instead of processing the sequence as a whole, the “TimeDistributed” layer allows each step of the sequence to be processed independently and in parallel, using the same operations.

In RNNs, the output at each time step is typically a sequence of activations that reflects the processing of the input over time. However, there are situations where it is necessary to apply additional processing layers, such as dense or convolutional layers, to each step of this sequence. The "TimeDistributed" layer facilitates this task by encapsulating these layers so that they are applied consistently at each time step. Thus, in an *LSTM* that processes a sequence of images, the "TimeDistributed" layer can be used to apply a convolutional or dense layer to each image in the sequence independently. This allows the same layer to be reused at each step of the temporal sequence, instead of duplicating the layer for each time step, which would be computationally inefficient.

2.6. Recurrence plot

The concept of recurrence is fundamental for the analysis of dynamical systems, especially in contexts where the behavior of time series is nonstationary or nonlinear. Recurrence, as described by [26], has become accessible thanks to advances in mathematical and computational techniques. Recurrence-based methods are particularly effective for analyzing time series that are short, nonstationary, and nonlinear, offering a powerful way to explore and understand the complex dynamics of these systems.

Nonlinear dynamical systems often exhibit complex and unpredictable behaviors, where state regression is a key feature. State regression can be visualized through recurrence plots (RP), a tool that allows for the detection and analysis of transitions in the dynamics of time series [27]. These plots help to identify underlying patterns and structures in the dynamics of time series, providing a more detailed view of the interactions and dynamic behavior.

An RP quantifies and visualizes the repetitions of a trajectory in phase space, representing these repetitions in a matrix. The central idea is that recurrence occurs when a trajectory revisits a neighborhood in phase space that has been previously visited. In practical terms, this means that if a system returns to a state or a condition similar to a previous state, this will be reflected as a recurrence structure in the plot. The formula for calculating the RP, as presented by [27], is expressed in (Equation 2.3):

$$R(i,j) = \begin{cases} 1 & \text{if } \left\| \vec{x}(i) - \vec{x}(j) \right\| \leq \varepsilon, \\ 0 & \text{otherwise.} \end{cases} \tag{2.3}$$

In this (Equation 2.3, $R(i, j)$) is a binary value. It is 1 if the Euclidean distance between vectors $\vec{x}(i)$ and $\vec{x}(j)$ is less than or equal to a specified threshold ε. This implies that points i and j are considered recurrent or close in the data space. If the distance exceeds ε, the recurrence is not met, and $R(i, j)$ is set to 0. Conversely, a matrix involving different trajectories is known as a cross-recurrence matrix. Analysis of RPs reveals essential properties of dynamic systems, thereby deepening our comprehension of complex behavioral patterns.

By using recurrence to extract information about the dynamic behavior of time series, it is possible to study and quantify behavioral interactions in the classification of HAR. Recurrence analysis can reveal patterns and regularities in time series, contributing to the identification of different regimes and behaviors in the analyzed system. The use of RPs enables a deeper understanding of the dynamics of complex systems, offering a valuable tool for the analysis and classification of temporal data. With the ability to reveal hidden patterns and dynamic transitions, recurrence analysis is essential for the interpretation and modeling of nonlinear and nonstationary time series.

3. Related work

This section reviews related works that have contributed to the development of *HAR* methodologies, ranging from converting 1D inertial signals to 2D images to integrate advanced neural network architectures for more effective feature extraction and activity classification.

In [28], a temporal series encoding approach using smartphone inertial sensors for data capture was employed. This technique converts 1D inertial signals into 2D images, integrating the computer vision domain for signal classification. Unlike most existing approaches, which focus on DL networks with 1D-DL (one-dimensional) data, this work proposes a 2D-DL approach, requiring an additional transformation of inertial signals to operate in the 2D space.

New structures for encoding temporal series using different types of images, such as *Gramian angular summation/difference fields* (GASF/GADF) and *Markov transition fields* (MTF), were proposed in [29]. This allows the application of computer vision techniques for the classification and imputation of temporal series. The experiment, which used *CNN* on 20 datasets, achieved highly competitive results compared to nine leading temporal series classification approaches.

Deep CNNs are recommended for automating feature extraction from raw inertial sensor data, presenting a generic structure for activity recognition based on *CNN* and LSTM. The *CNN+LSTM* structure proposed by [2] outperforms some comparative results by up to 9%, being applicable to homogeneous sensors and capable of fusing multimodal data to enhance performance. This study explores the complementarity between CNNs and LSTMs, where CNNs are used to extract spatial features from the data, and LSTMs to capture temporal dependencies. The proposed architecture replaces the conventional dense layers of the CNN with *LSTM* layers, following four *CNN-1D* layers, with the aim of leveraging the long-term capabilities of the *LSTM* to improve classification accuracy.

The neural network proposed by [30] combines *Conv1D* with *ConvLSTM2D*, allowing the automatic extraction of relevant features with few parameters. It introduces an innovative approach to *HAR* called batch normalization, which accelerates training convergence. Experimental results indicate an accuracy of 91.25% to 93.15% on the *UCI-HAR* dataset, highlighting the effectiveness of the approach in surveillance and monitoring scenarios.

In the study conducted in [31], an architecture with *LSTM* and *2D CNN* branches operating in parallel, receiving raw signals and their spectrograms, is proposed. The extracted features are concatenated for activity recognition, comparing their effectiveness with other common architectures. Hyperparameter tuning through Bayesian optimization and evaluation on public datasets considers not only classification performance but also network complexities, such as the number of parameters, size, time, and floating-point operations (FLOPs).

This study, compared to the mentioned works, differs in some aspects. Similar to [2], it employs deep neural networks for activity recognition but distinguishes itself by using the 2D-DL technique, converting inertial data into 2D images through RPs. In addition to architectures like *CNN 2D* and *2D CNN-LSTM*, it incorporates *EfficientNet-B0* and *InceptionResnet*, expanding the variety of evaluated networks. In line with [30], this work addresses *LSTM* challenges but adopts a time-distributed layer approach, evaluating the ability of *LSTMs* to predict future values in temporal sequences.

4. Materials and methods

This section outlines the data transformation steps within the *HAR* methodology, aiming to recognize human activities through visual representations created from mobile device inertial sensor data. This section treats the experimental operational steps, from database utilization to activity recognition, using the developed models.

4.1. Methodology summary

The comparative analysis aims to evaluate the effectiveness of the developed models using metrics such as accuracy, recall, precision, and F1-Score, along with the analysis of the confusion matrix for human activity recognition. The central goal is to achieve high percentages in these metrics, with a focus on detailed analysis of the confusion matrix to identify the precision in activity classification. The study is conducted from the researcher's perspective, considering the analysis of the metrics used, the replicability of the experiment, and the generalizability of the results. Figure 3 illustrates the operational steps of the experiment, from the use of the dataset to the recognition of activities by the developed models.

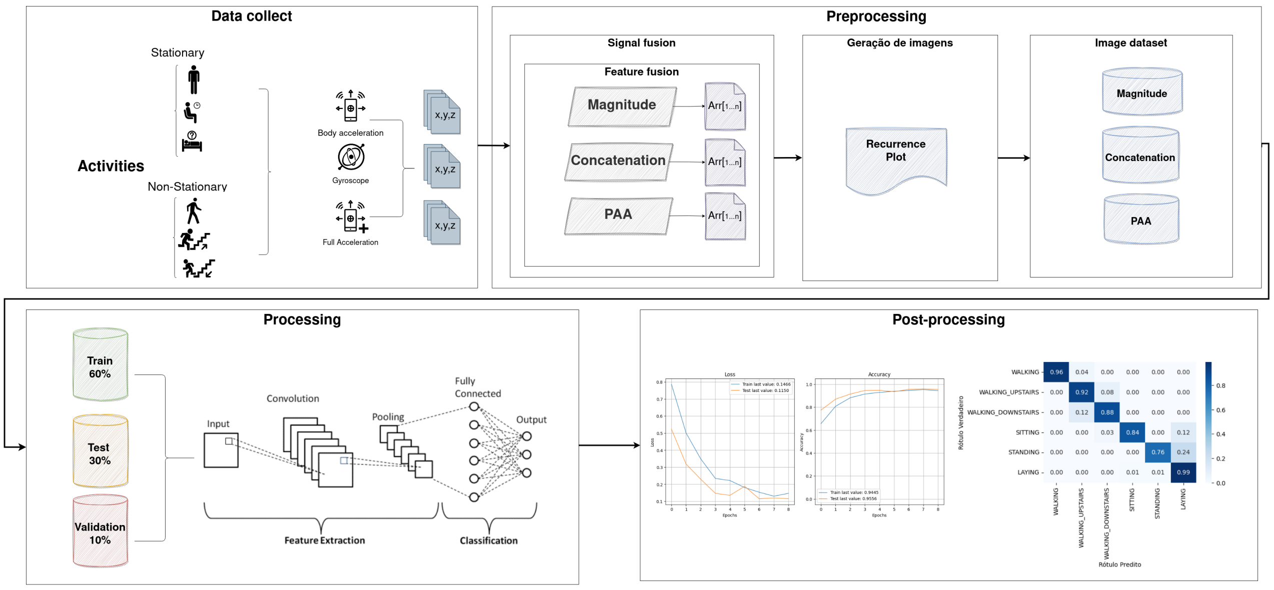


Figure 3. Operational steps of the experiment utilizing the *HAR* methodology.

Segmentation is the first step after data collection. Here, the data is organized according to the activity they represent. The collected data, typically in the form of time series, is stored in formats that facilitate analysis, such as tables or columns. The "windowing" process is used to divide the time series into smaller parts or windows. This is important because each windowing approach offers different advantages and disadvantages depending on the type of data and the objective of the analysis.

Preprocessing is a crucial phase where data is prepared to be used in machine learning models. It involves several steps:

- Removal of outliers: Eliminate data that falls outside the expected pattern. Removal of duplicate values: Remove repeated entries to avoid distortions.

- Normalization: Adjust the data values to a common range, usually between 0 and 1.
- Noise reduction: Apply filters, such as the low-pass filter, to smooth the data and eliminate unwanted noise.
- Data cleaning: Ensure that the data is free from inconsistencies.

Preprocessing is done both during the model training phase and in real-world application to ensure accuracy in activity classification.

4.2. Dataset

The *UCI-HAR* dataset served as the foundation for this study. Sourced from the University of California, Irvine's machine learning repository, the dataset comprises inertial sensor readings from an Android Galaxy SII smartphone. Thirty participants contributed to the dataset, performing six predefined movements: walking, going upstairs, going downstairs, sitting, standing, and lying down. The sensors recorded samples at a 50 Hz sampling rate, with each activity performed twice by participants. Figure 4 displays the sample sizes for each activity, highlighting the data volume used in model training.

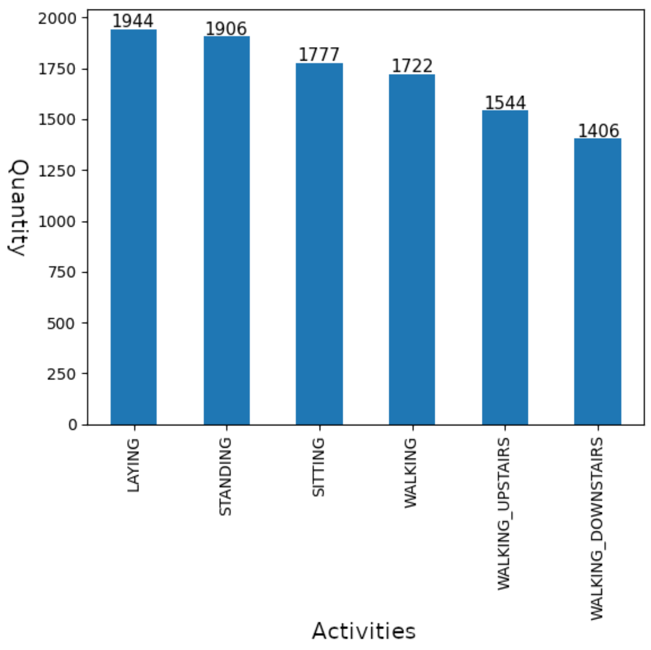


Figure 4. Quantity of data per activity class.

Data collection occurred in specific time windows, each lasting 2.56 seconds with a 50% overlap between adjacent windows, resulting in 128 readings per window. These windows include tri-directional *(x, y, z)* readings from both the accelerometer and gyroscope.

4.3. Preprocessing

Preprocessing prepares raw data for analysis or modeling, ensuring quality and consistency. It involves cleaning (removal of null or redundant values), transformation (normalization or standardization), feature extraction, and formatting to meet recognition requirements. This process reduces noise, enhances computational efficiency, and improves accuracy.

4.4. Methodology summary

The preprocessing stage is crucial for both preparing the data for training and for application in activity classification. In this stage, the raw signals from the *UCI-HAR* are used, as they have not undergone manual feature identification. The goal of preprocessing is to ensure that the data is clean, normalized, and ready to be transformed into suitable representations for the classification model.

Three methods were employed in the transformation of inertial raw data:

- **Concatenation of triaxial vectors:** This method concatenates the triaxial body acceleration ($acc_i = (x_i, y_i, z_i)$), gyroscope ($gyr_i = (x_i, y_i, z_i)$), and total acceleration ($accTotal_i = (x_i, y_i, z_i)$) bands, creating a more comprehensive representation of the forces acting in different directions. This combination is particularly useful for capturing the complex interaction between different axes of motion.
- **Magnitude of triaxial vectors:** Applies (Equation 4.1), calculating the magnitude of the sensor's triaxial signals, representing the combined total force of the movement, regardless of direction, which simplifies the analysis by reducing the three vector components to a single scalar value.

$$\text{magnitude} = \sqrt{(x)^2 + (y)^2 + (z)^2} \tag{4.1}$$

- **Piecewise Aggregate Approximation (PAA):** This is a dimensionality reduction technique for time series, dividing the series into segments and calculating the average of each one. PAA facilitates pattern detection in long series while preserving the main trends of the original series, allowing for more efficient analysis that is less susceptible to noise.

This method was inspired by the model proposed by [32], which uses the triaxial vector concatenation approach to encode time series data into *2D* images, aggregating this information into a single image for classification. The preprocessing steps for converting raw data into images proceed as outlined in Figure 5.

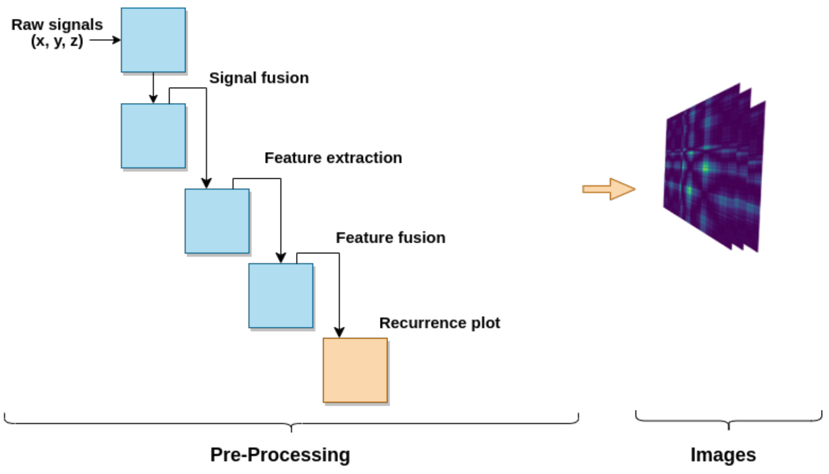


Figure 5. Steps performed in preprocessing to generate images for model training.

4.4.1. Data transformation into images

At this stage, the raw data from the *UCI-HAR* database are loaded into memory and organized into a *numpy* array, which contains both the labels and the images generated from these raw signals. The images are then resized using the *resize()* method from the Python Pillow library, with dimensions of *m x n* (32, 64, and 128).

The RP method converts the time series into images. This plot is generated based on the behavioral characteristics of the time series, such as periodicity, trends, and cyclicities. Each image, represented by an *m x n* matrix, is derived from the raw inertial data. Figure 6 illustrates the activities performed by a participant, highlighting the characteristic patterns in the images generated for each activity.

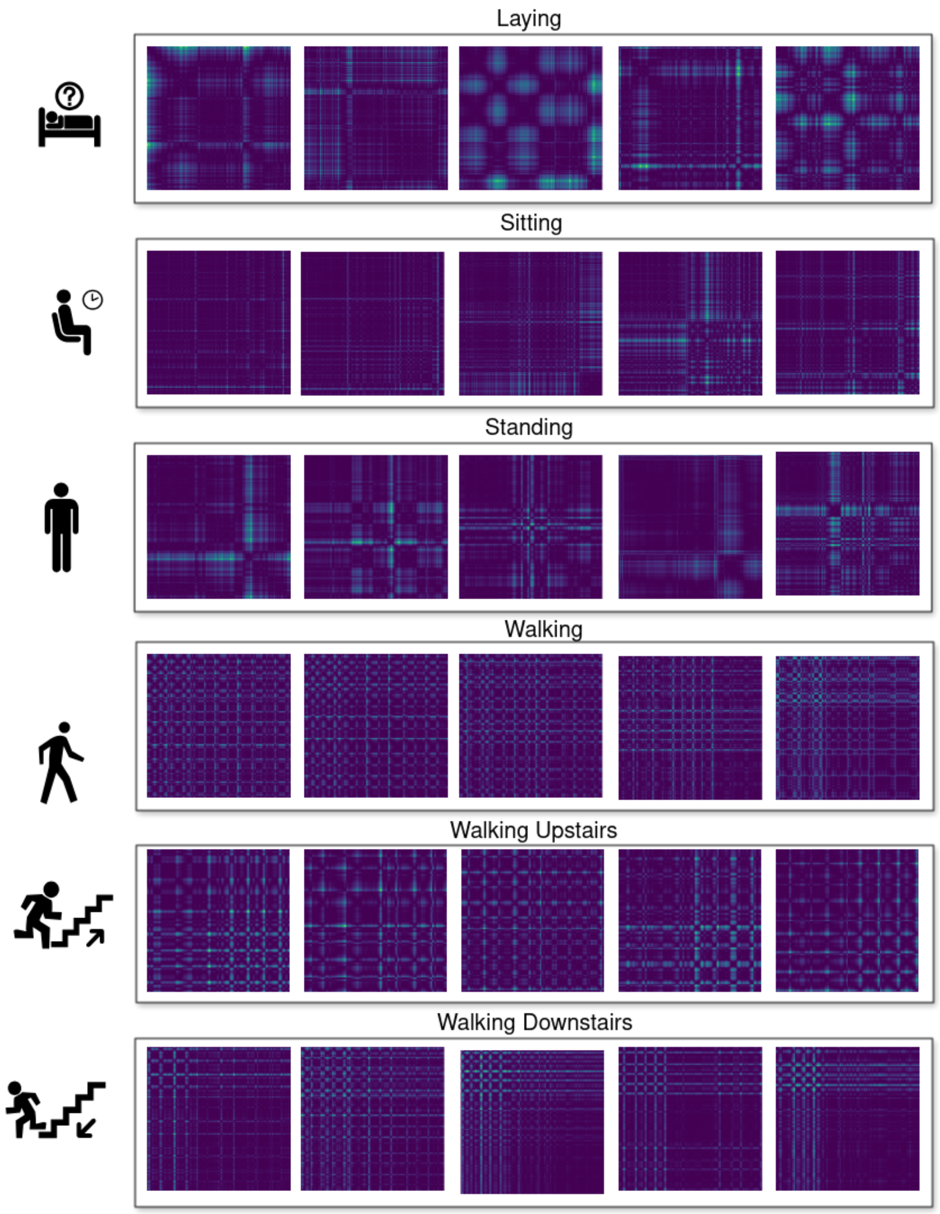


Figure 6. Transformation of inertial data into images depicting activities performed by a participant.

4.5. Processing

The classification in *HAR* utilizes supervised learning, relying on labeled data for prediction. Given a dataset $\{x^{(1)}, ..., x^{(m)}\}$ with corresponding labels $\{y^{(1)}, ..., y^{(m)}\}$, the objective is to predict $\hat{y}$ based on input x. This study explores DL approaches, including *CNNs* and RNNs, for their effectiveness in classification tasks. The main benefits of this approach can be defined as an improvement in data quality in the preprocessing stage by ensuring that raw signals are clean and normalized, making them suitable for modeling and reducing the impact of noise and outliers and the uses consist of sequences of two-dimensional maps, such as a series of video frames treated as a three-dimensional tensor. Using multiple images as input enhances the network's ability to capture temporal dynamics and spatial relationships, which is crucial for tasks in HAR, where movement patterns evolve over time.

This approach is innovative for HAR due to its ability to simultaneously process temporal and spatial data, enhancing the robustness and efficiency of activity recognition, and increasing accuracy in complex scenarios, capture of spatial and temporal patterns, multimodal data processing, and efficiency in complex tasks. Additionally, it provides a more comprehensive representation of data through the concatenation of triaxial vectors, offering a complete view of the forces acting in different directions. This capability captures the complexity of movements and enables more detailed analyses. Moreover, the technique simplifies analysis and reduces dimensionality by calculating the magnitude of triaxial vectors, condensing vector data into a single scalar value. This simplification retains essential information about the total force of movement. The PAA technique further facilitates pattern detection in time series, preserving key trends and enabling more efficient analysis that is less susceptible to noise.

4.6. Post-processing

This analysis aims to evaluate and compare the performance of different model training strategies, focusing on metrics such as accuracy, recall, precision, and F1-Score. It includes an examination of the confusion matrix to assess the precision of human activity recognition. The primary goal is to achieve high scores across these metrics and to scrutinize the confusion matrix for insights into the classification accuracy of human activities. This overview offers researchers a comprehensive understanding of the metrics used, the replicability of the study, and the generalization capability of the problem-solving approach. Figure 3 depicts the experimental operational steps, from database utilization to activity recognition with the developed models.

5. Experiments

Google Colab Pro was chosen for the experiments due to its robust online environment, offering support for the high-performance hardware needed for intensive tasks such as neural network training. *Python 3* was the central language, with specialized libraries for each task: Keras and TensorFlow for neural networks, *PIL* (Python image library) and *OpenCV* (Open source computer vision library) for image manipulation. Transforming raw data into images is a crucial step for recognizing temporal patterns, facilitating the analysis and training of machine learning models.

The models can be divided into several main sections:

1. Hyper-parameters:

- **Epoch**: represents a complete pass through the entire training set. A higher number of epochs allows the network to learn more from the data but can also increase the risk of overfitting.
- **Number of classes**: defines the number of output classes, which in this case is 6.
- **Batch size**: the number of samples processed before the model's weights are updated, set to 16.
- **Frames**: the number of frames per temporal sequence, set to 3, 4, and 5 frames per sequence.
- **Normalization**: the input image data was normalized by dividing the pixel values by 255.0, to scale the values between 0 and 1.

2. Optimizers:
   - The **SGD** optimizer was used and configured with parameters such as momentum, decay, learning rate, and Nesterov. Momentum helps avoid the training process getting stuck in local minima and accelerates convergence.
3. Main Layers:
   - **ConvLSTM2D**: combines convolutions and LSTMs, allowing the network to capture spatiotemporal dependencies in videos or temporal image sequences. Initialized with 128 filters, kernel size (3,3), strides (2,2), Padding 'same', and return sequences set to True.
     Dropout: helps prevent overfitting by forcing the network to learn more robust representations.
   - **TimeDistributed**: wraps other layers to apply the same operation to each frame of the temporal sequence.
   - **Flatten**: flattens the input to transform a matrix into a vector, preparing the data for subsequent dense layers.
   - **Dense**: Dense layers fully connect all neurons in the previous layer to all neurons in the next layer. The last layer uses 'softmax' to produce classification probabilities. The intermediate layers used ReLU (Rectified linear unit) activation, it introduces non-linearity to the network, allowing it to learn complex patterns and the model's final output layer employed softmax activation to output the activities mapped in the input.
4. Callbacks:
   - To prevent overfitting, the EarlyStopping callback was used, which stops training if the validation loss does not improve after a certain number of epochs. The argument patience=4 specifies that training will be halted if there is no improvement after 4 consecutive epochs.

The splitting strategy was defined to ensure the robustness of the models and to avoid overfitting; the data was divided into training, testing, and validation sets. The splitting proportion was such that a significant portion of the data was used for training, while a sufficient amount was reserved for testing and subsequently for validation. The typical proportion is 60% for training, 30% for testing, and 10% for validation. The following subsections detail the experiments conducted and the results achieved.

5.1. Experiment 1 - 2D CNN baseline

This experiment established a baseline *CNN* architecture, as depicted in Figure 7. The network begins with a `Conv2D` input layer featuring 128 filters, a `ReLU` activation function, a stride of (2, 2), and a dilation of (1, 1). A `MaxPooling2D` layer (2, 2) follows, along with a `Dropout` layer set at a rate of 0.2 to mitigate overfitting.

The architecture continues with two `Conv2D` layers, one with 128 filters and another with 64 filters, both employing `ReLU` activation. These are succeeded by `MaxPooling2D` and `Dropout` layers (rate of 0.2). A `Flatten` layer transforms the features into a one-dimensional vector, followed by two `Dense` layers: the first with 128 units and `ReLU` activation and a subsequent `Dropout` (rate of 0.5). The final `Dense` layer matches the number of classes and employs softmax activation for multi-class classification. We set the training for sixty epochs with early stopping to prevent *overfitting*, detailed in Table 1.

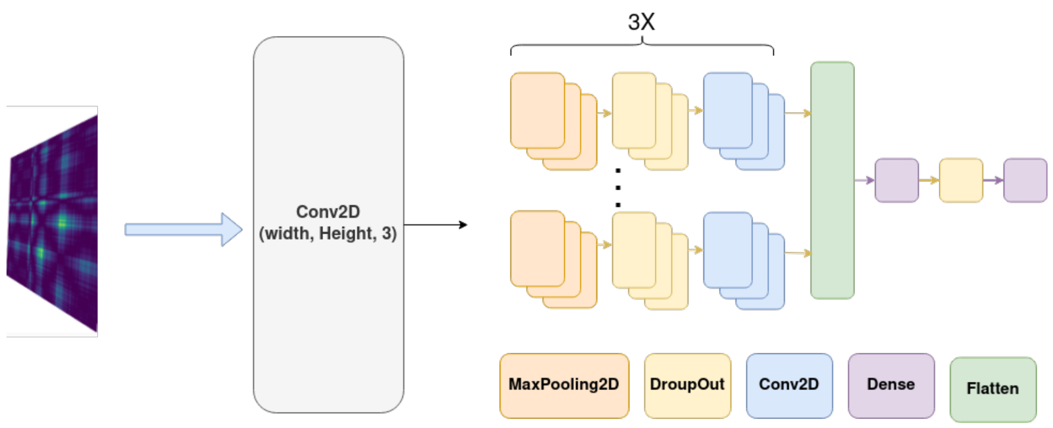


Figure 7. Experiment 1 - 2D CNN architecture.

Table 1. Early stop monitoring parameters.

	Monitoring			
Parameters	monitor	min delta	patience	Epochs
EarlyStopping	`val_loss`	1e-2	5	60

We compiled the model with a batch size of 10 using the `Adam` optimizer (learning rate of 0.001) and the categorical cross entropy loss function. Evaluated metrics included accuracy, F1-Score, precision, and recall.

5.2. *Experiment 2 - reuse of efficientnet architecture*

We adopted the EfficientNet model for this experiment, aiming for a balance between model size, computational efficiency, and accuracy in image classification [34]. We imported the `EfficientNetB0` architecture from the *tf.keras.applications* package, customizing the input layer and recompiling the final four layers with specific adjustments:

- Modified the input parameters to *input_shape=(128, 128, 3)*;
- Configured the output method as *classifier_activation="softmax"*;

- Specified the number of output classes as *classes=6*;
- Set the batch size to 10.

After incorporating the `Flatten` layer to transform features into a one-dimensional vector, we added a `Dense` layer with 128 units activated by the `ReLU` function. We then implemented a `Dropout` layer with a reduced rate of 0.25% based on prior results, particularly to improve the recognition of stationary activities in the confusion matrix. The final `Dense` layer, designed for the six classes, employs the `softmax` activation function to convert model outputs into probabilities, aligning with the activities identified. Adhering to the *CNN* Baseline Experiment 1's configuration, we conducted the training over 60 epochs with an `early stop` callback to curtail overfitting. The batch size remained at 10, and we compiled the model using the Adam optimizer. The categorical cross entropy function was chosen for the multi-class classification task.

This experiment's integration of the `EfficientNetB0` model, with tailored adjustments, aims not only for computational efficiency but also to enhance accuracy in the multi-class classification of human activities. Notably, we opted not to use `ImageNet` dataset weights for training, as preliminary findings suggested that doing so resulted in lower performance metrics than those achieved without them.

5.3. *Experiment 3 - reuse of inception-resnet architecture*

This experiment utilized the Inception ResNet-v2 model, implemented through the Keras library's `tf.keras.applications.InceptionResNetV2`.

We adjusted the input layer to `input_shape=(128, 128, 3)` and modified the last four layers in a manner similar to the EfficientNetB0 adaptation.

We introduced a `Flatten` layer to transform the features into a one-dimensional vector, followed by a `Dense` layer with 128 units using the `ReLU` activation function and a `Dropout` layer with a rate of 0.25. The concluding `Dense` layer, corresponding to the number of classes, utilizes the softmax activation function for multi-class classification.

5.4. *Experiment 4 - CNN-LSTM2D for sequential image recognition*

Aiming to analyze sequential image features for activity identification and prediction, this experiment leveraged a model that combines the strengths of *CNNs* and *LSTMs* networks in a layer `ConvLSTM2D`. *LSTMs* "memory" capabilities are crucial for recognizing patterns in time series data, allowing the network to predict future values based on observed sequences. Training with sequential images necessitated high-memory and high-processing capacity instances, such as T4 and V100 GPUs (Graphics processing unit). However, processing models with more than 5 images often exceeded GPU memory limits, causing processing interruptions.

To capture sequential data effectively, we adopted a time-distributed layer approach, applying consistent transformations to a list of input images. This method facilitated the exploration of features in sequential images for activity identification and prediction, particularly highlighting movement variability. Furthermore, *LSTMs* predictive capabilities in temporal sequences were evaluated for action detection, integrating *CNNs* for activities requiring multiple captures for accurate recognition. As noted by [35], multi-sensor time series methods represent a promising approach for monitoring, though feature extraction from such data remains a challenge for both traditional and DL models.

The input to the network consists of image sets with dimensions *(images, height, width, color channel)*, namely, *(X, 128, 128, 3)*. Each image, akin to a video frame within a sequence, facilitates visual activity recognition. Either multi-sensor or sequential single-sensor captures can enhance information extraction, leveraging additional feature extraction and shared weights between processed layers. Figure 8 displays a sequence of five movements within a 2.56-second overlap window.

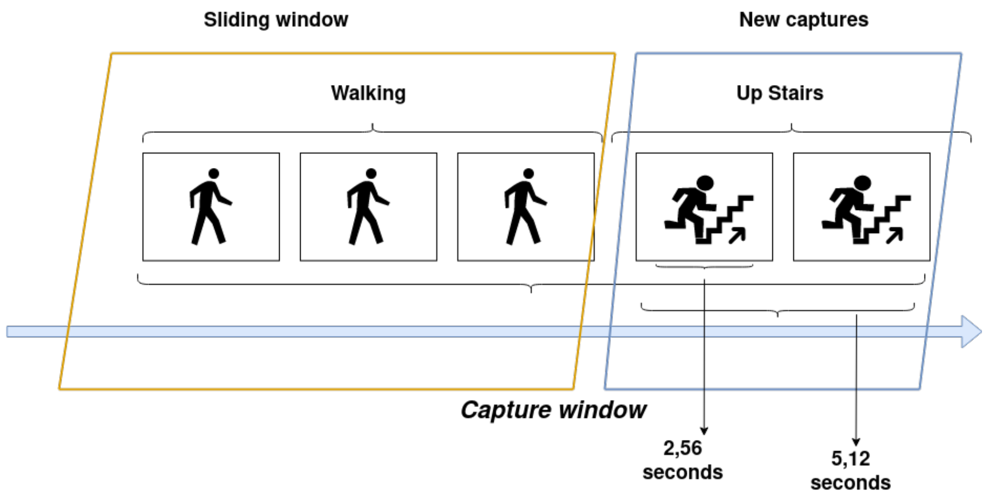


Figure 8. Sequential activity representation with a 2.56-second overlap window.

The architectural goal was to account for movement transition variations using a chronologically organized image set. The results were evaluated using metrics such as accuracy, recall, precision, F1-score, root mean square error (RMSE), and mean absolute error (MAE), as shown in Figure 9 for the architecture with a time-distributed layer.

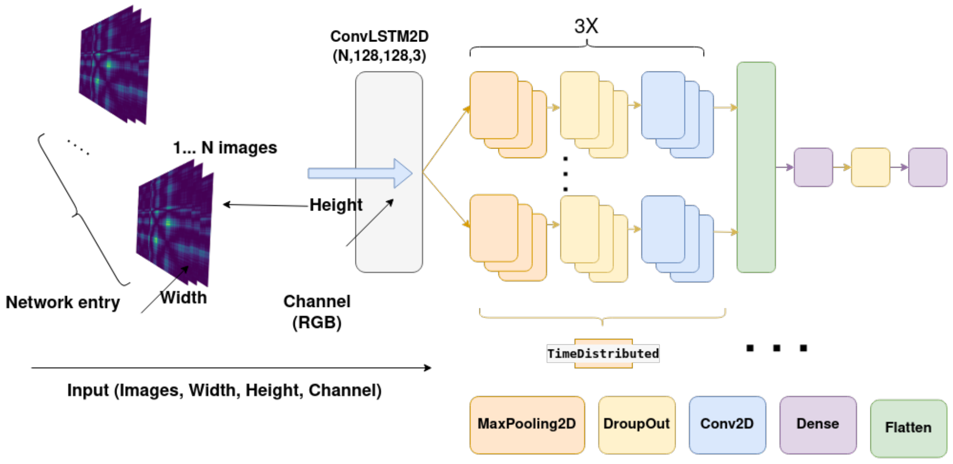


Figure 9. *CNN-LSTM2D* architecture with a time-distributed layer.

The `ConvLSTM2D` layer initiates the model by processing sequential images to capture dynamic features. This is followed by `TimeDistributed` layers that evenly distribute computational operations over time. After each `ConvLSTM2D` layer, the network implements a sequence of layers including `MaxPooling2D` and `Dropout`, reducing feature dimensions and mitigating overfitting, respectively. Subsequent `Conv2D` layers with `ReLU` activation further process the features, each followed by additional `MaxPooling2D` and `Dropout` layers to refine and regularize the model's outputs.

This structured approach ensures that the network efficiently handles sequential image data, making it adept at identifying nuanced activity patterns. The `Flatten` layer transitions the network from two-dimensional feature maps to a one-dimensional feature vector, preparing the data for dense network processing. Following this, the model employs a `Dense` layer with `ReLU` activation for high-level reasoning, and a subsequent `Dropout` layer reduces the risk of overfitting by randomly omitting a portion of the features. The final `Dense` layer, using `softmax` activation, classifies the activities into one of the predefined categories based on the learned features.

By integrating `ConvLSTM2D` with `TimeDistributed` layers, this architecture effectively captures and analyzes the temporal and spatial dimensions of the sequential image data. This methodical layering, as detailed in Table 2, underscores the model's capability to discern complex patterns in movement sequences, thereby enhancing the accuracy of activity classification.

Table 2. `ConvLSTM2D` architecture parameters with 5 input images.

	Layer parameters								
Layers	Input shape	Strides	Padding	Dilation rate	Return sequences	Kernel size	Activation	Pool size	Rate
ConvLSTM2D	(5, 128, 128, 3)	(2, 2)	same	(1, 1)	True	(3, 3)	relu	-	-
MaxPooling	-	-	-	-	-	-	-	(2, 2)	-
Dropout	-	-	-	-	-	-	-	-	0.2
Conv2D	-	-	-	-	-	(3,3)	relu	-	-
MaxPooling	-	-	-	-	-	-	-	(2, 2)	-
Dropout	-	-	-	-	-	-	-	-	0.2
Conv2D	-	-	-	-	-	(3,3)	relu	-	0.2
MaxPooling	-	-	-	-	-	-	-	(2, 2)	-
Dropout	-	-	-	-	-	-	-	-	0.2
Flatten	-	-	-	-	-	-	-	-	-
Dense	-	-	-	-	-	-	relu	-	-
Dropout	-	-	-	-	-	-	-	-	0.5
Dense	-	-	-	-	-	-	softmax	-	-

6. Results and discussion

This section presents and discusses the results of the experiments, highlighting performance metrics and providing an analysis of the observations made from these results. The choice of the UCI-HAR dataset is due to its widespread use in studies of human activity recognition, offering robust and diverse data captured from a smartphone. This dataset allows for detailed analysis and the training of machine learning models to classify different activities based on inertial signals. The UCI-HAR includes not only the raw data captured by the sensors but also nine triaxial signal tracks with corresponding labels and without any prior feature extraction. The use of this data in the experiments enabled a comparative evaluation of the methods employed with other studies that use similar approaches to transform inertial data into images.

A critical point to highlight is the imbalance in the distribution of activities within the UCI-HAR dataset. This imbalance can lead to bias in the trained models, making them less effective in classifying underrepresented activities, such as stationary activities (SITTING, STANDING, LAYING). The variability in the execution of the same activities by different individuals adds a level of difficulty since the model needs to capture consistent patterns. To mitigate these challenges, it was necessary to employ statistical methods that enhanced relevant features and isolated noise between similar and stationary activities.

Another significant challenge was dealing with the temporal component of some activities. This required models capable of effectively capturing and processing the sequence of events. Simple models failed to capture the temporal dynamics necessary to differentiate between activities that share similar movement patterns. When processing the model with a large number of sequential images (more than five images per input), the GPU memory consumption exceeded the available limits, resulting in interruptions in processing. This limitation imposed challenges in optimizing the model and efficiently managing computational resources.

Thus, the main challenges encountered during the experiments involved processing large volumes of data when transforming temporal data into images and training a model with high accuracy in sequential human activity recognition. The use of an architecture capable of handling temporal data and transforming it into visual representations was essential but also required significant computational resources. To train complex DL models based on images, it was necessary to use an environment with high processing capacity and memory, with a need for a GPU. The GPUs provided by Google Colab Pro, such as the *T4* and *V100*, were crucial to handling the intensive computational load and processing the models efficiently.

6.1. Experiment 1 - 2D CNN baseline

The experiment entailed developing deep neural network architectures. In the preprocessing phase, we applied signal fusion methods, feature extraction such as magnitude, raw signal concatenation, aggregated approximation, feature fusion, and data transformation into images via the RP, as detailed in Subsection 4.2. Exploring the impact of network input size, we experimented with three different image dimensions within the proposed architecture in the Experiment 1 - *2D CNN* Baseline section. We evaluated the model's loss and performance across these inputs, assessing performance metrics and interpreting the results through confusion matrices.

The Table 3 outlines the training and validation data, including the method, input dimensions, loss, accuracy, F1-Score, precision, recall, epochs, and runtime. We tested image sizes of *32* × *32*, *64* × *64*, and *128* × *128* pixels, finding that *128* × *128* pixels images yielded optimal performance. This efficiency is attributed to the *UCI-HAR* model's data collection methodology, which utilizes a window of 128 positions, thereby preserving information that smaller image sizes might lose.

The Figure 10 and Figure 11 show the metrics over 21 training epochs, indicating that using the *PAA* method and *128* × *128* pixels input images resulted in superior training outcomes for Experiment 1. Notably, activities such as "Walking" achieved a 100% classification rate, and "Laying" reached 99% accuracy in the training data. "Walking Upstairs" and "Walking Downstairs" recorded accuracy rates of 94% and 88%, respectively, while "Sitting" and "Standing" activities were classified with 93% and 88% accuracy, as depicted in the confusion matrix on test data.

The results indicate that, among the three feature fusion methods, the PAA method with a *128* ×

128 pixel input achieved the best performance, showing superior results in loss function, accuracy, F1-Score, recall, and precision. Overall, the baseline of the proposed model exhibits strong performance in classification tasks. Therefore, using a *128* × *128* pixel input, with an image constructed through the RP in the *Conv2D* input layer, represents the most suitable approach for feeding the data input in the proposed architecture concerning the *UCI-HAR* dataset.

The best results were obtained with the PAA approach, possibly due to its ability to capture the main trends of the time series more efficiently, reducing the impact of minor variations and noise that could confuse other methods.

Table 3. Training results of *Experimento-1*.

Method	Experimento-1							
	Input	Loss	Accuracy	F1-Score	Recall	Precision	Épocas	Time (min)
Magnitude	32 × 32	59.59	72.90	70.30	78.09	64.62	13	5.53
Magnitude	64 × 64	58.78	73.22	70.62	79.05	64.53	10	6.88
Magnitude	128 × 128	54.33	75.08	72.97	82.43	66.24	11	15.71
Concatenation	32 × 32	34.75	86.17	86.12	84.83	87.60	8	6.88
Concatenation	64 × 64	26.93	89.66	89.50	88.60	90.52	9	8.95
Concatenation	128 × 128	20.77	91.56	91.58	91.21	92.00	8	17.36
PAA	32 × 32	24.74	90.11	90.10	90.89	90.89	16	7.76
PAA	64 × 64	22.64	91.51	91.43	92.11	90.82	13	8.63
PAA	128 × 128	13.30	94.81	94.84	94.97	94.72	21	18.02

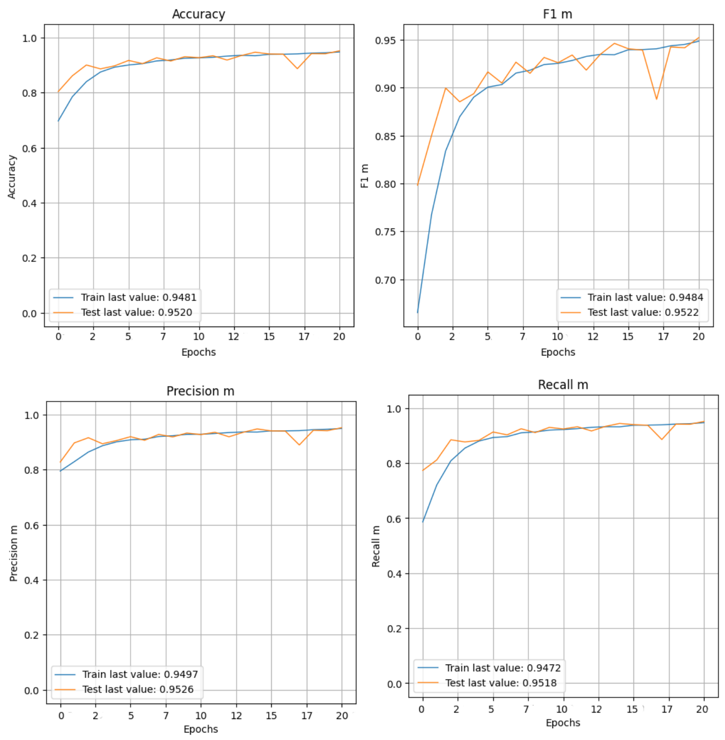


Figure 10. Performance of train/test metrics of accuracy, F1-score, recall, and precision for the PAA method with a 128 × 128 input from Experiment 1.

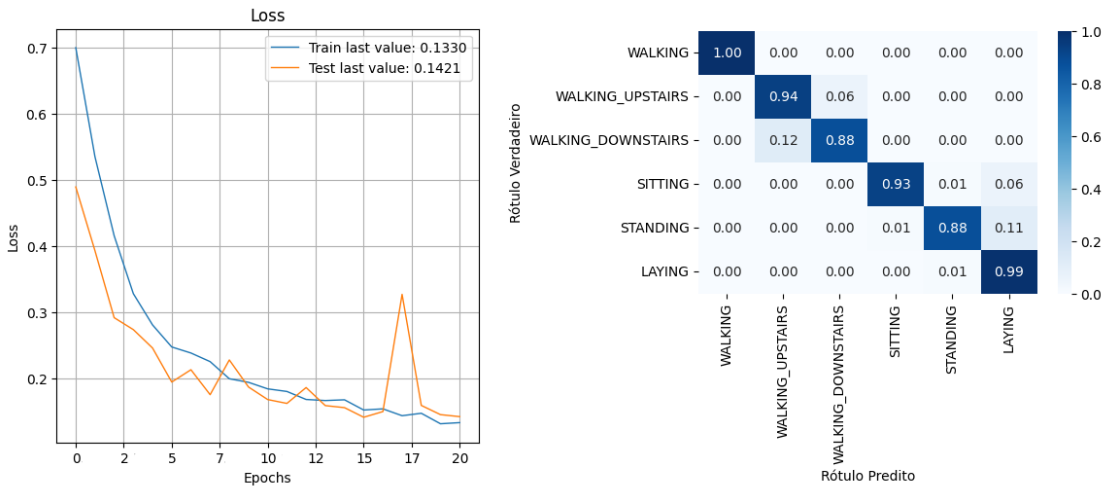


Figure 11. Loss and confusion matrix with test data for the PAA method of Experiment 1.

6.2. *Experiment 2 - efficientnet reuse*

This architecture aims to balance model size, computational efficiency, and accuracy in image classification tasks. We detail the results in Table 4 for a performance comparison with the baseline model from Experiment 1. Employing the `EfficientNet-B0` model with the PAA method over 12 epochs reached an accuracy of 89.0%. The training outcomes, depicted in Figure 12, include loss and evaluation metrics.

Table 4. Training results with EfficientNet-B0 and 128 × 128 image resolution.

	Experiment 2				
Method	Loss	Accuracy	F1-Score	Recall	Precision
Magnitude	86.96	59.44	52.45	52.45	43.86
Concatenation	41.77	83.03	83.10	85.45	81.11
PAA	27.84	89.04	89.06	89.90	88.31

However, Figure 13 shows evidence of overfitting, noted by the precision discrepancy between training and testing sets. These findings suggest the model's difficulty in robust feature extraction, impacting its generalization capability for unseen activities. Addressing overfitting through data augmentation, regularization techniques, or architectural adjustments could enhance model generalization. The potential of the `EfficientNet-B0` model potential for human activity image recognition necessitates further optimization tailored to this specific application.

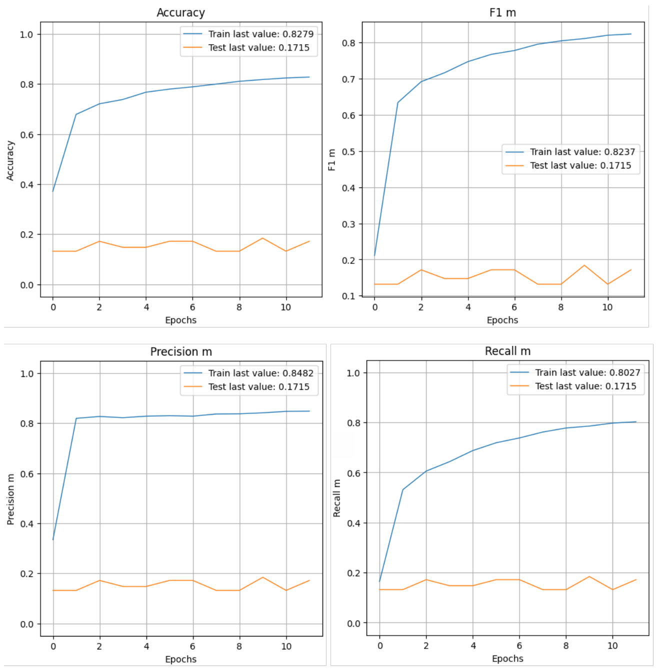


Figure 12. PAA method results from Experiment 2 with 128 × 128 input: Accuracy, F1-score, Recall, and Precision metrics.

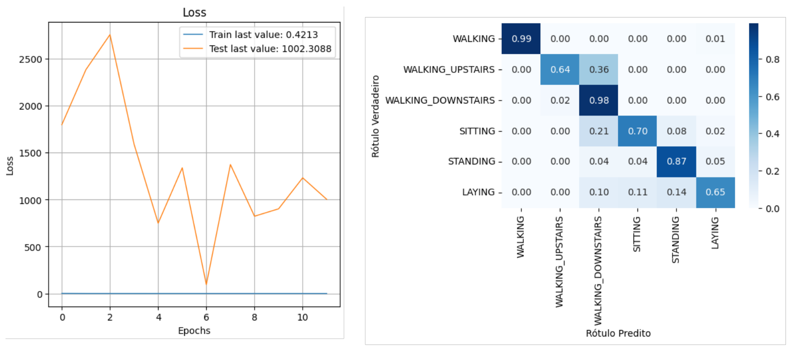


Figure 13. Loss function and confusion matrix of PAA method test data from Experiment 2 with 128 × 128 input.

6.3. *Experiment 3 - Reuse of InceptionResNetV2*

Utilizing the `InceptionResNetV2` architecture, we adjusted the input layer to *(128, 128, 3)* and reconfigured the final four layers for activity output mapping. The model demonstrated promising classification accuracy, reaching 92.75% as shown in Figure 14. The confusion matrix analysis, presented in Figure 15, further supports these findings. Dynamic activities like walking and using stairs were accurately recognized, alongside stationary activities such as sitting and laying, underscoring the model's capability in activity differentiation.

However, differentiating standing from laying proved challenging, reflecting in lower accuracy for standing and high false positives for laying. This indicates a potential area for model improvement in distinguishing low-movement activities. The results can be verified in Table 5 to illustrate the results achieved.

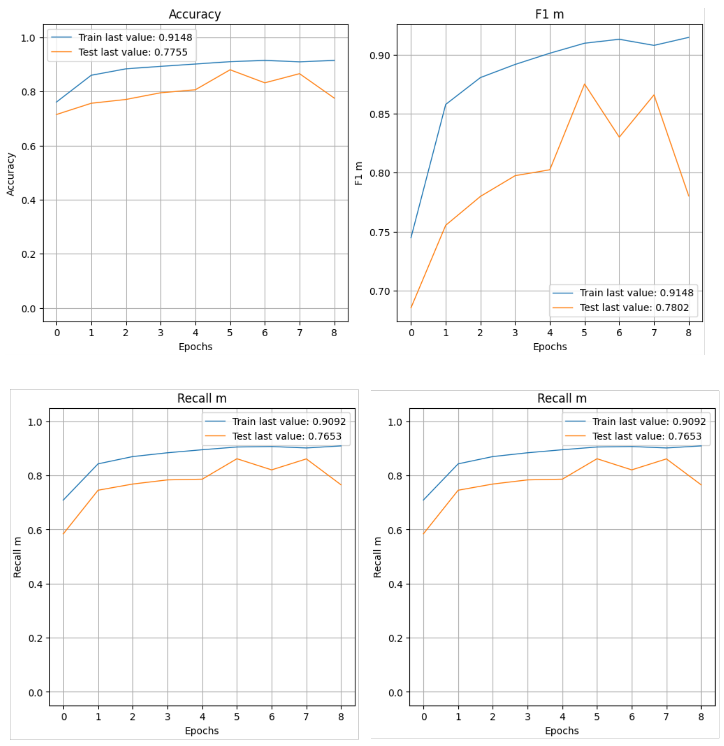


Figure 14. Performance metrics from Experiment 3 with PAA method and 128 × 128 input: accuracy, F1-score, recall, and precision.

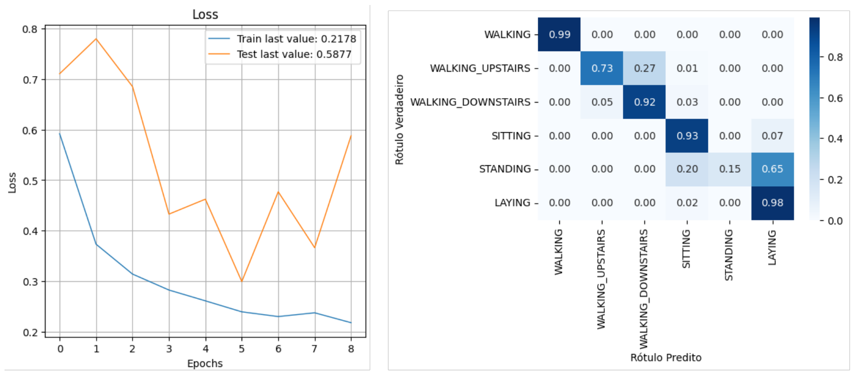


Figure 15. Loss function and confusion matrix from Experiment 3 with PAA method and 128 × 128 input.

Table 5. Results of Experiment 3 with PAA method and 128 × 128 image using InceptionResNetV2.

	Experiment 3				
Method	Loss	Accuracy	F1-Score	Recall	Precision
Magnitude	70.24	67.62	64.29	74.61	57.40
Concatenation	24.39	90.48	90.53	91.41	89.75
PAA	19.81	92.75	92.75	92.64	93.12

Experiment 3 demonstrates that `InceptionResNetV2`, combined with PAA and *128* × *128* pixels images, offers a viable approach for activity classification. Yet, to fully exploit its capabilities, further refinements are necessary, especially in distinguishing between static activities.

6.4. Experiment 4 - 2D CNN-LSTM

The Table 6 illustrates the performance of various methods across different image sequences. Notably, as the sequence length increases, the model's accuracy improves, indicating the effectiveness of *LSTM* layers in capturing temporal dynamics.

Table 6. Results of Experiment 4 training with 128 × 128 images.

	Experiment 4									
Method	Images	Loss	Accuracy	F1-Score	Recall	Precision	RMSE	MAE	Epochs	Time(min)
Magnitude	3	36.17	82.55	82.06	85.22	79.33	19.31	7.43	28	19.47
Magnitude	4	32.62	84.11	84.23	86.35	82.36	18.36	6.70	27	21.19
Magnitude	5	34.64	83.38	83.22	85.54	81.18	18.94	7.09	20	16.21
Concatenation	3	12.59	95.07	95.03	95.25	94.82	11.02	2.43	14	15.34
Concatenation	4	8.64	97.03	96.96	96.88	94.82	8.94	1.59	16	18.59
Concatenation	5	7.93	97.54	97.52	97.59	97.46	8.30	1.39	14	17.30
PAA	3	8.94	96.65	96.64	96.74	96.55	9.20	1.63	23	17.33
PAA	4	5.69	97.78	97.80	97.86	97.75	7.35	1.05	14	18.57
PAA	5	5.34	98.16	98.14	98.19	98.10	6.92	0.94	20	30.20

This experiment aimed to identify human actions or activities within a sequence of movements by considering their sequential order and transitions. We employed an architecture that merges *CNNs* with *LSTM* networks. *LSTMs* are crucial for analyzing features in sequential images, understanding patterns in time series blocks, and predicting future values based on observed sequences. This capability is vital for recognizing actions, taking into account not just the current activity but the sequence of preceding activities.

A time-distributed layer was introduced to address the sequential data capture challenge. This layer applies consistent transformations across a sequence of input images, including convolution, pooling, and dense layers, enabling efficient processing of image sequences. The best results were achieved with the PAA method applied to sequences of 5 images, achieving an accuracy of 98.16%, showcasing the potential of combining *CNN* and *LSTM* architectures for sequential image analysis. However, this enhanced performance comes at the cost of increased computational resources and processing time, particularly for longer image sequences.

The accuracy obtained in Experiment 4 using the PAA method with *128* × *128* pixels across sequences of 3, 4, and 5 images is illustrated in Figure 16. This comparison highlights the model's performance enhancement with an increasing number of images in the sequence, albeit at the cost of increased processing time and resource utilization.

The results demonstrate a direct correlation between the sequence length and the model's accuracy, with the highest accuracy of 98.16% achieved using 5 images in the input sequence. However, sequences with 4 images yielded the most significant and consistent outcomes for dynamic activities in Figure 17 and Figure 18.

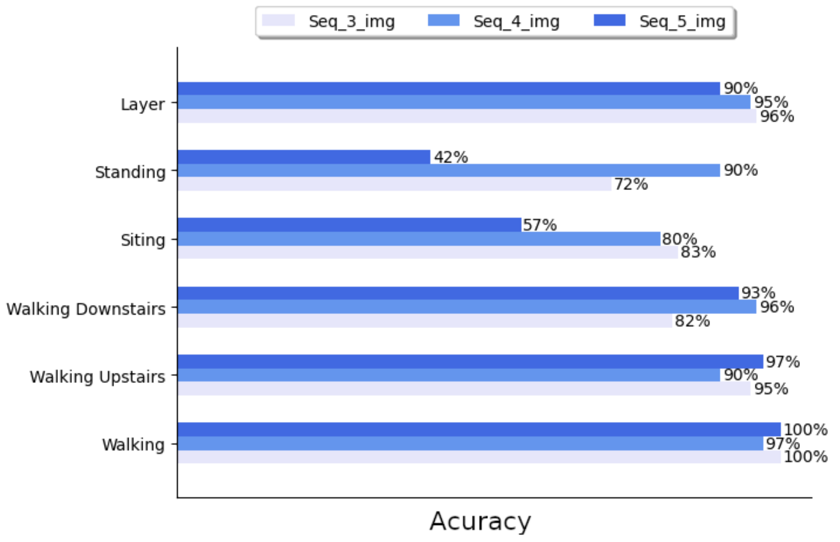


Figure 16. Accuracy percentages from the confusion matrix using the PAA method in Experiment 4 for sequences of 3, 4, and 5 images.

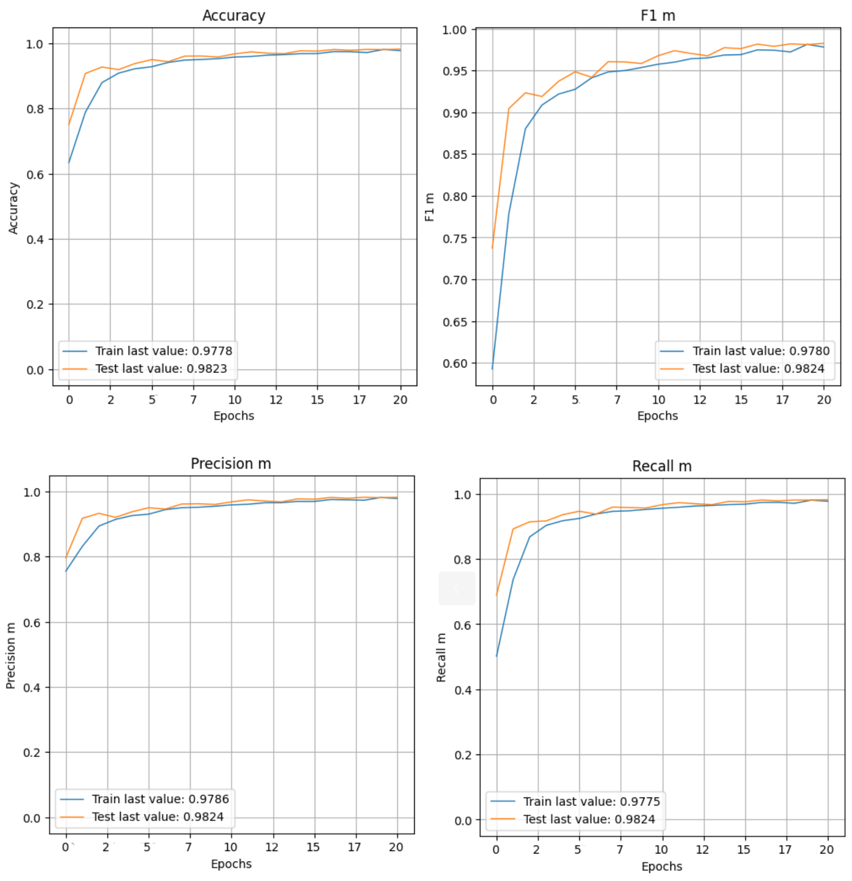


Figure 17. Train/Test accuracy, F1-Score, precision, and recall metrics for the PAA method with 4 input images from Experiment 4.

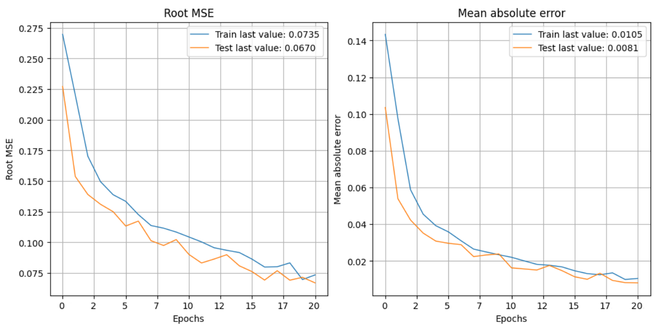
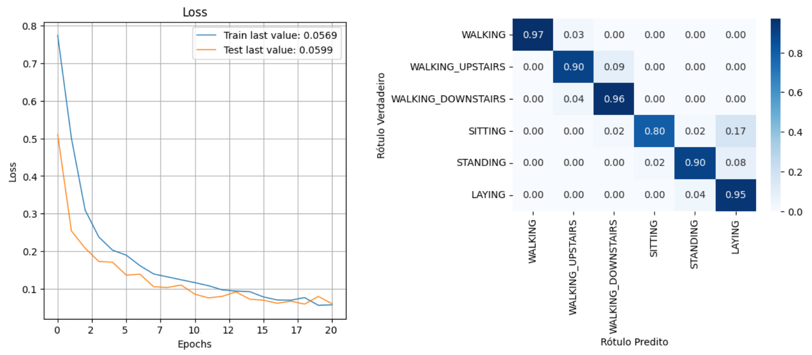


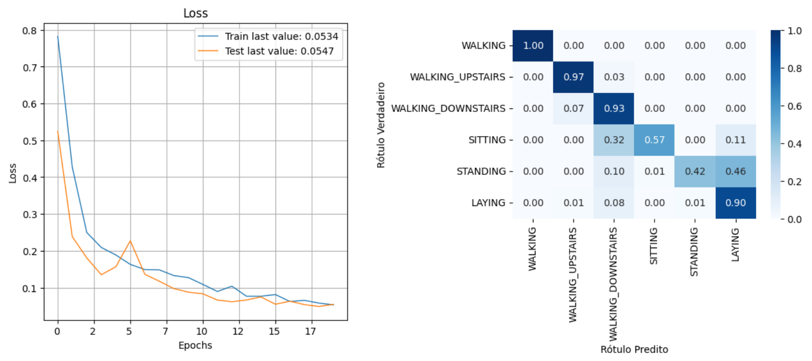
Figure 18. RMSE and MAE metrics for the PAA method with 4 input images from Experiment 4.

The "early stopping" criterion, which halts training after a specified number of epochs without improvement in loss, further underscores the model's efficiency and prevents overfitting, as evidenced by the minimal discrepancy between training and testing accuracies. The confusion matrix Figure 19 for sequences of 4 and 5 images Figure 20 indicates that the model is highly effective across all six activities. This effectiveness is particularly notable in dynamic activities such as "Walking", "Walking Upstairs", and "Walking Downstairs", as well as stationary activities like "Sitting", "Standing", and "Laying", despite the dataset's imbalance.



Rótulo Verdadeiro	WALKING	WALKING_UPSTAIRS	WALKING_DOWNSTAIRS	SITTING	STANDING	LAYING
WALKING	0.97	0.03	0.00	0.00	0.00	0.00
WALKING_UPSTAIRS	0.00	0.90	0.09	0.00	0.00	0.00
WALKING_DOWNSTAIRS	0.00	0.04	0.96	0.00	0.00	0.00
SITTING	0.00	0.00	0.02	0.80	0.02	0.17
STANDING	0.00	0.00	0.00	0.02	0.90	0.08
LAYING	0.00	0.00	0.00	0.00	0.04	0.95

Figure 19. Loss and confusion matrix for the PAA method with 4 input images from Experiment 4.



Rótulo Verdadeiro	WALKING	WALKING_UPSTAIRS	WALKING_DOWNSTAIRS	SITTING	STANDING	LAYING
WALKING	1.00	0.00	0.00	0.00	0.00	0.00
WALKING_UPSTAIRS	0.00	0.97	0.03	0.00	0.00	0.00
WALKING_DOWNSTAIRS	0.00	0.07	0.93	0.00	0.00	0.00
SITTING	0.00	0.00	0.32	0.57	0.00	0.11
STANDING	0.00	0.00	0.10	0.01	0.42	0.46
LAYING	0.00	0.01	0.08	0.00	0.01	0.90

Figure 20. Loss and confusion matrix for the PAA method with 5 input images from Experiment 4.

The data division followed a logic of maximizing the network's ability to generalize to new data, preventing overfitting. Similar to this study, [30, 31] also used the *UCI-HAR* dataset, which is widely known and used for HAR model validation, allowing for a direct comparison with the results obtained. In [30], a deep neural network is proposed that combines *Conv1D* with *ConvLSTM2D*. Additionally, the inclusion of a batch normalization layer is used to accelerate training. Experimental results demonstrated a high accuracy of 91.25% and 93.15% on the dataset, outperforming similar methods. This highlights the effectiveness of this approach in human activity surveillance and monitoring scenarios. The architecture proposes combining a one-dimensional CNN (Conv1D) and a two-dimensional CNN with LSTM (ConvLSTM2D) for human activity recognition. The model also incorporates techniques such as pooling and batch normalization to enhance convergence and performance.

7. Comparative results

The proposed models, particularly the 2D CNN-LSTM configuration with 4 images, achieved remarkable accuracy, F1-Score, recall, and precision metrics, indicating a highly effective approach for activity recognition. Comparing the proposed models with existing studies, as detailed in Table 7, showcases the proposed models' superior performance in recognizing human activities using the *UCI-HAR* dataset.

Table 7. Comparison of results from studies using the *UCI-HAR* dataset.

	Comparison					
Classifier	Loss	Accuracy	F1-Score	Recall	Precision	Study
DeepConvLSTM	–	-	89.50%	-	-	[2]
Conv-1D	1.025%	93.15%	93.00%	93.0%	93.0%	[30]
2D CNN-LSTM	1.182%	91.25%	91.25%	91.0%	92.0%	[30]
2D CNN-LSTM	–	93.21%	93.18%	93.15%	93.67%	[31]
2D CNN	1.330%	94.81%	94.84%	94.97%	94.72%	Proposed
2D CNN-LSTM (4 images)	0.569%	97.78%	97.80%	97.86%	97.75%	Proposed

On the other hand, [31] presents an innovative architecture that combines *LSTM* and *2D-CNN* branches in parallel. These branches process raw signals and spectrograms for activity recognition. The approach is compared to other common architectures, including CNNs and LSTMs, with hyperparameter tuning through Bayesian optimization and evaluation on two public datasets: *UCI-HAR* [36] and the *DSA dataset* [37]. This *2D CNN-LSTM* model achieves an average accuracy of 93.67% on both datasets. It features a hybrid architecture that combines a 2D CNN and an LSTM. The proposed network processes raw signals and their spectrograms in parallel, combining the extracted features for activity recognition. The architecture achieved average accuracies of 95.66% and 92.95% on the two datasets, outperforming other hybrid architectures and individual networks.

This study focuses on employing a CNN combined with an LSTM for HAR using inertial sensors. A hybrid CNN-LSTM architecture was utilized, leveraging the spatial features extracted by the CNN and the temporal dependencies captured by the LSTM. While the experimental results for both Conv 2D and CNN-LSTM 2D demonstrated notable improvements, the training and testing times using the *PAA* method (with inputs of 4 and 5 images, resulting in durations of 18.57 and 30.20 minutes, respectively), as illustrated in Table 7, indicate that more complex architectures, though more accurate, may require significantly more computational time. This trade-off between accuracy and computational cost should be carefully considered depending on the specific use case.

The study [31] compares with [2], also mentioned in the related studies (Related work). Both reinforce the importance of hybrid *CNN-LSTM* architectures for human activity recognition, demonstrating that the combination of spatial and temporal features leads to better performance. However, the study [2] focuses on replacing dense layers with *LSTM* to capture temporal dependencies, while the study [31] delves deeper into the impact of removing different components (CNN or LSTM), providing additional insights into the relative contribution of each to the final model performance. Although the study [31] compares with [2], it indicates an improvement of 0.14–0.28% in the performance metrics for the UCI HAR dataset, and its proposed hybrid 2D CNN-LSTM architecture allows extracting a broader range of representative features, resulting in higher activity recognition accuracy. We used spectrograms as input for the CNN-2D branch compared to the performance of existing DL models: *1D CNN*, *2D CNN*, *LSTM*, Standard *1D CNN-LSTM* model, *1D CNN-LSTM* proposed in [2].

The proposed and developed models in this study outperformed the mentioned models in terms of accuracy, F1-Score, recall, and precision. This indicates that the proposed models have superior performance in human activity recognition tasks based on the UCI-HAR dataset. Additionally, the proposed models demonstrated greater efficiency in terms of loss, indicating more effective optimization during training. Therefore, the results suggest that the proposed approach is effective and promising for this scenario.

It has been demonstrated that this study advances the use of hybrid CNN-LSTM architectures, with differences in implementations and approaches resulting in varying levels of accuracy and efficiency. While the study by [2] was pioneering in combining CNNs and LSTMs for HAR, more recent studies, such as those by [31] and [30], further explore these combinations, utilizing techniques such as batch normalization, spectrograms, and parallel approaches to enhance performance. The result is a clear progression in the effectiveness of these networks for HAR, with more recent architectures consistently surpassing earlier ones in key metrics.

8. Conclusion

The proposed architecture, which combines *CNN*, *LSTM*, and temporal distributed layers, excels in the effective identification and prediction of monitored activities, handling nuances in movement sequences well. The fusion of signals and features provides an integrated approach, enhancing performance by transforming inertial data into images and generating richer representations of activities. In summary, this combination offers an effective solution for dealing with complex data, bringing notable benefits in terms of performance and understanding. The dataset used, such as *UCI-HAR*, is specific to certain types of activities and environments. This dependency on specific data may limit the model's ability to generalize to other contexts or recognize activities not represented in the training data. The model may struggle with less frequent activities that present subtle differences or involve more complex interactions, potentially resulting in biases toward more common or simpler activities.

The RP is a powerful tool for transforming temporal data into visual representations, facilitating the identification of recurring patterns in monitored activities. By feeding images generated from the RP into neural networks, these visual representations can be processed in convolutional architectures, capturing complex relationships in situations where linear methods are inadequate.

The architectural approach that incorporates sequential data shows high effectiveness, resulting in improvements in performance metrics. The ability to analyze features in sequential images and consider variability in movements allows the model to successfully identify and predict monitored actions, overcoming the limitations of previous models that neglected the chronological order of movements, resulting in more accurate predictions. The application of *LSTMs* plays a crucial role in predicting future values in temporal sequences, contributing to a more effective sequential analysis of activities. Thus, collecting image sequences is essential, especially for activities involving varied movements, in tasks that require consideration of long-term dependencies. *LSTMs* have the ability to generalize well from training data, making them effective in predicting activities in new sequential data for tasks involving memorization and retention of patterns over time.

The architecture, combining *CNNs*, *LSTMs*, and other layers, increases the risk of overfitting, particularly with limited or unbalanced data. Although techniques such as batch normalization and pooling help mitigate this risk, the model may still exhibit reduced performance when exposed to new, unseen data, especially if the training data do not adequately represent the full range of possible activities.

The use of *CNNs* with multiple captures and operations improves recognition, while the application of the temporal distributed layer overcomes challenges in sequential data capture. Configuring the network input with a set of images provides more comprehensive and accurate analyses, resulting in enhanced performance. However, it is crucial to mention the limitation of the experiment, highlighting the need for a development environment with GPUs to process deep learning architectures with image-based data. The model architecture, while powerful, requires significant computational resources for both training and inference. This includes the need for GPUs to handle large-scale data processing, especially when dealing with image-based inputs derived from RPs. This limitation makes the model less accessible for deployment in environments with limited computational capabilities, such as mobile devices or real-time applications.

The model's ability to operate in real-time faces limitations due to the time required to process and classify the data. *LSTMs* have the capability to predict future values. However, the model may inherit biases from the training data, particularly if the dataset is not sufficiently diverse in terms of participants, activities, or environmental conditions. These biases may result in skewed predictions, favoring certain demographic groups, activity types, or contexts, which could compromise the accuracy and fairness of results in practical applications. To mitigate these issues, a potential solution is the use of transfer learning to perform customized training for each individual, tailoring the model to the specific characteristics of each user. In real-world scenarios, applying this approach can be combined with sensor monitoring and signal fusion in multi-sensor environments. By reducing the amount of collected data and using image recognition to generate sequential measurements and more efficiently determine the state of a monitored environment.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

The authors thank FAPERJ for partially funding this work.

Conflict of interest

The authors declare no conflict of interest, ensuring the objectivity and transparency of this research.

References

1. Ferrari A, Micucci D, Mobilio M, et al. (2021) Trends in human activity recognition using smartphones. *J Reliable Intell Environ* 7: 189–213. https://doi.org/10.1007/s40860-021-00147-0
2. Ordóñez FJ, Roggen D (2016) Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors* 16: 115. https://doi.org/10.3390/s16010115
3. Archana R, Jeevaraj PSE (2024) Deep learning models for digital image processing: a review. *Artif Intell Rev* 57: 11. https://doi.org/10.1007/s10462-023-10631-z
4. Wang CY, Bochkovskiy A, Liao HYM (2023) YOLOv7: trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, IEEE Computer Society, Los Alamitos, USA, 2023: 7464–7475. https://doi.org/10.1109/CVPR52729.2023.00721
5. Vaswani A, Shazeer N, Parmar N, et al. (2017) Attention is all you need, *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, Curran Associates Inc., Red Hook, NY, USA, 2017: 6000–6010. https://doi.org/10.5555/3295222.3295349
6. Li T, Zhang Y, Wang T (2021) SRPM–CNN: a combined model based on slide relative position matrix and CNN for time series classification. *Complex Intell Syst* 7: 1619–1631. https://doi.org/10.1007/s40747-021-00296-y
7. Hussain A, Khan SU, Khan N, et al. (2024) AI-driven behavior biometrics framework for robust human activity recognition in surveillance systems. *Eng Appl Artif Intell* 127: 107218. https://doi.org/10.1016/j.engappai.2023.107218
8. An G, Zheng Z, Wu D, et al. (2019) Deep spectral feature pyramid in the frequency domain for long-term action recognition. *J Vis Commun Image R* 64: 102650. https://doi.org/10.1016/j.jvcir.2019.102650
9. Torse DA, Khanai R, Desai VV (2019) Classification of epileptic seizures using recurrence plots and machine learning techniques, *2019 International Conference on Communication and Signal Processing (ICCSP)*, IEEE, 2019: 0611–0615. Available from: https://ieeexplore.ieee.org/document/869798.

10. Hurezeanu B, Ungureanu GM, Digulescu A, et al. (2013) Fetal heart rate variability study with recurrence plot analysis, *2013 E-Health and Bioengineering Conference (EHB)*, IEEE, 2013: 1–4. Available from: https://ieeexplore.ieee.org/document/6707310.

11. San-Um W, Potiwanna C, Jakborvornphan S (2018) Characterizations of critical heart disease in ECG signal features through recurrence plots as for medical imaging diagnostics, *2018 5th International Conference on Business and Industrial Research (ICBIR)*, Thailand, 2018: 183–188. Available from: https://ieeexplore.ieee.org/document/8391189.

12. Tian Y, Huang J, Sun Y (2023) Fault diagnosis for rolling bearings based on recurrence plot and convolutional neural wetwork, *2023 2nd International Conference on Big Data, Information and Computer Network (BDICN)*, China, 2023: 335–340. Available from: https://ieeexplore.ieee.org/document/10109955.

13. LeCun Y, Boser B, Denker JS, et al. (1989) Backpropagation applied to handwritten zip code recognition. *Neural comput* 1: 541–551. https://doi.org/10.1162/neco.1989.1.4.541

14. Krohn J, Beyleveld G, Bassens A (2019) *Deep Learning Illustrated: A Visual, Interactive Guide to Artificial Intelligence*, USA: Addison-Wesley Professional. Available from: https://www.deeplearningillustrated.com.

15. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. *Nature* 521: 436–444. https://doi.org/10.1038/nature14539

16. Kang X, Song B, Sun F (2019) A deep similarity metric method based on incomplete data for traffic anomaly detection in IoT. *Appl Sci* 9: 135. https://doi.org/10.3390/app9010135

17. Li Z, Liu F, Yang W, et al. (2021) A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE T Neur Net Lear* 33: 6999–7019. https://doi.org/10.1109/TNNLS.2021.3084827

18. Hochreiter S, Schmidhuber J (1997) Long short-term memory *Neur Comput* 9: 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

19. Gers FA, Schmidhuber J, Cummins F (2000) Learning to forget: continual prediction with LSTM. *Neur Comput* 12: 2451–2471. https://doi.org/10.1162/089976600300015015

20. Arif S, Wang J, Ul Hassan T, et al. (2019) 3D-CNN-based fused feature maps with LSTM applied to action recognition. *Future Int* 11: 42. https://doi.org/10.3390/fi11020042

21. Ercolano G, Rossi S (2021) Combining CNN and LSTM for activity of daily living recognition with a 3D matrix skeleton representation. *Intel Serv Robot* 14: 175–185. https://doi.org/10.1007/s11370-021-00358-7

22. Deng F, Chen Z, Liu Y, et al. (2022) A novel combination neural network based on convlstm-transformer for bearing remaining useful life prediction. *Machines* 10: 1226. https://doi.org/10.3390/machines10121226

23. Shi X, Chen Z, Wang H, et al. (2015) Convolutional LSTM network: A machine learning approach for precipitation nowcasting, In: Cortes C, Lawrence N, Lee D, et al., *Advances in Neural Information Processing Systems*, NY: Curran Associates, 802–810. Available from: https://proceedings.neurips.cc/paper/2015/hash/07563a3fe3bbe7e3ba84431ad9d055af-Abstract.html.

24. Yu C, Ma X, Ren J, et al. (2020) Spatio-Temporal Graph Transformer Networks for Pedestrian Trajectory Prediction. *Computer Vision – ECCV 2020: 16th European Conference*, Proceedings, Part XII. Springer-Verlag, United Kingdom, Springer International Publishing, 507–523. https://doi.org/10.1007/978-3-030-58610-2_30

25. Gupta S (2021) Deep learning based human activity recognition (HAR) using wearable sensor data. *Int J Inf Manage Data Insights* 1: 100046. https://doi.org/10.1016/j.jjimei.2021.100046

26. Marwan N (2008) A historical review of recurrence plots. *Eur Phys J Spec Top* 164: 3–12. https://doi.org/10.1140/epjst/e2008-00829-1

27. Marwan N, Carmen Romano M, Thiel M, et al. (2007) Recurrence plots for the analysis of complex systems. *Phys Rep* 438: 237–329. https://doi.org/10.1016/j.physrep.2006.11.001

28. Daniel N, Klein I (2021) INIM: inertial images construction with applications to activity recognition. *Sensors* 21: 4787. https://doi.org/10.3390/s21144787

29. Wang Z, Oates T (2015) Imaging time-series to improve classification and imputation, *Proceedings of the 24th International Conference on Artificial Intelligence*, Buenos Aires, Argentina, AAAI Press, 2015: 3939–3945. https://doi.org/10.48550/arXiv.1506.00327

30. Pandey A, Kumar P, Prasad S (2023) 2d convolutional lstm-based approach for human action recognition on various sensor data, *International Conference on Frontiers of Intelligent Computing: Theory and Applications*, Singapore, Springer Nature Singapore, 327: 405–417. https://doi.org/10.1007/978-981-19-7524-0_36

31. Koşar E, Barshan B (2023) A new CNN-LSTM architecture for activity recognition employing wearable motion sensor data: enabling diverse feature extraction. *Eng Appl Artif Intel* 124: 106529. https://doi.org/10.1016/j.engappai.2023.106529

32. Yang C-L, Yang C-Y, Chen Z-X, et al. (2019) Multivariate time series data transformation for convolutional neural network, *2019 IEEE/SICE International Symposium on System Integration (SII)*, IEEE, 2019: 188–192. https://doi.org/10.1109/SII.2019.8700425

33. Bisong E (2019) Google Colaboratory, In: Bisong, E., *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, Berkeley: Apress, 59–64. https://doi.org/10.1007/978-1-4842-4470-8_7

34. Tan M, Le Q (2019) Efficientnet: Rethinking model scaling for convolutional neural networks, *Proceedings of the 36th International Conference on Machine Learning*, PMLR, 97: 6105–6114. Available from: https://proceedings.mlr.press/v97/tan19a.html.

35. Qiao H, Wang T, Wang P, et al. (2018) A time-distributed spatiotemporal feature learning method for machine health monitoring with multi-sensor time series. *Sensors* 18: 2932. https://doi.org/10.3390/s18092932

36. Blunck H, Bhattacharya S, Prentow T, et al. (2015) Heterogeneity activity recognition. *UCI Mach Learn Repos* 10: C5689X. https://doi.org/10.24432/C5689X

37. Barshan B, Altun K (2013) Daily and sports activities. *UCI Mach Learn Repos* 10: C5C59F. https://doi.org/10.24432/C5C59F