Assessment and Optimization of 1D CNN Model for Human Activity Recognition

Rafael Schild Reusch, Leonardo Rezende Juracy, Fernando Gehm Moraes
PUCRS – School of Technology, Porto Alegre, Brazil
{rafael.reusch, leonardo.juracy}@edu.pucrs.br, fernando.moraes@pucrs.br

2022 XII Brazilian Symposium on Computing Systems Engineering (SBESC) | 978-1-6654-7425-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/SBESC56799.2022.9964520

***Abstract*—Artificial Intelligence (AI) solves complex tasks like human activity and speech recognition. Accuracy-driven AI models introduced new challenges related to their applicability in resource-scarce systems. In Human Activity Recognition (HAR), state-of-the-art presents proposals using complex multi-layer LSTM networks. The literature states that LSTM networks are suitable for treating temporal-series data, a key feature for HAR. Most works in the literature seek the best possible accuracy, with few evaluating the overall computational cost to run the inference phase. In HAR, low-power IoT devices such as wearable sensors are widely used as data-gathering devices, but little effort is made to deploy AI technology in these devices. Most studies suggest an approach using edge devices or cloud computing architectures, where the end-device task is to gather and send data to the edge/cloud device. Most voice assistants, such as Amazon's Alexa and Google, use this architecture. In real-life applications, mainly in the healthcare industry, relying only on edge/cloud devices is not acceptable since these devices are not always available or reachable. The objective of this work is to evaluate the accuracy of convolutional networks with a simpler architecture, using 1D convolution, for HAR. The motivation for using networks with simpler network architectures is the possibility of embedding them in power- and memory-constrained devices.**

***Index Terms*—Artificial Intelligence, Machine Learning, Human Activity Recognition, 1D Convolutional CNNs**

I. Introduction

With the ever-increasing elder population, falls were accepted as an illness for older adults in the International Classification of Disease-9 (ICD-9) and ICD-10 [1]. Monteiro et al., in their research, found that more than 30% of deaths along people older than 60 years old are due to a hard fall. Much is done to prevent emergencies in elderly homes, from hold bars in bathrooms to 24/7 nurse care. In a real-world environment, it's impossible to prevent all emergencies, thus the need to detect these situations in real-time and act accordingly.

Human Activity Recognition (HAR) is the process in which data is analyzed and processed to determine the person's activities, such as walking, running, sitting, and showering. Historically, sensor data of human activity was scarce, and retrieval was expensive and complex. Currently, where smartphones and wearable devices are affordable and filled with different sensors such as accelerometers and gyroscopes, data can be easily obtained, and recorded [2].

The growth and high availability of mobile phones and IoT sensors with complex sensor arrays led to many proposals related to the HAR subject. Therefore, embedded devices became the target platform for HAR. Data gathering for HAR can be done in various ways with wearable sensors, such as the use of smartwatches and smart bracelets. With the high availability of smartphones, even in the elderly population, recent studies used these devices to record and classify human activity [1].

Recent studies proposed neural network models to be applied in HAR tasks, including complex architectures that presented low real-life applicability due to the computational costs demanded. This is an issue in various AI tasks. More complex tasks such as speech recognition for virtual assistants are mainly cloud-based [3], where the only task of local devices is to work as input data of the neural network located off-site in a dense server dedicated to AI.

HAR is essentially a time series classification problem in which data should be processed in time frames, resembling signal processing problems. The classification involves predicting the movement of a person based on raw sensor data.

HAR can also be treated as a pattern recognition problem using machine learning (ML) approaches. Examples of ML approaches for HAR include decision trees, support vector machines (SVM), and Markov models. Recent developments in Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNN) lead to a high-accuracy activity recognition [4]. The complexity and overhead (processing time and memory footprint) of RNNs, in particular LSTMs (Long Short-Term Memory) implementations, lead to hybrid approaches, mixing CNNs with LSTMs [5]. The goal is to combine the CNN feature extraction properties with the time series processing ability of LSTM.

Recent advances in AI adopt the CNN-only approach to various problems, seeking overhead reduction during the inference phase while maintaining a state-of-the-art accuracy. CNNs have shown promising results in applications such as image processing in battery-powered devices [6]. In a sensor-based HAR scenario, battery-powered devices are the only option, and only a few studies evaluate how a low-power CNN-only approach would perform in this scenario.

The *goal* of this work is to evaluate and optimize a 1D Convolutional Neural Network for its use in resource-scarce embedded systems. To meet this objective, this work evaluates and extends a 1D CNN for HAR [7].

This paper is organized as follows. Section II presents the state-of-the-art of AI in HAR tasks. Section III details the reference 1D CNN model. Section IV details the optimizations carried out in the 1D CNN model with the goal of improv-

TABLE I
RELATED WORKS ON ML APPLIED TO HAR.

Work	Model	Modeling	Goals
LSTM Networks Using Smartphone Data for Sensor-Based Human Activity Recognition in Smart Homes [2]	LSTM	TensorFlow	Compare different LSTM approaches
Human activity recognition from inertial sensor time-series using batch normalized deep LSTM recurrent networks [8]	LSTM	Keras Python	LSTM accuracy for temporal correlation analysis
Convolutional Neural Networks for Human Activity Recognition using Mobile Sensors [1]	CNN	not defined by the Authors	Novel CNN approach to achieve state-of-the-art accuracy in HAR problem
Pre-Impact Fall Detection with CNN-Based Class Activation Mapping Method [9]	CNN	PyTorch	Pre-impact fall detection, combining CNNs and threshold-based methods to reduce the computational cost to run on the wearable device
All binarized convolutional neural network and its implementation on an FPGA [10]	Binarized CNN	Pytorch	State-of-the-art accuracy in a FPGA using low-power and low-area techniques
Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition [11]	CNN + LSTM	Lasagne	Improve accuracy of LSTM approach using CNN as Feature Extractor
Towards effective detection of elderly falls with CNN-LSTM neural networks [12]	CNN + LSTM	not defined by the Authors	Accurate fall detection using wearable sensors in elderly population

ing accuracy without increasing the CNN model complexity. Section VI concludes this paper and point-out directions for future work.

II. RELATED WORK

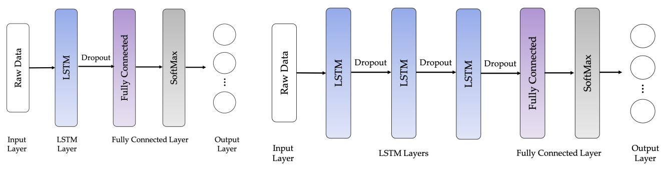
Table I presents related work, where each row color corresponds to an ML method: LSTM in red, CNN in green and hybrid approaches in blue. Section II-A and Section II-B present LSTM and CNN approaches, respectively. Hybrid approaches are described in Section II-C. Section II-D presents how this work fills the gaps observed in the literature.

A. LSTM Approaches

Standard Recurrent Neural Networks (RNNs) suffer from short-term memory due to a vanishing gradient problem that emerges when working with longer data sequences [8]. More advanced versions of RNNs, such as Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU), can preserve information from earlier sequence parts and carry it forward. LSTM contains “memory cells”, allowing to make predictions based on prior information [2].

Mekruksavanich et al. [2] compare LSTM architectures in activities of daily living recognition: (i) Vanilla LSTM; (ii) 2-stacked LSTM; (ii) 3-stacked LSTM. The original LSTM model (Vanilla LSTM) consists of an individual LSTM layer followed by a classifier with a Dropout Layer, a Fully Connected Layer, and a SoftMax layer. The stacked LSTM architectures contain the same classifier layers but include more LSTM layers, taking advantage of the temporal feature extraction of each layer. Figure 1 shows these architectures. Mekruksavanich et al. [2] also proposes a 4-layer CNN-LSTM, presenting the best performance of all evaluated networks with the UCI-HAR dataset. The model achieved 99.39% accuracy, improving 2.24% from the baseline LSTM.

Zebin et al. [8] propose a stacked LSTM approach, with 2 LSTM layers added before the classifier. Differently from [2], this architecture uses Batch Normalization inside the classifier



(a) Vanilla LSTM
(b) 3-Stacked LSTM

Fig. 1. Vanilla and 3-Stacked LSTM Network architectures [2].

to reduce training epochs needed to increase the accuracy, as demonstrated by [2]. This work uses a waist-worn sensor with two sensors: (i) accelerometer; (ii) gyroscope. Each sensor contains three axes. Sensors data are recorded at a 50 Hz sampling frequency. The raw data from the dataset was reshaped to allow a 3-D structure, as required for the LSTM Layer 1 input shape. In this work, 128 1D samples of each sensor axis are grouped. This work achieved a 92% average recognition accuracy for 6 daily-life activities. The Authors highlighted the reduction in training epochs and added robustness due to the Batch Normalization and Dropout layers.

B. CNN Approaches

The feature extraction is a key component of CNN approaches to the HAR problem. Zeng et al. [1] propose a traditional CNN approach using one Convolutional Layer, a Max-Pooling layer, and two fully connected layers (hidden layers). Figure 2 shows the proposed architecture. The Authors use a sliding window of 64 samples with a certain percentage of overlap to extract input data for the CNN. Different than other works, the Authors use only one sensor: an accelerometer with three axes. Input data is shaped into a 2-dimensional array with 64 samples of each of the three axes. This work showed the superiority of the CNN approach against other traditional methods. The Authors highlighted that their results are experimental, and more experiments with larger datasets, such as MobiAct [13] need to be made to study the robustness of the proposed architecture.

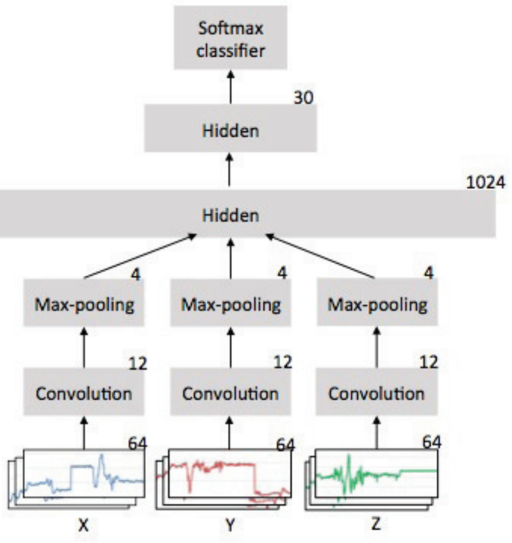


Fig. 2. CNN Network architecture proposed by Zent et al. [1].

Shi et al. [9] propose a similar architecture, with three 1D convolutional layers with 32 output channels, 64 and 128 channels, respectively. The classifier contains a pooling layer, a fully connected layer, and a SoftMax layer. The Author's goal is to detect preimpact movement using the MobiAct dataset, with a sensor worn on the waist, with a sampling rate of 200 Hz. In the training and validation phases, Shi et al. [9] used data of falls and activities of daily living (ADLs), such as walking, jumping, and jogging. This work does not use all MobiAct classes because there are activities unrelated to falls, such as elevator riding. This work achieved an accuracy of 95% using a combination of CNN and Class Activation Mapping. Unlike other papers that used private datasets, this study used the MobiAct dataset.

Shimoda et al. [10] present a binarized CNN which treats binarized values (+1/-1) for the weights and the activation value. Only the first convolutional layer calculates in integer precision since the input values are 8-bit RGB pixels. The proposed method enables a binarized CNN to use bitwise operation in all layers and shares a binarized convolutional circuit among all convolutional layers. The Authors argue that the area is smaller and 1.2 times faster than the typical CNNs, with accuracy reaching 82.8%.

C. Hybrid CNN+LSTM Approaches

Implementations that combine CNN and LSTM are common in many areas, such as video classification and fall detection [12, 14]. This hybrid approach combines the feature extraction of CNNs, with the temporal analysis of LSTMs.

Ordóñez et al. [11] propose a new Deep Neural Network framework designed for activity recognition using wearable sensors. This architecture named DeepConvLSTM combines convolutional and recurrent layers. Similar to other implementations, this work uses the convolutional layer as the automated feature extractor, and the recurrent layer analyses the temporal dynamics of the feature maps created by the convolutional layer. The authors highlighted that the proposed framework improved the accuracy by 4% on average compared to a baseline CNN approach. The paper did not discuss power and memory overheads when comparing the hybrid approach to the standalone CNN approach.

García et al. [12] use this approach to detect elderly falls. Their model includes two CNN layers and one LSTM layer, presented in Figure 3. An accelerometer sensor worn in the waist area produces data. Raw data is processed using a sliding window of 1.28 seconds with 50% overlap to extract time frames from time-series raw sensor data. Using the UCI-FALL [15] dataset, the Authors concluded that combining CNN and LSTM improves accuracy compared to standalone models, achieving 95% in the test dataset. The Authors highlighted that accuracy improvement could be obtained in the raw data processing phase.

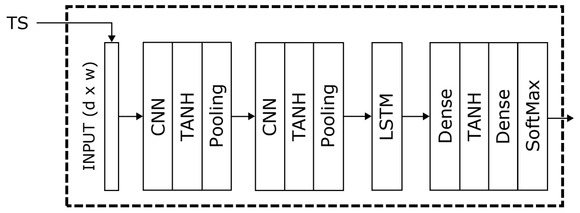


Fig. 3. CNN+LSTM network proposed by García et al. [12].

D. Final Remarks

This Section presented works related to ML frameworks and models, almost all using IoT sensors to capture data. We observe a lack of studies using IoT devices as the target device for ML inference. Optimization techniques were proposed to minimize the accuracy loss without power and memory footprint analyses, key parameters for embedded systems. Mekruksavanich et al. [2] mention using smartphones as a data capture source, but no effort is made to use the smartphone as the target for the inference phase.

Many of the proposed models [5, 8, 11, 12] consist of complex LSTM stacked structures, which require the use of resources that are scarce in embedded systems, such as volatile memory area. From one side, these works achieved state-of-the-art accuracy for many datasets. On the other side, they do not apply to real-time, battery-powered devices, requiring an edge or cloud device to process this information. In practice, edge or cloud devices are unavailable everywhere and every time, creating the risk of not detecting an emergency, such as a hard fall event.

Due to the limitations of embedded systems in processing and energy, it is necessary to study networks with simpler architectures, such as 1D convolutions. The next Section presents the 1D reference architecture, which uses three convolutional layers, organized as vectors and not matrices.

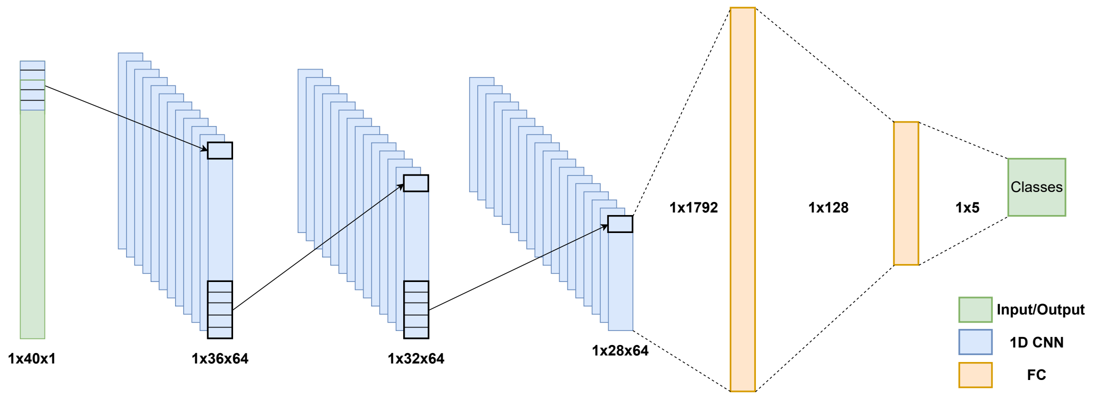


Fig. 4. 1D CNN Reference Model.

III. Baseline 1D CNN Model

Paszke et al. [16] introduced the Pytorch framework. It is an open-source Python library that performs tensor computation with GPU acceleration capability. Despite Pytorch being a Python framework, most of this library is written in C++ to improve performance. It is compatible with most commonly used GPU technologies such as CUDA and recently received support for Apple Silicon GPU. One of the key features of the Pytorch framework is the ability to execute dataflow on GPU asynchronously.

Figure 4 presents the baseline 1D CNN model [7]. It contains three convolutional layers and two fully connected layers. It is a relatively simple architecture by today's standards. Its goal is not to achieve state-of-the-art accuracy but to demonstrate the effectiveness of simple CNN models in HAR tasks.

```
1      def __init__(self, input_size,
       ↪  num_classes):
2          super().__init__()
3
4          # Extract features, 1D conv layers
5          self.features = nn.Sequential(
6              nn.Conv1d(input_size, 64, 5),
7              nn.ReLU(),
8              nn.Dropout(),
9              nn.Conv1d(64, 64, 5),
10             nn.ReLU(),
11             nn.Dropout(),
12             nn.Conv1d(64, 64, 5),
13             nn.ReLU(),
14             )
15         # Classify output, fully connected
           ↪  layers
16         self.classifier = nn.Sequential(
17             nn.Dropout(),
18             nn.Linear(1792, 128),
19             nn.ReLU(),
20             nn.Dropout(),
21             nn.Linear(128, num_classes),
22         )
```

Fig. 5. Pytorch network modeling.

Figure 5 presents the reference CNN modeled using the Pytorch framework. Lines 6 to 13 present the Feature Extractor. Each line corresponds to a specific layer inside this network. Some layers, such as ReLu and Dropout, are hidden in Figure 4. *Conv1d* function specifies a 1D convolutional layer. This function accepts many parameters. In this implementation, the used parameters are: (i) input size; (ii) number of filters; (iii) kernel size. Line 17 to 21 model the classifier. Lines 18 and 21 specify the Fully Connected layers, called "Linear" by Pytorch. The parameters are: (i) input size; (ii) output size.

The reference model adopts ReLu as the activation function. The dropout layer reduces the probability of overfitting by adding noise to the ReLu output. This last layer is useful in models that use a low volume of sensors on each training iteration [7].

This CNN uses a public dataset, similar to the MobiAct dataset, containing 20,000 sensor readings from 6 people, each performing five different actions. Each sensor reading has three axes (roll, pitch, yaw) of 4 sensors (accelerometer, gyroscope, pose, magnetometer). A total of 4 sensor positions are included (belt, arm, dumbbell, forearm). The accelerometer sensor also outputs a virtual axis designed to specify the total acceleration measured in a given time. Magnetometers are often used to normalize the raw data of other sensors. This CNN model does not include it as an input feature, using the accelerometer, gyroscope, and pose inputs. The division between training and evaluation data is as follows: 80% for training and 20% for evaluation.

Unlike other implementations, the 1D CNN presented in this reference model can only process single time step, resulting in a unidimensional, 1x40 input. Figure 6 shows the input shape of the reference model.

Although processing a single time step do not allow precise temporal analysis, accelerometer and gyroscope sensors contain temporal characteristics embedded in raw data, thus allowing the model to detect human activities with 75% accuracy.

In this model, the training parameters are set in a class named *params* and later inserted into the training function. The

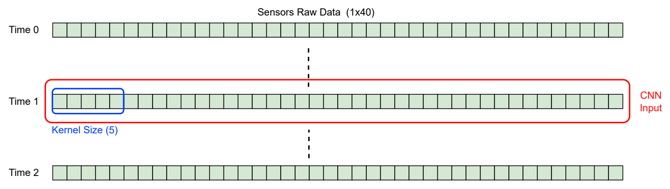


Fig. 6. Reference CNN input shape.

method *pytorch.train* is responsible for starting the training phase. It receives the model defined in Figure 5, training parameters, and training and evaluation data from the dataset.

IV. Optimizations of the Reference Model

A simple 1D CNN model has benefits such as reduced processing power requirements during training and evaluation and reduced memory usage. This CNN does not provide complex optimization methods, such as pruning and quantization, neither it is optimized to achieve state-of-the-art accuracy. This model achieves 75% accuracy in randomly selected activities from the dataset.

Our optimization process aims to improve accuracy without substantially increasing the CNN complexity. The following subsections present the modifications done to the model and evaluated in the sequel.

A. Classifier Changes

The Fully Connected (FC) layer is present before the Flatten layer. The FC has an important effect on the accuracy and computational cost. The first layer of the FC nodes is a function of the number of output nodes of the previous layer. While changing the number of output nodes is possible, increasing it may improve the accuracy with the cost of increasing the number of weights. Initially, the number of output nodes was changed from 128 to 256 and later to 512.

B. Feature Extraction Changes

The reference model has 3 1D Convolutional layers. Ordóñez et al. [11] proposed a hybrid approach that included four CNN layers, taking advantage of its feature extraction capabilities. We modified the reference model using this knowledge to include a fourth Convolutional layer. This modification results in a smaller output feature map at the end of the feature extraction phase, requiring adaptation on the FC layer to be compatible with the new feature map input.

C. Kernel size

The reference model adopts a kernel size equal to 1x5 (Figure 6). With the stride defined at its default value (1), the convolution calculation generates 64 filters with this kernel. Kernel size choice is commonly chosen during the training phase. We evaluate the impact on the accuracy using 1x3, 1x5, and 1x7 kernels.

D. Extended 1D Model

The major disadvantage of this 1D CNN model is the lack of temporal analysis. Human activities are highly related to previous and future movements. Even though temporal characteristics are embedded in sensor data, it is insufficient to detect complex human activities accurately. Recent approaches [1, 5, 12] process data in time frames, extracted from the raw sensors data with a sliding-window method. Each work uses a different window width and overlap percentage. These approaches increase accuracy by reducing the effect of unintended data (noise or random human motions) on the readings.

To add temporal awareness to the reference model while keeping its reduced complexity, the convolutional layer was modified to allow multiple time steps to be included simultaneously in a modified input vector. Most HAR datasets use a time-per-row topology. Thus, a single row in the dataset contains the sensor data of this single time step, together with the activity label. A Python data repackager algorithm was written to convert the dataset into a timeframe-per-row topology. The data repackager joins a parameterizable number of time steps in a single row, adding a temporal correlation to the CNN without modifying its topology. This process is done for training and inference data at runtime, loaded to the GPU via PyTorch's data loader. Figure 7 presents how the 120-width array is created using three different time steps.

Using a single time step as input (40-with array), the resulting Feature Map size is 1x28 , combined with the number of Filters (64), the Flatten Layer transforms these multiple arrays into a single 1x1792 vector (Figure 4). In the Extended 1D Model, the newly formed 120-width timeframe requires the input shape to be changed to accommodate the new array size. One of the benefits of using a 1D Convolution is that the input array can be increased without increasing the number of weights, reducing memory area, but that is not true for the Classifier. Using the proposed model, the resulting feature map size is 1x108, containing roughly 3.8 times more information than the reference model, consequently improving accuracy. To accommodate the larger Feature Map, the new Fully Connected (FC) input size is 1x6912. Despite the increased number of weights in the FC layer, it requires fewer weights than other 1D CNN evaluated optimizations while achieving higher accuracy.

V. Results

The training and evaluation phases were done using PyTorch using a CUDA-enabled GPU. This model is cross-compatible with the new Apple Silicon (arm64) architecture. The design phase was done in an ARM-Powered MacBook Pro, and the training and evaluation phase in a PC equipped with 16GB RAM, i7 7700k, and a GTX 1080.

Table II presents results using the optimizations made in the reference model, evaluating the number of parameters, the model size (i.e., memory footprint), and accuracy. The 2nd row presents results related to the baseline 1D model.

The 3rd and 4th rows evaluate the impact of increasing the fully connected (FC) layer. The accuracy increases by increas-

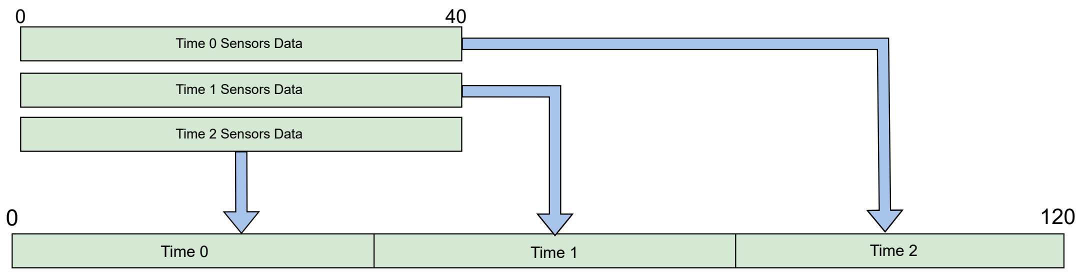


Fig. 7. Input data for the Extended 1D Model.

TABLE II
REFERENCE AND OPTIMIZED MODELS ACCURACY. REFERENCES [17] AND [18] ARE STATE-OF-THE-ART REFERENCES.

Model	**Number of Params**	**Model Size (MB)**	**Accuracy**
Baseline 1D model	271,621	1.18	**74%**
FC - 256 output nodes	501,765	2.06	76%
FC - 512 output nodes	962,053	3.82	79%
Kernel size = 7	238,981	1.04	72%
Kernel size = 3	304,261	1.32	76%
Added Convolutional Layer	259,397	1.17	70%
Extended 1D Model - Accurate	926,981	4.03	**88%**
Extended 1D Model - Reduced	484,229	2.34	**85%**
T. S. Jordan CNN [17]	738,000	-	78%
ResNet-18 [18]	11,000,000	-	94.8%

ing the number of output nodes of the first fully connected layer at the cost of raising the number of weights in this layer. The number of weights in the fully connected layer doubled and quadrupled using 256 and 512 output nodes, respectively. The quadrupled memory area is hardly justified by just a 5% accuracy increase in a memory-constrained device.

The 5th and 6th rows evaluate the impact of the kernel size. Increasing the kernel size results in a smaller output feature map, consequently with less information and reduced accuracy. A smaller kernel produces larger feature maps and allows a better feature extraction phase, resulting in an accuracy increase at the cost of a larger output feature map and increased computational cost. More study needs to be made to decide if the memory overhead increase justifies small increments in performance.

Increasing the convolutions while not changing the kernel size could result in a smaller feature map to be used in the classifier, negatively affecting the accuracy. It is not clear the reason explaining the accuracy reduction to 70%. One reason may be the unbalanced ratio between the feature extraction and the classificator. It is necessary to further investigate the model to take advantage of this extra convolutional layer.

The 8th and 9th rows present the accuracy of the proposed extended 1D models. This result corroborates how relevant the time-series analysis is to HAR. The accuracy improved 14% when adding two extra time steps in the input shape. Even though the number of weights increased due to the change on the FC layer, it is roughly 5% smaller than the FC with 512 output nodes approach, with 9% higher accuracy.

The FC is responsible for more than 95% of the total number of weights in the model. We evaluated a reduced version of the proposed model by minimizing the number of weights in the FC layer. This version consists of changes in the classifier, focusing on reducing the number of parameters and model size while achieving similar accuracy. Compared to the "accurate" version, the number of parameters and model size was reduced by roughly 50%, with only a 3% accuracy reduction. Both 1D extended models increased the accuracy at the cost of a larger number of parameters and model size compared to the baseline model.

For comparison purposes, the network proposed in [17], contains 5 2D layers (64x64x128, 32x32x128, 16x16x128, 8x8x64, 4x4x64) and 2 FC (256 and 6 classes). This network achieves an overall accuracy of 78% in all activities, achieving 94% in specific test sets, using 738,000 parameters. State-of-the-art CNNs, as ResNet-18 [18], achieves consistently 94.79% with 11M parameters.

VI. CONCLUSION AND FUTURE WORK

This work reviewed works using machine learning for HAR. The proposals available in the literature achieve high accuracy ($> 90\%$) at the cost of complex networks (2D CNNs and LSTMs). In the context of embedded systems, computing power and memory footprint are scarce resources. As a result, we evaluated a network with a simpler architecture, 1D CNN. As expected, this CNN presented a low accuracy (74%). Our main contribution was a design space exploration using the 1D network to increase its accuracy without making the project more complex. The result was the "extended 1D" network through the inclusion of temporal analysis to the 1D model. A precision between 85% and 88% was reached by doubling the size of the original model. This work paves the way to embed ML on the end device, as a wearable, without using edge or cloud devices.

The main direction of future work is the CNN optimization for energy- and memory-constrained devices. This phase involves studying and applying optimization techniques, such as quantization and pruning, and evaluating the impact on power and memory area usage. Next, to codify the CNN models using 32-bit integers in C language to run inference and assess accuracy. It is necessary to adopt the C language since the CNN model is to be used in embedded processors with energy and memory constraints. We consider adopting abstract (such as OVP) or RTL models to evaluate performance, power, and memory footprint.

The final result expected of this research project is a framework starting from, e.g., Pytorch, and according to the power and memory footprint constraints, generate C descriptions for the CNNs targeting embedded processors.

Acknowledgment

This work was financed in part by CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico), grant 309605/2020-2; and CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior), Finance Code 001.

References

[1] M. Zeng, L. T. Nguyen, B. Yu, O. J. Mengshoel, J. Zhu, P. Wu, and J. Zhang, "Convolutional Neural Networks for Human Activity Recognition Using Mobile Sensors," in *Mobile Computing, Applications and Services (MobiCASE)*, 2014, pp. 197–205. [Online]. Available: https://doi.org/10.4108/icst.mobicase.2014.257786

[2] S. Mekruksavanich and A. Jitpattanakul, "LSTM Networks Using Smartphone Data for Sensor-Based Human Activity Recognition in Smart Homes," *Sensors*, vol. 21, no. 5, pp. 1–25, 2021. [Online]. Available: https://doi.org/10.3390/s21051636

[3] J. Janak, T. Tseng, A. Isaacs, and H. Schulzrinne, "An Analysis of Amazon Echo's Network Behavior," in *IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 1–6. [Online]. Available: https://doi.org/10.1109/GLOBECOM46510.2021.9685138

[4] A. Hayat, M. Dias, B. P. Bhuyan, and R. Tomar, "Human Activity Recognition for Elderly People Using Machine and Deep Learning Approaches," *Information*, vol. 13, no. 6, pp. 1–12, 2022. [Online]. Available: https://doi.org/10.3390/info13060275

[5] J. Xu, Z. He, and Y. Zhang, "CNN-LSTM Combined Network for IoT Enabled Fall Detection Applications," *Journal of Physics: Conference Series*, vol. 1267, pp. 1–6, 2019. [Online]. Available: https://doi.org/10.1088/1742-6596/1267/1/012044

[6] C. Zhang, Z. Tang, T. Guo, J. Lei, J. Xiao, A. Wang, S. Bai, and M. Zhang, "SaleNet: A low-power end-to-end CNN accelerator for sustained attention level evaluation using EEG," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2022, pp. 1–5.

[7] J. Chiang, "Simple 1D CNN approach to human-activity-recognition (HAR) in PyTorch," 2022. [Online]. Available: https://github.com/jchiang2/Human-Activity-Recognition/

[8] T. Zebin, M. Sperrin, N. Peek, and A. J. Casson, "Human activity recognition from inertial sensor time-series using batch normalized deep LSTM recurrent networks," in *IEEE Engineering in Medicine and Biology Society (EMBC)*, 2018, pp. 1–4. [Online]. Available: https://doi.org/10.1109/EMBC.2018.8513115

[9] J. Shi, D. Chen, and M. Wang, "Pre-Impact Fall Detection with CNN-Based Class Activation Mapping Method," *Sensors*, vol. 20, no. 17, 2020. [Online]. Available: https://doi.org/10.3390/s20174750

[10] M. Shimoda, S. Sato, and H. Nakahara, "All binarized convolutional neural network and its implementation on an FPGA," in *Field Programmable Technology (ICFPT)*, 2017, pp. 291–294. [Online]. Available: https://doi.org/10.1109/FPT.2017.8280163

[11] F. J. Ordóñez and D. Roggen, "Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition," *Sensors*, vol. 16, no. 1, pp. 1–25, 2016. [Online]. Available: https://doi.org/10.3390/s16010115

[12] E. García, M. Villar, M. Fáñez, J. R. Villar, E. de la Cal, and S.-B. Cho, "Towards effective detection of elderly falls with CNN-LSTM neural networks," *Neurocomputing*, vol. 500, pp. 231–240, 2022. [Online]. Available: https://doi.org/10.1016/j.neucom.2021.06.102

[13] G. Vavoulas, C. Chatzaki, T. Malliotakis, M. Pediaditis, and M. Tsiknakis, "The MobiAct Dataset: Recognition of Activities of Daily Living using Smartphones," in *Information and Communication Technologies for Ageing Well and e-Health, ICT4AgeingWell*, (2016, pp. 143–151. [Online]. Available: https://doi.org/10.5220/0005792401430151

[14] J. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 4694–4702. [Online]. Available: https://doi.org/10.1109/CVPR.2015.7299101

[15] D. Podareanu, V. Codreanu, S. Aigner, C. Leeuwen, and V. Weinberg, "Best Practice Guide - Deep Learning," 2019. [Online]. Available: https://doi.org/10.13140/RG.2.2.31564.05769

[16] A. Paszke *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," *CoRR*, vol. abs/1912.01703, pp. 1–12, 2019. [Online]. Available: http://arxiv.org/abs/1912.01703

[17] T. S. Jordan, "Using convolutional neural networks for human activity classification on micro-Doppler radar spectrograms," in *Sensors, and Command, Control, Communications, and Intelligence (C3I)*, 2016, pp. 1–9. [Online]. Available: https://doi.org/10.1117/12.2227947

[18] H. Du, Y. He, and T. Jin, "Transfer Learning for Human Activities Classification Using Micro-Doppler Spectrograms," in *IEEE International Conference on Computational Electromagnetics (ICCEM)*, 2018, pp. 1–3. [Online]. Available: https://doi.org/10.1109/COMPEM.2018.8496654