Learning and Nonlinear Models - Journal of the Brazilian Society on Artificial Intelligence (SBIA), Vol. 24, Iss. 1, pp. 4–17, 2026
https://doi.org/10.21528/lnlm-vol24-no1-art1

ENERGY EFFICIENCY IN HUMAN ACTIVITY RECOGNITION ON SMARTPHONES WITH ADAPTIVE SAMPLING AND DEEP LEARNING MODELS WITH EARLY EXITS

Marcel Franco de Oliveira
Programa de Pós-graduação em Engenharia de Produção e Sistemas Computacionais
Universidade Federal Fluminense - UFF, Campus Rio das Ostras - RJ
marcelfranco@id.uff.br

Alessandro Copetti
Programa de Pós-graduação em Engenharia de Produção e Sistemas Computacionais
Universidade Federal Fluminense - UFF, Campus Rio das Ostras - RJ
alessandro_copetti@id.uff.br

Roberto Pacheco
Programa de Pós-graduação em Engenharia de Produção e Sistemas Computacionais
Universidade Federal Fluminense - UFF, Campus Rio das Ostras - RJ
robertopacheco@id.uff.br

Lauro A. G. Sampaio
Programa de Pós-graduação em Engenharia de Produção e Sistemas Computacionais
Universidade Federal Fluminense - UFF, Campus Rio das Ostras - RJ
laurosampaio@id.uff.br

Luciano Bertini
Universidade Federal de Itajubá - UNIFEI - MG
lbertini@unifei.edu.br

Abstract – Human Activity Recognition (HAR) plays a crucial role in applications such as health monitoring, physical training, and therapeutic support. Smartphones equipped with embedded sensors provide a convenient platform for these applications. However, running deep learning models continuously can drain device energy autonomy. Despite recent advances, few studies have explored how executing deep learning-based HAR models across different computational environments affects energy consumption, especially when combined with strategies that leverage application-level contextual information. This work develops and evaluates optimized deep learning models, including variants with early exits, in both local and remote execution scenarios. It also incorporates an adaptive sampling strategy that dynamically adjusts sensor reading frequency according to movement intensity (contextual information). Results show that the proposed approach reduces energy consumption and inference time while preserving model accuracy.

Keywords – Deep learning, early exit, human activity recognition, adaptive sampling, energy efficiency, mobile devices.

1. Introduction

Human Activity Recognition (HAR) has been extensively studied over the past decades [1, 2]. HAR integrates pervasive sensing technologies that continuously and ubiquitously collect sensor data from various environments to identify human activities. Wearable devices, widely adopted and equipped with inertial sensors (such as accelerometers, gyroscopes, and magnetometers), capture inertial motion data [3]. The collected data usually undergoes several stages, including signal preprocessing and classification with machine learning methods, to recognize activities. Recent surveys have highlighted the rapid evolution of HAR driven by advances in machine learning and deep learning techniques, particularly in mobile and wearable environments [4]. Deep learning has significantly advanced HAR, with recent studies focusing on improving classification techniques to enhance both accuracy and efficiency. For instance, [5] achieved 99% accuracy with a sequential model that combines local pattern extraction and temporal memory (CNN-LSTM), while [6] reported 96% accuracy using a Multi-Layer Perceptron (MLP) neural network. These results highlight the growing emphasis on high-performance classification models in HAR research.

https://doi.org/10.21528/lnlm-vol24-no1-art1

A key challenge in HAR on mobile devices is achieving energy efficiency, since real-time systems follow specific power consumption patterns and must maintain enough autonomy to operate without interruption. For mobile-based HAR, this often translates into frequent battery recharges, which degrades user experience [2]. Executing deep learning models directly on mobile devices leads to high energy consumption due to intensive resource demands. These demands include constant use of processing units (CPU, GPU, or NPU), memory, and embedded sensors, which considerably reduce battery life. A common alternative is to offload processing to the cloud, which alleviates the device's computational burden. However, this strategy requires higher data transmission, increasing communication-related energy consumption and introducing latency—undesirable in applications that demand real-time responses.

Edge computing approaches, such as those discussed in [7], seek to balance efficiency and performance by enabling local processing at the edge. Furthermore, many HAR applications fail to explore application-layer strategies to optimize energy consumption, such as adapting operational modes according to motion intensity, which could enhance both efficiency and model performance. The Early Exit (EE) technique has drawn attention as a way to improve the efficiency of deep learning models, particularly in resource-constrained environments. In HAR, EE introduces multiple exit points within the network, enabling intermediate classifications. Pacheco and Couto [8] explore this strategy by applying early exits to reduce inference costs on edge devices, allowing predictions to terminate early when confidence is high. This mechanism enables models to complete inference faster, lowering computational load and energy consumption. BranchyNet [9] formally introduced the concept, marking one of the first applications of EE in deep neural networks. By balancing inference latency and accuracy, EE improves HAR performance on mobile and wearable devices. Incorporating these strategies is essential for building energy-efficient HAR models.

This study investigates an energy-efficient HAR model optimized for mobile devices. To design an accurate and low-power system, we developed two applications—one running locally and the other remotely in the cloud—based on models with early exits and adaptive sampling. A systematic comparison of these applications, in terms of accuracy and energy efficiency, demonstrates the advantages of embedded over cloud-based processing. The remainder of this article is structured as follows: Section 2 presents related work; Section 3 covers the theoretical background; Section 4 details the proposed method; Section 5 discusses results; and Section 6 concludes the paper and outlines future directions.

2. Related Work

HAR has advanced significantly with the integration of deep learning techniques aimed at improving both accuracy and energy efficiency in mobile devices. Several studies have proposed strategies to reduce energy consumption while maintaining high performance in activity recognition applications. This section reviews key approaches and innovations in energy-efficient HAR systems, focusing on adaptive sampling, context-aware classification, and early exit strategies that optimize deep learning models for smartphones and wearables, both on-device and in the cloud.

[10] demonstrated the feasibility of achieving high recognition accuracy with reduced sampling rates while preserving energy efficiency. The authors introduced a hierarchical classifier, HSVMCC, which combines Support Vector Machines (SVM) with context-based classification. By incorporating past and future contextual variables around each activity, the technique improved accuracy even at low sampling rates. As a result, their approach achieved up to 59.6% energy savings compared to standard methods, without compromising recognition accuracy. Similar to [10], which leverages context to offset low sampling rates, our solution combines adaptive sampling (based on RMS) with dynamic decision mechanisms. While HSVMCC relies on predefined rules, our system automatically adjusts to variations in motion intensity and activity complexity.

[11] addressed energy efficiency in continuous accelerometer usage, emphasizing the high sensitivity of these sensors to prolonged activity monitoring in healthcare applications. To lower energy consumption, the study varied sampling rates, window sizes, and feature vector dimensionality across six motion types. This strategy reduced energy use by 44–79% while maintaining classification accuracy.

In the context of smartphone-based HAR, [12] proposed the SFCN model, which employs a convolutional neural network to balance feature extraction and accuracy. The system integrates a clustering-center-based pre-classification module that reduces the number of calls to the main deep learning model, thereby saving energy. By lowering the load on the primary model and controlling sampling frequency, this approach improves overall efficiency.

[2] introduced AHAR (Adaptive Human Activity Recognition), an adaptive CNN model designed for low-power edge devices. Built on an edge-computing architecture, the solution reduces latency and energy consumption by processing part of the data locally while preserving accuracy. The CNN architecture supports early exits, dynamically adjusting network depth according to prediction confidence to further optimize efficiency. Unlike that work, which performs all inference locally with an exclusive focus on edge devices, our approach adopts a hybrid design that combines local and cloud execution. We also compare different architectures (CNN, EE-CNN, and EE-LSTM) and integrate adaptive sampling to adjust sensor reading frequency according to motion intensity.

[13] proposed a computationally efficient method based on segment-level change detection. Instead of processing all samples continuously, the system targets transitions, reducing energy use. Using a fully convolutional neural network (FCN), the approach enabled deployment on embedded devices with sixfold energy savings compared to conventional CNNs, without sacrificing accuracy. In wearable devices, [14] proposed a binary convolutional neural network (BCNN) to optimize energy consumption in continuous health monitoring applications. By binarizing weights and activation functions, BCNN reduces memory requirements and improves efficiency compared to conventional networks. This model consumed up to 12 times less energy while maintaining

https://doi.org/10.21528/lnlm-vol24-no1-art1

90.29% accuracy.

In the IoT context, [15] introduced AE-HAR (Accuracy and Energy-Aware HAR), a hybrid model that alternates between local and cloud processing. The system incorporates modules for monitoring energy consumption and connectivity, dynamically deciding where to execute inference. This strategy balances efficiency and accuracy by using local processing for simpler tasks and cloud resources for more complex analyses. Among the reviewed works, AE-HAR is the closest to our approach, since both integrate deep learning across local and cloud environments with a focus on accuracy and energy efficiency. However, while AE-HAR employs dynamic switching based on prediction confidence and energy metrics, our study performs a controlled quantitative comparison between local and remote execution using different models and optimization strategies, including early-exit networks and adaptive sampling. More recently, Tokas et al. [16] proposed an attention-driven CNN–LSTM framework optimized for real-time deployment on wearable devices, further demonstrating the potential of hybrid architectures for efficient HAR inference. Table 1 summarizes the main strategies proposed in related work for energy-efficient HAR.

Author	Device	Deployment	Strategy
[14]	Wearable	L	BCNN + TMEX; binary CNN with Early Exit; 12× energy savings
[2]	Edge	L	Adaptive CNN with Early Exit; dynamic block selection; high accuracy and energy savings
[15]	Smartphone	L+C	Hybrid model (local/cloud) with decision based on accuracy and energy efficiency
[10]	Smartphone	L	Hierarchical H-SVM + context-based classification; low sampling (1 Hz); 17–60% energy savings
[11]	Smartphone	L	Lightweight ML (VARD); dynamic adjustment of window and frequency; energy savings without accuracy loss
[12]	Smartphone	L	DL with pre-classification (clustering) and sampling control; 49% energy savings
[13]	Wearable	L	FCN with change detection; classification only during transitions; 6.5× energy savings
[16]	Wearable	L	Attention-driven CNN–LSTM; real-time deployable model with attention mechanism for feature fusion and temporal modeling; optimized for on-device inference vs. energy savings
This Study	Smartphone	L+C	CNN, EE-CNN, EE-LSTM, and adaptive sampling; hybrid processing with early exit

Table 1: Comparison of strategies from each related work. Legend: *L = Local C = Cloud.*

3 Theoretical Background

This section presents the theoretical foundations that support the models and techniques adopted in this study, including Convolutional Neural Networks (CNN), Early Exit Convolutional Networks (EE-CNN), Early Exit LSTM (EE-LSTM), and adaptive sampling.

3.1 CNN Model

CNNs are a specialized class of deep neural networks widely used to process grid-structured data, such as images, videos, and time series [17]. They are grounded in the fundamental principles of deep learning and hierarchical representation learning described in standard deep learning literature [18]. Inspired by the organization of the human visual cortex, CNNs consist of convolutional layers that apply filters (or kernels) responsible for automatically detecting local patterns, such as edges and textures. As data moves through the layers, these representations become progressively more abstract, allowing the model to learn high-level features.

Furthermore, pooling layers reduce dimensionality and computational cost, while fully connected layers perform the final classification. A traditional CNN model is composed of a sequence of layers: convolutional layers that apply filters to extract local features; max-pooling layers that reduce dimensionality while preserving the most relevant information; a flatten layer that transforms the feature map into a one-dimensional vector; fully connected layers that combine the extracted features; the dropout mechanism, applied to reduce overfitting; and finally, the softmax layer, which produces the probability distribution across the output classes. Figure 1 illustrates the architecture of a standard CNN.

https://doi.org/10.21528/lnlm-vol24-no1-art1

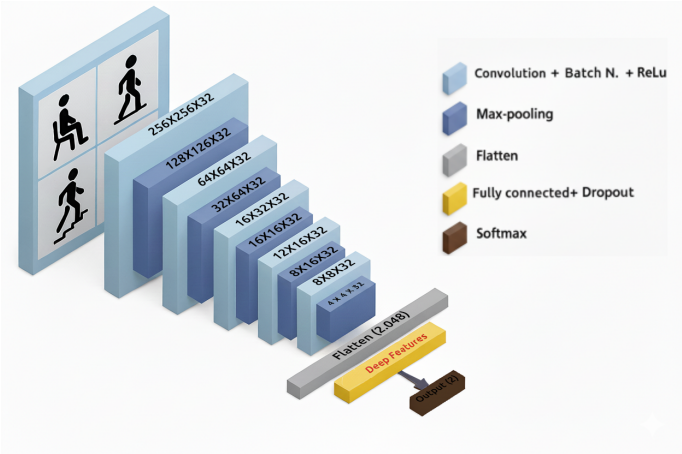

Figure 1: Standard CNN architecture. Adapted from [19].

3.2 EE-CNN and EE-LSTM Models

The EE-CNN model is built similarly to a conventional CNN, but with intermediate classifiers inserted along the network [20]. This technique accelerates neural network processing by reducing execution time—especially on resource-constrained devices such as smartphones and embedded systems. The goal is to enable early predictions, before all network layers are processed, whenever a reliable output can already be obtained [21]. Figure 2 illustrates this scenario. The auxiliary predictor is highlighted in green.

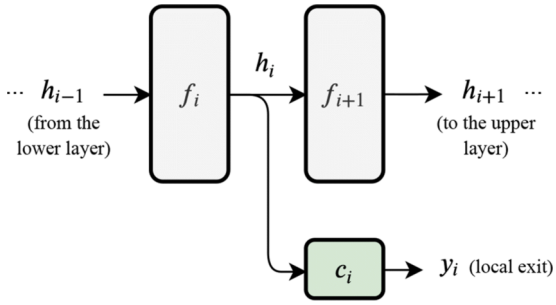

Figure 2: Graphical representation of a generic early exit in neural network architectures, proposed in [22].

In conventional networks, data passes through all layers until the final output, creating high computational demand, especially in tasks such as image or activity recognition. In the EE approach, exit points are added in intermediate layers, acting as auxiliary classifiers that generate preliminary predictions based on confidence thresholds [2]. EE therefore improves efficiency by avoiding full network processing, saving resources and reducing latency in constrained devices. While CNN-based EE models typically place exit points after convolutional blocks, recurrent architectures such as LSTMs can also benefit from early exits. The EE-LSTM introduces exit points at intermediate timesteps or hidden states, allowing the model to generate predictions without processing the full sequence. This strategy reduces computational cost and inference time, particularly in time-series tasks like HAR. By leveraging partial temporal dependencies, the network dynamically adapts the depth of sequential processing according to the confidence of intermediate outputs [22]. Such flexibility makes EE-LSTM especially suitable for streaming sensor data, where early and accurate decisions are crucial for energy efficiency.

3.3 Adaptive Sampling

Adaptive sampling dynamically adjusts the sampling interval according to signal variations, optimizing data collection and reducing resource consumption. Jon [23] proposed a method for wireless sensor networks based on a Kalman filter, in which the sampling frequency is adjusted according to differences between measurements, ensuring energy efficiency without loss of accuracy. In the context of HAR, statistical measures such as the Root Mean Square (RMS) can be used to quantify signal magnitude, serving as a criterion to guide adaptive sampling and ensuring that significant activity changes are captured [24]. The RMS value of a discrete signal $x[n]$ is calculated according to Equation 1, which provides a measure of its effective magnitude.

https://doi.org/10.21528/lnlm-vol24-no1-art1

$$x_{\text{RMS}} = \sqrt{\frac{1}{N}\sum_{n=0}^{N-1}[x[n]]^2} \tag{1}$$

Where $x[n]$ represents the value of the discrete signal at sample n, and N corresponds to the total number of samples considered. In this work, RMS values are computed from the triaxial accelerometer signal and accumulated over non-overlapping temporal windows of 30 seconds. For each window, the average RMS value is calculated and compared against an empirically defined threshold of 5.9 to distinguish between static and dynamic activities. When the average RMS is below this threshold, the accelerometer sampling frequency is reduced to 5 Hz, whereas values equal to or above the threshold trigger a higher sampling frequency of 100 Hz. This adaptive mechanism allows the acquisition rate to be dynamically adjusted according to activity intensity, reducing unnecessary energy consumption during periods of low motion.

4 Materials and Methods

The methodology consisted of several stages, beginning with system instrumentation, in which a smartphone was used as an accelerometer to collect inertial data from users. Next, the experimental protocol was defined, involving four standard activities: standing, sitting, climbing stairs, and descending stairs. After data acquisition, preprocessing was performed, followed by the definition of predictive models and the configuration of both local and remote execution environments. In the final phase, predictive applications were tested, energy consumption was measured for each application, and results were extracted.

In the local application, the evaluation focused on the impact of different sampling frequencies using the adaptive sampling technique combined with LRT (Lite Run-Time) models. In the remote application, models with early exits (EE-CNN and EE-LSTM) were employed. The experiment systematically assessed the trade-off between energy efficiency and accuracy in both execution environments. Figure 3 illustrates the complete workflow of the application, from data collection to the model-generated response.

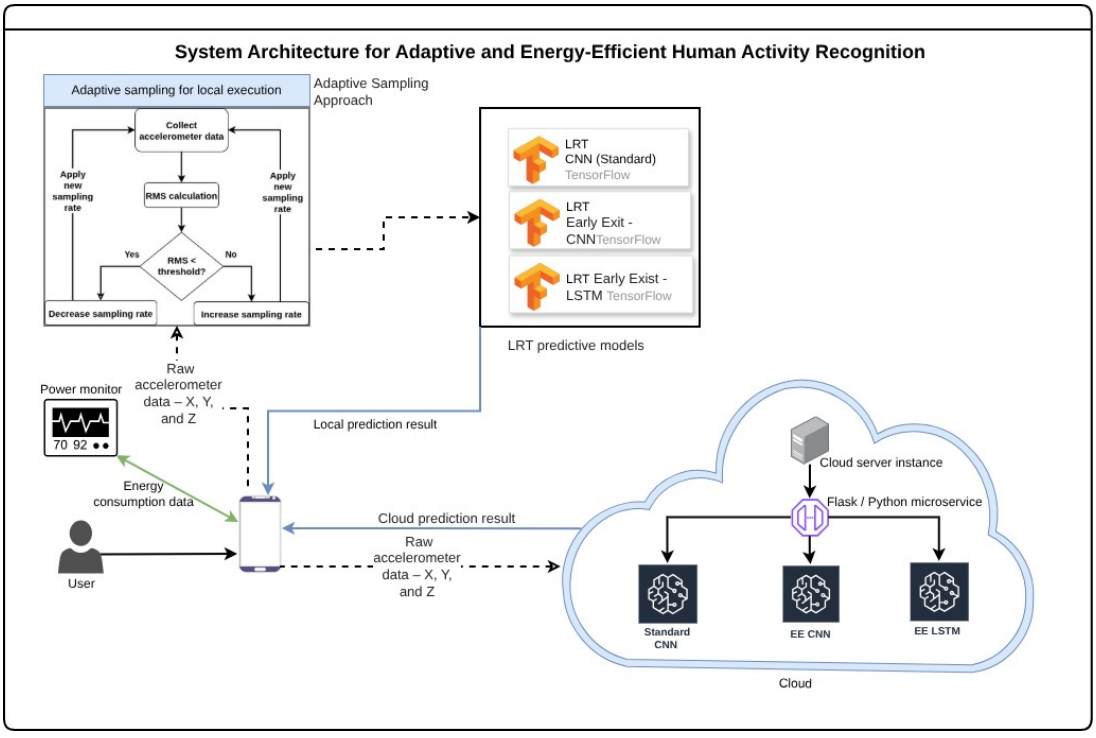

Figure 3: Hybrid local and cloud model architecture.

The proposed solution consists of two complementary applications: an embedded version optimized for resource-constrained devices, and a remote cloud-based version executed on the AWS platform. In the latter, raw data collected locally is transmitted to the cloud server for processing. In this environment, data undergoes preprocessing before being submitted to a neural network model, which may be: a conventional CNN (baseline predictor), an EE-CNN, or an EE-LSTM, both incorporating early exits. In parallel, a local application operates with models compiled in LRT format, derived from the aforementioned architectures. The solution also includes an additional implementation layer: the adaptive sampling module, which continuously monitors motion intensity and dynamically adjusts the sampling frequency every 30 seconds. To validate the obtained results, an additional monitoring layer was incorporated to measure real-time energy consumption, generating quantitative log data.

https://doi.org/10.21528/lnlm-vol24-no1-art1

The cloud-based experiments were conducted using a standard client–server configuration, in which the smartphone transmitted the collected data to the cloud server over a stable Wi-Fi connection. The remote execution environment was hosted on an AWS EC2 t2.micro instance, equipped with 1 vCPU, 4 GB of RAM, and 30 GB of storage. The São Paulo region was selected due to its lower communication latency. GPU acceleration was not employed, as the objective was solely to contextualize cloud-based inference rather than to optimize server-side processing performance.

4.1 Data Acquisition

An experiment was conducted with 8 healthy participants (6 men and 2 women), aged between 15 and 50 years. The study was approved by the Ethics Committee of the Fluminense Federal University, Niterói/RJ, Brazil, and each participant signed an informed consent form prior to participation (CAAE 31189520.3.0000.5243). Data were collected using a Galaxy J2 Prime smartphone positioned at the participant's waist. A custom dataset was collected instead of using a public HAR dataset to allow precise control of sampling rates during acquisition and to reflect the resource limitations of the smartphone employed in the experiments.

The Phyphox application (http://phyphox.org) was used to obtain linear acceleration on the three axes during two dynamic activities (climbing and descending stairs) and two static activities (sitting and standing). Each activity was performed for 10 minutes, resulting in a total data collection time of 40 minutes per participant. To assess the energy impact of local and remote inference, the *TuneUp Kit* software tool, developed by Qualcomm for analyzing and measuring energy consumption on mobile devices, was used.

4.2 Preprocessing

The data preparation process involved several steps to ensure the quality and representativeness of the dataset used for model training and evaluation.

Z-score normalization. Raw data normalization was performed using the Z-score technique to standardize acceleration signals and mitigate the effect of outliers in the segments [2]. This method transforms the data such that the mean is zero and the standard deviation equals one, as shown in Equation 2:

$$Z_i = \frac{X_i - \bar{X}}{S} \tag{2}$$

Where: Z_i represents the standardized value, X_i the original value, $\bar{X}$ the mean, and S the standard deviation of the series.

Class balancing. The initial analysis revealed a significant imbalance among activity classes, with a predominance of Descending Stairs and Standing, as shown in Table 2. To reduce bias during training and improve model generalization, a subsampling technique was applied to the majority classes, equalizing the number of samples across all activities.

Activity	Samples	%
Descending Stairs	596504	33.92
Standing	506435	28.81
Climbing Stairs	327691	18.64
Sitting	327689	18.64

Table 2: Original distribution of samples per activity.

Table 3 presents the new distribution obtained after balancing, with all classes adjusted to contain the same number of samples.

Activity	Original (Samples)	Original (%)	Balanced (Samples)	Balanced (%)
Descending Stairs	596,504	33.92	327,689	25.00
Standing	506,435	28.81	327,689	25.00
Climbing Stairs	327,691	18.64	327,689	25.00
Sitting	327,689	18.64	327,689	25.00

Table 3: Distribution of samples before and after class balancing by subsampling.

Segmentation. After balancing, the data were segmented using a sliding window of 80 samples with 50% overlap. This procedure is essential for capturing the continuous temporal characteristics of acceleration signals. **Data augmentation.** The initial application of the model indicated that the dataset was still insufficient to adequately represent activity variability. To address this limitation, data augmentation was applied exclusively to the training set. For each original segmented window, six additional synthetic samples were generated using time-series data augmentation techniques, namely: additive Gaussian noise (noise factor of 0.01), jittering with uniform noise in the range [-0.01, 0.01], random amplitude scaling with scaling factors

https://doi.org/10.21528/lnlm-vol24-no1-art1

between 0.8 and 1.2, time warping based on temporal distortion with standard deviation $\sigma = 0.2$ and linear interpolation, sign flipping, and segment shuffling into 10 segments. As a result, the training set was expanded to 160,559 windows, while the validation and test sets (both with 4,915 windows) remained unchanged. **Label encoding.** Finally, label encoding was applied using the *LabelEncoder* technique, converting the textual categories of activities into numerical values. The resulting order was: 'Descending Stairs' (0), 'Climbing Stairs' (1), 'Standing' (2), and 'Sitting' (3), enabling efficient inference by the model.

4.3 Data Segmentation and Partitioning

The raw accelerometer signals were segmented into sliding windows of fixed length containing 80 samples, using a 50% overlap, which results in a shift of 40 samples between consecutive windows. Each window was labeled according to the dominant class among the samples it contains, determined by maximizing the frequency of activity labels within the window interval. After the windowing process, the resulting segments were randomly partitioned into training (60%), validation (20%), and test (20%) sets, with the partitioning performed at the window level rather than at the participant level. Since the partitioning was applied after windowing, overlapping windows may share samples across different sets; however, this protocol was consistently adopted across all experiments in order to preserve a sufficient number of training samples and to ensure comparability of the results among the evaluated models.

4.4 Training of the Standard CNN Classifier

The standard CNN network model was developed in TensorFlow/Keras for multiclass classification, following the dataset partitioning protocol described in Section 4.3 (Data Segmentation and Partitioning), with 60% of the windows used for training, 20% for validation, and 20% for testing. The architecture includes two convolutional layers with 8 and 16 filters (kernel 3×3, ReLU), followed by max pooling, as well as a dense layer with 32 neurons, L2 regularization ($\lambda = 0.01$), and Dropout (0.5).

Empirical tests with different learning rates (0.01, 0.005, 0.001) showed that 0.001 combined with the Adam optimizer provided the best performance. The model was trained for up to 100 epochs using the EarlyStopping technique, which monitors the validation metric and automatically stops training if no improvement occurs after three consecutive epochs. This strategy helps prevent overfitting and ensures better overall performance.

4.5 Training of the EE-CNN Classifier

The architecture of the Early-Exit Convolutional Neural Network (EE-CNN) includes two convolutional layers with 8 and 16 filters, followed by a pooling layer. Several hyperparameters were tuned empirically, such as the number of filters (8, 16, 32), kernel size (3×3, 5×5), and activation functions (ReLU, tanh). The best configuration was obtained with 8 and 16 filters, kernel 3×3, and ReLU activation.

To implement the early-exit technique, in addition to the final predictor output, an intermediate exit point called *exit-1* was added immediately after the first convolutional and pooling layers. At this stage of the model, the data already undergo an initial feature extraction, which may be sufficient to classify simpler samples, such as static activities. If the probability distribution generated by the Softmax function at this point exhibits low entropy—indicating high confidence in the prediction—the model performs early inference.

In this work, early inference is triggered when the maximum Softmax probability at the intermediate exit (exit-1) exceeds a confidence threshold empirically set to 0.98. This value was selected after evaluating multiple thresholds in order to balance early-exit frequency and classification accuracy. As shown in Table 4, with this threshold approximately 96.8% of the test samples are classified at the intermediate exit, while only 3.2% require full processing by the network.

Threshold (t)	Exit rate (%)	Hybrid accuracy (%)
0.90	99.94	99.95
0.95	99.86	99.95
0.98	**96.77**	**99.95**
0.99	77.68	99.95
0.995	71.53	99.95

Table 4: Relationship between confidence threshold, early-exit rate, and hybrid accuracy.

This approach reduces processing time. However, if the input is more complex, the data continue through the remaining network layers. This principle is analogous to hierarchical decision systems, in which a preliminary response is sufficient for low-complexity events, whereas more detailed decisions require additional processing. Several methods can be used to define an appropriate threshold for interrupting processing. In [21], three criteria for early-exit thresholds are discussed; this study adopts the entropy-based criterion to measure uncertainty and evaluate confidence in the dominant class. The entropy of the probability distribution is defined as:

$$H(P) = -\sum_{i} P(i) \log P(i)$$

https://doi.org/10.21528/lnlm-vol24-no1-art1

where $P(i)$ represents the probability of the i-th class.

For model compilation, the Adam optimizer and the sparse-categorical-crossentropy loss function were used, with accuracy as the evaluation metric. As with the other models in the experiment, training included the EarlyStopping technique. To validate the impact of the early-exit mechanism, the same dataset partitioning as in the Standard CNN was employed, ensuring comparable conditions between the EE-CNN and EE-LSTM predictors.

4.6 Training of the EE-LSTM Classifier

The EE-LSTM architecture consists of three stacked LSTM layers, with hyperparameters tuned empirically, including the number of units per layer (16, 32, 64), activation functions (tanh, ReLU), dropout rates (0.3, 0.5), and optimizers (Adam, RMSprop). The best-performing configuration included 32 units in the initial and final layers, 16 units in the early-exit layer, tanh activation, and a dropout rate of 0.5. An intermediate early-exit branch is introduced after the first LSTM stage. At this point, the model produces a Softmax probability distribution that is used to estimate prediction confidence.

As in the EE-CNN model, early inference is triggered when the maximum Softmax probability exceeds a predefined confidence threshold. For the EE-LSTM, a stricter threshold of 0.995 was empirically adopted to enforce meaningful hierarchical computation. As summarized in Table 5, this threshold results in early exit for approximately 79.2% of the test samples while preserving a hybrid classification accuracy of 99.96%. Samples that do not satisfy the confidence criterion continue through the remaining LSTM layers for final classification. This adaptive mechanism allows the network to allocate additional computational effort only to more complex inputs, improving efficiency without sacrificing predictive performance. It is worth noting that, similarly to the other neural models evaluated in this study, the EE-LSTM employed the same dataset partitioning strategy and the EarlyStopping technique to prevent overfitting and to ensure fair comparison across architectures.

Threshold (t)	Exit rate (%)	Hybrid accuracy (%)
0.90	99.99	99.96
0.95	99.97	99.96
0.98	99.97	99.96
0.99	99.97	99.96
0.995	**79.21**	**99.96**

Table 5: Relationship between confidence threshold, early-exit rate, and hybrid accuracy for the EE-LSTM model.

4.7 Adaptive Sampling

The Adaptive Sampling technique aims to dynamically adjust the data collection frequency of inertial sensors according to user motion variability.

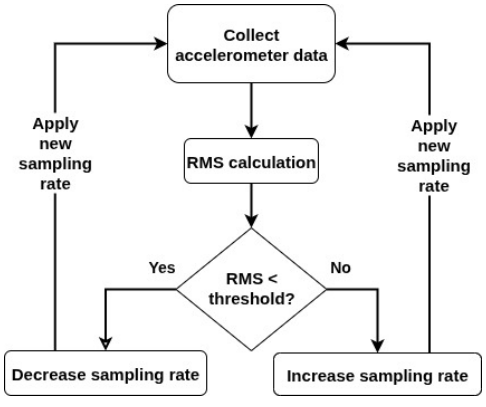

Figure 4: Flow of the RMS-based adaptive sampling algorithm for dynamic frequency adjustment.

The process relies on analyzing the average RMS (Root Mean Square) value, computed every 30 seconds. If the average RMS is below the predefined threshold (5.9, obtained empirically), corresponding to static activities, the system reduces the sampling frequency, minimizing energy consumption. Otherwise, if the RMS is equal to or above the threshold, the sampling frequency increases to capture more details from the data. This adaptive adjustment optimizes energy use by enabling more efficient data collection only when necessary.

https://doi.org/10.21528/lnlm-vol24-no1-art1

5 Results and Discussion

The predictive models were evaluated in the two scenarios, local and remote. Results were analyzed from three main perspectives: accuracy, energy consumption, and energy efficiency. The analysis focused on the impact of adaptive sampling frequency and early-exit models (EE-CNN and EE-LSTM), aiming to identify the best balance between predictive performance and efficient resource usage. Table 5 summarizes the results, highlighting execution environment, models, operating frequencies, energy consumption, and accuracy.

Experiment 1–18 (Frequency 10 Hz)		
Model	**Avg. Cons. (mW)**	**Acc. (%)**
Standard CNN	712	98.47
EE-CNN	649	98.89
EE-LSTM	571	98.89
LRT Standard CNN	286	94.67
LRT EE-CNN	237	90.00
LRT EE-LSTM	226	93.33
Experiment 19–36 (Frequency 30 Hz)		
Standard CNN	725	99.31
EE-CNN	656	99.03
EE-LSTM	640	99.17
LRT Standard CNN	285	95.33
LRT EE-CNN	238	91.00
LRT EE-LSTM	245	94.67
Experiment 37–54 (Frequency 50 Hz)		
Standard CNN	720	100.00
EE-CNN	672	99.44
EE-LSTM	663	99.31
LRT Standard CNN	302	95.67
LRT EE-CNN	250	92.33
LRT EE-LSTM	245	95.17
Experiment 55–72 (Frequency 100 Hz)		
Standard CNN	760	100.00
EE-CNN	700	99.58
EE-LSTM	665	99.72
LRT Standard CNN	320	96.83
LRT EE-CNN	264	93.83
LRT EE-LSTM	273	95.00

Table 6: Energy consumption and accuracy metrics by sampling frequency.

In the experiments carried out, two sets of neural network models were evaluated: the original models executed in a cloud environment (Standard CNN, EE-CNN, and EE-LSTM) and their optimized versions for mobile devices using LRT (LRT Standard CNN, LRT EE-CNN, and LRT EE-LSTM). Each set was executed in its respective environment, enabling a comparative performance analysis between the two settings. The tests were conducted under controlled conditions, in which, for each combination of predictive models (local and cloud) and their respective operating frequencies (10 Hz, 30 Hz, 50 Hz, and 100 Hz), three independent runs of 10 minutes each were performed.

To illustrate this case, consider the optimized EE-CNN model (LRT) executed in the local environment. For the 10 Hz frequency, three independent runs (T1, T2, and T3) were carried out, each lasting 10 minutes. At the end of each run, in addition to inference time, two main indicators were recorded: device energy consumption and the predictive accuracy achieved by the model. The experiments resulted in a total of 72 iterations, with 36 in the cloud environment and 36 in the local environment.

5.1 Accuracy – Standard CNN vs. Early-Exit Models in Local and Cloud Environments

Models were evaluated at four operating frequencies (10 Hz, 30 Hz, 50 Hz, and 100 Hz) in both cloud and local environments. As shown in Figures 5 and 6, model accuracy remained consistently high across all evaluated sampling frequencies, exceeding 98% in most scenarios and reaching 100% at 50 Hz and 100 Hz. Although cloud-based models demonstrated greater stability under frequency variations, local models exhibited a slight increase in variability as the sampling rate increased. This variability may reflect local computational constraints, such as limited processing or resource management. Among the predictors, the Standard CNN achieved the best accuracy in all configurations.

Its superior performance is explained by processing the complete input data, which allows a more detailed analysis. By contrast, the EE-CNN showed accuracy reductions, especially at lower frequencies (10 Hz and 30 Hz), possibly due to fewer data being available in early layers, leading to premature decisions. The EE-LSTM delivered intermediate performance, surpassing EE-CNN in many cases but remaining below the Standard CNN. Overall, accuracy improved with higher sampling frequencies, although the values remained very close.

https://doi.org/10.21528/lnlm-vol24-no1-art1

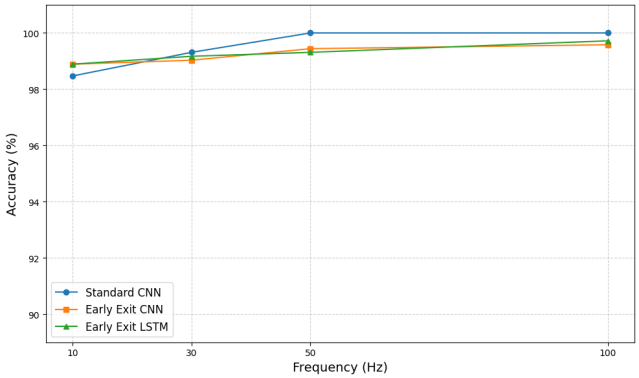

Figure 5: Accuracy by frequency in the cloud.

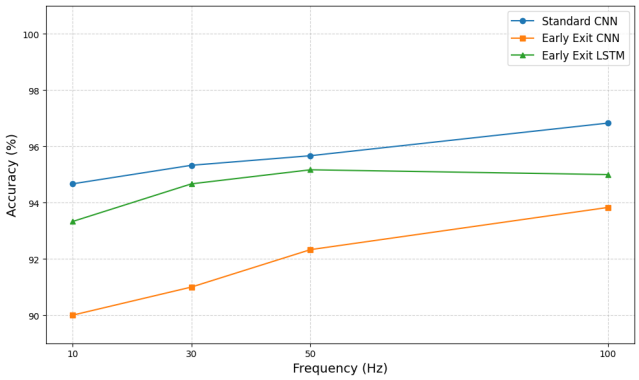

Figure 6: Accuracy by frequency in the local environment.

5.2 Energy Consumption – Local vs. Cloud

Energy consumption was measured in both environments under controlled tests, varying sampling frequency and model architecture. The local application used adaptive sampling, while the cloud application ran with fixed frequencies for experimental control and comparability. It is important to note that the observed differences between the local and cloud scenarios reflect both the execution location (on-device versus remote processing) and the use of adaptive sampling in the local configuration. Therefore, the reported gains result from the combined effects of reduced communication overhead and sampling optimization. The analysis showed clear differences between environments. Average energy consumption in the local environment was 264.22 mW, compared to 677.67 mW in the cloud. Thus, cloud execution consumed 61.01% more energy on average. Adaptive sampling proved effective in reducing local energy use, especially during static activities with low signal variability. Lower sampling frequencies generally stabilize energy consumption by reducing data volume and computational demand. In contrast, systems with fixed high sampling rates consume more energy, even when complexity does not require it, as in the cloud case. Communication with the cloud adds an additional energy overhead on the device. Table 7 reports the statistical comparison.

Environment	Energy Cons. (mW)	Std. Dev. (mW)
Local	264.22	29.16
Cloud	677.67	49.53

Table 7: Statistical comparison of energy consumption between environments.

Cloud execution consumed approximately 2.56 times more energy than local execution. The lower standard deviation observed locally reflects adaptive sampling, which smooths computational load during periods of low activity. These results indicate that the higher energy consumption observed in the cloud environment is associated with communication overhead between the device and the server, as well as the absence of adaptive sampling in that configuration. In addition, the processes required to prepare, transmit, and receive data impose extra computational and communication costs that exceed those of optimized local execution. Together, these factors substantially increase the total energy usage in the cloud scenario.

https://doi.org/10.21528/lnlm-vol24-no1-art1

5.3 Energy Efficiency – Local vs. Cloud

Energy efficiency was measured as the ratio of accuracy to energy consumption, following [25]:

$$\eta = \frac{\text{Accuracy}}{\text{Energy}} \tag{3}$$

where η: energy efficiency of the application, *Accuracy*: predictive model performance (%), *Energy*: energy consumption during execution (mW). As shown in Table 8, the local application achieved 2.43 times higher efficiency than the cloud application. Even with lower accuracy, the local model produced more correct predictions per unit of energy consumed, indicating more effective energy use.

Environment	Efficiency	Std. Dev.	Accuracy (%)
Local	0.359	0.033	93.99
Cloud	0.148	0.011	99.32

Table 8: Average energy efficiency, standard deviation, and accuracy in local and cloud environments.

Cloud processing ensured higher accuracy and more stable efficiency across frequencies, but local execution demonstrated substantially higher energy efficiency, making it more suitable for scenarios where battery autonomy is critical.

5.4 Energy Efficiency – EE vs. Standard CNN Models

Comparing model efficiencies (Standard CNN, EE-CNN, EE-LSTM) revealed significant differences by frequency. Overall, EE-LSTM achieved the best results, peaking at 0.4136 efficiency at 10 Hz.

Freq. (Hz)	Standard CNN	EE-CNN	EE-LSTM
10	0.3314	0.3797	0.4136
30	0.3345	0.3818	0.3869
50	0.3168	0.3688	0.3879
100	0.3026	0.3554	0.3484

Table 9: Comparison of model efficiencies.

This result suggests that when energy consumption is critical, EE-LSTM is the best choice, although its efficiency approaches other models at higher frequencies. EE-CNN achieved intermediate efficiency, consistently outperforming the Standard CNN, confirming the benefits of early exits for saving energy without compromising accuracy. These findings confirm the effectiveness of early-exit strategies in reducing energy use while maintaining high accuracy. EE-LSTM achieved an average accuracy of 94.54%, supporting its suitability for edge and mobile applications.

5.5 Impact of Early Exit on Inference Time – EE-LSTM vs. Standard CNN in Cloud Environment

Accelerometer data from X, Y, and Z axes were grouped into 80-sample vectors for inference in the cloud. The total response time included data capture, vector storage, wireless transmission, reception, preprocessing, and model inference hosted on AWS. On average, the EE-LSTM model required 3.02 seconds for static activities and 3.20 seconds for dynamic activities. The Standard CNN required 3.30 seconds for static and 3.58 seconds for dynamic activities. This represents reductions of 8.48% and 10.61%, respectively, averaging 9.59% overall. Other studies reported higher cloud inference latencies depending on system optimization. For example, [26] found that a complete CNN-based action recognition system required 7.1 seconds per prediction, while its optimized Fast-HEAR version reduced latency to 2.4 seconds without accuracy loss. Table 10 compares average inference time and accuracy for both models.

Activity	Inference Time (s)		Reduction (%)	Accuracy (%)	
	Standard CNN	EE-LSTM		Standard CNN	EE-LSTM
Static	3.30	3.02	8.48	98.3	95.9
Dynamic	3.58	3.20	10.61	97.8	96.9
Average	3.44	3.11	9.59	98.05	96.4

Table 10: Comparison of Standard CNN and EE-LSTM in average inference time (s) and accuracy (%).

Although early exit slightly reduced accuracy for static activities, the gain in inference speed outweighed this drawback. For dynamic activities, EE-LSTM accuracy was closer to the Standard CNN, suggesting that early exit was less frequently triggered in more complex sequences. These results support early exit as a practical strategy when fast responses are required with limited computational cost.

https://doi.org/10.21528/lnlm-vol24-no1-art1

5.6 Energy Performance Efficiency Metrics

To provide a comparison among the evaluated models, the energy consumption analysis was complemented with energy-performance efficiency metrics. In addition to reporting the average power consumption, energy per inference and the accuracy-per-Joule ratio were considered. The total energy consumption was obtained from the fundamental relationship between power and time:

$$E = P \times T \tag{4}$$

where P represents the average power and T the execution time. Since power was measured in milliwatts (mW) and time in seconds (s), the resulting energy is expressed in millijoules (mJ).

$$E_{total} = P_{mean} \cdot T \tag{5}$$

The energy per inference was calculated by dividing the total energy by the total number of performed predictions:

$$E_{inf} = \frac{P_{mean} \cdot T}{N} \tag{6}$$

where N corresponds to the total number of processed inferences. Additionally, the accuracy-per-Joule ratio was defined as:

$$Accuracy/J = \frac{Accuracy}{E_{inf}/1000} \tag{7}$$

These metrics enable a direct evaluation of the trade-off between predictive performance and energy consumption in the embedded environment. Table 11 presents the obtained results. As observed in Table 11, the LRT early-exit models reduce the energy consumed per inference compared to the LRT standard CNN. Although the standard architecture achieves the highest absolute accuracy, the LRT EE-LSTM provides the highest accuracy-per-Joule ratio, demonstrating a superior energy-performance trade-off in the embedded scenario.

Model	Mean power (mW)	Energy per inference (mJ)	Accuracy	Accuracy/J
LRT (CNN Standard)	298.17	894.50	0.9563	1.071
LRT (EE CNN)	247.42	742.25	0.9179	1.238
LRT (EE LSTM)	247.08	741.25	0.9454	1.281

Table 11: Energy-performance efficiency metrics for embedded models

5.7 Factor Isolation Analysis

Table 12 presents the isolation of the effects of the evaluated techniques. The results indicate that the *early exit* mechanism is the primary factor responsible for the reduction in energy consumption. When applied independently, it achieves an approximate 17% reduction in average power consumption, with a moderate impact on accuracy. In contrast, *adaptive sampling*, when evaluated independently, shows a limited impact on energy reduction under the experimental conditions considered, while keeping accuracy virtually unchanged. Although the combined approach preserves the energy savings achieved by early exit, it does not significantly reduce consumption beyond that level. This indicates that early exit accounts for most of the energy reduction observed in the experiments.

Scenario	Adaptive	Early Exit	Avg. Acc. (%)	Avg. Power (mW)	Δ Power	Δ Acc.
Baseline	No	No	95.75	300.50	–	–
Adaptive Only	Yes	No	95.63	298.25	-2.25 (-0.7%)	-0.12
Early Exit Only	No	Yes	91.83	248.00	-52.50 (-17%)	-3.92
Combined	Yes	Yes	91.79	247.25	-53.25 (-17.7%)	-3.96

Table 12: Isolation of the Effects of Adaptive Sampling and Early Exit

5.8 Impact of Execution Environment on Average Inference Time – Local vs. Cloud

Table 13 presents latency and energy consumption for recognition applications executed locally and in the cloud. Local execution achieved lower latency (2.9 s) and lower energy consumption (264.22 mW) compared to cloud execution (677.67 mW). This confirms the suitability of local inference for applications requiring rapid predictions and extended autonomy in energy-constrained devices.

Metric	Local	Cloud
Latency	2.9 s	3.23 s
Energy Consumption	264.22 mW	677.67 mW

Table 13: Inference metrics in local and cloud environments.

As reported by [27], local inference with model LRT TensorFlow can achieve low energy consumption and millisecond-level response times. Both latency and energy usage depend strongly on sensor update rate. Higher sampling frequencies produce more data, requiring faster processing but increasing energy demand. Lower frequencies reduce system load, but may compromise real-time response. These findings highlight the viability of embedded solutions with optimized models such as early-exit architectures for smartphones, edge computing, wearables, and IoT applications.

6 Conclusion

This work developed and evaluated energy-efficient HAR models for smartphones by combining adaptive sampling and deep learning architectures with early exits (EE-CNN and EE-LSTM). Two complementary applications were implemented and compared: one running locally on the device and another executed remotely in the cloud. The experiments demonstrated that local execution using adaptive sampling and LRT optimization reduced energy consumption by approximately 61% compared to cloud inference (264.22 mW vs. 677.67 mW) while maintaining accuracy above 93%. The EE-LSTM model achieved the best balance between accuracy (94.5% on average) and efficiency (0.41 accuracy/mW), confirming the advantage of combining recurrent structures with early-exit mechanisms for resource-constrained environments. These results validate the feasibility of embedded inference as an alternative to full cloud dependency in mobile HAR systems.

The results indicate that local execution tends to be more advantageous in scenarios where energy autonomy, low latency, and data privacy are critical requirements. Embedded inference, combined with adaptive sampling and models optimized with early exit mechanisms, showed significantly lower energy consumption and shorter response times, making it particularly suitable for continuous HAR applications on smartphones or in contexts with limited or unstable connectivity. Conversely, cloud-based execution may be preferable in situations where reliable connectivity and low network latency are available, as well as when more complex models with higher computational costs are required, demanding greater processing capacity and higher levels of accuracy. Thus, the choice between local or cloud execution should consider the trade-offs among energy consumption, latency, data privacy, and application performance requirements.

Despite the promising results, some limitations must be acknowledged. The experimental evaluation was conducted with only eight healthy participants, which limits statistical representativeness and the diversity of movement patterns. In addition, data collection and energy consumption measurements were performed using a single smartphone model, whose hardware characteristics directly influence energy consumption and inference latency. Therefore, the reported energy savings and efficiency gains should be interpreted within the context of the evaluated device and experimental setup. Future work includes extending the experimental evaluation to a larger and more heterogeneous group of participants and assessing the proposed approach across different smartphone models and hardware generations. In addition, early-exit strategies can be further explored by incorporating multiple exit points in deeper architectures, such as ResNet or Transformer-based models, as well as in hybrid approaches that combine CNNs with LSTMs or other temporal networks. These directions may enable finer control of inference depth and contribute to further improvements in energy efficiency.

References

[1] H. Junker, P. Lukowicz and G. Troster. “Sampling frequency, signal resolution and the accuracy of wearable context recognition systems”. In *Eighth International Symposium on Wearable Computers*, pp. 176–177, 2004.

[2] N. Rashid, B. U. Demirel and M. A. Al Faruque. “AHAR: Adaptive CNN for energy-efficient human activity recognition in low-power edge devices”. *IEEE Internet of Things Journal*, vol. 9, no. 15, pp. 13041–13051, 2022.

[3] M. Straczkiewicz, P. James and J.-P. Onnela. “A systematic review of smartphone-based human activity recognition methods for health research”. *NPJ Digital Medicine*, vol. 4, no. 1, pp. 148, 2021.

[4] M. A. Hossen and P. E. Abas. “Machine Learning for Human Activity Recognition: State-of-the-Art Techniques and Emerging Trends”. *Journal of Imaging*, vol. 11, no. 3, 2025.

[5] R. Mutegeki and D. S. Han. “A CNN-LSTM approach to human activity recognition”. In *2020 international conference on artificial intelligence in information and communication (ICAIIC)*, pp. 362–366. IEEE, 2020.

[6] D. Khan, M. Alonazi, M. Abdelhaq, N. Al Mudawi, A. Algarni, A. Jalal and H. Liu. “Robust human locomotion and localization activity recognition over multisensory”. *Frontiers in Physiology*, vol. 15, pp. 1344887, 2024.

[7] R. Singh and S. S. Gill. “Edge AI: a survey”. *Internet of Things and Cyber-Physical Systems*, vol. 3, pp. 71–92, 2023.

[8] R. G. Pacheco, R. S. Couto and O. Simeone. "On the impact of deep neural network calibration on adaptive edge offloading for image classification". *Journal of Network and Computer Applications*, vol. 217, pp. 103679, 2023.

[9] S. Teerapittayanon, B. McDanel and H. T. Kung. "BranchyNet: Fast inference via early exiting from deep neural networks". In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 2464–2469. IEEE, 2016.

[10] L. Zheng, D. Wu, X. Ruan, S. Weng, A. Peng, B. Tang, H. Lu, H. Shi and H. Zheng. "A Novel Energy-Efficient Approach for Human Activity Recognition". *Sensors*, vol. 17, no. 9, 2017.

[11] J. Lee and J. Kim. "Energy-Efficient Real-Time Human Activity Recognition on Smart Mobile Devices". *Mobile Information Systems*, vol. 2016, pp. 1–12, July 2016.

[12] J. Shi, D. Zuo and Z. Zhang. "An Energy-Efficient Human Activity Recognition System Based on Smartphones". In *2020 7th International Conference on Soft Computing Machine Intelligence (ISCMI)*, pp. 177–181, 2020.

[13] C. Y. Jeong and M. Kim. "An Energy-Efficient Method for Human Activity Recognition with Segment-Level Change Detection and Deep Learning". *Sensors*, vol. 19, no. 17, 2019.

[14] N. Rashid and M. A. Al Faruque. "Energy-efficient Real-time Myocardial Infarction Detection on Wearable Devices". In *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pp. 4648–4651, 2020.

[15] D. N. Jha, Z. Chen, S. Liu, M. Wu, J. Zhang, G. Morgan, R. Ranjan and X. Li. "A Hybrid Accuracy- and Energy-Aware Human Activity Recognition Model in IoT Environment". *IEEE Transactions on Sustainable Computing*, vol. 8, no. 1, pp. 1–14, 2023.

[16] P. Tokas, V. B. Semwal and S. Verma. "A Real-Time Deployable Attention-Driven CNN–LSTM Framework for Human Activity Recognition using Wearable Sensor". *IEEE Sensors Journal*, pp. 1–1, 2025.

[17] Y. LeCun, Y. Bengio and G. E. Hinton. "Deep learning". *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[18] C. M. Bishop and H. Bishop. *Deep Learning: Foundations and Concepts*. Springer Nature, 2023.

[19] J. Yun, Y. Cho, S. M. Lee, J. Hwang, J. Lee, Y.-M. Oh, S.-D. Lee, L.-C. Loh, C.-K. Ong, J. B. Seo and N. Kim. "Deep radiomics-based survival prediction in patients with chronic obstructive pulmonary disease". *Scientific Reports*, vol. 11, pp. 15144, July 2021.

[20] R. G. Pacheco, F. D. V. R. Oliveira and R. S. Couto. "Early exit deep neural networks for distorted images on edge environments". *ITU Journal on Future and Evolving Technologies*, vol. 5, no. 3, pp. 344–355, 2024. © International Telecommunication Union, CC BY-NC-ND 3.0 IGO.

[21] E. Lattanzi, C. Contoli and V. Freschi. "Do we need early exit networks in human activity recognition?" *Engineering Applications of Artificial Intelligence*, vol. 121, pp. 106035, 2023.

[22] S. Scardapane, M. Scarpiniti, E. Baccarelli and A. Uncini. "Why Should We Add Early Exits to Neural Networks?" *Cognitive Computation*, vol. 12, September 2020.

[23] Y. Jon. "Adaptive Sampling in Wireless Sensor Networks for Air Monitoring System". Master's thesis, Uppsala University, Department of Information Technology, Uppsala, Sweden, 2016. Master's thesis.

[24] J. Chen, Y. Sun and S. Sun. "Improving Human Activity Recognition Performance by Data Fusion and Feature Engineering". *Sensors*, vol. 21, no. 3, 2021.

[25] E. Cueto-Mendoza and J. D. Kelleher. "A Framework for Measuring the Training Efficiency of a Neural Architecture". *Artificial Intelligence Review*, vol. 57, no. 12, pp. 349, 2024.

[26] M. Kim, X. Jiang, K. Lauter, E. Ismayilzada and S. Shams. "Secure human action recognition by encrypted neural network inference". *Nature Communications*, vol. 13, no. 1, pp. 4799, 2022.

[27] G. Rodrigues and R. Rabelo. "Human Activity Recognition on Embedded Devices: An Edge AI Approach". pp. 973–979, January 2025.