

An automatic network structure search via channel pruning for accelerating human activity inference on mobile devices

Junjie Liang, Lei Zhang^{*}, Can Bu, Dongzhou Cheng, Hao Wu, Aiguo Song

School of Electrical and Automation Engineering, Nanjing Normal University, Nanjing, 210023, China

School of Information science and Engineering, Yunnan University, Kunming, 650000, China

School of Instrument Science and Engineering, Southeast University, Nanjing, 210023, China

ARTICLE INFO

Keywords:

Automatic structure search
Convolutional neural networks
Channel pruning
Mobile devices
Human activity recognition
Wearable sensors

ABSTRACT

During recent years, deep convolutional neural networks (CNNs) have become increasingly popular in sensor-based human activity recognition (HAR), due to powerful feature extraction capacity. However, the required high computational cost has hindered their practical deployment in real-world applications. To address this issue, this paper presents a novel approach named ABCSearch to prune channels through an artificial bee colony (ABC) algorithm, which is a meta-heuristic optimization algorithm inspired by the foraging behavior of honeybees. Unlike traditional pruning strategies that prioritize channel importance, the proposed method seeks to identify the optimal number of channels at each layer, i.e., the optimal pruning structure. To lessen human interference, we transform the search problem of the optimal pruning structure into a simple optimization problem and solve it automatically through the ABC algorithm, which can significantly decrease both model size and computational cost without compromising performance. Specifically, we validate the effectiveness of ABCSearch for a standard five-layer CNN backbone architecture, which achieves 97.05%, 98.45%, 91.71%, 78.89% classification accuracies on four HAR benchmarks, i.e., UCI-HAR, WISDM, PAMAP2, UniMiB-SHAR. Our approach can yield 87.44%, 89.41%, 89.06%, 89.59% FLOPs reductions, and 89.54%, 88.70%, 88.38%, 90.45% less memory consumptions, as well as 4.4×, 5.0×, 5.8×, 6.1× practical speedups on real hardware devices, while maintaining comparable accuracy.

1. Introduction

The last ten years have seen a significant advancement in internet of things and sensor technologies. Due to their advantages of high precision, small size, and low cost, sensors are widely used in portable devices such as mobile phones or smart watches. The use of sensors for human activity recognition (HAR) (Andrade-Ambriz et al., 2022; Panja et al., 2023; Sekaran et al., 2023) has emerged as a new research trend. Nowadays, sensor-based HAR is widely used in healthcare, exercise tracking, gaming console technology, and other fields (Bulling et al., 2014; Lima et al., 2021; Zhang et al., 2021). Essentially, HAR can be seen as a common pattern recognition problem. However, traditional pattern recognition algorithms like naive Bayes, random forests, and SVM have been limited to manually designed shallow features that rely heavily on domain-specific knowledge and artificial experience. This greatly hampers the classification performance and model generalization of HAR. To tackle the aforementioned challenges, many deep learning methods equipped with automated feature extraction

capabilities have been developed. In particular, deep convolutional neural networks (CNNs) (Khan et al., 2020; Krizhevsky et al., 2017; Lin et al., 2021) have become a mainstream technique in various activity recognition tasks, which has been proven to be more effective than traditional machine learning methods in inferring activity from sensor data.

However, despite the stronger feature extraction capability of deep CNNs, their larger size requires more memory and computational resources, making it difficult to deploy HAR on mobile or wearable devices. For real-time activity recognition systems, low-latency prediction is crucial, and the high computational cost has to be addressed. To achieve real-time and reliable activity identification on mobile or wearable devices, lighter and faster CNN models are necessary. The commonly used model compression techniques include quantification (Gholami et al., 2021; Yang et al., 2019), distillation (Gou et al., 2021; Park et al., 2019), and network pruning (Liu et al., 2017; Molchanov et al., 2019), among which channel pruning has been recognized as one of the most effective solutions. The objective of channel

^{*} Corresponding author at: School of Electrical and Automation Engineering, Nanjing Normal University, Nanjing, 210023, China.

E-mail addresses: JunjieLiang85@gmail.com (J. Liang), leizhang@njnu.edu.cn (L. Zhang), bucanya123@gmail.com (C. Bu), 211843002@njnu.edu.cn (D. Cheng), haowu@ynu.edu.cn (H. Wu), a.g.song@seu.edu.cn (A. Song).

<https://doi.org/10.1016/j.eswa.2023.122180>

Received 30 June 2023; Received in revised form 12 October 2023; Accepted 13 October 2023

Available online 16 October 2023

0957-4174/© 2023 Elsevier Ltd. All rights reserved.

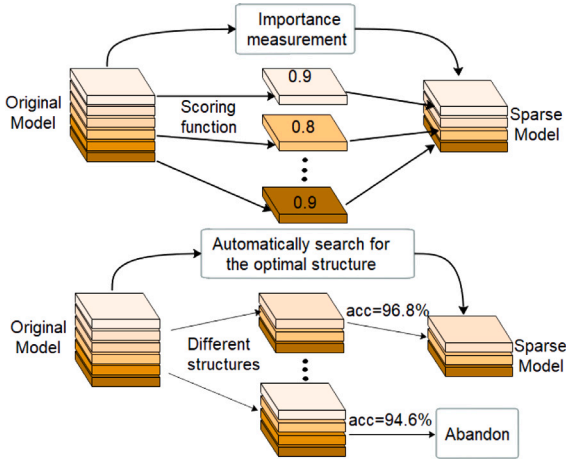


Fig. 1. A main difference between the previous pruning solution (above) and the pruning solution in this article (below).

pruning is to eliminate entire channels in each layer. However, this is a challenging task as removing a channel in one layer may significantly alter the input in the subsequent layer. As shown in Fig. 1, most existing pruning methods preserve important channels based on importance metric, which are suboptimal because they potentially ignore impact of the model's structure on performance. Liu et al. (2018) demonstrates that the key factor to channel pruning is to identify the optimal pruning structure, which refers to the number of channels per-layer, rather than selecting significant channels. However, how to search for the optimal pruning structure still remains a challenging task. In deep CNNs, there are countless combinations of pruning structures. Given a CNN model with an L -layer convolution, the combination of pruning structures can be formulated as $\prod_{j=1}^L C_j$, where C_j represents the number of preserved channels in the j th layer. In a deep CNN, the combination overhead is extremely large and intractable, making it impossible for HAR to be deployed on resource-limited devices.

In this paper, we present a channel pruning method that utilizes the Artificial Bee Colony (ABC) algorithm (Bansal et al., 2013; Karaboga & Akay, 2009; Liu et al., 2023; Yang et al., 2023). Our method not only finds the optimal pruning structure but also solves the problem of dealing with a huge combination of pruning structures in deep networks. Our ABCSearch employs two strategies. To begin with, we restrict the number of channels preserved in each layer to αC_j , where α is a value between 0.1 and 1, indicating the proportion of channels preserved in every convolutional layer. This interpretable operation reduces the number of channel combinations, enabling an efficient search for optimal pruning structures. Secondly, we employ the ABC algorithm to automatically derive the optimal pruning structure, which mitigates time-consuming or laborious human interference by deciding the number of channels. In fact, raw time series sensor data needs to be first measured and collected through wearable inertial sensors attached to different body positions, and then segmented into fixed-size windows as input to the subsequent deep model. In some cases, this deep model can take advantage of a spectrogram representation to implement an efficient activity inference. As we have known, a spectrogram of such inertial sensor signal can be represented as a function of time and frequency. To be specific, the spectrogram is equivalent to the magnitude squared of the short-time Fourier transform (STFT). In previous literature, Jiang and Yin (2015), Ravi et al. (2016) and Ravi et al. (2017) have highlighted the importance of a suitable signal domain while applying a deep learning model to time-series sensor data. In particular, they have shown that the spectrogram representation is suitable to extract discriminative activity features that emphasize the intensity differences between nearest sensor data

points. The spectrogram representation is also able to provide a form of sampling rate invariance, which enables deep model to be more robust against potential changes in sampling frequency. In addition, proper frequency selection in such signal domain can also provide an implicit way of noise filtering for time series sensor data. On this basis, Fig. 2 presents a schematic workflow of the proposed method, which consists of the following main steps: (1) Initializing a structure set that contains multiple structures, with each structure having elements representing the number of channels preserved in each convolutional layer (also indicated by corresponding numbers); (2) Randomly selecting filter weights from a pre-trained full model and allocating them to each structure, measuring their fitness (i.e., accuracy) through training for a certain number of epochs; (3) Introducing the ABCSearch algorithm to update the structure set and recalculating the fitness of each structure through above step (2); (4) Iteratively updating the structure set to automatically search for the pruning structure with the highest fitness and fine-tuning this structure as the final prediction model; (5) After obtaining the optimal pruning structure, one can evaluate the model's predictive performance on each action category in the test set. We validate the effectiveness of the proposed ABCSearch on four publicly available benchmark HAR datasets (UCI-HAR, WISDM, PAMAP2, UniMiB-SHAR) and demonstrate its practical speedup effect on resource-limited embedded platforms. The main contributions of this article are as follows:

1. We introduce an ABCSearch that automatically searches for the optimal pruning structure to reduce manual intervention and use it for efficient and rapid activity inference of sensor data.
2. Our experiments suggest that the key factor to model pruning is to determine the optimal number of channels for each convolutional layer, which focuses on determining the optimal structure of the model rather than just selecting important channels.
3. Given a certain α , the optimal number of channels per-layer can be automatically searched using the ABC algorithm, thereby reducing computational complexity. Numerous experiments conducted on multiple benchmark HAR datasets have shown that our ABCSearch can effectively reduce memory consumption and accelerate activity inference for different α values.

Below, in Section 2, we first review previous studies and outline the main differences between them and the introduced ABCSearch. Section 3 gives an overview of the structural design of ABCSearch, and introduces the implementation details of our method. Section 4 introduces the main experimental results of four popular HAR benchmark datasets and a series of ablation studies. Finally, Section 5 summarizes and discusses this paper.

2. Related work

Deep CNNs have made significant progress in recent years, which have replaced traditional manual feature extraction. As a result, they are now widely used in HAR area (Bulling et al., 2014; Rashidi & Cook, 2009; Zeng et al., 2018). Ronao and Cho (2016), Yang et al. (2015), and Zeng et al. (2018) were the first to apply CNN models that can automatically extract features for activity recognition. Although deeper networks can achieve better performance, the explosive growth of parameters and computational complexity still hinders practical deployment of HAR on mobile or wearable devices. For example, the traditional VGGNet-16 model (Simonyan & Zisserman, 2015), which is composed of 13 layers of convolutional layers and 3 layers of fully connected layers, has a parameter volume of about 138M and memory consumption of about 526M. It is difficult to be deployed for sports watches or mobile bracelets in real-world situations. Therefore, it is essential to solve the problem of high computing costs caused by deep models for fast and reliable activity recognition on wearable devices. In recent years, there has been a lot of literatures on model compression (Buciluă et al., 2006; Li et al., 2021; Lin et al., 2021), but

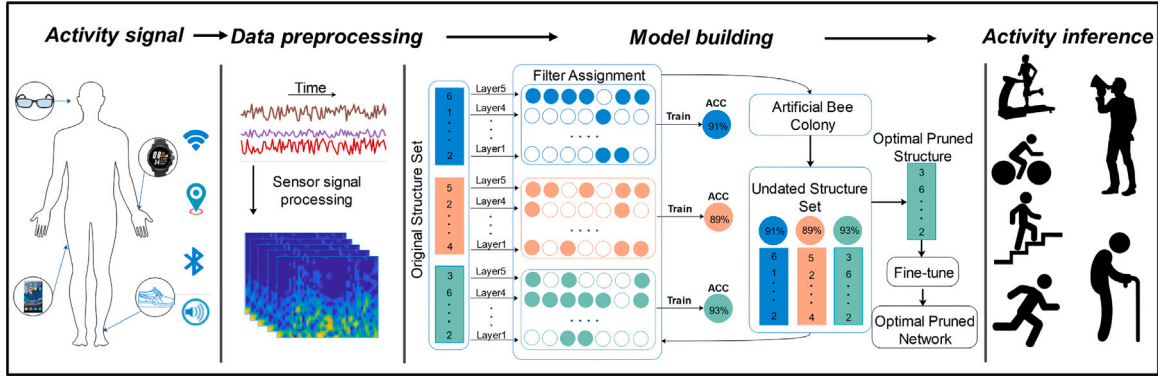


Fig. 2. A schematic workflow of the proposed ABCSearch algorithm for activity recognition.

few have focused on HAR based on CNN acceleration. Among various model compression methods such as tensor decomposition, knowledge distillation, quantization, network pruning, etc., network pruning has been one of the most widely employed compression methods.

Network pruning can be classified into two categories: structured pruning (also known as channel pruning) (Li et al., 2021; Lin et al., 2021; Liu et al., 2017) and unstructured pruning (also known as weight pruning) (Aghasi et al., 2017; Frankle & Carbin, 2019; Han et al., 2015). Unstructured pruning involves removing unimportant neurons in different layers, causing most of the CNN weights to be zero. This reduces memory consumption and computing cost through the sparse array model, but requires special hardware and software for actual acceleration. Because channel pruning can directly remove redundant filters, it is more applicable for various hardware devices without requiring special software for acceleration. In practical applications of HAR, we prefer channel pruning.

Channel pruning has been widely used due to its hardware compatibility. Many previous studies aim to select important channels based on filter importance, such as L_1 -norm (Liu et al., 2017) and activation output sparsity (He et al., 2017; Luo et al., 2017). In Li et al. (2021), channel pruning has been formulated as an optimization problem, by combining filter information loss to reduce channels. However, this method requires iterative optimization and fine-tuning, which would be challenging to implement end-to-end. These methods typically rely on human interference for hyper-parameter tuning, hindering practical application of HAR. Therefore, automatic pruning has gained a lot of attention. However, most existing automatic pruning methods are implemented layer-by-layer or from bottom to up. For instance, He, Lin et al. (2018) employs reinforcement learning, and He, Kang et al. (2018) uses automatic feedback loop. Our ABCSearch differs from previous automatic pruning approaches. Instead of selecting important channels or pruning layer-by-layer, this paper focuses on the correlation between different layers and automatically search for the optimal structure from the full network by determining the number of preserved channels in each layer.

3. Propose method

The traditional pruning method selects important channels according to the importance metric, which is usually sub-optimal (He, Lin et al., 2018). On the contrary, what makes our ABCSearch apart from these methods is that the ABC algorithm can automatically determine the optimal model structure, e.g., the number of channels in each layer.

3.1. Optimal pruned structure

Inspired by the study of Lin et al. (2021), Liu et al. (2018) and Yang et al. (2023), our method aims to find the optimal pruning

structure for activity recognition. For a given full model M with L -layer convolution, the number of channels for each convolutional layer is $C = \{C_1, C_2, \dots, C_L\}$, where C_j represents the number of preserved channels in layer j while pruning model M . Our goal is to search for the best channel combination of C using the test set S_{test} and training set S_{train} . We then fine-tune the pruning model M on S_{train} to achieve the highest accuracy. Therefore, we formulate the channel pruning problem as follows:

$$C^* = \arg \max_C \text{accuracy} (M(C, W; S_{train}); S_{test}), \quad (1)$$

where W is the weights of the fine-tuned model M on S_{train} , and accuracy ($\sim$) represents the S_{test} precision of the construct C . As depicted in Fig. 2, we update the pruning weights directly on the pruned model M . However, given the very large combination of possible structures in C , it naturally causes high computational cost in channel pruning. Therefore, optimizing Eq. (1) is difficult. In Section 3.2, we further shrink the combination to address this issue.

3.2. Shrink the combination

Given a full model M , the combinations of pruned structures can be expressed as $\prod_{j=1}^L C_j$, which is computationally expensive for resource-limited wearable devices. Therefore, we further propose to optimize Eq. (1) as follows:

$$C^* = \arg \max_C \text{accuracy} (M(C, W; S_{train}); S_{test}), \quad (2)$$

$$s.t. \quad C_j \in \{0.1C_j, 0.2C_j, \dots, \alpha C_j\}^L$$

where $\alpha = \{0.1, 0.2, \dots, 1\}$ is a pre-defined hyperparameter that represents the maximum number of channels preserved in j -layer network. We have two motivations to introduce α : First, during the pruning process, one network has to preserve a certain number of channels, and α can be used as the upper limit of the channels preserved in each layer. Second, introducing α significantly reduces the combination of trimmed structures. Although the introduction of α reduces the computational costs, it is still suboptimal and requires human intervention. To address this problem, we introduce an automated structure search algorithm in Section 3.3.

3.3. Automatic structure search

We recommend an automatic search for the optimal structures, instead of manual structure design or enumerating all possible structures (i.e., the number of channels in each layer). Specifically, we initialize a set of n pruned structures $\{C_j\}_{j=1}^n$ from the i -element C_{ji} of C_j randomly sampled in $\{0.1C_j, 0.2C_j, \dots, \alpha C_j\}$. Correspondingly, we can obtain a set of pruned networks $\{M_j\}_{j=1}^n$ and a set of pruned weights $\{W_j\}_{j=1}^n$, where each pruned structure C_j serves as a solution

Algorithm 1: Artificial bee colony algorithm.

Input: Max proportion: α , Cycles: T , Number of pruned structures: n , Max times: m , Counter: $\{t_i\}_{i=1}^n$.

Output: The model M optimal structure C^* .

```

1 Set  $s = 1, j = 1$ , initialize the set of structures:  $\{C_j\}_{j=1}^n$ .
2 while  $s < T$  do
3   while  $j \leq n$  do
4     Produce potential structures  $G_j$  via Eq. (3);
5     Evaluate the fitness of  $C_j$  and  $G_j$  via Eq. (4);
6     if  $fit(C_j) \geq fit(G_j)$  then
7        $t_j = t_j + 1$ ;
8     else
9        $C_j = G_j$ ;
10       $t_j = 0$ ;
11    end
12  end
13  while  $j \leq n$  do
14    Compute the  $P_j$  via Eq. (5);
15    Produce a random number  $\mu_j \in [0, 1]$ ;
16    if  $\mu_j \leq P_j$  then
17      Produce potential  $G_j$  via Eq. (3);
18      Assess the performance of  $G_j$  via Eq. (4)
19      if  $fit(C_j) \geq fit(G_j)$  then
20         $t_j = t_j + 1$ ;
21      else
22         $C_j = G_j$ ;
23         $t_j = 0$ ;
24      end
25    end
26  end
27  while  $j \leq n$  do
28    if  $t_j > m$  then
29      Re-initialize  $C_j$ ;
30    end
31  end
32   $s = s + 1$ ;
33 end
34  $C^* = \arg \max_{C_j} \text{Acc}(M_j(C_j, W_j; S_{\text{train}}); S_{\text{test}})$ .

```

for the optimization problem. Our objective is to progressively update the structure set and ultimately determine the most optimal pruning structure. To achieve this goal, we demonstrate that by incorporating the ABC algorithm (Lin et al., 2021; Liu et al., 2023; Yang et al., 2023), the optimal structure search can be automated to find the pruning structure. We first outline our ABC algorithm in Alg. 1. The ABC algorithm will be described in detail below.

Employed Bee (line 3 - line 12): The employed bees can generate a new candidate structure G_j for each candidate structure C_j , and the i th element of G_j can be defined as:

$$g_{ji} = [C_{ji} + r(C_{ji} - C_{gi})], \quad (3)$$

where r is the random number between $[-1, 1]$, and $g \neq j$ represents the g th pruned structure. $[\sim]$ returns the value closest to $\{0.1C_j, 0.2C_j, \dots, \alpha C_j\}$.

The employed bee will then decide whether to replace C_j based on the performance of the generated candidate G_j , and this process is implemented by the following formula:

$$fit(G_j) = \text{accuracy}(M_j(G_j, W_j; S_{\text{train}}); S_{\text{test}}). \quad (4)$$

If G_j performs better than C_j , one can replace C_j , otherwise remains C_j unchanged.

Observing Bee (line 13–line 26): The observing bees continue to update $\{C_j\}_{j=1}^n$ according to Eqs. (3) and (4). Differently, the selection probability of the trimmed structure C_j is related to its performance, which can be described as:

$$P_j = 0.9 \times \frac{fit(C_j)}{\max(fit(C_j))} + 0.1. \quad (5)$$

Therefore, the better the performance of C_j , the higher the probability of selecting C_j , which is then updated to a better pruned structure. In this way, the best pruning structure can be automatically and gradually generated through the ABC algorithm.

Scout Bee (line 27–line 31): If the pruning structure C_j is not updated more than m times, the scout bee reinitializes it to further generate a new prune structure.

4. Experiments

In this section, we evaluate the performance of ABCSearch on a variety of different HAR benchmark datasets. To indicate the advantages of the proposed method, we compare it with the traditional importance-based pruning method. In addition, we also perform a series of ablation studies to further verify the effectiveness of this method. Finally, we demonstrate the effect of our ABCSearch in real-world by measuring the actual activity inference time on an embedded Raspberry Pi 4 B+ platform.

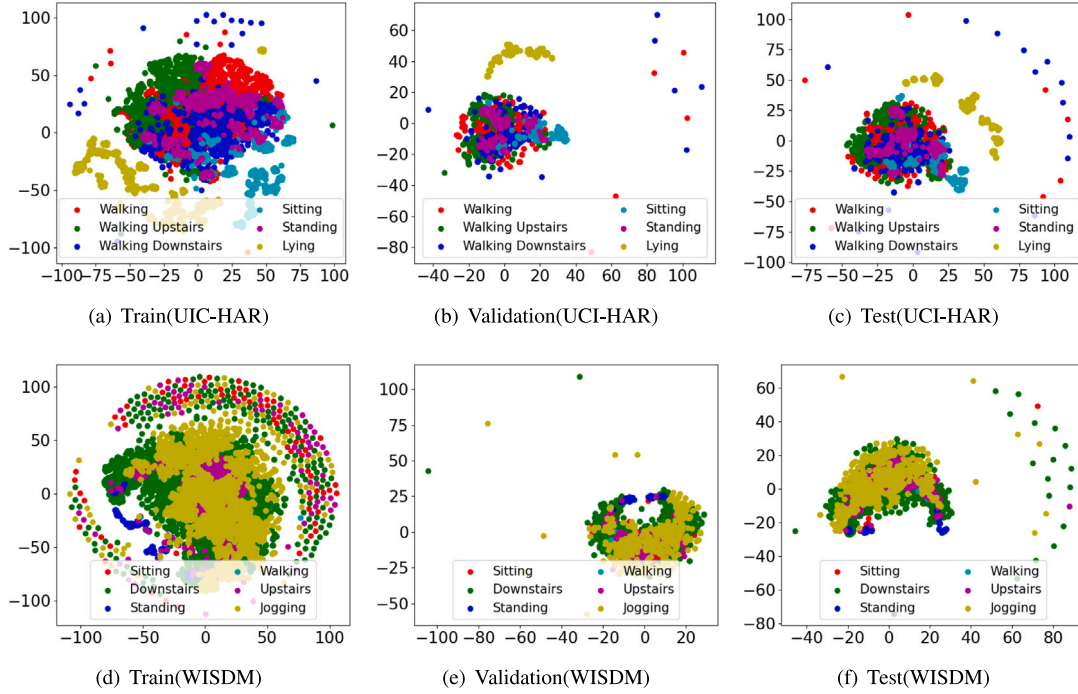
4.1. Benchmark datasets

It is well known that benchmark datasets always play a vital role in estimating model performance and ensuring fair comparisons with other approaches. With recent rapid development of HAR, there has been a wide range of benchmark datasets in common use. Without loss of generality, we comprehensively evaluate the performance of the proposed approach on the four popular HAR benchmark datasets including UCI-HAR, WISDM, PAMAP2, and UniMiB-SHAR, which are publicly available and can be freely download from internet. The benchmark datasets have been chosen according to the frequency of their wide usage in the HAR research community. During the past two decades, the selected four datasets have been widely used as benchmark and cited by hundreds and thousands of previously published papers (also see recent survey papers Chen et al., 2021; Dang et al., 2020; Nweke et al., 2018; Qiu et al., 2022). The data is collected through multiple sensor types that are attached to different body locations under various conditions and environments. Table 1 displays the details of the data preprocessing process, including window size and sampling rate, etc. In fact, the preprocessing steps to segment time series sensor data are very tedious. Due to implementation simplicity, the sliding window has been the most widely applied in HAR area, where the continuous time series sensor data may be divided into fixed-size windows. So far, there have been two kinds of sliding windows: overlapping and non-overlapping windows. For the former case, an overlap can be allowed between two adjacent windows for preserving the continuity of sensor signals. Such overlapping window is more beneficial for UCI-HAR, WISDM, and PAMAP2 including continuous or periodic activities (e.g., “Walking” and “Running”), which often last a long time period. Intuitively, a certain overlap rate can yield the higher accuracy, but take longer to segment the sensor data. Previous works (Bao & Intille, 2004; Jobanputra et al., 2019) have witnessed the success of such overlapping window in UCI-HAR and PAMAP2 by an 50% overlap rate. In particular, to better maintain the continuity of sensor signals on WISDM dataset, a higher 95% overlap rate is allowed, since its designed window is relatively large (i.e., 200 that corresponds to 10 s) (Kwapisz et al., 2011). For the latter case, i.e., UniMib-SHAR, we adopt this non-overlapping sliding window rather than overlapping one because the dataset is mainly focused on instant motion-related recognition of “falls” that take place in a short time period, where each

Table 1

Data preprocessing details (A = accelerometer, G = gyroscope, M = magnetometer).

Dataset	Attribute					
	Activity classes	Windows size	Overlap rate	Sampling rates	Sensor types	Number of subjects
UCI-HAR	6	128	50%	50 Hz	A, G	30
WISDM	6	200	95%	20 Hz	A	29
PAMAP2	12	171	50%	100 Hz	A, G, M	9
UniMiB-SHAR	17	151	–	50 Hz	A	30

**Fig. 3.** Visualization of raw sensor data from different datasets.

window is only centered around its peak (Micucci et al., 2017). Overall, following previous literatures (Anguita et al., 2012; Kwapisz et al., 2011; Micucci et al., 2017; Reiss & Stricker, 2012), we still utilize the same window setting to ensure fair comparisons. Recently, t-Distributed Stochastic Neighbor Embedding (t-SNE) has been a well-established dimensionality-reduction algorithm, which is widely employed to observe data distribution in a two or three-dimensional space (Van der Maaten & Hinton, 2008). Before moving forward to activity classification stage, we first take a close look at raw sensor data distribution by using the t-SNE to visualize training, validation and test activity samples in the UCI-HAR and WISDM datasets, as illustrated in Fig. 3. Here each point represents an activity sample, and different colors represent different activity classes. Given that the samples of different classes are apart away from each other while the same class samples are close together, the situation can be seen as an ideal case for activity classification. However, as can be seen from the figure, that is obviously not the case for the two datasets. Different activity classes are highly confused, which leads to a challenging task for activity recognition.

UCI-HAR dataset: The dataset was collected by the research team at the University of California Irvine from 30 volunteers aged from 19 to 48 years old. Each volunteer had a Samsung Galaxy S2 at the waist and was required to perform six daily activities, including “lying down”, “standing up”, “walking”, “going downstairs” and “going upstairs”. The sensor signals were sampled by a gyroscope and a triaxial accelerometer at 50 Hz. We partitioned the entire dataset into three subsets in a random manner: 70% for training, 10% for validation, and 20% for testing purposes (Anguita et al., 2012).

WISDM dataset: The dataset was collected by the WISDM research team at Fordham University from 29 volunteers. Each volunteer wore an Android phone with a three-axis accelerometer in his/her pocket and was instructed to perform six daily activities, including “upstairs”, “downstairs”, “jogging”, “sitting”, “walking” and “standing”. The accelerometer collected 20 samples per second. The dataset was split into three parts, with 80% being used as the training set, 10% being used as the validation set, and the remaining 10% being used as the test set (Kwapisz et al., 2011).

PAMAP2 dataset: The dataset was collected by the German AI Research Center from 18 kinds of sports activities performed by 9 volunteers. This includes 12 basic activities (“running”, “standing”, “walking”, etc.) and several optional activities (“playing” “football”, “car driving”, etc.). Each volunteer wore a Colibri wireless inertial measurement unit, which was placed on their hand, ankle and chest. Data were recorded at 100 Hz during the experiment (Reiss & Stricker, 2012).

UniMiB-SHAR dataset: This dataset was collected by Micucci et al. from the University of Milan. Thirty volunteers between the age of 18 and 60 were selected. The dataset contained 11,771 samples that belong to 17 different activity categories. There are eight daily activities, such as “standing”, “sitting” and “walking”, etc., as well as nine fall movements, such as “fall forward”, “fall right”, and “fall left”, etc. Each volunteer wore a Samsung Galaxy Nexus I9250 smartphone in his/he pocket. Sensor signals were sampled by a three-axis BMA220 accelerometer at 50 Hz (Micucci et al., 2017).

4.2. Experiment details

This section introduces the implementational details of the experiment. We follow the previous convolutional architectures described in Ronao and Cho (2016), Yang et al. (2015) and Zeng et al. (2014) to construct a CNN backbone model of five convolutional layers as the baseline. The network structure is described in Fig. 4, which includes the convolutional layer, ReLU activation function, and pooling layer, etc. All the layers are arranged in the following order: C-B-R-C-B-R-P-C-B-R-C-B-R-P-C-B-R-AP-FC (C: Conv, B: Batch Normalization, R: ReLU, P: Pooling, AP: AdaptiveAvgPool, FC: Full Connection Layer). The AdaptiveAvgPool layer is applied after convolutional neural networks and before the fully connected layer to normalize feature maps of different sizes into a uniform size. This can simplify the feature maps fed to the fully connected layer, and reduce the number of model parameters, leading to improved inference speed of the model. For different datasets, we use different convolution kernel sizes. For example, the size of the convolution kernel for the PAMAP2 dataset is 9×1 , and for the WISDM dataset, the kernel size is 7×1 . All the networks are trained by stochastic gradient descent (SGD) with a momentum of 0.9. The baseline model (pre-trained full model) for each dataset is trained for 200 epochs using a small batch size of 64. The initial learning rate is set to 0.001, which is divided by 10 after every 50 epochs. For the pruned models, we fine-tune them using an Adam optimizer with the learning rate set to $1e-5$. Experiments show that we only need to fine-tune 20 epochs to reach or even exceed the performance of the full model, which greatly saves the time required for retraining the new model from scratch. To obtain the desired model, we only need to prune and fine-tune the pre-trained model, which offers significant advantages for practical HAR deployment. We conduct experiments on four different benchmark datasets of HAR (UCI-HAR, WISDM, PAMAP2, UniMiB-SHAR). α is a key hyper-parameter during the experiment, which represents the percentage of the preserved channels in each layer. For example, $\alpha = 0.8$ indicates that a maximum of 80% of the channel number can be maintained in each layer. We compare the performance of the pruned models under different α . Unlike conventional pruning methods, ABCSearch aims to automatically search for the optimal pruning architecture. The experimental results show that ABCSearch is superior to conventional pruning methods. We also compare it with a three-layer convolution model with the same number of parameters. The model is trained from scratch for 200 epochs. The experiments show that ABCSearch can achieve better performance with the same number of parameters.

On the one hand, we compare with a regular convolution model with the same number of parameters, as shown in Table 2. The regular convolution model is trained for 200 epochs, while ABCSearch only fine-tunes for 20 epochs. The experimental results show that ABCSearch not only greatly saves training time but also achieves better performance. For example, on UCI-HAR dataset, our ABCSearch requires only 0.32M parameters to achieve 97.05% accuracy, while the traditional three-layer convolution model achieves a lower 96.08% accuracy. Similarly, on WISDM dataset, ABCSearch achieves a performance improvement of 1.27% (98.45% vs. 97.18%) over the conventional three-layer convolution model, with only 0.27M parameters. With the same 0.36M parameters, the performance improvement of ABCSearch on PAMAP2 dataset is 1.89% (91.71% vs. 89.82%) higher than the conventional three-layer convolution model. On UniMiB-SHAR dataset, with 0.23M parameters, our method achieves a performance improvement of 2.60% (78.89% vs. 76.29%). This shows that our method not only saves a lot of training time but also achieves better performance with the same number of parameters, further proving the advantages and effectiveness of ABCSearch.

On the other hand, we measure the performance of the traditional importance-based pruning method (Liu et al., 2017), further demonstrating the performance advantage of ABCSearch in HAR. As shown in Table 3, our method not only achieves a greater pruning rate,

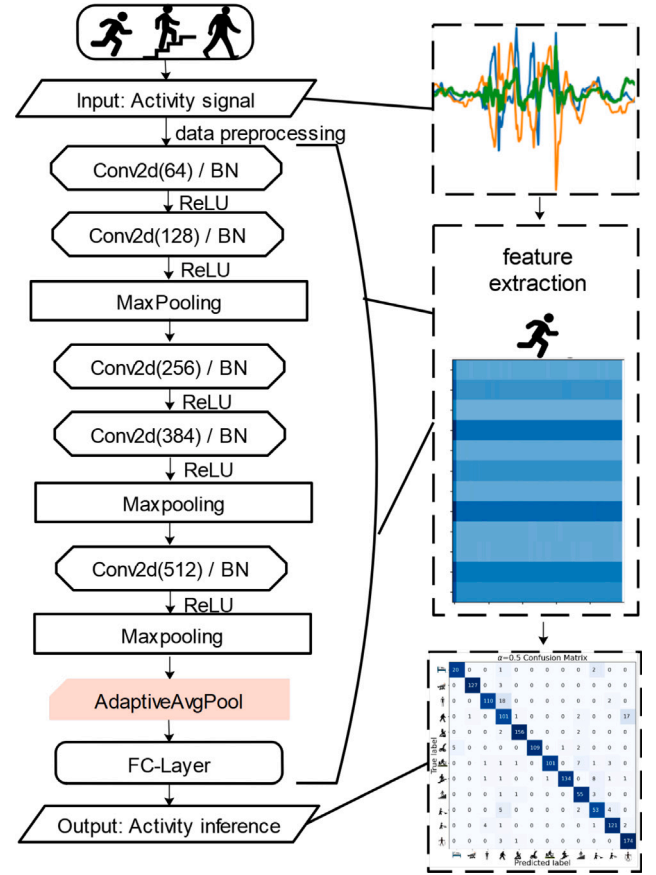


Fig. 4. Brief description for architectures.

and also performs better than the traditional pruning methods. First, our ABCSearch saves significant computing resources (e.g., 107.37M VS. 208.29M on UCI-HAR dataset; 100.37M VS. 183.31M on WISDM dataset; 246.47M VS. 251.41M on PAMAP2 dataset; and 43.98M VS. 85.28M on UniMiB-SHAR dataset). Second, ABCSearch greatly reduces model complexity compared to conventional methods (e.g., 0.44M VS. 0.55M on UCI-HAR dataset; 0.42M VS. 0.58M on WISDM dataset; 0.47M VS. 0.51M on PAMAP2 dataset; and 0.23M VS. 0.47M on UniMiB-SHAR dataset). Finally, our method achieves better performance at greater pruning rates (e.g., 97.25% VS. 96.97% on UCI-HAR dataset; 98.54% VS. 98.38% on WISDM dataset; 92.29% VS. 91.41% on PAMAP2 dataset, and 79.45% VS. 78.23% on UniMiB-SHAR dataset). Overall, our ABCSearch can not only achieve better performance, and also save more computing resources, which further demonstrates that searching for optimal pruning architectures has advantages over traditional importance-based pruning methods.

4.3. Ablation study

4.3.1. AdaptiveAvgPool and AdaptiveMaxPool

We investigate the impact of AdaptiveMaxPooling and AdaptiveAvgPooling on model performance. Ablation studies are conducted on the four HAR benchmark datasets. As shown in Table 4, it can be seen that AdaptiveAvgPooling is consistently superior to AdaptiveMaxPooling, which may be explained as follows. In fact, the output of AdaptiveMaxPooling is the maximum value in the pooling region, which can emphasize the most discriminative activity features and be suitable to focus on local features. Instead, the output of AdaptiveAvgPooling is the average value in the pooling region, which tends to smooth the activity features and be suitable to capture non-local or global

Table 2

Comparisons with a regular CNN at the same number of parameters (Accuracy & F1-score & Parameters).

Method	Dataset			
	UCI-HAR	WISDM	PAMAP2	UniMiB-SHAR
Baseline(5 layers CNN)	97.40%&97.66%&3.06M	98.45%&98.44%&2.39M	92.36%&95.18%&3.10M	79.26%&81.81%&2.41M
ABCSearch($\alpha = 0.5$)	97.05%&97.33%&0.32M	98.45%&98.40%&0.27M	91.71%&94.73%&0.36M	78.89%&81.42%&0.23M
ABCSearch($\alpha = 0.7$)	97.25%&97.53%&0.44M	98.54%&98.49%&0.42M	92.29%&95.02%&0.47M	79.45%&81.92%&0.61M
3 layer CNN	96.08%&96.13%&0.32M	97.18%&97.16%&0.27M	89.82%&93.06%&0.36M	76.29%&80.05%&0.23M
5 layer CNN	96.42%&96.55%&0.32M	97.41%&97.38%&0.27M	90.21%&93.32%&0.36M	76.56%&80.65%&0.23M

Table 3

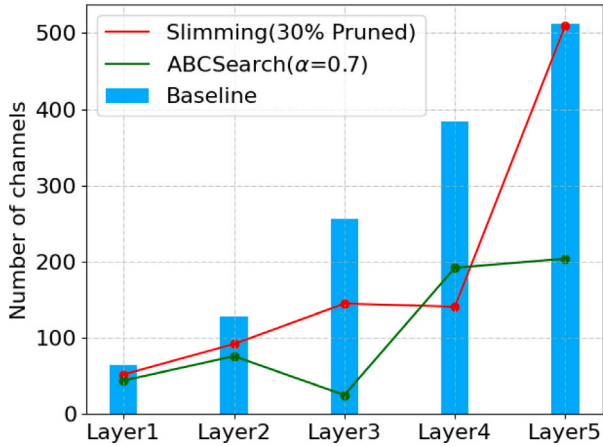
Accuracy & Parameters & FLOPs of models on various datasets.

Method	Dataset			
	UCI-HAR	WISDM	PAMAP2	UniMiB-SHAR
Baseline	97.40%&3.06M&641.8M	98.45%&2.39M&554.13M	92.36%&3.10M&1.08G	79.26%&2.41M&422.86M
ABCSearch(0.5)	97.05%&0.32M&80.56M	98.45%&0.27M&58.65M	91.71%&0.36M&117.23M	78.89%&0.23M&43.98M
ABCSearch(0.7)	97.25%&0.44M&107.37M	98.54%&0.42M&100.37M	92.29%&0.47M&246.47M	79.45%&0.61M&110.65M
Slimming(50%)	96.97%&0.55M&208.29M	98.38%&0.58M&183.31M	91.41%&0.51M&251.41M	78.23%&0.47M&85.28M
Other Method	95.41% (Dong et al., 2021) 96.37% (Ignatov, 2018)	98.27% (Wang et al., 2021) 98.23% (Han et al., 2022)	89.30% (Ma et al., 2019) 89.96% (Zeng et al., 2018)	75.42% (Huang et al., 2021) 75.33% (Wang et al., 2021)

Table 4

Performances comparison between AdaptiveMaxpool and AdaptiveAvgpool.

Dataset	AdaptiveAvgPool	AdaptiveMaxPool	Accuracy
UCI-HAR	✓	×	97.05%
	×	✓	96.74%
WISDM	✓	×	98.45%
	×	✓	97.83%
PAMAP2	✓	×	91.71%
	×	✓	91.33%
UniMiB-SHAR	✓	×	78.89%
	×	✓	78.37%

**Fig. 5.** Number of channels in different layers.

semantics. In such activity recognition tasks, one has to handle the non-local temporal dependencies within human activity datasets. Therefore, to improve the long-range dependencies learning ability of the network, the AdaptiveAvgPooling module (instead of AdaptiveMaxPooling) is finally added to model long-range temporal dependencies in time series sensor data and form the output for activity recognition.

4.3.2. The impact of preserved channel number

We count the number of channels in each layer on PAMAP2 dataset, to further investigate the effect of the channel number on model performance. Our approach aims to search the optimal number of channels preserved in each layer and prune them through the ABC algorithm.

The traditional pruning method (Liu et al., 2017) involves first assessing the importance of each channel and then pruning the unimportant ones. Compared to the traditional pruning methods, we focus more on the overall architecture of the full model, i.e., the number of channels in each convolutional layer. As shown in Fig. 5, our method has different channel numbers in different layers as compared to the conventional methods, which further validates the performance advantage of the ABCSearch in searching for the best pruning architecture.

4.3.3. t-SNE visualization of clustering results

In order to intuitively show the classification performance of the pruned model, we further compare and analyze the t-SNE visualization results of the four datasets for activity classification. As can be seen, Figs. 6(a), 6(b), 6(c) and 6(d) respectively represent the raw data distributions, while Figs. 6(e), 6(f), 6(g) and 6(h) show the clustering effects of the pruned model by ABCSearch for different activity categories. One can observe that there are confused areas, where raw sensor samples of different activity classes highly overlap, which will be often hard to recognize correctly. After the pruned model is applied to handle raw sensor samples, most of the homogeneous activity samples are clustered together, creating a more favorable feature representation for the deep classifier than raw sensor signals. In fact, the visualization results are able to convey in-depth information about the pruned model's success in recognizing individual activity categories. For example, the pruned model can perform very well while predicting activity samples in WISDM, UCI-HAR, and PAMAP2, where there is a clear clustering structure that closely trails the relatively high classification accuracy score. In contrast, one can observe high overlap in most regions of UniMiB-SHAR dataset, which only provides an accuracy of 78.89%. The pruned model finds it difficult to classify those samples that belong to "falls" classes such as "FallingBack", "FallingForward", "FallingRight", and "FallingBack", hence resulting in worse performance. This is because various "falls" activity samples have very similar waveform, indicating the visualization results' interpretability. Overall, the visualization results suggest that the pruned model by ABCSearch can exhibit a clear and compact cluster structure that then feeds back good performance.

4.3.4. The effect of different α on performance

We further analyze the influence of the introduced factor α within the range [0.1, 1], which denotes the percentage of preserved channel at every layer. To analyze, we display the overall accuracy curves for all HAR benchmark datasets in Fig. 7. Intuitively, larger α leads to higher accuracy but less pruning rate, consuming more FLOPs or parameter budgets. The experimental results in Fig. 7 well support the assumption. As can be seen, the proposed ABCSearch is still able to maintain a

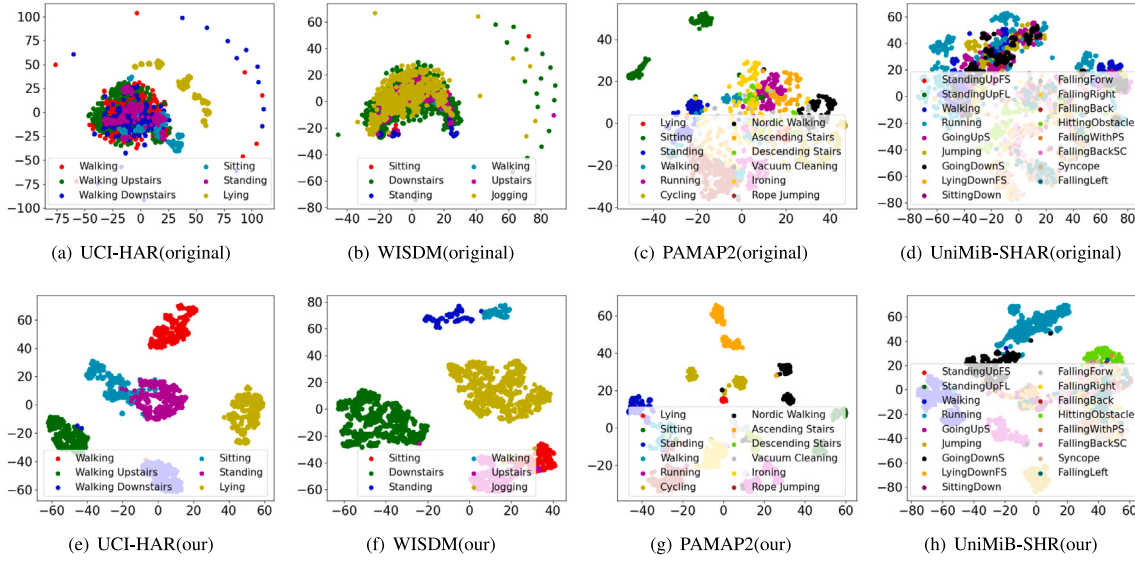
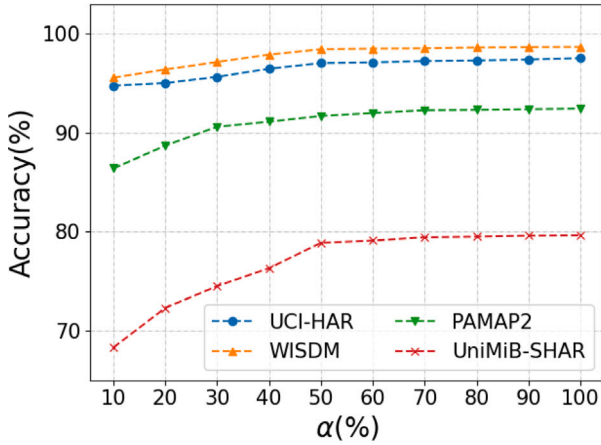


Fig. 6. t-SNE visualization results.

Fig. 7. The effect of different α on performance.

comparable accuracy performance against the full model, when 50% percentage of the channels are preserved at every layer. To achieve a better accuracy-cost trade-off, in this paper, we can uniformly set $\alpha = 50\%$ as illustrated in Table 3.

4.3.5. Confusion matrix

The confusion matrix in Fig. 8 shows the classification performance of PAMAP2 dataset under different values of α . The main diagonal line in the figure indicates the number of correctly predicted samples, and darker colors indicate higher accuracy. Therefore, we can easily identify prediction errors for each activity in PAMAP2 dataset. For example, in Fig. 8(b), there have been 17 samples labeled as “walking”, which are incorrectly predicted as “rope jumping”. From Fig. 8, we observe that increasing the pruning rate (decreasing α) does not result in a significant decline in performance. This suggests that our pruning method can significantly reduce memory consumption and computational load while maintaining similar accuracy, enabling real-time human activity inference and energy savings.

4.3.6. Inference-time analysis

Though FLOPs has been a well-established metric (cost indicator) to measure pruning efficiency, it does not directly translate to inference latency. Practical inference time or latency is another crucial

metric, which directly affect the user experience in HAR applications. In this section, to better reflect the pruning efficiency in real-world environment, we further investigate practical inference time of HAR on a Raspberry Pi 4B+ platform equipped with a quad-core Cortex-A72 (ARM v8) 64-bit processor and up to 4 GB of SDRAM, where the installed Raspberry Pi OS has a good compatibility with current popular PyTorch deep learning library. The original full models are pre-trained on a standard workstation (Intel Xeon Gold 6258R CPU, 128G memory, and NVIDIA RTX3090 GPU). Pruning is performed offline from the original full models after training and before inference, and fine-tuning is employed to recover accuracy. Without loss of generality, we utilize the WISDM dataset as an example to compare the inference times between the full model and the pruning model through loading them to the Raspberry Pi 4B+ platform. Fig. 9 shows the user interface of activity inference, which contains graphical representation and prediction probability of each human action, as well as practical inference time under different α . The main program is written in Python language, which uses torch.save() and torch.load() to save and load models (.pt file), and then calls model.eval() to set batch normalization layers to evaluation mode before implementing activity inference. Fig. 10 summarizes the actual inference time of 200 active samples under different α , where the blue one represents the actual inference time of the full model, and the other colors represent the actual inference time of our pruning method under different α . As we can see from the figure, ABCSearch can greatly lessen the inference time and save energy consumption. We also summarize the inference times of the different α values for one sample randomly selected from UCI-HAR, WISDM, PAMAP2, and UniMiB-SHAR. For example, in the case of WISDM, it can be seen that the full model takes round 201.9 ms to process a sample, while our method only needs 55.5 ms, 40.1 ms, and 26.3 ms when α is set to 0.7, 0.5, and 0.3, respectively. After pruning the full model with $\alpha = 0.7$, our method even can result in a slight increase of 0.09% in accuracy through fine-tuning, while achieving practical 3.6 $\times$ speedup. When $\alpha = 0.5$, our method is still able to realize 5.0 $\times$ speedup, while maintaining almost the same accuracy. If α is further decreased to 0.3, an extremely high acceleration (i.e., 7.7 $\times$) can be attained. Similar case can also be seen in UCI-HAR, PAMAP2, and UniMiB-SHAR. The experiment results indicate that the pruned method is general and can effectively accelerate activity inference without significant loss of accuracy, which does not require any special hardware and libraries (see Table 5).

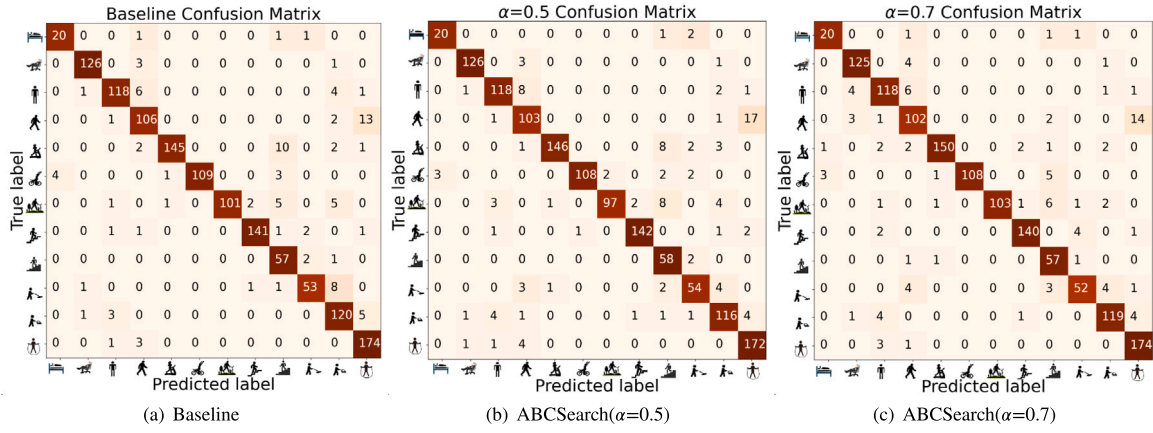


Fig. 8. Confusion matrices.

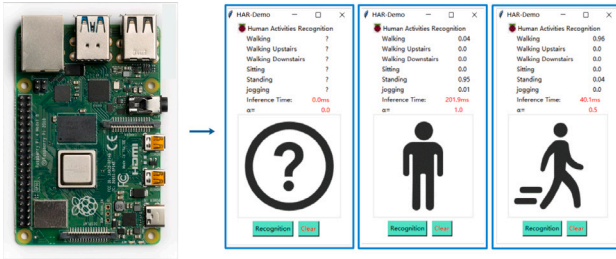


Fig. 9. Real-time HAR system on Raspberry Pi 4 B+.

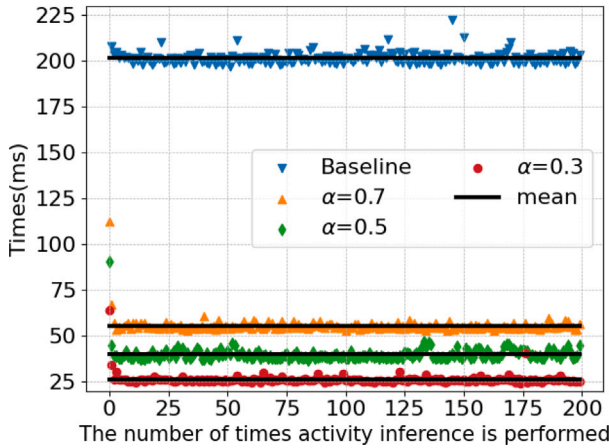


Fig. 10. Inference speed over 200 times.

5. Conclusion

During the past decade, deep CNNs have been widely applied in sensor-based HAR domain because of their automatic feature extraction ability for multivariate time series sensor data. Due to high computational cost and memory footprint, it is often hard to implement deep models over various resource-limited mobile devices, which hinders their practical deployment in a larger of variety of mobile HAR applications. To address this problem, this paper presents an efficient channel pruning method named ABCSearch for activity recognition, where an artificial bee colony (ABC) algorithm is used to automatically find optimal pruned structure (i.e., channel number at every layer), instead of identifying more “important” channels as previous most works have done. However, the potential combinations of these pruned structure

Table 5

Inference time.

Dataset	Method	Inference time (ms)
UCI-HAR (ms/windows)	Baseline	240.2
	ABCSearch($\alpha=0.7$)	73.1
	ABCSearch($\alpha=0.5$)	54.7
	ABCSearch($\alpha=0.3$)	25.3
WISDM (ms/windows)	Baseline	201.9
	ABCSearch($\alpha=0.7$)	55.5
	ABCSearch($\alpha=0.5$)	40.1
	ABCSearch($\alpha=0.3$)	26.3
PAMAP2 (ms/windows)	Baseline	478.5
	ABCSearch($\alpha=0.7$)	176.9
	ABCSearch($\alpha=0.5$)	82.2
	ABCSearch($\alpha=0.3$)	48.8
UniMiB-SHAR (ms/windows)	Baseline	190.4
	ABCSearch($\alpha=0.7$)	75.6
	ABCSearch($\alpha=0.5$)	31.2
	ABCSearch($\alpha=0.3$)	8.7

are extremely large, which makes exhaustively searching them infeasible. Therefore, to avoid the intractably huge combinations, we shrink the combinations of the preserved channels to a small limited space, resulting in an efficient search for optimal pruned structure. On this basis, the search of optimal pruned structure could be transformed into an optimization problem, and then the structure search may be solved through the ABC algorithm in an automatic manner, to lessen human interference. Overall, instead of enumerating all potential combinations (channel number at every layer), our approach can automatically search the optimal pruned structure for activity recognition, which does not rely on any empirical setting or manual design. Extensive experiments on multiple standard HAR benchmarks demonstrate that ABCSearch can efficiently accelerate activity inference, while maintaining high activity recognition performance. Practical speedups in activity inference are evaluated on an embedded mobile device.

CRedit authorship contribution statement

Junjie Liang: Conceptualization, Data curation, Investigation, Methodology. **Lei Zhang:** Writing – original draft, Writing – review & editing. **Can Bu:** Formal analysis, Software. **Dongzhou Cheng:** Formal analysis, Validation. **Hao Wu:** Methodology, Supervision. **Aiguo Song:** Investigation, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

The work was supported in part by the National Natural Science Foundation of China under Grant 62373194 and in part by the Natural Science Foundation of Jiangsu Province, China under grant BK20191371. Lei Zhang is the corresponding author.

References

- Aghasi, A., Abdi, A., Nguyen, N., & Romberg, J. (2017). Net-Trim: convex pruning of deep neural networks with performance guarantee. In *Proceedings of the 31st international conference on neural information processing systems* (pp. 3180–3189).
- Andrade-Ambroz, Y. A., Ledesma, S., Ibarra-Manzano, M.-A., Oros-Flores, M. I., & Almanza-Ojeda, D.-L. (2022). Human activity recognition using temporal convolutional neural network architecture. *Expert Systems with Applications*, 191, Article 116287.
- Anguita, D., Ghio, A., Oneto, L., Parra, X., & Reyes-Ortiz, J. L. (2012). Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In *Ambient assisted living and home care: 4th international workshop, IWAAL 2012, Vitoria-Gasteiz, Spain, December 3-5, 2012. Proceedings 4* (pp. 216–223). Springer.
- Bansal, J. C., Sharma, H., & Jadon, S. S. (2013). Artificial bee colony algorithm: a survey. *International Journal of Advanced Intelligence Paradigms*, 5(1-2), 123–159.
- Bao, L., & Intille, S. S. (2004). Activity recognition from user-annotated acceleration data. In *International conference on pervasive computing* (pp. 1–17). Springer.
- Bucilua, C., Caruana, R., & Niculescu-Mizil, A. (2006). Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 535–541).
- Bulling, A., Blanke, U., & Schiele, B. (2014). A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys*, 46(3), 1–33.
- Chen, K., Zhang, D., Yao, L., Guo, B., Yu, Z., & Liu, Y. (2021). Deep learning for sensor-based human activity recognition: Overview, challenges, and opportunities. *ACM Computing Surveys*, 54(4), 1–40.
- Dang, L. M., Min, K., Wang, H., Piran, M. J., Lee, C. H., & Moon, H. (2020). Sensor-based and vision-based human activity recognition: A comprehensive survey. *Pattern Recognition*, 108, Article 107561.
- Dong, Y., Li, X., Dezert, J., Zhou, R., Zhu, C., Wei, L., & Ge, S. S. (2021). Evidential reasoning with hesitant fuzzy belief structures for human activity recognition. *IEEE Transactions on Fuzzy Systems*, 29(12), 3607–3619.
- Frankle, J., & Carbin, M. J. (2019). The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *ICLR*.
- Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M. W., & Keutzer, K. (2021). A survey of quantization methods for efficient neural network inference. In *Low-power computer vision* (pp. 291–326). Chapman and Hall/CRC.
- Gou, J., Yu, B., Maybank, S. J., & Tao, D. (2021). Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6), 1789–1819.
- Han, S., Pool, J., Tran, J., & Dally, W. J. (2015). Learning both weights and connections for efficient neural networks. In *Proceedings of the 28th international conference on neural information processing systems-Volume 1* (pp. 1135–1143).
- Han, C., Zhang, L., Tang, Y., Huang, W., Min, F., & He, J. (2022). Human activity recognition using wearable sensors by heterogeneous convolutional neural networks. *Expert Systems with Applications*, 198, Article 116764.
- He, Y., Kang, G., Dong, X., Fu, Y., & Yang, Y. (2018). Soft filter pruning for accelerating deep convolutional neural networks. In *Proceedings of the 27th international joint conference on artificial intelligence* (pp. 2234–2240).
- He, Y., Lin, J., Liu, Z., Wang, H., Li, L.-J., & Han, S. (2018). Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 784–800).
- He, Y., Zhang, X., & Sun, J. (2017). Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE international conference on computer vision* (pp. 1389–1397).
- Huang, W., Zhang, L., Gao, W., Min, F., & He, J. (2021). Shallow convolutional neural networks for human activity recognition using wearable sensors. *IEEE Transactions on Instrumentation and Measurement*, 70, 1–11.
- Ignatov, A. (2018). Real-time human activity recognition from accelerometer data using Convolutional Neural Networks. *Applied Soft Computing*, 62, 915–922.
- Jiang, W., & Yin, Z. (2015). Human activity recognition using wearable sensors by deep convolutional neural networks. In *Proceedings of the 23rd ACM international conference on multimedia* (pp. 1307–1310).
- Jobanputra, C., Bavishi, J., & Doshi, N. (2019). Human activity recognition: A survey. *Procedia Computer Science*, 155, 698–703.
- Karaboga, D., & Akay, B. (2009). A comparative study of artificial bee colony algorithm. *Applied Mathematics and Computation*, 214(1), 108–132.
- Khan, A., Sohail, A., Zahoor, U., & Qureshi, A. S. (2020). A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review*, 53(8), 5455–5516.
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6), 84–90.
- Kwapisz, J. R., Weiss, G. M., & Moore, S. A. (2011). Activity recognition using cell phone accelerometers. *ACM SIGKDD Explorations Newsletter*, 12(2), 74–82.
- Li, Y., Lin, S., Liu, J., Ye, Q., Wang, M., Chao, F., Yang, F., Ma, J., Tian, Q., & Ji, R. (2021). Towards compact cnns via collaborative compression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 6438–6447).
- Lima, W. S., Bragança, H. L., & Souto, E. J. (2021). NOHAR-NOvelty discrete data stream for human activity recognition based on smartphones with inertial sensors. *Expert Systems with Applications*, 166, Article 114093.
- Lin, M., Ji, R., Zhang, Y., Zhang, B., Wu, Y., & Tian, Y. (2021). Channel pruning via automatic structure search. In *Proceedings of the twenty-ninth international conference on international joint conferences on artificial intelligence* (pp. 673–679).
- Liu, J., Ang, M. C., Chaw, J. K., Kor, A.-L., & Ng, K. W. (2023). Emotion assessment and application in human-computer interaction interface based on backpropagation neural network and artificial bee colony algorithm. *Expert Systems with Applications*, 232, Article 120857.
- Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., & Zhang, C. (2017). Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international conference on computer vision* (pp. 2736–2744).
- Liu, Z., Sun, M., Zhou, T., Huang, G., & Darrell, T. (2018). Rethinking the value of network pruning. In *International conference on learning representations*.
- Luo, J.-H., Wu, J., & Lin, W. (2017). Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision* (pp. 5058–5066).
- Ma, H., Li, W., Zhang, X., Gao, S., & Lu, S. A. (2019). Multi-level attention mechanism for multimodal human activity recognition. In *Proceedings of the twenty-eighth international joint conference on artificial intelligence, Macao, China* (pp. 10–16).
- Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(86), 2579–2605.
- Micucci, D., Mobilio, M., & Napoletano, P. (2017). Unimib shar: A dataset for human activity recognition using acceleration data from smartphones. *Applied Sciences*, 7(10), 1101.
- Molchanov, P., Mallya, A., Tyree, S., Frosio, I., & Kautz, J. (2019). Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 11264–11272).
- Nweke, H. F., Teh, Y. W., Al-Garadi, M. A., & Alo, U. R. (2018). Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges. *Expert Systems with Applications*, 105, 233–261.
- Panja, A. K., Rayala, A., Agarwala, A., Neogy, S., & Chowdhury, C. (2023). A hybrid tuple selection pipeline for smartphone based Human Activity Recognition. *Expert Systems with Applications*, Article 119536.
- Park, W., Kim, D., Lu, Y., & Cho, M. (2019). Relational knowledge distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 3967–3976).
- Qiu, S., Zhao, H., Jiang, N., Wang, Z., Liu, L., An, Y., Zhao, H., Miao, X., Liu, R., & Fortino, G. (2022). Multi-sensor information fusion based on machine learning for real applications in human activity recognition: State-of-the-art and research challenges. *Information Fusion*, 80, 241–265.
- Rashidi, P., & Cook, D. J. (2009). Keeping the resident in the loop: Adapting the smart home to the user. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 39(5), 949–959.
- Ravi, D., Wong, C., Lo, B., & Yang, G.-Z. (2016). Deep learning for human activity recognition: A resource efficient implementation on low-power devices. In *2016 IEEE 13th international conference on wearable and implantable body sensor networks (BSN)* (pp. 71–76).
- Ravi, D., Wong, C., Lo, B., & Yang, G.-Z. (2017). A deep learning approach to on-node sensor data analytics for mobile or wearable devices. *IEEE Journal of Biomedical and Health Informatics*, 21(1), 56–64.
- Reiss, A., & Stricker, D. (2012). Introducing a new benchmarked dataset for activity monitoring. In *2012 16th international symposium on wearable computers* (pp. 108–109). IEEE.
- Ronao, C. A., & Cho, S.-B. (2016). Human activity recognition with smartphone sensors using deep learning neural networks. *Expert Systems with Applications*, 59, 235–244.

- Sekaran, S. R., Han, P. Y., & Yin, O. S. (2023). Smartphone-based human activity recognition using lightweight multiheaded temporal convolutional network. *Expert Systems with Applications*, 227, Article 120132.
- Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *ICLR*.
- Wang, X., Zhang, L., Huang, W., Wang, S., Wu, H., He, J., & Song, A. (2021). Deep convolutional networks with tunable speed–accuracy tradeoff for human activity recognition using wearables. *IEEE Transactions on Instrumentation and Measurement*, 71, 1–12.
- Yang, J., Nguyen, M. N., San, P. P., Li, X., & Krishnaswamy, S. (2015). Deep convolutional neural networks on multichannel time series for human activity recognition. In *Ijcai*, Vol. 15 (pp. 3995–4001). Buenos Aires, Argentina.
- Yang, J., Shen, X., Xing, J., Tian, X., Li, H., Deng, B., Huang, J., & Hua, X.-s. (2019). Quantization networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 7308–7316).
- Yang, J., Xia, X., Cui, J., & Zhang, Y.-D. (2023). An artificial bee colony algorithm with a cumulative covariance matrix mechanism and its application in parameter optimization for hearing loss detection models. *Expert Systems with Applications*, 229, Article 120533.
- Zeng, M., Gao, H., Yu, T., Mengshoel, O. J., Langseth, H., Lane, I., & Liu, X. (2018). Understanding and improving recurrent networks for human activity recognition by continuous attention. In *Proceedings of the 2018 ACM international symposium on wearable computers* (pp. 56–63).
- Zeng, M., Nguyen, L. T., Yu, B., Mengshoel, O. J., Zhu, J., Wu, P., & Zhang, J. (2014). Convolutional Neural Networks for human activity recognition using mobile sensors. In *6th international conference on mobile computing, applications and services* (pp. 197–205).
- Zhang, D., Liao, Z., Xie, W., Wu, X., Xie, H., Xiao, J., & Jiang, L. (2021). Fine-grained and real-time gesture recognition by using IMU sensors. *IEEE Transactions on Mobile Computing*, 22(04), 2177–2189.