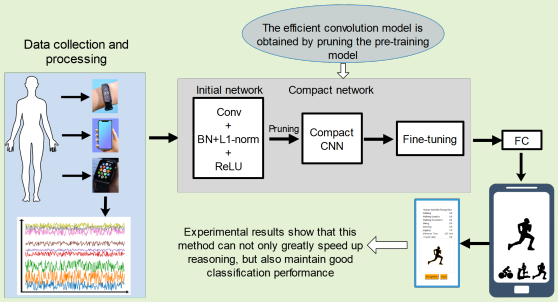
Efficient Convolutional Neural Network for Human Activity Recognition

Junjie Liang, Lei Zhang ✉, Can Bu, Dongzhou Cheng, Hao Wu

***Abstract*— In recent years, deep convolutional neural networks have been widely used in sensor-based human activity recognition (HAR) due to their powerful feature extraction capabilities, but their high computational cost hinders their deployment in real-world applications. In this paper, we propose an energy-efficient network pruning method that we call Efficient Convolutional Neural Network (ECNN) that reduces both model size and memory footprint at runtime, and reduces computational complexity without compromising accuracy. Unlike many existing methods, our approach works directly on modern CNN architectures and does not require specialized deep learning libraries or hardware acceleration. During pre-training, unimportant channels in the CNN are automatically identified and subsequently pruned according to a preset threshold, and finally recovering the network performance by fine-tuning. We demonstrate the effectiveness of our method on several benchmark HAR datasets. For the WISDM dataset, we have reduced the computation cost by 66.97% and the memory consumption by 75.73% while maintaining a fairly high accuracy, and achieved a 4.2× acceleration effect in practical applications.**



***Index Terms*— Sensor, convolution neural networks, human activity recognition, channel pruning, wearable device.**

I. Introduction

In the last 20 years, the Internet of Things and sensor technologies have been greatly developed. Human activity recognition (HAR) [1] [2] [3] based on wearable sensors has become a research hotspot. Complex human activity can be analyzed by sensors such as accelerometers and gyroscopes. Due to its advantages of high accuracy, small size and low cost, sensors have been widely used in various fields of life, such as smart home, motion tracking, medical and health care, etc [4] [5]. Essentially, wearable sensor-based human activity recognition can be addressed as a multi-channel time-series classification problem, which has recently attracted great interest in universal computational scenarios. Traditional machine learning algorithms such as random forest, KNN, and SVM rely heavily on manual feature extraction by manual intervention, which is not only time-consuming, but also largely hinders the classification performance and model generalization of HAR. Many deep learning methods for automated feature extraction have been proposed to solve the above problems. In particular, convolutional neural networks have been widely used for human activity recognition. In the classification tasks of sensor data, convolutional neural networks (CNNs) [7] [8] are more rapid and more efficient than conventional ML methods.

However, while convolutional neural networks have stronger feature representation capabilities, larger CNN require more memory and computational resources, making it difficult to deploy to resource-limited devices, such as mobile phones and smartwatches. As the number of convolutional neural network layers deepens, the parameters and computational costs will also explode, which severely limits the HAR application scenarios. Therefore, in order for HAR to be used more widely in our real life, it is necessary to develop lighter and faster CNN models.

In the last decade, many research work has been devoted to the compression and acceleration of CNN models, proposing many model lightweight methods, such as pruning [9] [10], knowledge distillation [11] [12], tensor decomposition [13] [14], etc. In particular, pruning is the most widely used technique in model compression, which removes the inherent redundancy in the parameters according to different strategies, making the network more sparse. Although the network sparse method proposed by [15] can greatly compress the model and speed up the inference speed, it requires special software or hardware to accelerate it. Such unstructured pruning severely limits the deployment of HAR on wearable devices. In this paper, we propose a simple and effective network compression method. The structure is shown in Figure 1. We convert the extracted human activity features into tensors that can be processed by the convolution neural network, and then send

Junjie Liang, Lei Zhang (E-mail: leizhang@njnu.edu.cn), Can Bu and Dongzhou Cheng 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.

Corresponding author: Lei Zhang

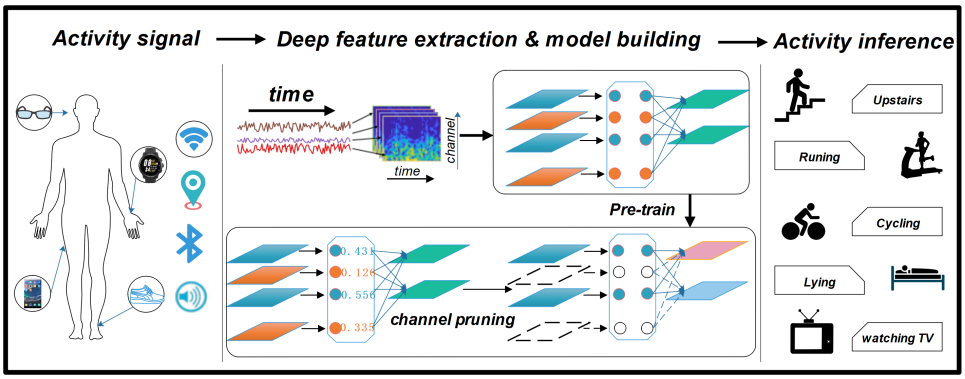


Fig. 1. Application of ECNN in model structure and human activity recognition.

the human activity (sensor) data into the original convolution model for training. During the training, each channel is scored, and then the low scoring channels are pruned. Finally, the network is fine tuned and the network performance is tested. Our method can greatly compress the CNN model while maintaining the classification accuracy of the sensor data, and it does not require a specific software or hardware to accelerate it. Therefore, our compression method is more conducive to the deployment of HAR on wearable devices. Our approach is to add the L1 regularization to the scaling factor in the batch normalization layer, which is therefore easy to implement, and does not require any changes to the original CNN architecture. By adding regularization to the scaling factor of the BN layer, we were able to identify unimportant channels and score each channel. This helped us to crop the channels. The regularization terms we added do not have much effect on the classification accuracy of the sensor data. In fact, sometimes proper regularization can produce higher classification accuracy. We will discuss different regularization values in the following experiments. Pruning channels may reduce classification performance, but we can restore performance through fine tuning. Our main contributions are as follows:

1) This paper proposes a structured pruning method based on L1 regularization, which aims to use sensor data for fast and efficient human activity recognition.
2) Our method accurately scores the importance of each channel in each layer, and then pruning the channel according to the importance, so our method can truly and effectively pruned redundant channels, and maintain good classification accuracy.
3) Our method does not destroy the original convolution structure. And through extensive experiments on HAR dataset, it is proved that our method can perform classification tasks more quickly and accurately, and does not require special deep learning library or hardware.

II. Related Work

In recent years, the deep CNN model has been widely used in sensor based human activity recognition. Zeng et al. [18], Ronao et al. [3] and Yang et al. [19] first used the CNN model that can automatically extract features for human activity recognition. Although the CNN model can achieve higher classification accuracy, it also requires more computing resources. The early CNN model like AlexNet [7] consists of five convolutional layers and three full connection layers. It has 61M parameters and 249M memory usage, and requires 1.5B of computation (FLOPs). Therefore, it is very important to solve the problem of high computing cost caused by deep CNN model for fast and reliable human activity recognition on wearable devices. In recent years, there are a lot of literatures on model compression [20] [10] [21], but few people pay attention to human activity recognition based on CNN acceleration. In particular, according to the existing literature, few people apply model compression methods such as channel pruning or knowledge distillation to sensor based human activity recognition. In fact, although many CNN models can improve the classification performance of sensor data, excessive memory consumption and floating point operations still limit the deployment of HAR on small devices such as smart watches. Therefore, in order to make the application of HAR closer to life, the lightweight model is essential. Below we will introduce several traditional model lightweight methods.

Tensor decomposition: Singular value decomposition (SVD) [13] is the most commonly used in tensor decomposition. This technique uses a low rank matrix to approximate the weight matrix in the neural network. This method is particularly effective on the full connection layer. However, the floating point operations in the CNN model mainly come from the convolution layer, so this method is not applicable to our convolution model.

Pruning: Pruning is the most commonly used method in CNN model compression. Pruning can also be divided into structured pruning and unstructured pruning. Although unstructured pruning can achieve greater compression rate, it needs special deep learning library or hardware to accelerate it. Therefore, structural pruning [23] is most commonly used in practical applications.

Unstructured pruning: Unstructured pruning [15] [24] removes unimportant neurons, and accordingly, the connection

between the pruned neurons and other neurons will be ignored during calculation. Because the model after pruning is usually sparse and destroys the structure of the original model, this kind of method is called unstructured pruning. Unstructured pruning can greatly reduce the parameter amount and theoretical calculation amount of the model, but the calculation method of the existing hardware architecture cannot speed it up, so the actual running speed cannot be improved, and specific hardware needs to be designed to speed it up. Therefore, unstructured pruning is not very friendly to the deployment of HAR model on wearable devices.

Structured pruning: As mentioned earlier, because unstructured pruning is incompatible with off the shelf hardware, structured pruning [10] [25] is the first choice in practical applications. Structural pruning usually takes filters or the entire network layer as the basic unit for pruning. If a filter is pruned, the previous feature map and the next feature map will change accordingly, but the structure of the model is not damaged and can still be accelerated through CPU or other hardware. So in the practical application of HAR, we give priority to structural pruning. [26] imposes sparsity at the neuron level during training, so it can trim some neurons to obtain a compact network. [23] proposed a Structured Sparse Learning (SSL) method for sparse processing of different levels of CNN structures, such as channels, filters or layers. Both of these methods use sparsity regularization to obtain structured sparsity during training. Our method is different. In the training process, we use simple L1 regularization on the scaling factor in the channel direction, which is simply to score each channel in the training process, then rank each channel according to the score, and finally pruning the channel under the control of the preset compression ratio.

III. Propose Method

In this section, we introduce our method, and explain in detail how we use the scaling layer (BN layer) in batch normalization to effectively identify and prune unimportant channels in the CNN model.

A. Setting of the scale factor γ

Our idea is to add a scaling factor γ to each channel during training and multiply the output of the channel. Then we combine the network weight and scaling factor, and sparse regularize the scaling factor. Finally, we prune the channels of small factors and fine tune the compressed model. To sum up, the objective function of our method is:

$$L = \sum_{(x,y)} l(f(x, W), y) + \lambda \sum_{\gamma \in \Gamma} g(\gamma) \tag{1}$$

where (x, y) represents the input and target of training, W represents the trainable weight, the first sum-term is training loss of the original CNN, $g(\sim)$ is the sparsity-induced penalty for the scaling factor, λ balancing both terms. In our experiments, we chose $g(s) = |s|$, which is called the L1-norm, and is widely used to achieve sparsity.

B. Use the scaling factor in the BN layer

Batch normalization layer (BN layer) has become an important part of convolutional neural network [10] [7] [8], just like activation layer and pooling layer. The batch normalization layer can achieve fast convergence and better generalization performance. We design a simple and efficient approach to combine channel scaling factors using the BN normalized activation manner. We assume that t_{in} and t_{out} for the input and output of the BN layer, and that B represents the current mini-batch, then the operation of the BN layer can be expressed as:

$$\bar{t} = \frac{t_{\text{in}} - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}; t_{\text{out}} = \gamma \bar{t} + \beta \tag{2}$$

where σ_B and μ_B are the standard deviation and mean of the input activation on B, γ and β are the trainable affine transformation parameters (scale and displacement) that provide the possibility of converting the normalized activation linear transformation back to any scale.

In the design of the convolution model, we inserted a BN layer behind each convolution layer, and used the channel scaling parameters. We directly use the parameters λ in the BN layer as the scaling factor required for the model compression. It has the advantage of introducing no additional computational overhead to the model.

IV. Experiments

In this section, we evaluate the performance of ECNN on various HAR benchmark datasets based on our experiments. We first select the optimal λ in Eq. 1 through experiments, and the subsequent experiments are conducted on the optimal λ basis. We then compare the performance of different datasets at different compression percentages and demonstrate the effectiveness of ECNN. In addition, we performed a series of ablation studies that further demonstrate the effectiveness of our method on HAR. Finally, we demonstrate the advantages of our method in practical application by measuring the actual activity inference time on the Raspberry Pi 4B plus platform.

A. Benchmark Datasets

We conducted experiments on four popular benchmark HAR datasets, including UCI-HAR, PAMAP2, WISDM, and UniMiB-SHAR. These datasets were collected through different sensors connected to different positions on the body. As shown in Figure 1, we must preprocess the collected data before training the CNN model. For different datasets, there is still no clear consensus on the optimal window size. Too large a window will consume more time for the HAR system, and too small a window may affect the actual classification performance. In fact, the window currently designed usually depends on the size of the window used in previous studies. Therefore, for fair comparison, we still use the window size in the previous literature. Table I shows the details of the preprocessing process.

UCI-HAR: This dataset was established by a research team of the University of California Irvine and its volunteers to test the performance of various machine learning algorithms in HAR tasks [27]. Thirty volunteers aged 19 to 48 were

TABLE I
DATA PREPROCESSING (A=ACCELEROMETER, G=GYROSCOPE, M=MAGNETOMETER)

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

chosen, each with a Samsung Galaxy S2 at their waist. Each volunteer was asked to perform six daily activities, including "lying ", "standing", "walking", "going downstairs" and "going upstairs". The sensor signals were sampled by a gyroscope and a triaxial accelerometer at 50Hz. Following the segmentation method in [36], the entire dataset was randomly split into two parts, 70% for training and 30% for testing.

PAMAP2: This dataset was collected by the German Artificial Intelligence Research Center. Nine volunteers were selected to participate in 18 sports activities, including 12 specific activities (running, standing, walking, etc.) and several optional activities (watching TV, driving, etc.). Each volunteer wore a Colibri wireless inertial measurement (IMU) node on their hands, feet and chest. During the experiment, the data were recorded at a sampling frequency of 100Hz [28].

WISDM: This dataset was collected by the WISDM research team of Fordham University. Twenty nine volunteers were selected [29]. Each volunteer wore an Android phone with a three-axis accelerometer in his pocket and was instructed to perform six daily activities, including "going upstairs", "going downstairs", "jogging", "sitting", "walking" and "standing". The accelerometer collects 20 samples per second. The data set is divided into three parts, 70% of which are used as training sets, 10% as validate sets, and the remaining 20% as test sets.

UniMiB-SHAR: This dataset was collected by 30 volunteers aged between 18 and 60. The 11771 samples in the dataset can be divided into 17 categories, including 8 kinds of daily activities, such as "standing", "sitting", "walking", and 9 types of falls such as "falling forward", "falling rightward", and "falling leftward". Each volunteer wears a Samsung Galaxy Nexus I9250 smartphone in their pocket. The sensor data is sampled by a three-axis BMA220 accelerometer at a frequency of 50Hz. We use five fold cross validation to evaluate the performance of the model on this dataset [30].

B. Experimental details

Our method is implemented using PyTorch. Our model is similar to the convolution structure described in [7] [8], using five convolution layers and a full connection layer. Since our method is to insert a regularization term into the scale factor of BN layer, we insert a BN layer after each convolution layer. All layers are arranged in the following order: C-B-R-C-B-R-P-C-B-R-C-B-R-P-C-B-R-P-AP-FC (C: Conv, B: Batch Normalization, P: Pooling, AP: AdaptiveAvgPool, FC: Full Connection layer). AdaptiveAvgPool is used to solve the

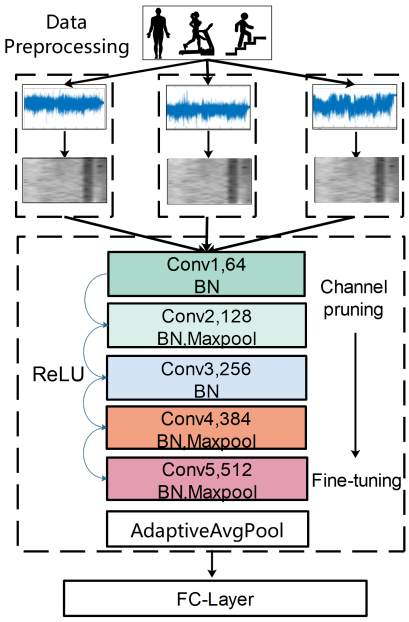


Fig. 2. Brief description for architecture.

problem of input dimension alignment of the full connection layer after pruning. Each convolution layer has a certain number of channels or neurons, as shown in Figure 2. For example, on PAMAP2, the third layer has 128 input channels and 256 output channels, and the filter size is 9×1; On UniMiB-SHAR, the fifth layer has 384 input channels and 512 output channels, and the filter size is 7×1. All networks are trained by random gradient descent (SGD) with momentum of 0.9. Each dataset uses a small batch size of 64 to train 200 epochs. The initial learning rate is set to 0.001, and every 50 epochs are divided by 10. After the training, we get the scaling factor of each channel, and then rank them according to the size of the scaling factor. Before pruning the model, we need to set a threshold to control the pruning percentage. Then the channel is pruned according to the size of the threshold and the scaling factor. Finally, in order to obtain high-performance models, we need to fine tune the pruned models. In the experiment, we can fine tune the pruned model by 20 epoch to achieve good performance.

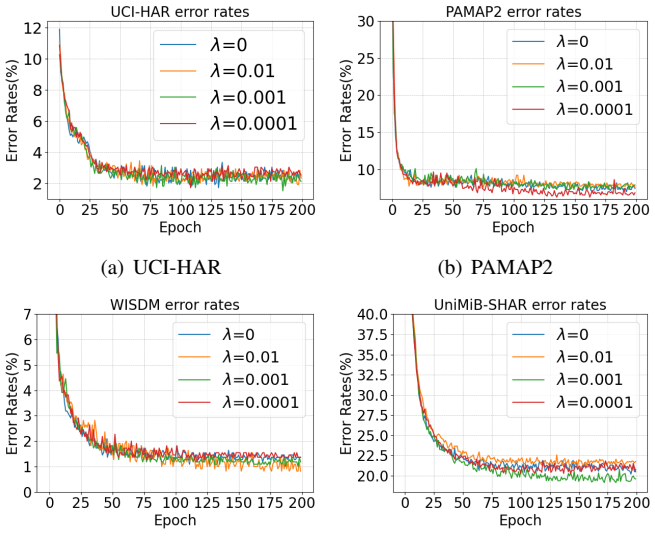
C. Experimental results

1) Effect of the hyper-parameters λ on the performance of the pre-trained model: The λ in Eq. 1 is a very important hyper-

TABLE II
PERFORMANCE OF FOUR BENCHMARK HAR DATASETS AT DIFFERENT PRUNING RATES

Method	Accuracy	↑↓	F1-score	↑↓	Channel	Pruned	FLOPs	Pruned	Parameters	Pruned
				UCI-HAR						
Baseline	97.35%	–	97.43%	–	1344	–	641.32M	–	3.05M	–
ECNN(30% Pruned)	97.51%	0.16%↑	97.63%	0.20%↑	940	30.01%	332.94M	48.09%	1.17M	61.64%
ECNN(50% Pruned)	96.95%	0.40%↓	96.99%	0.44%↓	671	50.07%	207.81M	67.60%	0.53M	82.62%
				PAMAP2						
Baseline	91.56%	–	94.34%	–	1344	–	1.08G	–	3.10M	–
ECNN(30% Pruned)	91.20%	0.36%↓	93.92%	0.42%↓	940	30.01%	449.96M	58.34%	1.07M	65.48%
ECNN(50% Pruned)	90.84%	0.72%↓	93.54%	0.80%↓	671	50.07%	92.76M	91.41%	0.36M	88.39%
				WISDM						
Baseline	98.59%	–	98.49%	–	1344	–	553.73M	–	2.39M	–
ECNN(30% Pruned)	98.68%	0.09%↑	98.60%	0.11%↑	940	30.01%	326.36M	41.06%	1.20M	49.79%
ECNN(50% Pruned)	98.36%	0.23%↓	98.26%	0.23%↓	671	50.07%	182.91M	66.97%	0.58M	75.73%
				UniMiB-SHAR						
Baseline	79.23%	–	79.76%	–	1344	–	422.56M	–	2.41M	–
ECNN(30% Pruned)	78.56%	0.67%↓	79.25%	0.51%↓	1142	15.03%	243.28M	42.43%	1.32M	45.23%
ECNN(50% Pruned)	77.37%	1.86%↓	78.33%	1.43%↓	940	30.01%	95.18M	77.48%	0.45M	81.33%

parameter, which we study on four different datasets. The role of λ is to balance the network loss and the regularization term, and the appropriate regularization can appropriately improve the network performance. Therefore, in order for the network to achieve a better performance, it is necessary to optimize the regularization term. During the experiment, we found that the optimal λ used by the different datasets was different. We used test error rates to measure the effect of different on performance, as shown in Figure 3. For the WISDM dataset, the optimal λ is 0.01; for UCI-HAR and UniMiB-SHAR, the optimal λ is 0.001, and for PAMAP2, the optimal λ is 0.0001. Experiments show that the appropriate regularization can achieve better performance, and after adding the regularization term, the error rate is lower than the baseline model ($\lambda = 0$). The subsequent experiments were carried out on the optimal λ.



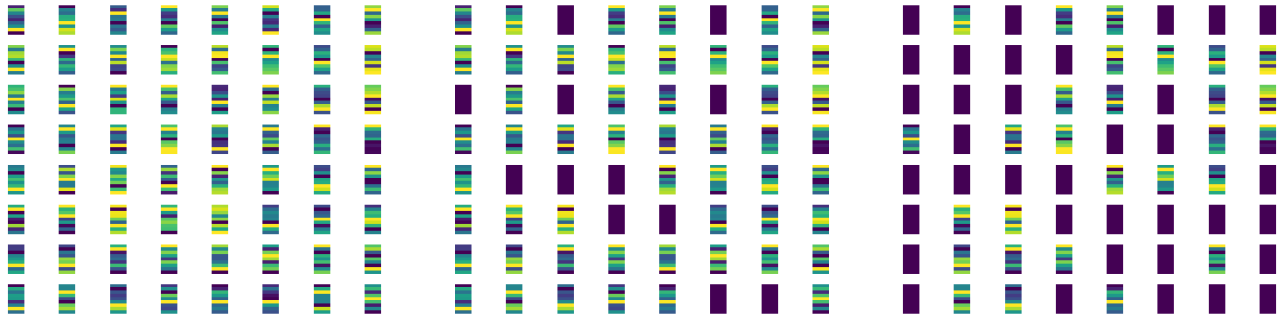
(a) UCI-HAR (b) PAMAP2

(c) WISDM (d) UniMiB-SHAR

Fig. 3. Effect of different λ on performance

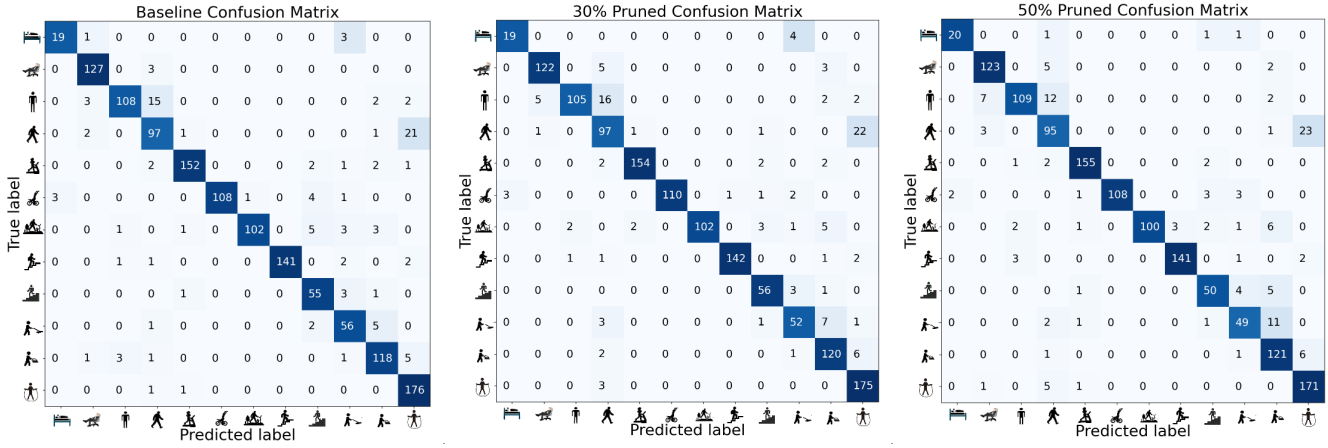
2) Main results: After selecting the optimal λ value and completing the pre-training, we prune the model according to the set threshold. In [9], different layers are pruned according to different proportions, which requires a lot of manual work to set the pruning rate of each layer. In contrast, our method uses a global pruning threshold. The prune threshold is determined by the percentile of all scaling factors, for example, 30% or 50% of the channels are pruned. We compared the performance of different datasets under different compression ratios. It can be seen from Table II that our method can greatly reduce the amount of parameters and calculations. For example, on UCI-HAR, we have reduced 67.60% of FLOPs (207.81M vs. 641.32M) and 82.62% of parameters (0.53M vs. 3.05M), and the accuracy has only decreased by 0.40% (96.95% vs. 97.35%); On PAMAP2, we can achieve greater acceleration and smaller memory consumption, because we have reduced 58.34% of the computing cost (449.96M vs. 1.08G) and 65.48% of the memory consumption (1.07M vs. 3.10M), and only 0.36% (91.20% vs. 91.56%) of the accuracy is reduced; For WISDM dataset, the accuracy rate only decreased 0.23% (98.36% vs 98.59%) when the calculation amount and the parameter amount were reduced by 66.97% (182.91M vs. 553.73M) and 75.73% (0.58M vs. 2.39M) respectively; On the UniMiB-SHAR dataset, our accuracy rate only decreased by 0.67% (78.56% vs. 79.23%), and we reduced 42.43% (243.28M vs. 422.56M) of the computing expenses and 45.23% (1.32M vs. 2.41M) of the memory consumption. Overall, our pruning method has fewer parameters and lower inference cost, which has great potential to accelerate activity identification better without sacrificing too much accuracy.

During the experiment, we further counted the number of channels of the model before and after pruning, and the experimental results are shown in Table II. We found that the pruning percentage of the number of channels differs greatly from the pruning percentage of the amount of computation and parameters. For example, the UCI-HAR pruning the number of channels by 50.07% (671 vs. 1344), but the actual number of parameters decreases by 82.62% (0.53M vs. 3.05M); PAMAP2



(a) Baseline (b) 30% Pruned (c) 50% Pruned

Fig. 4. Channel visualization in first convolution of PAMAP2.



(a) Baseline (b) 30% Pruned (c) 50% Pruned

Fig. 5. Confusion matrix

reduced the number of parameters by 65.48% (1.07M vs. 3.10M) and the amount of computation by 58.34% (449.96M vs. 1.08G), but the number of channels only decreased by 30.01% (940 vs. 1344); For WISDM, the number of channels decreased by 50.07% (671 vs. 1344), the amount of computation and parameters decreased by 66.97% (182.91M vs. 553.73M) and 75.73% (0.58M vs. 2.39M) respectively; UniMiB-SHAR only reduced the number of channels by 15.03% (1142 vs. 1344), while the number of parameters decreased by 45.23% (1.32M vs. 2.41M), and the computation decreased by 42.43% (243.28M vs. 422.56M). We speculate that this is because the number of parameters and channels in each layer is different, and the deep convolution often has more parameters. Therefore, the parameters of the model mainly come from deep convolution. During the experiment, we found that a small amount of channel pruning for deep convolution can reduce most parameters of the model.

3) Channel visualization: Our method is to determine the channel to be pruned by the scaling factor. But in fact, we can't see the channels that have been pruned, and there is no further study on the channels that have been pruned in previous work. In this section, we have visualized the channel, as shown in Figures 4 . We visualized the first convolution of PAMAP2 . The shape of the rectangular box in the figure represents the shape of the convolution kernel. Because we use the long strip convolution kernel, the rectangular box is also a long strip shape; The number of rectangles represents the number of convolution kernel, that is, the number of output channels; Dark rectangles represent pruned channels, while colored rectangles represent reserved channels. We compared the number of channels under two different pruning percentages. Compared with the baseline (Figure 4 (a)), the number of channels pruned will increase with the increase of the compression percentage.

4) Confusion matrix: The confusion matrix in Figure 5 shows the classification performance of PAMAP2 dataset under different pruning rates. The main diagonal line in the figure indicates the number of correctly predicted samples. The darker the color, the more correctly predicted samples. Therefore, from the figure, we can clearly observe which action prediction errors occurred in PAMAP2. For example, 22 activity samples marked as 'walking' in Figure 5 (b) were incorrectly predicted as 'rope jumping'. From Figure 5, we can see that the increase of pruning rate does not lead to a sharp decline in performance. This also shows that our pruning method can significantly reduce memory consumption and computing overhead at the cost of lower accuracy, and greatly save energy consumption.

5) Inference time analysis: FLOPs is a commonly used indicator to evaluate model performance. It is usually used

Fig. 6. Real-time HAR system on Raspberry Pi Model 4B plus

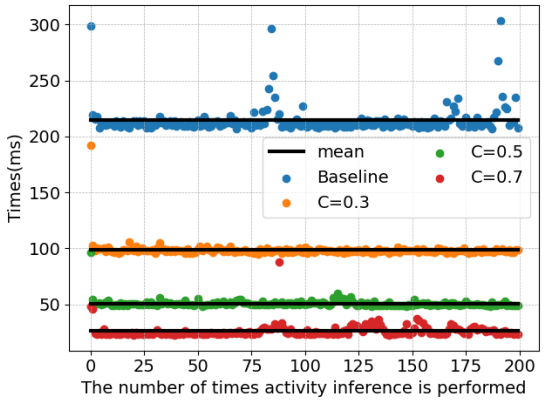


Fig. 7. Inference speed of 200 times

TABLE III
INFERENCE TIME

Dataset	Method	Inference Time(ms)
WISDM (windows/ms)	Baseline	214.4
	ECNN(30% Pruned)	98.4
	ECNN(50% Pruned)	50.7
	ECNN(70% Pruned)	26.0

to refer to computing cost. The lower the FLOPs, the lower the computing cost. Although unstructured pruning can also reduce computing costs, it requires special libraries or hardware to speed it up. Our pruning method is applicable to any hardware or library, and has a wider range of applications. Therefore, our pruning method is more suitable for HAR deployment on mobile devices or wearables. In this section, we further study the inference time of HAR in real hardware. Taking WISDM dataset as an example, we deployed the trained baseline model and pruning model to Raspberry Pi 4B plus platform. Figure 6 is the user interface of an application for activity inference developed in python, which contains the graphical representation and prediction probability of each human action, as well as the inference time under different pruning rates. Figure 7 summarizes the actual running time of 200 active samples under different pruning rates. The blue color represents the actual running time of the baseline model, and the other colors represent the actual running time of our pruning method under different pruning rates. As can be seen from the figure, our method can greatly shorten inference time and save energy consumption. Table III shows the inference time of different pruning rates under one sample. The baseline model needs 214.4ms to process a sample, while our model only needs 98.4ms, 50.7ms and 26.0ms. When the pruning rate is 30%, our pruning model not only improves the performance by 0.09% through fine-tuning, but also realizes the actual acceleration of 2.18×; When the pruning rate is 50%, our model achieves 4.2× acceleration, and only 0.23% accuracy reduction; When the pruning rate is set at 70%, 8.2× acceleration is achieved. Facts have proved that our method can recognize human activities faster than the baseline model, which is very advantageous in practical application. Faster inference time can be better applied to any scenario, and can greatly save energy consumption.

D. Conclusion

Neural network is widely used in various tasks due to its powerful feature representation capability. However, due to its high computing cost and memory consumption, it is difficult to deploy to various small devices, which hinders the practical application of HAR on mobile devices with limited resources. In this paper, we propose a reasonable solution to this problem. The method in this paper is to add sparse regularization to the scaling factor of the Batch Normalization (BN) layer, so that unimportant channels can be automatically identified in the pre-training process, and then pruned, and finally the network performance can be recovered through fine-tuning. Our method achieves higher pruning rate and faster inference time with minimal performance loss.

ACKNOWLEDGMENTS

The work was supported in part by the National Nature Science Foundation of China under Grant 61962061 and in part by the Natural Science Foundation of Jiangsu Province under grant BK20191371. Lei Zhang is the corresponding author.

REFERENCES

[1] E. Ramanujam, T. Perumal, and S. Padmavathi, "Human activity recognition with smartphone and wearable sensors using deep learning techniques: A review," *IEEE Sensors Journal*, vol. 21, no. 12, pp. 13 029–13 040, 2021.

[2] H. Yan, Y. Zhang, Y. Wang, and K. Xu, "Wiact: A passive wifi-based human activity recognition system," *IEEE Sensors Journal*, vol. 20, no. 1, pp. 296–305, 2020.

[3] 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.

[4] N. A. Choudhury, S. Moulik, and D. S. Roy, "Physique-based human activity recognition using ensemble learning and smartphone sensors," *IEEE Sensors Journal*, vol. 21, no. 15, pp. 16 852–16 860, 2021.
[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] P. E. Plsek and T. Greenhalgh, "The challenge of complexity in health care," *Bmj*, vol. 323, no. 7313, pp. 625–628, 2001.
[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.
[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
[9] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," *arXiv preprint arXiv:1608.08710*, 2016.
[10] 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.
[11] 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.
[12] 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.
[13] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.
[14] 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.
[15] S. Changpinyo, M. Sandler, and A. Zhmoginov, "The power of sparsity in convolutional neural networks," *arXiv preprint arXiv:1702.06257*, 2017.
[16] M. Schmidt, G. Fung, and R. Rosales, "Fast optimization methods for l1 regularization: A comparative study and two new approaches," in *European Conference on Machine Learning*. Springer, 2007, pp. 286–297.
[17] M. Schmidt, A. Niculescu-Mizil, K. Murphy *et al.*, "Learning graphical model structure using l1-regularization paths," in *AAAI*, vol. 7, 2007, pp. 1278–1283.
[18] 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.
[19] 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.
[20] 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.
[21] 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.
[22] A. G. Akritas and G. I. Malaschonok, "Applications of singular-value decomposition (svd)," *Mathematics and computers in simulation*, vol. 67, no. 1-2, pp. 15–31, 2004.
[23] 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.
[24] S. Srinivas, A. Subramanya, and R. Venkatesh Babu, "Training sparse neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2017, pp. 138–145.
[25] S. Anwar, K. Hwang, and W. Sung, "Structured pruning of deep convolutional neural networks," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 13, no. 3, pp. 1–18, 2017.
[26] H. Zhou, J. M. Alvarez, and F. Porikli, "Less is more: Towards compact cnns," in *European conference on computer vision*. Springer, 2016, pp. 662–677.
[27] 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.
[28] 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.
[29] 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.
[30] 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.

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.

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.

Dongzhou Cheng received the B.S. degree from the Tianjin University of Technology and Education, Tianjin, China, in 2021. He is currently pursuing the M.S. degree with Nanjing Normal University, Nanjing, China. His research interests include activity recognition 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.