This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3271306

A Collaborative Compression Scheme for Fast Activity Recognition on Mobile Devices Via Global Compression Ratio Decision

Junjie Liang, Lei Zhang✉, Chaolei Han, Can Bu, Hao Wu, Aiguo Song, *Senior Member, IEEE*

Abstract—Despite strong representation ability, deep convolutional neural networks (CNNs) are largely hindered in practical human activity recognition (HAR) deployment due to high computational cost, which is often unaffordable on resource-limited wearable devices. In this paper, to bridge the gap between on-device HAR and deep learning, we present a collaborative compression scheme to reduce the runtime of HAR with an acceptable performance degradation, which combines channel pruning and tensor decomposition to simultaneously handle sparsity and low-rankness when fully considering mutual interference in one network consisting of efficient 1-dimensional convolutional kernels. Our method includes two main stages. Concretely, given a target compression ratio, a global compression ratio decision optimization is first performed to automatically decide per-layer compression ratio by measuring compression sensitivity, without requiring labor-exhaustive human intervention. Then a multi-step collaborative compression is iteratively implemented to remove the least important compression unit based on an improved importance metric until the per-layer target compression ratio is attained. Extensive experiments on multiple HAR benchmarks show that our approach considerably outperforms previous compression strategies. For example, it can achieve around 50% FLOPs reduction with only an accuracy drop of 0.25% and 0.15% on UCI-HAR and PAMAP2, respectively. Actual implementation is evaluated on an embedded platform.

Index Terms—Sensor, convolutional neural networks, activity recognition, channel pruning, tensor decomposition, wearable device

1 Introduction

1.1 Motivation

The past 20 years have witnessed rapid development of the Internet of Things and sensor technology [1] [2]. Due to several prominent advantages such as higher accuracy, smaller size, and lower manufacturing prices, various sensors have been integrated into wearable devices such as mobile phones and watches, which make them smarter and more useful. Since human activity is unique, extracting useful knowledge from raw sensor data has been proved to be very critical in a broad range of application domains including smart homes, personal fitness, sports tracking, game console designing, and human behavior (activities of daily living like running, walking, sleeping, etc.) monitoring for health purpose to name a few [3] [4] [5]. In essence, human activity recognition (HAR) [6] [7] [8] based on wearable sensors could be handled as multichannel time series classification problem, which has recently attracted great interest in ubiquitous computing scenario. Traditional machine learning (ML) algorithms such as support vector machine, random forests, naive Bayes heavily depend on heuristic handcrafted feature extraction requiring specific domain knowledge and time-consuming human intervention [9] [10], which are largely hindered with regard to classification performance and model generalization in HAR. Plenty of deep learning methods equipped with an automatic feature extraction ability have recently been exploited to address above challenges. In particular, deep convolutional neural networks (CNNs) [11] [12] have been a dominant technique for a variety of activity recognition tasks, which have been verified to be more effective than the conventional ML approaches in inferring activity from sensor data.

However, despite stronger feature representation ability, larger CNNs are more resource-intensive, which is unlikely to be affordable on resource-limited platforms such as mobile and wearable devices. There inevitably exists an explosive growth of parameters and computational cost compared with traditional ML models, which severely restrict the deployment of CNNs in practical HAR applications. With regard to online or real-time activity recognition systems, an immediate response or low-latency prediction would be required. Therefore, the high computational cost issue has to be considered in HAR area. It is necessary to develop lightweight CNN models for implementing real-time and reliable activity recognition on mobile and wearable devices.

During the past decade, considerable research efforts have been devoted to investigating model compression or acceleration, including but not bound to, quantization [13] [14], knowledge distillation [15] [16], parameter pruning [17] [18], tensor decomposition [19] [20], etc. It is well known that both channel pruning and tensor decomposition are two most widely employed techniques in model compression, which seek to remove inherent redundancy in weight parameters according to different strategies. Specifically, the former yields sparse weight structures by identifying the important channels and removing the redundant or

- *Junjie Liang, Lei Zhang (E-mail: leizhang@njnu.edu.cn), Chaolei Han and Can Bu are with the School of Electrical and Automation Engineering, Nanjing Normal University, Nanjing, 210023, China.*
- *Hao Wu is with the School of Information Science and Engineering, Yunnan University, Kunming, 650500, China.*
- *Aiguo Song is with Department of Instrument Science and Engineering, Southeast University, Nanjing, 210096, China.*
- *Corresponding author: Lei Zhang*

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3271306

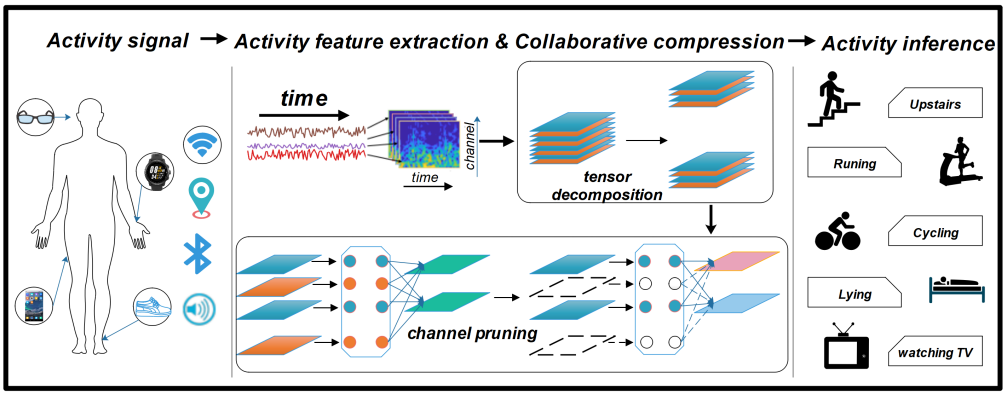


Fig. 1: The overview of the *Collaborative Compression* scheme.

unimportant ones, while the latter exploits tensor decomposition technique to approximate convolutional filters according to an intrinsic low-rank characteristic of weight parameters. Motivated by these properties, a natural idea is to combine the benefits of two compression strategies towards a fast and real-time activity inference. To pursue high as possible as compression ratio, [21] at the earliest time combines channel pruning and tensor decomposition sequentially to compress weight parameters, but process both of them in an independent or separate manner, which leads to a considerable accuracy drop. The main reason is that the two compression strategies are not completely orthogonal and might have a mutual inference, which makes it impossible to fully exploit the complementary nature of channel pruning and tensor decomposition. How one leverages the benefits of channel pruning and tensor decomposition to construct lightweight CNN models have rarely been explored in ubiquitous HAR computing scenario.

1.2 Our Approach and Contributions

In this paper, to maximize the benefits of two compression strategies, we propose a collaborative compression framework for HAR, which combines channel pruning and tensor decomposition to simultaneously handle sparsity and low-rankness in weights, as shown in Fig. 1. The mutual interference between both is fully considered for computational efficiency while minimizing potential accuracy drop. In particular, prior training-aware compression methods [22] [23] generally require to train networks from scratch, which are very difficult to control compression ratio. To decide an optimal global compression ratio, one has to tune hyper-parameters with different compression ratios along each layer to strike an ideal trade-off between model accuracy and compression ratio. Such exhaustive trial-and-error process inevitably leads to labor-intensive human intervention, which is computationally expensive and infeasible for real-time HAR with tiny or shallow networks. To avoid the drawback, the proposed approach seeks to simultaneously handle both sparsity and low-rankness in pre-trained networks. That is to say, such collaborative compression would be an inference-time or post-training algorithm, which is easier to implement in practical HAR environment compared with training-aware ones. To this end, the compression sensitivity of each layer is in detail analyzed through exploring the relationship between the compression ratio and information loss on sensor data. It can be found that an exponential decay curve can well fit the relationship, in which the compression sensitivity can be approximately modeled as a first-order derivative of exponential function. On this basis, a global compression ratio optimization process is constructed to generate the best possible compression ratio of each layer under a global target compression ratio, which allows an automatic control in the compression ratio. Once the compression ratio of each layer is determined by above optimizing process, we could easily compress every layer according to an importance metric in an independent and synchronous manner. The effectiveness of the proposed collaborative compression scheme is validated on four publicly available HAR benchmark datasets including WISDM, PAMAP2, UCI-HAR, and UniMiB-SHAR. Practical implementation is evaluated on a resource-limited embedded platform. We summarize our main contributions as follows:

- We propose a new collaborative compression framework towards a fast and efficient activity inference using sensor data, which combines channel pruning and tensor decomposition to simultaneously handle sparsity and low-rankness in weights when fully considering mutual inference between both, hence leading to better trade-off between model accuracy and compression ratio compared with a single compression operation.
- Since analyzing exactly compression sensitivity of each layer is extremely expensive even for tiny networks, we approximate it with an exponential function under a target compression ratio, resulting in a global optimization criterion to yield an automatic control in compression ratio of each layer, which is easy to implement in existing HAR environment with minimal computational overhead over training.
- Given the whole network compression ratio, a comprehensive analysis is provided to identify the level of compression sensitivity in different layers. Extensive experiments on several HAR benchmark datasets verify that our collaborative compression approach could decrease memory footprint and accelerate activity inference at difference compression ratios, without requiring any specialized deep learning libraries or hardware.

Below, Section 2 first reviews previous literatures of and

outlines main distinctions between them and our collaborative compression approach. Section 3 then gives an overview of the collaborative compression framework and introduces the details of our approach. The main experiment results and a series of ablation studies on four popular HAR benchmark datasets are presented in Section 4. Finally, Section 5 concludes this paper with the summary and discussion.

2 Related Work

During recent years, deep CNN models have been extensively applied in sensor-based human activity recognition community [6] [4] [5] [10]. Zeng *et al.* [10], Ronao *et al.* [6], and Yang *et al.* [7] at the earliest time exploit CNN models to identify human activities by automatically extract discriminative features from multivariate time series sensor data. Despite superior accuracy, deep CNN models are typically resource-hungry, the early CNN model like AlexNet [11] consists of five convolutional layers and three fully connected layers, which has 61M parameters (249 MB memory footprint) and needs to implement 1.5B high precision operations to make a prediction. Thus, it is important and essential to address the issue of high computational cost caused by deep CNN models to realize fast and reliable human activity recognition on mobile or wearable devices. For example, [24] provides a feasible solution by mingling handcrafted and deep features to reduce computational cost, in which handcrafted spectral features are integrated with only one convolutional layer and two fully connected layers for real-time activity recognition. The feasibility of such hybrid usage is validated on two smartphones, which requires only tens of milliseconds inference time for one prediction. References [24] [25] also shows the combination of deep neural networks and handcrafted features is a potential strategy to implement real-time activity inference on resource-limited mobile devices. Another intuitive strategy is network quantization, where both memory usage and model size are greatly reduced in contrast to full-precision networks [24]. Though there has been a great amount of literature on model compression [26] [18] in the latter years, little attention has been devoted to accelerating activity inference based on deep learning. In particular, the direct application of channel pruning or tensor decomposition to sensor-based activity recognition that has not been seen before according to existing literature. The main reason is that such compression strategies are labor-intensive and very costly, and require time-consuming human analysis with trial-and-error to obtain a satisfactory compression ratio. The whole network compression ratio is difficult to control, which restricts their potential use in ubiquitous HAR scenario.

During recent years, there have been several appealing studies including Ding et al. [27], Liebenwein et al. [28], Dong et al. [29], which perform automatic per-layer compression ratio decisions for deep neural networks. Built on these prior studies, in this paper, we focus on both channel pruning and tensor decomposition, which are most related to our work. Channel pruning suggested by Wen *et al.* [30], Luo *et al.* [31], Liu *et al.* [18], etc. mainly aims to reduce model size and accelerate inference via removing redundant channels or the related filters from the network, resulting in structured sparsity. Therefore, the pruned network has almost the same architecture but with fewer channels, which could yield smaller memory footprint and faster inference without needing any specialized hardware or software support. From another perspective, tensor decomposition [19] [20] [22] [26] primarily seeks to factorize large convolutional kernels into a series of tensors with fewer parameters, which makes network smaller and faster. Intuitively, tensor decomposition might be complementary to channel pruning, and both could be combined together to yield a higher compression ratio than using a single compression operation. To this end, previous works [21] [32] ensemble multiple compression strategies and sequentially perform one compression operation at each stage in a separate manner, but inevitably ignore the mutual interference between different compression operations, which leads to a considerable accuracy drop. In this paper, we focus our attention on extending previous studies in HAR environment with a new collaborative compression criterion, which combines channel pruning and tensor decomposition to handle sparsity and low-rankness from the trained model. Moreover, unlike applications in vision domain, how to control the whole network compression ratio still remains a challenging problem, which has rarely been explored in HAR area. Through in detail analyzing the compression sensitivity between information loss and compression ratio, we construct a global compression ratio optimization process based on sensor data to resolve this issue.

3 Model

3.1 Preliminaries

Deep CNNs can be seen as multi-layer feed-forward architectures that transform raw sensor input into certain output tensors through a sequence of operations. Formally, given the l-th convolutional layer that transforms an input tensor $X^l \in \mathbb{R}^{c^l \times h^l_{in} \times w^l_{in}}$ to an output tensor $Y^l \in \mathbb{R}^{n^l \times h^l_{out} \times w^l_{out}}$, let $W^l \in \mathbb{R}^{n^l c^l k^l 1^l}$ be the network weights with regard to the l-th convolutional layer, where c^l and n^l denote the number of input and output channels respectively. Here h^l_{in} and w^l_{in} represent the height and width of the input tensor, while h^l_{out} and w^l_{out} represent the height and width of the output tensor. In fact, heterogenous sensor modalities always remain a critical concern in HAR. To remove the impact of different sensing modalities, following previous literatures including Yang et al. [7], Zeng et al. [10], and Ronao et al. [6], we only perform one-dimensional (1D) convolutions along temporal dimension of multivariate time series sensor signals, where all the feature maps from different sensor modalities are merged before feeding into final fully-connected layer. Thus, the filter size is set to $k^l \times 1^l$. The normal convolutional operation can be expressed as:

$$Y^l = W^l \otimes X^l, \tag{1}$$

in which $\otimes$ denotes the convolutional operation and bias terms are ignored in our formulation for simplicity.

Channel Pruning: Without loss of generality, to induce sparsity, we treat the compression process as a function $\widehat{W}^l = f(W, o)$, in which o is equivalent to a corresponding compression unit index that would be removed [17] [18]. Based on above equation, when every channel is regarded as an individual compression unit, the channel pruning could be easily formulated as:

$$f(W,o) = \widehat{W}^l = \widehat{W}^l_{:,j,:,:} = \begin{cases} W^l_{:,j,:,:}, & j \neq o, \\ 0, & j = o. \end{cases} \tag{2}$$

Tensor Decomposition: The main goal of tensor decomposition scheme is to find a low-rank approximation of W^l that facilitates more efficient computation over CNN models according

to a simple and natural idea: replace the original convolution with two light convolutions by $W_2^l \otimes W_1^l \otimes X$. The model weights could be decomposed by using singular value decomposition (SVD) [19] [33] in a stable way. The original tensor W^l is first transformed into a matrix $M^l \in \mathbb{R}^{n^l \times (c^l k^l 1^l)}$, and the SVD of M^l is then denoted as $U^l \Sigma^l V^{l^{\mathrm{T}}}$, in which $U^l \in \mathbb{R}^{n^l \times n^l}$ and $V^l \in \mathbb{R}^{(c^l k^l 1^l) \times (c^l k^l 1^l)}$. The number of singular values, i.e., non-zero diagonal elements of M^l is equivalent to the rank $r^l = \min\{n^l, c^l k^l 1^l\}$. In such manner, the weight matrix M^l could be decomposed into $M_2^l = U^l \sqrt{\Sigma^l}$ and $M_1^l = \sqrt{\Sigma^l} V^{l^T}$, whose number of non-zero singular values, i.e., $\hat{r}^l$ is much smaller than $r^l = \min\{n^l, c^l k^l 1^l\}$. Finally, both M_2^l and M_1^l are mapped back to $W_2^l \in \mathbb{R}^{n^l \times \hat{r}^l \times 1 \times 1}$ and $W_1^l \in \mathbb{R}^{\hat{r}^l \times c^l \times k^l \times 1^l}$ for implementation [19] [20]. Similar to above Eq. (2), when the index of non-zero singular values is regarded as the compression unit index o, the compression process is formulated as:

$$f(W,o) = \widehat{W}^l = \varphi\left(U^l \hat{\Sigma}^l V^{l^{\mathrm{T}}}\right), \hat{\Sigma}^l = \begin{cases} \Sigma_{j,j}^l, & j \neq o, \\ 0, & j = o, \end{cases} \tag{3}$$

where φ is an operator that is in charge of mapping $M^l \in \mathbb{R}^{n^l \times (c^l k^l 1^l)}$ back to $W^l \in \mathbb{R}^{n^l c^l k^l 1^l}$.

Combining Channel Pruning and Tensor Decomposition: Different from previous compression methods [26] [34] [32], instead of sequentially performing one compression operation at each time in an isolated manner, we adopt a new collaborative compression scheme to accelerate online activity inference. For notational continence, let K^l be the set of indices for all candidate compression units at the l-th layer. Noting that there is overall $c^l + r^l$ candidate compression units, one could achieve the compressed network weights by simultaneously removing both t_1 input channels and t_2 singular values according to Eq. (2) and Eq. (3):

$$f(W,o) = \widehat{W}^l = \begin{cases} \widehat{W}_1^l \in \mathbb{R}^{(r^l - t_2) \times (c^l - t_1) \times k^l \times 1^l}, \\ \widehat{W}_2^l \in \mathbb{R}^{n^l \times (r^l - t_2) \times 1 \times 1}. \end{cases} \tag{4}$$

In an extreme case when $t_2 = 0$, only the channel pruning, i.e., $\widehat{W}^l \in \mathbb{R}^{n^l \times (r^l - t_1) \times k^l \times 1^l}$ is employed to compress the network weights. On this basis, the whole compression ratio at the l-th convolutional layer can be represented as:

$$R^l = 1 - \frac{(r^l - t_2) \times \left[(c^l - t_1) \times k^l \times 1^l + n^l\right]}{n^l \times c^l \times k^l \times 1^l}. \tag{5}$$

Moreover, if $t_2 = 0$, the compression ratio could be further simplified into $\frac{t_1}{c^l}$.

3.2 Collaborative Compression

As mentioned before, the collaborative compression seeks to remove the unimportant t_1 and t_2 compression units from a well-trained CNN model for model acceleration while minimizing an accuracy drop. Overall, our method consists of three main steps: (1) perform a fast per-layer compression sensitivity analysis; (2) determine the best compression ratio of each layer by solving a simple optimization problem under the target global compression ratio; (3) implement a multi-step collaborative compression according to an improved importance metric. The overall working pipeline of the algorithm is illustrated in Fig. 2. The details of the three steps are provided as follows.

Per-layer Compression Sensitivity: We first perform per-layer compression sensitivity analysis. As indicated in [35] [36], the compression sensitivity is possibly variable across different layers, where the compression ratio could be higher for the insusceptible layers while lower for the sensitive layers. Following

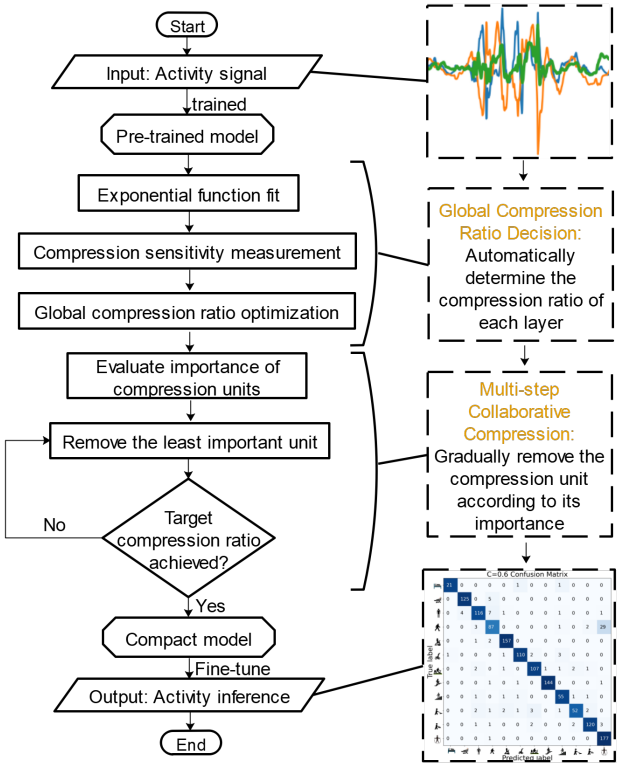


Fig. 2: Flow-chart of *Collaborative Compression* procedure.

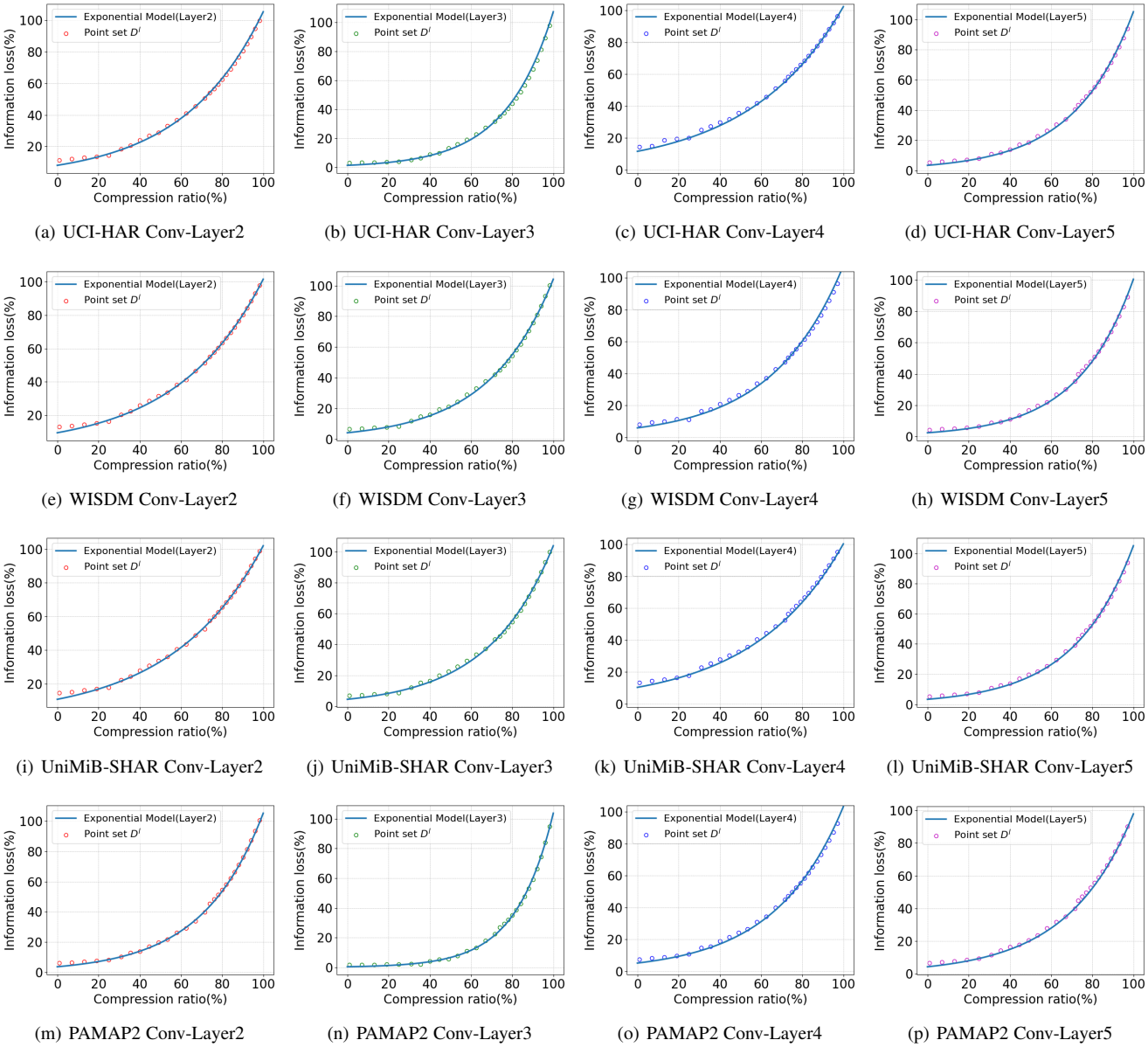
previous literature [34] [37] [17], we adopt a first-order Taylor expansion to approximate the information loss of the l-th layer on the compressed weights W^l:

$$\begin{aligned} I^l = \left[L\left(\widehat{W}^l\right) - L\left(W^l\right)\right]^2 &\approx \sum_{i \in Q^l} \left(\frac{\partial L}{\partial W_i^l} * \left(\widehat{W}_i^l - W_i^l\right)\right)^2 \\ &= S\left[\left(G^l * \left(\widehat{W}^l - W^l\right)\right)^2\right], \end{aligned} \tag{6}$$

in which Q^l denotes indices of all entries in W^l and the operator $S[\cdot]$ represents the sum of all elements. $G^l \in \mathbb{R}^{n^l \times c^l \times k^l \times 1^l}$ is the average gradient of the information loss with regard to the network weights W^l already pre-trained over the whole dataset. Based on above simplified first-order Taylor approximation, previous methods [35] [36] generally analyze the compression sensitivity of each layer by removing a certain ratio of compression units and then determining which compression units really contribute more to the discriminative ability of CNN model according to their importance. One compression unit could be regarded as an important one only when the information loss would drastically increase without it. However, as mentioned before, such process is very costly and labor-intensive, which needs to iteratively compute the information loss under different compression ratios.

To address this issue, taking an inspiration from [34] [37] [17] [35], we employ a greedy algorithm to fast measure compression sensitivity of each layer under different compression ratios, where the compression sensitivity can be treated as a point set D^l that is used to describe the relationship between information loss and compression ratio. In contrast to full combinatorial search, we refer to the simplified approximation as a first-order greedy search. Let D^l be empty at first. To obtain the information loss under different compression ratios, one could progressively remove candidate compression units based on their importance, i.e., the

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3271306



(a) UCI-HAR Conv-Layer2 (b) UCI-HAR Conv-Layer3 (c) UCI-HAR Conv-Layer4 (d) UCI-HAR Conv-Layer5

(e) WISDM Conv-Layer2 (f) WISDM Conv-Layer3 (g) WISDM Conv-Layer4 (h) WISDM Conv-Layer5

(i) UniMiB-SHAR Conv-Layer2 (j) UniMiB-SHAR Conv-Layer3 (k) UniMiB-SHAR Conv-Layer4 (l) UniMiB-SHAR Conv-Layer5

(m) PAMAP2 Conv-Layer2 (n) PAMAP2 Conv-Layer3 (o) PAMAP2 Conv-Layer4 (p) PAMAP2 Conv-Layer5

Fig. 3: Compression sensitivity analysis of different datasets.

information loss in Eq. (6). When one candidate compression unit is removed, both the corresponding information loss I^l and compression ratio R^l can easily be computed according to Alg. 1. After that, the compression sensitivity curves from the point set D^l are plotted in Fig. 3, where both I^l and R^l are normalized into an interval [0, 1]. By measuring the compression sensitivity of all the layers across various HAR benchmarks, we empirically verify that the curve of point set D^l could be well fit by an exponential function $I^l = a^l e^{b^l R^l}$, where a^l and b^l are learnable parameters through least squares. This is in well line with previous observation in [34] as well as common intuition.

Global Compression Ratio Optimization: Given the whole network compression ratio C, the compression ratio of every layer can be determined by solving the following optimization problem:

$$\min_{R} \left(\sum_{l=1}^{L} R_l \cdot F_l - C \cdot F \right)^2, \forall l \in \{1, 2, \ldots, L\}, \tag{7}$$

in which the set $R = \{R_1, ..., R_L\}$ represents the compression ratio at different layers. F and F_l denote the computational cost (i.e., FLOPs) of the whole network and l-th layer respectively. However, it is difficult to directly optimize Eq. (7). Instead, the optimization problem could be simplified through transforming the optimized variables from R^l to $\bar{I}^l$ according to the following formula:

$$\bar{I}^l = \frac{\partial I^l}{\partial R^l} = a^l b^l e^{b^l R^l} \Rightarrow R^l = \frac{1}{b^l} \log \left(\frac{\bar{I}^l}{a^l b^l} \right). \tag{8}$$

Keeping $\bar{I}^l$ the same for all layers, one could fit above exponential function with a^l and b^l across different layers based on the point set D^l in order to ensure the sensitive layers have lower compression ratio while the insusceptible layers higher compression ratio. Thus, the problem (7) could be reformulated as:

$$\min_{\bar{I}^l} \left(\sum_{l=1}^{L} \frac{F_l}{b^l} \log \left(\frac{\bar{I}^l}{a^l b^l} \right) - C \cdot F \right)^2, \tag{9}$$

which is easier to optimize by a gradient descent algorithm, as shown in Alg. 2. On this basis, one could determine the compression ratio of every layer according to Eq. (9) under global target compression ratio.

Algorithm 1 Compression sensitivity learning algorithm.

Input: Pre-trained weights W^l at the l-th layer; The corresponding average gradient G^l; The set K^l of indices for $c^l + r^l$ candidate compression units; The empty set A; The point set of compression sensitivity D^l.

Output: An exponential function $I^l = a^l e^{b^l R^l}$.

Set $\widehat{W}^l \leftarrow W^l, D^l \leftarrow \phi, A^l \leftarrow \phi, t_1 \leftarrow 0, t_2 \leftarrow 0$.

for $t \leftarrow 0$ to $c^l + r^l$ **do**

Find the index:

$$o = \arg\min_{o \notin A^l} \left\{ \left\| G^l * \left(f\left(\widehat{W}^l, o\right) - \widehat{W}^l \right) \right\|^2 \right\}.$$

$\widehat{W}^l = f\left(\widehat{W}^l, o\right)$.

$I^l = \frac{\left\| G^l * \left(\widehat{W}^l - W^l\right) \right\|_2^2}{\left\| G^l * W^l \right\|_2^2}$.

if compression unit index o in the c^l **then**

$t_1 \leftarrow t_1 + 1$;

else

$t_2 \leftarrow t_2 + 1$;

end if

Compute R^l based on Eq. (5).

Let $A^l \leftarrow A^l \cup \{o\}, D^l \leftarrow D^l \cup \left\{\left(R^l, I^l\right)\right\}$.

$t \leftarrow t + 1$.

end for

Exploit the exponential function $I^l = a^l e^{b^l R^l}$ to fit D^l based on least squares.

Algorithm 2 Compression ratio decision algorithm.

Input: The FLOPs of the whole network F; The FLOPs of each layer F=$\{F^1,...,F^L\}$; Global compression ratio C; Learning rate η; The exponential function of each layer $\{I^1 = a^1 e^{b^1 R^1}, \ldots, I^L = a^L e^{b^L R^L}\}$; The compression sensitivity $\bar{I}'$.

Output: The optimized compression ratio of each layer $\{R^1,...,R^L\}$.

Set $\bar{I}' \leftarrow 0.1$.

do

$$g = 2\left(\sum_{l=1}^{L} \frac{F^l}{b^l} \log\left(\frac{\bar{I}'}{a^l b^l}\right) - CF\right)\left(\sum_{l=1}^{L} \frac{F^l}{b^l \bar{I}'}\right),$$

$\bar{I}' = \bar{I}' - \eta g$.

while $\left(\sum_{l=1}^{L} \frac{F^l}{b^l} \log\left(\frac{\bar{I}'}{a^l b^l}\right) - CF\right)^2 > 10^{-4}$:

Return: $\left\{R^l = \frac{1}{b^l} \log\left(\frac{\bar{I}'}{a^l b^l}\right)\right\}_{l=1}^{L}$.

Multi-step Collaborative Compression: Once the compression ratio is determined, a multi-step collaborative compression scheme is then employed to compress each layer's weights. Through calculating the importance of each compression unit and then sorting them ascendingly, we progressively remove the least important unit step by step. Such compression process would be repeated until the obtained compression ratio is smaller than or equal to the target compression ratio at the l-th layer. In fact, prior most works assume that both channel pruning and tensor decomposition are totally orthogonal, which separately deal with the sparsity and low-rankness in an isolated manner, without considering their mutual inference. On the contrary, our approach seeks to jointly handles the sparsity on channel pruning and low-rankness on tensor decompensation in a unified compressed framework. However, as aforementioned, it is worth noting that removing an individual candidate compression unit might potentially affect the importance of the remaining compression units due to the mutual influence when two compression operations are mingled. To resolve this concern, we have to perform a multi-step compression process by iteratively removing the less important compression units step-by-step. However, the aforementioned importance metric, i.e., Eq. (6) is static, which only exploits fixed weights to compute the importance metric once at the beginning, hence making Eq. (6) to be no longer effective for such dynamic multi-step process. To avoid the drawback, taking an inspiration from the value function in Markov Decision Process [38] [34] [39], we further modify above Eq. (6) as a dynamic importance metric:

$$S_o^{l,(t)} = I_o^{l,(t)} + \gamma \frac{1}{\left|K^{l,(t)}\right| - 1} \sum_{i \in K^{l,(t)} \setminus o} I_{i|o}^{l,(t)}, \tag{10}$$

where $K^{l,(t)}$ is an index set used to represent the remaining compression units that have not been removed after t-1 steps. Both $I_o^{l,(t)}$ and $I_{i|o}^{l,(t)}$ are respectively denoted as follows:

$$I_o^{l,(t)} = S\left[\left(G^l * \left(\widehat{W}_o^{l,(t)} - W^l\right)\right)^2\right], \widehat{W}_o^{l,(t)} = f\left(\widehat{W}^{l,(t-1)}, o\right), \tag{11}$$

and

$$I_{i|o}^{l,(t)} = S\left[\left(G^l * \left(\widehat{W}_{i|o}^{l,(t)} - W^l\right)\right)^2\right], \widehat{W}_{i|o}^{l,(t)} = f\left(\widehat{W}_o^{l,(t)}, i\right). \tag{12}$$

Specifically, in Eq. (11), the first term is used to denote the importance of the o-th compression unit over $\widehat{W}^{l,(t-1)}$ at the $(t-1)$-th step, while the second term is used to estimate an average information loss that is calculated by iteratively removing each compression unit from the remaining compression space. γ is a hyper-parameter used to trade-off mutual influence between them. At each step, one can dynamically calculate the importance of one candidate compression unit according to the improved importance metric while fully considering the mutual inference between the removed compression unit at current step and the remaining compression space (i.e., these unremoved compression units). That is to say, the unified collaborative compression pipeline can keep more information to progressively reach a given target compression ratio by iteratively pruning less important channels or removing smaller singular values in a mingled way. In such a way, one can compress all layers parallel, and then fine-tune the compressed network to regain accuracy.

4 Experiments

In this section, we empirically evaluate the performance of the collaborative compression scheme on a variety of HAR benchmark datasets. To investigate the effectiveness of the proposed approach, we adopt the following state-of-the-art methods as the baselines for study: channel pruning, tensor decomposition, and the combination of two compression operations. Furthermore, we perform ablation studies with variations of several crucial hyper-parameters in order to analyze their effects and explore the best variant. Finally, we demonstrate the effectiveness of our collaborative compression approach via measuring practical activity inference time over Raspberry Pi 4 B+ platform.

TABLE 1: Data preprocessing (A=accelerometer, G=gyroscope, M=magnetometer)

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

4.1 Benchmark Datasets

We conduct our experiments on four popular HAR benchmark datasets including UCI-HAR [40], WISDM [41], UniMiB-SHAR [42] and PAMAP2 [43], which have been collected through multiple sensors attached to different body positions in various environments and conditions. As shown in Fig. 1, raw sensor data has to be first preprocessed to filter out noise signals and normalized into the range of [0, 1]. Then the preprocessed data is segmented with proper window lengths that are likely to contain the whole information about target activities, where an overlap rate between two adjoining windows is allowed to maintain the continuity of activity data. During training stage, the ground truth labels and activity features extracted from each window are used as input to train one CNN classifier. For comparison purpose, following the previous literature [40] [41] [42] [10], we still adopt the same parameter settings such as window length, overlap rate, etc., as summarized in Table 1.

UCI-HAR dataset [40]. This dataset was built by a research team from the University of California Irvine to benchmark various machine learning algorithms for activity recognition task. There have been thirty volunteer subjects whose ages range from 19 to 48 years old asked to perform 6 types of activities including 'walking', 'walking upstairs', 'walking downstairs', 'sitting', 'standing', and 'lying'. Raw sensor signals were recorded at a constant sampling frequency of 50 Hz through triaxial accelerometer and gyroscope embedded a Samsung Galaxy S II smartphone wore on the waist. Following the original segmentation method in [40], the whole dataset is randomly split into two parts, where 70% for training and 30% for test.

WISDM dataset [41]. The dataset was collected by the WISDM research team from Fordham University. In a supervised condition, twenty-nine subjects are instructed to perform 6 kinds of daily activities consisting of 'walking', 'jogging', 'upstairs', 'downstairs', 'sitting', and 'standing'. Raw sensor data is collected by triaxial accelerometer embedded in several brands of Android phones such the Nexus One, HTC Hero placed in front pants leg pockets. The accelerometer data is recorded every 50 ms, which yields 20 samples per second. All data is randomly partitioned into three parts, of which 70% are used as the training set, 10% as the validation set, and 20% as the test set.

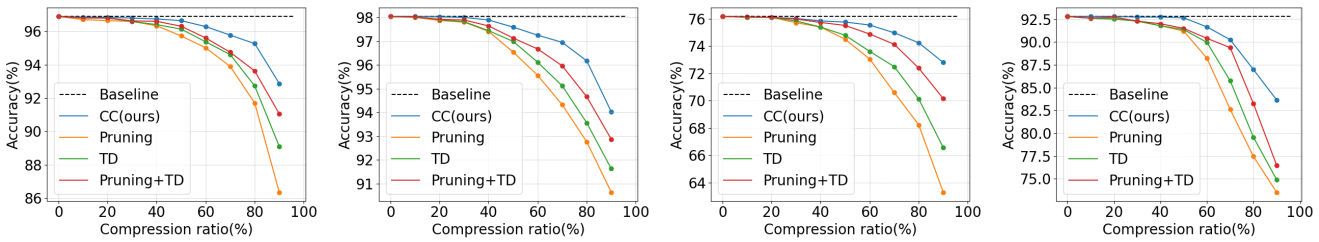
UniMiB-SHAR dataset [42]. The dataset has been designed to benchmark activity recognition and fall detection, which includes total 11,771 samples from 30 persons whose ages range between 18 and 60 years old. All samples could be divided into 17 fine grained categories, which include 8 types of activities of daily living (ADL) such as 'walking', 'standing', 'sitting', etc., and 9 types of falls such as 'falling leftward', 'falling rightward', 'falling backward', etc. Each person wore a Samsung Galaxy Nexus I9250 phone in his/her front trouser pocket. The sensor data was sampled at a fixed frequency of 50 Hz by an embedded triaxial BMA220 accelerometer. We use five-fold cross-validation to evaluate the performance of the model on this dataset.

PAMAP2 dataset [43]. The dataset has been collected from the whole nine volunteers, who were instructed to perform 18 types of physical activities, including 12 types of specific activities (walking, cycling, rope jumping, etc.) and several types of optional activities (watching TV, playing football, driving, etc.). Every volunteer wore three Inertia Measurement Units (IMUs) nodes placed on each dominant's chest, hand, and ankle. The data was recorded at a sampling rate of 100 Hz, which was then sub sampled to 33.3 Hz for further study. A heart rate monitor with a sampling rate of 9 Hz was used to estimate motion intensity. The splitting strategy is consistent with previous successful work, where Participant 6 constitutes the test set, Participant 5 constitutes the validation set, and the remaining participants used to train the model.

4.2 Implementation details

The proposed collaborative compression approach is implemented by using PyTorch. We apply the convolutional architecture described in [6] [7] [44] for our evaluation. The network consists of five convolutional and one linear layers with ReLU activation and Batch Normalization (BN) operation inserted between them, where the linear layer has a small input size and less hidden units, which does not occupy large portions of parameters. All the layers are arranged in the following order: C-B-R-C-P-R-C-B-R-C-P-R-C-B-P-R-L (C: Conv, R: ReLU, B: Batch Normalization, P: Pooling, L: Linear) where each convolutional layer has a certain number of channels or neurons, as shown in Fig. 5. For example, on UCI-HAR, the second layer has 32 input and 64 output channels and filters of size 9×1; On WISDM, the third layer has 64 input and 128 output channels with filter size of 3×1, ect. All networks are trained for 300 epochs by using the mini-batch size of 64. The weight parameters are optimized via stochastic gradient descent (SGD) with momentum 0.9, where the initial learning rate is set to 0.01 that is divided by 10 every 50 epochs. Based on the pre-trained models, the proposed collaborative compression scheme is employed to determine the desired compression units. We consider only four intermediate convolutional layers that constitute around 80% of inference time without compressing the first and the last layers. Once the global compression ratio is set, one could decide what percentage of compression units to be removed from each layer, where the number of removed compression units has a direct correlation with acceleration. However, it would be destructive to model performance after removing too many compression units. We adopt above multi-step compression and layer-wise iterative fine-tuning (retraining) strategy to recovery accuracy. Specifically, the compressed network is retrained (fine-tuned) on validation data for 20 epochs to regain the accuracy. In particular, γ to set to 0.5 for controlling balance in the importance metric. Finally, the

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3271306



(a) UCI-HAR Accuracy (b) WISDM Accuracy (c) UniMiB-SHAR Accuracy (d) PAMAP2 Accuracy

Fig. 4: Comparison of the accuracy under different compression ratios, TD means tensor decomposition.

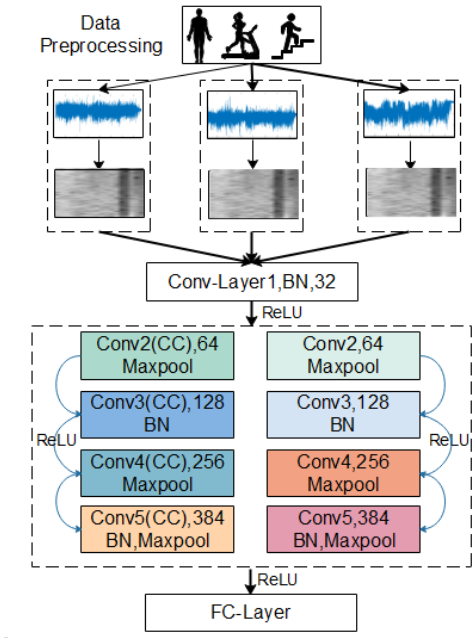


Fig. 5: Brief description for architecture.

performance is evaluated by both accuracy and F1-score due to potential class imbalance in activity data [41].

4.3 Main Results

We analyze the performance of collaborative compression scheme via comparing against a single compression operation (either channel pruning or tensor decomposition) and two compression operations. To investigate the effect of setting different target compression ratios, we remove a certain portion of candidate compression units and measure the accuracy obtained from the compressed models on UCI-HAR, WISDM, UniMiB-SHAR, and PAMAP2. Experimental results are shown in Fig. 4. It can be seen that the performance of the compressed models becomes worse with the increase of compression ratio. However, our compression method with a 50% rate is close to the pre-trained model, with only 0.15% reduction in accuracy. Overall, the performance degradation of our collaborative compression is smaller than that of other methods at the same compression ratio. To ensure fair comparisons, we measure the accuracy while fixing similar reductions of parameters and FLOPs, i.e., the same target compression ratio, or measure the reductions of parameters and FLOPs while fixing the accuracy to be close to the baselines.

On one hand, we perform the single view evaluation by measuring the accuracy while fixing the same compression ratio, as shown in Table 2. Here the results of "channel pruning + tensor decomposition" are obtained via first pruning those candidate channels (removing t_1 compression units) and then implementing singular value decomposition (removing t_2 compression units), which is regards as an intuitive combination of two compression operations without considering their mutual interference, i.e., Eq. (10). It can be seen that our approach achieves the best performance, which is more accurate than either channel pruning or tensor decomposition alone, also outperforms its simple combination. For example, under the same compression ratio, our collaborative compression method maintains a higher accuracy compared with channel pruning (e.g., 96.64% vs. 95.72% on UCI-HAR, 97.59% vs. 96.53% on WISDM, 75.77% vs. 74.51% on UniMiB-SHAR, and 92.65% vs. 91.20% on PAMAP2). Meanwhile, our method outperforms tensor decomposition, which leads to 0.51%, 0.60%, 0.97%, and 1.30% improvement in accuracy (e.g., 96.64% vs. 96.13% on UCI-HAR, 97.59% vs. 96.99% on WISDM, 75.77% vs. 74.80% on UniMiB-SHAR, and 92.65% vs. 91.35% on PAMAP2). Lastly, it shows better performance than Channel Pruning + Tensor Decomposition (e.g., 96.64% vs. 96.30% on UCI-HAR, 97.59% vs. 97.13% on WISDM, 75.77% vs. 75.52% on UniMiB-SHAR, and 92.65% vs. 91.49% on PAMAP2), which emphasize the benefit of considering mutual interference between two compression operations.

On the other hand, we measure the reductions of parameters and FLOPs while maintaining the comparable accuracy. As shown in Table 3, our collaborative compression method surpasses its counterparts in two aspects including FLOPs and parameters reduction. For example, our collaborative compression yields larger FLOPs reduction (e.g., 63.2M vs. 105.3M on UCI-HAR, 44.4M vs. 61.5M on WISDM, 13.6M vs. 35.4M on UniMiB-SHAR, and 62.1M vs. 99.3M on PAMAP2) while greatly reducing the model complexity (e.g., 0.43M vs. 0.72M on UCI-HAR, 0.41M vs. 0.57M on WISDM, 0.18M vs. 0.47M on UniMiB-SHAR, and 0.44M vs. 0.70M on PAMAP2 for parameters), which significantly outperforms channel pruning. Compared with tensor decomposition, it further relieves memory overload (e.g., 0.43M vs. 0.63M on UCI-HAR, 0.41M vs. 0.52M on WISDM, 0.18M vs. 0.45M on UniMiB-SHAR, and 0.44M vs. 0.58M on PAMAP2) and accelerates the computation (e.g., 63.2M vs. 91.9M on UCI-HAR, 44.4M vs. 55.5M on WISDM, 13.6M vs. 33.9M on UniMiB-SHAR, and 62.1M vs. 82.3M on PAMAP2). Finally, in comparison with Channel Pruning + Tensor Decomposition, it leads to faster acceleration (e.g., 63.2M vs. 86.2M on UCI-HAR, 44.4M vs. 52.2M on WISDM, 13.6M vs. 22.6M on UniMiB-

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3271306

TABLE 2: Comparison of the performance at the same compression ratio C=0.5

Method	Accuracy	Accuracy.Drop	F1-score	F1-score.Drop
		UCI-HAR		
Baseline	96.89%	–	97.02%	–
Pruning	95.72%	1.17%	95.67%	1.35%
TD	96.13%	0.76%	96.16%	0.86%
Pruning+TD	96.30%	0.59%	96.34%	0.68%
CC(ours)	**96.64%**	**0.25%**	**96.62%**	**0.40%**
		WISDM		
Baseline	98.04%	–	97.96%	–
Pruning	96.53%	1.51%	96.51%	1.45%
TD	96.99%	1.05%	97.05%	0.91%
Pruning+TD	97.13%	0.91%	97.08%	0.88%
CC(ours)	**97.59%**	**0.45%**	**97.52%**	**0.44%**
		UniMiB-SHAR		
Baseline	76.18%	–	79.43%	–
Pruning	74.51%	1.67%	78.20%	1.23%
TD	74.80%	1.38%	78.34%	1.09%
Pruning+TD	75.52%	0.66%	79.02%	0.41%
CC(ours)	**75.77%**	**0.41%**	**79.28%**	**0.15%**
		PAMAP2		
Baseline	92.80%	–	94.77%	–
Pruning	91.20%	1.60%	93.74%	1.03%
TD	91.35%	1.45%	93.85%	0.92%
Pruning+TD	91.49%	1.31%	93.87%	0.90%
CC(ours)	**92.65%**	**0.15%**	**94.55%**	**0.22%**

TABLE 3: Comparison of the computational complexity under the close accuracy

Method	Accuracy	FLOPs	FLOPs.Drop	Param	Param.Drop
			UCI-HAR		
Baseline	96.89%	191.5M	–	1.31M	–
Pruning	96.00%	105.3M	86.2M	0.72M	0.59M
TD	96.07%	91.9M	99.6M	0.63M	0.68M
Pruning+TD	96.13%	86.2M	105.3M	0.59M	0.72M
CC(ours)	**96.17%**	**63.2M**	**128.3M**	**0.43M**	**0.88M**
			WISDM		
Baseline	98.04%	111.0M	–	1.04M	–
Pruning	97.02%	61.5M	49.5M	0.57M	0.47M
TD	97.10%	55.5M	55.5M	0.52M	0.52M
Pruning+TD	97.15%	52.2M	58.8M	0.49M	0.55M
CC(ours)	**97.20%**	**44.4M**	**66.6M**	**0.41M**	**0.63M**
			UniMiB-SHAR		
Baseline	76.18%	75.3M	–	0.99M	–
Pruning	74.01%	35.4M	39.9M	0.47M	0.52M
TD	74.04%	33.9M	41.4M	0.45M	0.54M
Pruning+TD	74.10%	22.6M	52.7M	0.30M	0.69M
CC(ours)	**74.18%**	**13.6M**	**61.7M**	**0.18M**	**0.81M**
			PAMAP2		
Baseline	92.80%	206.9M	–	1.46M	–
Pruning	90.02%	99.3M	107.6M	0.70M	0.76M
TD	90.11%	82.3M	124.6M	0.58M	0.88M
Pruning+TD	90.16%	72.5M	134.4M	0.51M	0.95M
CC(ours)	**90.23%**	**62.1M**	**144.8M**	**0.44M**	**1.02M**

SHAR, and 62.1M vs. 72.5M on PAMAP2) and more compact parameters (e.g., 0.43M vs. 0.59M on UCI-HAR, 0.41M vs. 0.49M on WISDM, 0.18M vs. 0.30M on UniMiB-SHAR, and 0.44M vs. 0.51M on PAMAP2), which is fast enough to be deployed on embedded platform.

We also compare our approach with other compression strategies based on single compression operations including knowledge distillation [16] and network quantization [45]. We implement all the results. As shown in the following Table 4, the collaborative compression scheme can yield superior performance gains over previous ones. For example, compared to knowledge distillation [16], the collaborative compression scheme with $C = 0.5$ leads to better performance (75.77% vs. 74.72%) with higher FLOPs reduction (49.9% vs. 43.7%) on UniMiB-SHAR. In the mean-

TABLE 4: Compare with knowledge distillation and model quantification (PR denotes pruning ratio)

Benchmark	Method	Accuracy	FLOPs(PR)	#Param.(PR)
UCI-HAR	Baseline	96.89%	191.5M	1.31M
	KD	96.03%	97.8M(48.9%)	0.67M(48.9%)
	CC(C=0.5)	**96.64%**	**95.8M(50.0%)**	**0.66M(49.6%)**
	Quantization	94.67%	49.2M(74.3%)	0.34M(74.0%)
	CC(C=0.76)	**95.34%**	**45.9M(76.0%)**	**0.32M(75.6%)**
WISDM	Baseline	98.04%	111.0M	1.04M
	KD	96.91%	57.8M(47.9%)	0.54M(48.1%)
	CC(C=0.5)	**97.59%**	**55.5M(50.0%)**	**0.52M(50.0%)**
	Quantization	96.03%	28.2M(74.6%)	0.27M(74.0%)
	CC(C=0.76)	**96.47%**	**26.6M(76.0%)**	**0.26M(75.0%)**
UniMiB-SHAR	Baseline	76.18%	75.3M	0.99M
	KD	74.72%	42.4M(43.7%)	0.56M(43.4%)
	CC(C=0.5)	**75.77%**	**37.7M(49.9%)**	**0.54M(45.5%)**
	Quantization	73.14%	19.3M(74.4%)	0.25M(74.7%)
	CC(C=0.76)	**74.43%**	**18.0M(76.1%)**	**0.24M(75.8%)**
PAMAP2	Baseline	92.80%	206.9M	1.46M
	KD	91.93%	107.2M(48.2%)	0.82M(43.8%)
	CC(C=0.5)	**92.65%**	**103.2M(50.1%)**	**0.76M(47.9%)**
	Quantization	85.98%	53.3M(74.2%)	0.36M(75.3%)
	CC(C=0.76)	**87.36%**	**49.6M(76.0%)**	**0.36M(75.3%)**

while, compared to 8-bit quantized network [45], our method with $C = 0.76$ also achieves higher FLOPs reduction (76.0% vs. 74.2%) with higher accuracy (87.36% vs. 85.98%) on PAMAP2. Overall, our collaborative compression method has significantly fewer parameters and inference costs than using a single compression operation, which has a great potential to better accelerate activity recognition without sacrificing too much accuracy, showing that the sparsity and low-rankness of model weights could serve as a discriminative property for removing the redundant compression units.

4.4 Ablation Study

There are several crucial hyper-parameters that affect our proposed collaborative compression scheme. In this section, we investigate the effect in more detail via conducting extensive ablation studies on their variants.

4.4.1 *Effect of two main components*

In this part, we analyze the effectiveness of the Global Compression Ratio Optimization (GCRO) in PAMAP2 dataset, as well as the proposed Multi-Step Compression (MSC). The main goal of collaborative compression (CC) is to reduce the number of parameters and FLOPs. To the best of our knowledge, there have been several state-of-the-art works that set a target fraction to uniformly remove the C fraction of compression units from each layer. For fair comparisons, we first run the uniform compression strategy via removing the same C fraction of compression units from each layer in the pre-trained model and retraining the compressed model (i.e., CC w/o GCRO). Besides, we study the effect of the non-uniform compressed strategy obtained by GCRO (i.e., CC). Then we replace above multi-step compression (MSC) and again run the uniform compression strategy via compute the conventional importance metric, i.e., Eq. (6) only once rather than $T = r^l + c^l$ steps (i.e., CC w/o MSC). Fig. 6 presents the results of collaborative compression on PAMAP2 dataset with three variants for different target compression ratios, which shows that there is a considerable performance degradation of the compressed model when removing either GCRO or MSC. It can be seen that MSC has

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3271306

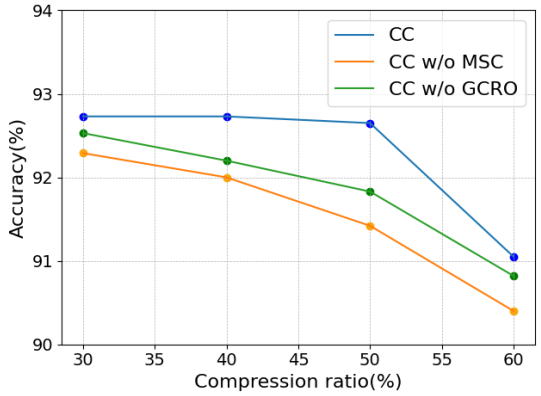


Fig. 6: Independent influence of Compression Ratio Optimization (GCRO) algorithm and Multi-Step Compression (MSC) on PAMAP2.

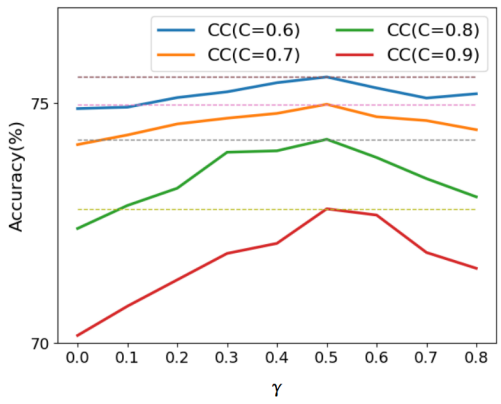


Fig. 7: Influence of different γ on UniMiB-SHAR.

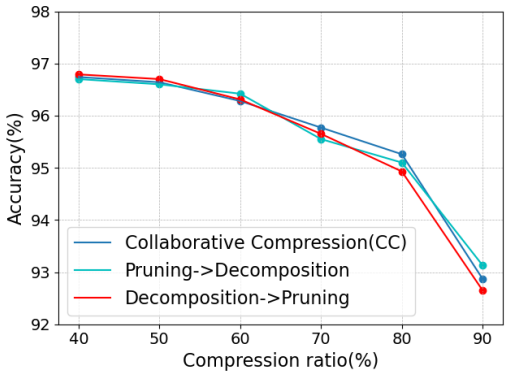


Fig. 8: Compression of different orders of CC on UCI-HAR.

a larger effect than GCRO in our unified collaborative compression scheme, which implies that the multi-step compression strategy is more important than the global compression ratio decision.

4.4.2 *Effect of the trade-off hyper-parameter γ*

We study the impact of the trade-off hyper-parameter γ via setting different global compression ratios on UniMiB-SHAR dataset with different γ. Here, a larger γ suggests that one would put more emphasis on the average information loss, which is able to more fully consider mutual interference between channel pruning and tensor decomposition (Also see Eq. (10)). From Fig. 7, it can be seen that the performance of the compressed model first improves and then gradually decreases with increasing γ. It is worth noting that exploiting the improved importance metric could better contribute to performance of the compressed model when γ is larger than 0, which strongly supports our motivation to fully consider mutual interference between two compression operations. On the whole, since all compression ratios achieve an acceptable accuracy drop when γ is set to 0.5, we use it as an initialization value of γ in all our experiments.

4.4.3 *Impact of the compression order*

As mentioned before, one is able to obtain the architecture of the compressed network after iteratively removing the t_1 channels and t_2 singular values during collaborative compression. We further study the impact of different compression orders with regard to the performance of the compressed network by comparing our method with its two variants. The first variant, i.e., "Tensor Decomposition→Channel Pruning" means that one first decomposes the network and then prunes the decomposed network, while the second variant, i.e., "Channel Pruning →Tensor Decomposition" represents a reverse order. As shown in Fig. 8, it can be easily found that the compression order has a negligible effect on final classification performance. The main reason is that the two variants could yield the same compressed network architecture as our collaborative compression, which use the same t_1 and t_2 , but with only difference between compression order. That is to say, the compressed network architecture is more dominant than the compressed weights in the collaborative compression scheme.

4.4.4 *Compression percentage analysis*

Under different target compression ratios, Fig. 9 illustrates the per-layer compression ratio from bottom to top layers generated by the global compression ratio decision method on PAMAP2 dataset. We further measure the proportion of removed compression units within each layer caused by channel pruning and tensor decomposition respectively, as shown in Fig. 10. Overall, the first and second layers contribute more than the third and fourth layers. For the same compression ratio, it can be seen that the first convolution layer in the compressed model pays more attention to generate sparse weights rather than low-rank weights, which reveal that the diversity of input features is more important for achieving the desired compression objective. Instead, the final layer with the different compression ratios generally tends to generate more compact features.

4.4.5 *Confusion matrices*

We further show the classification performance of the collaborative compression method via computing the confusion matrices on PAMAP2 dataset at different compression ratios. In fact, it is easy to quantitively inspect the confusion matrices for wrong predictions. The results clearly indicate that there is total 25 activity samples labeled as 'walking' that has been wrongly predicted as 'rope jumping' by the pre-trained baseline model because of their very similar signal waveforms. Increasing the compression ratio does not yield a drastic decrease in the value. For example, the collaborative compression scheme only leads to overall 27 and 29 samples of wrong predictions at and without a considerable decrease in prediction performance. This verifies

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3271306

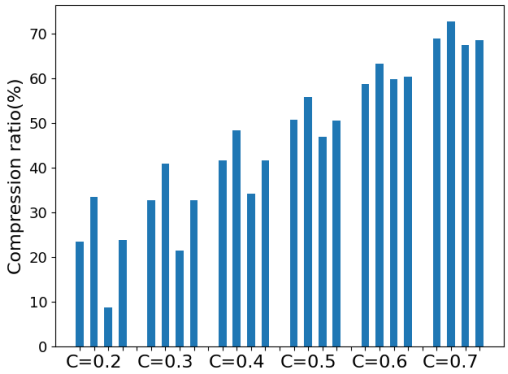


Fig. 9: Compression ratios of different layers.

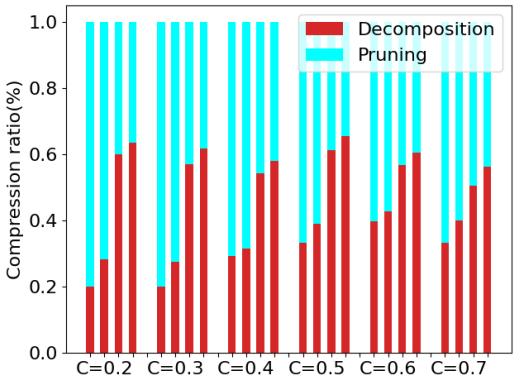
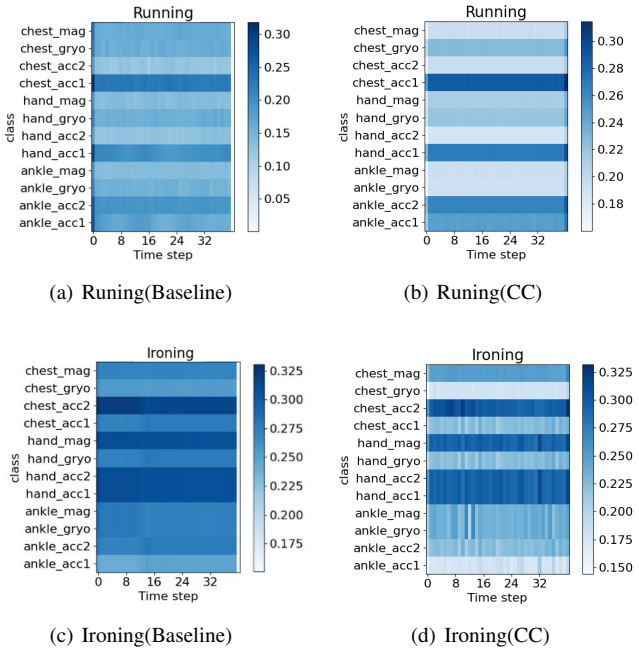


Fig. 10: Compression ratios of different compression operations in different layers of PAMAP2.

that our collaborative compression method is able to significantly decrease the computational overhead at the cost of lower accuracy drop.

4.4.6 *Visualization of feature maps*

To figure out to what extent the proposed collaborative compression method benefits actual activity inference, we conduct an additional visualizing analysis with regard to PAMAP2 dataset to analyze the effect of different sensor modalities attached to different body positions. We visualize the feature maps generated by the second convolutional layer (i.e., conv2) from the compressed network. The results are shown in Fig. 11. We compare the feature maps of the 'Runing' and 'Ironing' activity according to different sensor modalities in Fig. 11, where both x and y axis respectively denote different sensor modalities and time steps that are down-sampled via pooling operation. Deeper color represents larger value. After the collaborative compression, some feature values might become smaller (brighter). It can be seen that the feature maps of the compressed model (See Fig. 11(b) and Fig. 11(d)) are still informative compared with those of the pre-trained ones (See Fig. 11(a) and Fig. 11(c)). The pre-trained model treats all the sensor modalities more equally, but places higher emphasis on the ankle sensor (Acc1, Acc2) and the arm sensor (Acc1) after collaborative compression. This observation is reasonable and in well line with common intuition, which proves that the proposed collaborative compression could remove redundant or useless information and at the same time maintains a strong discriminative ability for activity inference.



(a) Runing(Baseline) (b) Runing(CC)

(c) Ironing(Baseline) (d) Ironing(CC)

Fig. 11: The effectiveness of *Collaborative Compression*.

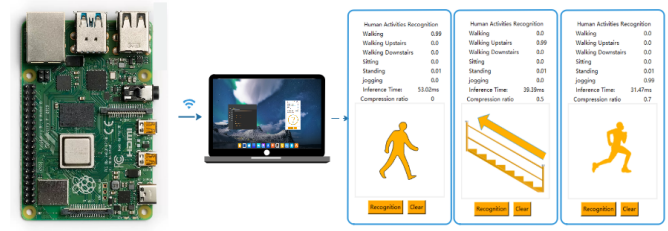


Fig. 12: Real-time HAR system on Raspberry Pi 4 B+.

4.4.7 *Runtime Analysis*

FLOPs has always been a popular performance metric commonly used to compare the computational cost, which is simple to compute and could be done statically. However, such indirect performance metric is somewhat irrespective of the actual software implementations and target hardware platforms. Therefore, we present a study of runtime analysis in activity inference. Because saving and loading models across different mobile devices is relatively straightforward using PyTorch, we experiment the proposed collaborative compression scheme on WISDM dataset via running PyTorch models on the Raspberry Pi 4 B+ platform. To this end, an application program is developed in Python for activity inference, where its main window of user interface is shown in Fig. 12, which contains a graphic representation and predicted probability of current performed activity, as well as the corresponding runtime. In order to measure actual runtime of activity inference, we implement two main steps: 1) pre-training our CNN model and then implement the collaborative compression with different compression ratios; 2) save and load the compressed networks. Then a timer would automatically start when launching the compressed model to read an activity sample, and then stop once a predicted result is returned. Fig. 13 and Table 5 summarize actual runtime over total 200 activity samples (Fig. 13). One could observe that the native uncompressed runtime is: 53.02 ms per-sample (i.e., per-window), while our actual compressed runtimes

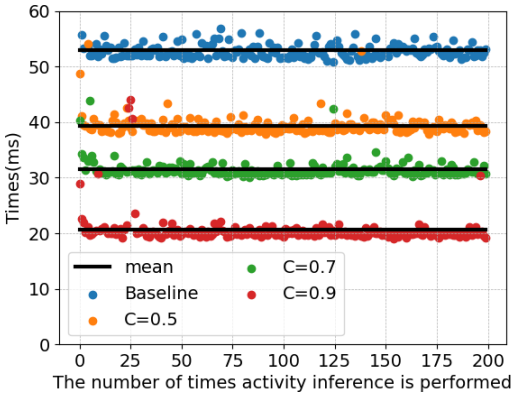


Fig. 13: Inference speed over 200 times.

TABLE 5: Inference Time

Dataset	Method	Inference Time (ms)
WISDM (windows/ms)	Baseline	53.02
	CC(C=0.5)	39.39
	CC(C=0.7)	31.47
	CC(C=0.9)	20.64

are 39.39 ms, 31.47 ms and 20.64 ms respectively at different compression ratios. Due to memory access and other overheads, though theoretical FLOPs reduction does not indicate an actual speedup, our method achieves around 1.34×, 1.68×, and 2.56× acceleration rates, which is considerably faster on the hardware and could better meet the real-time requirement.

5 Conclusion

Although deep neural networks have shown a powerful ability for activity recognition, they are computationally expensive and memory intensive, which impedes their practical usage on resource-limited mobile devices. In this paper, we propose a new collaborative compression scheme that aims at decreasing the runtime and memory footprint for HAR with tolerable accuracy drop, which is achieved by simultaneously enforcing the channel-level sparsity and low-rankness in a simple but effective way. Specifically, the proposed method mainly consists of two stages: Global Compression Ratio Decision Optimization and Multi-Step Collaborative Compression, where the former is in charge of deciding the compression ratio of each layer via solving a simple optimization problem according to compression sensitivity measurement, while the latter is in charge of independently compressing each layer via iteratively removing the least important compression units one-by-one until the target compression is achieved. Extensive experiment analyses on various HAR benchmarks show obvious advantages of collaborative compression over a single compression operation, which would be useful for further understanding and compressing the network architectures for accelerating activity inference.

During recent years, sensor-based HAR has gained a lot of attention due to its wide range of applications in activity monitoring, healthcare, etc. The prominent drawback of deep learning-based HAR lies in its resource-intensive nature, which renders practical HAR deployment onto resource-limited devices often infeasible. Such on-device HAR requires lightweight models so as to be compatible with the low-power computational platform and tight memory requirement in fitness-tracker-like wearable devices [4]. To the best of our knowledge, different compression strategies could provide an effective solution for image classification problem based on deep learning. However, the usage of these combined techniques for HAR has been rarely explored. Though there are a series of works showing the theoretical effectiveness of deep learning for HAR in terms of recognition accuracy Yang et al. [7], Zeng et al. [10], and Ronao et al. [6], these approaches have never been actually deployed on-device, because of their limited memory requirements or large number of operations for realtime activity inference. For this reason, the development of resource-efficient HAR powered by deep learning still remains an interesting and challenging problem in the research field [46]. To bridge the gap between on-device HAR and deep learning, this paper explores a new set of optimizations for model compression targeted at on-device HAR. Different from most existing works [34], we propose to tackle above challenge with compact one-dimensional (1D) Convolutional Neural Networks (CNNs), which analyzes different orthogonal strategies (i.e., channel pruning and tensor decomposition) to remove redundant and irrelevant compression units based on input sensor data, thus resulting in a more lightweight solution for on-device HAR. While our approach concentrates on collaborative compression that enables realtime activity prediction with low inference cost and simple software implementation, it could potentially be combined with other compression strategies for pursuing higher computational efficiency at inference stage. In particular, beside theoretical efficiency, actual on-device HAR implementation is evaluated. In a future study, we will incorporate the per-layer compression ratio into a more delicate global optimization, and combine our method with other compression strategies to further reduce the and runtime and memory footprint for online activity inference.

Acknowledgments

The work was supported in part by the National Nature Science Foundation of China under Grant 61962061 and the Industry-Academia Cooperation Innovation Fund Projection of Jiangsu Province under Grant BY2016001-02, and in part by the Natural Science Foundation of Jiangsu Province under grant BK20191371. Lei Zhang is the corresponding author.

References

[1] A. Gupta, H. P. Gupta, B. Biswas, and T. Dutta, "A fault-tolerant early classification approach for human activities using multivariate time series," *IEEE Transactions on Mobile Computing*, vol. 20, no. 5, pp. 1747–1760, 2020.

[2] Z. Chen, C. Cai, T. Zheng, J. Luo, J. Xiong, and X. Wang, "Rf-based human activity recognition using signal adapted convolutional neural network," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2021.

[3] D. Zhang, Z. Liao, W. Xie, X. Wu, H. Xie, J. Xiao, and L. Jiang, "Fine-grained and real-time gesture recognition by using imu sensors," *IEEE Transactions on Mobile Computing*, 2021.

[4] A. Bulling, U. Blanke, and B. Schiele, "A tutorial on human activity recognition using body-worn inertial sensors," *ACM Computing Surveys (CSUR)*, vol. 46, no. 3, pp. 1–33, 2014.

[5] P. Rashidi and D. J. Cook, "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*, vol. 39, no. 5, pp. 949–959, 2009.

[6] C. A. Ronao and S.-B. Cho, "Human activity recognition with smartphone sensors using deep learning neural networks," *Expert systems with applications*, vol. 59, pp. 235–244, 2016.

[7] J. Yang, M. N. Nguyen, P. P. San, X. L. Li, and S. Krishnaswamy, "Deep convolutional neural networks on multichannel time series for human activity recognition," in *Twenty-fourth international joint conference on artificial intelligence*, 2015.

[8] C. Han, L. Zhang, Y. Tang, S. Xu, F. Min, H. Wu, and A. Song, "Understanding and improving channel attention for human activity recognition by temporal-aware and modality-aware embedding," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–12, 2022.

[9] O. D. Lara and M. A. Labrador, "A survey on human activity recognition using wearable sensors," *IEEE communications surveys & tutorials*, vol. 15, no. 3, pp. 1192–1209, 2012.

[10] M. Zeng, H. Gao, T. Yu, O. J. Mengshoel, H. Langseth, I. Lane, and X. Liu, "Understanding and improving recurrent networks for human activity recognition by continuous attention," in *Proceedings of the 2018 ACM international symposium on wearable computers*, 2018, pp. 56–63.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[12] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artificial intelligence review*, vol. 53, no. 8, pp. 5455–5516, 2020.

[13] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," in *Low-Power Computer Vision*. Chapman and Hall/CRC, pp. 291–326.

[14] J. Yang, X. Shen, J. Xing, X. Tian, H. Li, B. Deng, J. Huang, and X.-s. Hua, "Quantization networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7308–7316.

[15] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *International Journal of Computer Vision*, vol. 129, no. 6, pp. 1789–1819, 2021.

[16] W. Park, D. Kim, Y. Lu, and M. Cho, "Relational knowledge distillation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3967–3976.

[17] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, "Importance estimation for neural network pruning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 264–11 272.

[18] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2736–2744.

[19] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

[20] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3551–3582, 2017.

[21] A. Dubey, M. Chatterjee, and N. Ahuja, "Coreset-based neural network compression," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 454–470.

[22] Y. Li, S. Gu, C. Mayer, L. V. Gool, and R. Timofte, "Group sparsity: The hinge between filter pruning and decomposition for network compression," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 8018–8027.

[23] X. Yu, T. Liu, X. Wang, and D. Tao, "On compressing deep models by low rank and sparse decomposition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7370–7379.

[24] Z. Chen, L. Zhang, Z. Cao, and J. Guo, "Distilling the knowledge from handcrafted features for human activity recognition," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4334–4342, 2018.

[25] A. Ignatov, "Real-time human activity recognition from accelerometer data using convolutional neural networks," *Applied Soft Computing*, vol. 62, pp. 915–922, 2018.

[26] L. Deng, G. Li, S. Han, L. Shi, and Y. Xie, "Model compression and hardware acceleration for neural networks: A comprehensive survey," *Proceedings of the IEEE*, vol. 108, no. 4, pp. 485–532, 2020.

[27] X. Ding, X. Zhou, Y. Guo, J. Han, J. Liu *et al.*, "Global sparse momentum sgd for pruning very deep neural networks," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[28] L. Liebenwein, A. Maalouf, D. Feldman, and D. Rus, "Compressing neural networks: Towards determining the optimal layer-wise decomposition," *Advances in Neural Information Processing Systems*, vol. 34, pp. 5328–5344, 2021.

[29] X. Dong, S. Chen, and S. Pan, "Learning to prune deep neural networks via layer-wise optimal brain surgeon," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[30] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," *Advances in neural information processing systems*, vol. 29, 2016.

[31] J.-H. Luo, J. Wu, and W. Lin, "Thinet: A filter level pruning method for deep neural network compression," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5058–5066.

[32] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *International Con- ference on Learning Representations (ICLR)*, 2016.

[33] X. Zhang, J. Zou, K. He, and J. Sun, "Accelerating very deep convolutional networks for classification and detection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 10, pp. 1943–1955, 2015.

[34] Y. Li, S. Lin, J. Liu, Q. Ye, M. Wang, F. Chao, F. Yang, J. Ma, Q. Tian, and R. Ji, "Towards compact cnns via collaborative compression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 6438–6447.

[35] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," 2016.

[36] S. Lin, R. Ji, Y. Li, C. Deng, and X. Li, "Toward compact convnets via structure-sparsity regularized filter pruning," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 2, pp. 574–588, 2019.

[37] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient

inference," 2016.

[38] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang, "Filter pruning via geometric median for deep convolutional neural networks acceleration," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4340–4349.

[39] M. Lin, R. Ji, Y. Wang, Y. Zhang, B. Zhang, Y. Tian, and L. Shao, "Hrank: Filter pruning using high-rank feature map," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 1529–1538.

[40] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine," in *International workshop on ambient assisted living*. Springer, 2012, pp. 216–223.

[41] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," *ACM SigKDD Explorations Newsletter*, vol. 12, no. 2, pp. 74–82, 2011.

[42] D. Micucci, M. Mobilio, and P. Napoletano, "Unimib shar: A dataset for human activity recognition using acceleration data from smartphones," *Applied Sciences*, vol. 7, no. 10, p. 1101, 2017.

[43] A. Reiss and D. Stricker, "Introducing a new benchmarked dataset for activity monitoring," in *2012 16th international symposium on wearable computers*. IEEE, 2012, pp. 108–109.

[44] 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 *6th international conference on mobile computing, applications and services*. IEEE, 2014, pp. 197–205.

[45] M. Nagel, M. v. Baalen, T. Blankevoort, and M. Welling, "Data-free quantization through weight equalization and bias correction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1325–1334.

[46] K. Chen, D. Zhang, L. Yao, B. Guo, Z. Yu, and Y. Liu, "Deep learning for sensor-based human activity recognition: Overview, challenges, and opportunities," *ACM Computing Surveys (CSUR)*, vol. 54, no. 4, pp. 1–40, 2021.

Junjie Liang received the B.S. degree (2021) from Tianjin University of Technology, Tianjin, China. He is currently pursuing the M.S. degree with Nanjing Normal University.
His research interests include human activity recognition, computer vision, and machine learning.

Lei Zhang received the B.Sc. degree (2001) in computer science from Zhengzhou University, China, and the M.S. degree (2004) in pattern recognition and intelligent system from Chinese Academy of Sciences, China. He received the Ph.D. degree (2007) from Southeast University, China, in 2011. He was a Research Fellow (2008) with IPAM, UCLA, Los Angeles.
He is currently an Associate Professor with the School of Electrical and Automation Engineering, Nanjing Normal University. His research interests include machine learning, human activity recognition and computer vision.

Chaolei Han received the B.S. degree from the Yancheng Institute of Technology, Yancheng, China, in 2020. He is currently pursuing the M.S. degree with Nanjing Normal University, Nanjing, China.
His research interests include activity recognition, computer vision, and machine learning.

Can Bu received the B.S. degree from Huaiyin Institute of Technology, Huaian, China, in 2021. He is currently pursuing the M.S.degree with Nanjing Normal University.
His research interests include activity recognition, federated learning, and machine learning.

Hao Wu received the B.S degree (2001) in computer science from Zhengzhou University, He received M.S degree (2004) and Ph.D. degree (2007) in computer science from Huazhong University of Science and Technology. Now, he is an associate professor at School of Information Science and Engineering, Yunnan University, China. He has published more than 50 papers in peer-reviewed international journals and conferences, such as JASIST, FGCS, J.Supercomputing, KBS and PUC.
He has coauthored a monograph published in World Scientific. He has also served as reviewers and PC members for many venues. His research interests include service computing, information filtering and recommender systems.

Aiguo Song (Senior Member, IEEE) received the Ph.D. degree (1996) in measurement and control from Southeast University, Nanjing. He is currently a Professor with the School of Instrument Science and Engineering, Southeast University.
He is the chair of the China Chapter of the IEEE Robotics and Automation Society. His current research interests include teleoperation, haptic display, the Internet telerobotics, distributed measurement systems and machine learning.