Model Compression in Low Performance Edge Computing

Jingxuan Zhang[1], Yue Li[1(✉)], Jiaxun Wen[2], Kun Tian[1], and Mingze Zhao[1]

[1] Heilongjiang University, Harbin, China
{20213922,20222281,20215321}@s.hlju.edu.cn, 2017021@hlju.edu.cn
[2] Nanjing University, Nanjing, China
211220091@nju.edu.cn

Abstract. Due to the limited memory and computing resources in low performance edge computing, traditional large models are difficult to meet the high demand for latency in computational tasks. This paper builds upon the Pose ResNet18 Body model for human pose estimation, and analyses the hardware limitations of the low performance embedded device Jetson Orin Nano 4G, including limited memory and processing capability, as well as strict requirements for energy consumption and power. Then two model compression methods are proposed to address these limitations: compression based on INT8 quantization and compression based on weight sparsity pruning. Finally, the compressed models are validated on the Jetson Orin Nano 4G platform, and experimental results demonstrate significant advantages in inference time, storage space, power consumption, while maintaining the required prediction accuracy.

Keywords: Edge Computing · Model Compression · Embedded

1 Introduction

With the rapid development of edge computing technology [1, 7], more and more artificial intelligent applications are being deployed on edge devices, which have limited computing resources and storage capacity. In such environments, the deployment and execution of deep learning models face significant challenges [2], as traditional large models require a large amount of memory and computing costs. However, pretrained models running on edge computing devices have relatively lower accuracy requirements. Therefore, model compression techniques are applied to reduce the memory usage and computational complexity of models, thus achieving efficient model deployment and execution on low performance edge devices. In response to these challenges, various innovative model compression techniques have emerged. Weights pruning reduces the size of the model by removing unnecessary weights [3]; weight sharing further compresses the model by utilizing redundancy between weights; quantization converts the weights of the model from floating point numbers to low precision fixed point numbers, thereby reducing storage and computational requirements [4]. The application of model quantization techniques also faces challenges, as suitable quantization strategies are required to balance model size and performance [5].

H.-H. Chen and W. Meng (Eds.): WiSATS 2024, LNICST 606, pp. 140–150, 2025.
https://doi.org/10.1007/978-3-031-86203-8_12

Although research in the field of model compression has been extensive [18], there is less research specifically targeting embedded devices with limited memory and computational resources, and the model compression techniques are often realized at the cost of accuracy loss [6]. Such techniques typically include: lightweight model design [12, 13], Huffman coding [8], hardware accelerator support [15, 16], as well as INT8 quantization and pruning technologies discussed in this paper. In addition to model compression, edge computing and cloud computing in embedded devices are also complementary [14, 19]. Edge computing in embedded devices is often limited by computing power, storage space, and energy consumption, while cloud computing provides powerful computing and storage resources to support large scale model training and inference [17]. Through model compression techniques, deployment on edge devices and coordination with the cloud can be achieved to alleviate the pressure on cloud servers.

This paper focuses on the characteristics of the Jetson Orin Nano 4G embedded platform in terms of memory, processing power, and power consumption. Then based on the human pose estimation model, two model compression algorithms are proposed, including compression model based on INT8 quantization and compression model based on weight sparsity pruning.

2 Platform and Model Analysis

2.1 Platform Performance Analysis and Model Requirements

The model used in this paper is Pose ResNet18 Body, which is a skeleton key point detection model published and open sourced by Nvidia for human pose estimation. It utilizes ResNet18 as a feature extractor, and on this basis performs key point detection and supports multi person recognition. Its dependent ResNet architecture introduces Residual Block and skip connection, which can prevent the gradient vanishing and gradient explosion problems and facilitate the training of deeper networks.

The platform used in this study is the Jetson Orin Nano 4GB embedded platform. The performance parameters of this platform are shown in Table 1.

Table 1. Performance parameters of the platform

Hardware performance	Value
GPU	512core NVIDIA Ampere architecture GPU
GPU Max Frequency	625MHz
AI Performance	20 TOPS
CPU	6core Arm® Cortex®A78AE v8.2 64bit CPU
CPU Max Frequency	1.5GHz
Memory	4GB 64bit LPDDR5 34 GB/s

Its computational performance reflects the foundational performance of current embedded artificial intelligence, and the available memory is relatively small. After

deducting the memory reserved for system operation and the memory particles that cannot be utilized at the system level, only 1GB of space can be reserved for the model. The platform's computational performance can meet the model requirements. However, since the basic memory requirement for the model is 1.4GB, it is challenging for the platform to meet the model requirements, so swapping space must be used, and the model must be made lightweight.

2.2 Compression Methods and Objectives

Currently available methods include hash compression, Huffman coding, quantization, and pruning. Considering the performance limitations of embedded platforms, this study needs to select compression methods suitable for this environment. Hash compression is influenced by the load factor and is difficult to effectively reduce memory usage. While although Huffman coding optimizes memory pressure, it needs longer time to look up the table. Therefore, this study designs lightweight models based on INT8 quantization and weight sparsity pruning, which are suitable for embedded environments.

INT8 quantization can reduce model storage space to one fourth of the original size and is applicable to most embedded devices, while pruning not only reduces storage space but also improves model response speed. This study designs the compressed models based on the following targets: Firstly, shorten the inference time, which refers to the time taken by the model to process each group of incoming information from CSI (Camera Serial Interface) and output the corresponding inference results; Secondly, compress the memory and storage usage; Thirdly, reduce the power consumption; Finally, maintain the prediction precision of the test platform.

3 Lightweight Model Based on INT8 Quantization

The basic principle of quantization is to convert model weights from floating point format to fixed point or low bit width integer format. This reduces the precision of weight representation, thereby reducing the storage space required by the model.

Table 2. Dynamic ranges and minimum positive values for FP32, FP16, and INT8.

Quantization types	Dynamic range	Minimum positive value
FP32	$-3.4 \times 10^{38} \sim +3.4 \times 10^{38}$	1.4×10^{-45}
FP16	$-65504 \sim +65504$	5.96×10^{-8}
INT8	$-128 \sim +127$	1

The precision and data volume decrease in the order are shown in Table 2, with FP32 and FP16 being close in precision, while INT8 quantization offers the best compression effect. The quantization process can be formalized with the following formula: Let F represent floating point numbers, I represent integers, s represent scale, and p represent

zero point. Equations (1) and (2) respectively represent the quantization and dequantization processes shown in Fig. 1, where Eq. (1) deals with floating point data, Eq. (2) with integer data.

$$F = s(I - p) \tag{1}$$

$$I = round\left(\frac{F}{s} + p\right) \tag{2}$$

And Eqs. (3), (4) and (5) represents the rounding function, scale and zero point in Eqs. (1) and (2).

$$round\,(x) = \lfloor x + 0.5 \rfloor \tag{3}$$

$$s = \frac{F(\max) - F(\min)}{I(\max) - I(\min)} \tag{4}$$

$$p = round\,(I(\max) - \frac{F(\max)}{s}) \tag{5}$$

The rounding function shown in Eq. (3) can help control and manage the errors in the calculation, so that the floating point numbers are classified into closer integers during quantization. Equation (4) corresponds to the space ratio between floating point numbers and integers, INT8 quantization essentially maps floating point numbers into an integer space to reduce computer storage and computational complexity, and Eq. (5) corresponds to the position of the zero point of floating point numbers in the integer.

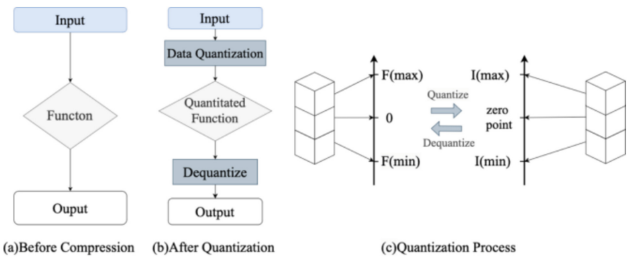


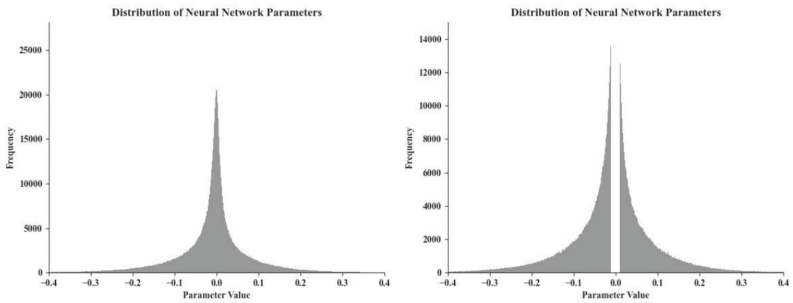
Fig. 1. The principle of quantization

Figure 1 shows the process of Data Quantization and its role in the system before and after compression. The figure includes three parts: **(a) Before compression**, Input: receives raw data input. Function: This performs processing on the input data. Output: The output data after processing. **(b) After quantization**, Input: receives raw data input. Data Quantization: The input data is quantized, and quantization converts floating point data into integer data. Quantitated function: Processes quantized data. Dequantize: This returns the quantized data to floating point numbers. Output: The output data after processing. **(c) Quantization process**, F(max) & F(min): maximum and minimum value of floating point data. I(max) & I(min): Maximum and minimum values for integer data

zero point: This is the pivot point of integer data; that is, the integer value corresponding to a floating point value of zero. It is obvious that INT8 quantization has advantages in reducing memory and storage pressure, and leads to a decrease in model precision inevitably. Although the zero point obtained from Eq. (5) does not affect the zero point after the dequantization process, Eq. (2) and Eq. (3) indicate that the INT8 quantization process exists rounding, which introduces precision loss. And the negative impact of increased computation caused by quantization makes the stability of reasoning time worse, so INT8 quantization is not suitable for environments with real time requirements, such as the control field. Hence, this paper fixes the model to operate with 30 frames as one computational cycle during runtime. It ensures that one cycle is outputted within one second, with each frame's computation time corresponding to a different frame rate. Within one computational cycle, only 5 frames are involved in pose estimation and person tracking, while the rest are assigned random values to introduce interference. This is done to simulate the need for filtering to eliminate interference from the system (a more common scenario). Accurate sampling and analysis of external environment is very important for the application of automatic control and other aspects [10, 11]. Moreover, due to the extended inference time, although quantization can compress peak power consumption, overall energy consumption may be higher compared to pruning or even the original model.

4 Lightweight Model Based on Weight Sparsity Pruning

Besides quantization, another proposed compressed model is based on pruning. The basic principle of pruning is to reduce the number of weights and the amount of calculation of the model by removing redundant connections and weights in the model. Pruning is usually applied to the trained model.



(a) Before Pruning (b) After Pruning

Fig. 2. Weights distribution of Pose ResNet18 Body model before and after pruning

According to the distribution of model weights of the Pose ResNet18 Body shown in Fig. 2(a), like other neural networks, it exhibits weight sparsity [9]. It is observed that most of the weights in the network are close to zero, indicating the presence of many redundant weights in the neural network. As shown in Fig. 2(b), weights and connections

with extremely low absolute weights were removed during pruning. Such process results in the loss of some features and information of the model.

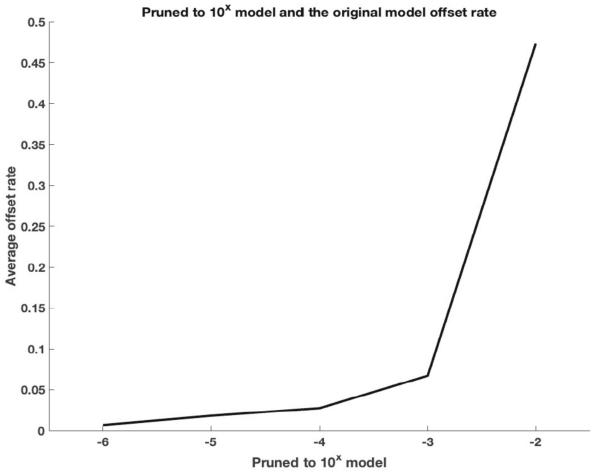


Fig. 3. Offset rates with different pruning thresholds

But a significant reduction in the number of weights can be observed. In this paper, weights are pruned under different thresholds 10^x, where the absolute values of weights that are smaller than 10^x will be pruned. Figure 3 shows the relationship between the model offset rate and the pruning threshold. The offset rate is defined by the following equation.

$$r = \frac{1}{n} \sum\nolimits_{S\alpha} \frac{a-b}{a} \tag{6}$$

In Eq. (6), r represents the average offset rate, n is the number of output points that are counted in the offset rate, and $S\alpha$ is the corresponding output points set. The range of $S\alpha$ is less than -1.4×10^{-45} and greater than 3.4×10^{-38}. In the output matrix, b and a are the values before and after pruning corresponding to the same element in two model outputs. After removing weights smaller than 10^{-2}, the model offset rate rises sharply, prompting this paper to prune weights using a threshold of 10^{-3}. Pruning can only reduce model size and memory usage to a limited extent, but it can improves inference time by saving more compute nodes in larger models.

5 Experimental Result

5.1 Experimental Platform

A four wheeled platform is used for model testing in this paper, where a camera is used to capture human orientation and a PID algorithm is employed to enable the model to recognize the human body and keep it at the center of the camera's field of view. The

tracking principle of the embedded platform in this paper is that when the PID value is greater than 0.1, it rotates and adjusts the angle of the vehicle in place to make it move toward the human body and then advance to the position 30cm in front of the human body. The platform, by leveraging the characteristics of the PID algorithm, demonstrates the critical importance of sampling and output for controlling devices.

In the following, we will assess the performances of the two proposed compressed models in the perspectives of inference time, memory and storage usage, power consumption and prediction precision (Fig. 4).

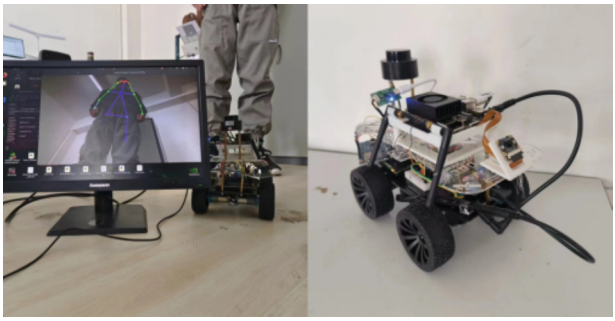


Fig. 4. Jetson Orin Nano 4GB Embedded Platform

5.2 Inference Time

In the assessment of inference time, this paper adopts the frame rate corresponding to each frame of the image as a metric. Since 30 images per second are obtained from the outside world, this means that even if the model processes an image in advance in a single frame time corresponding to 30FPS, it will have to wait for the end of this 30FPS frame time before the next image input is processed. In this way, the frame rate corresponding to a single frame image can reflect the speed of the model inference (although the output is still 30FPS, if the inference speed is slow, the frame processing time corresponding to a picture is longer, and the corresponding frame rate will be lower, and vice versa).

From Fig. 5 and Fig. 6, it can be observed that INT8 quantization has a negative impact on the frame rate, with an average decrease of 7.5%. Pruning operations have shown a positive effect, with an average increase in frame rate of 10.36%. Figure 6 effectively distinguishes the image that did not participate in the pose estimation. Because they have a higher frame rate without being processed by the model, which corresponds to the data near 160 FPS in Fig. 6. It can also be observed in Fig. 6 that the data near 160 FPS is more clustered after pruning, and the frame rate distribution of the model after pruning is also more concentrated in other areas. The clustered feature, which is also mentioned as inference time stability, makes it easy to distinguish between the activity of recognizing a human or processing interference data, because the frame with higher frame rate means that it is an interference frame, and the frame with lower frame rate means that it is the frame used for human recognition. The stability of inference time

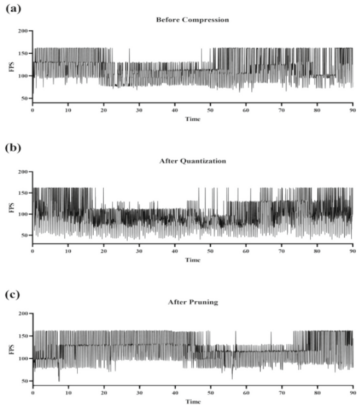


Fig. 5. Effect of two compressed methods on frame rate

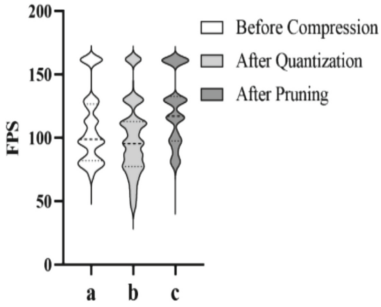


Fig. 6. Box plot of the effect of two compressed methods on the frame rate

with pruning greatly facilitates the use of frame rate to filter out interference data. In contrast, quantization requires the design of a more complex filtering scheme to meet the control requirements.

5.3 Storage and Memory Usage

As indicated in Fig. 7, the memory usage of the model after quantization has been reduced by an average of 46.92% compared to the unprocessed model. In contrast, the memory pressure after pruning has only been reduced by an average decrease of 6.55%. Although pruning can reduce memory usage, its effectiveness is comparatively mild. Figure 8 reveals that the INT8 quantization operation has a very pronounced impact on reducing storage size, reaching about 74.97% reduction, while the impact of pruning on storage size is not remarkable, reaching only 5.4% decrease.

Although theoretically, the compression ratio of a model in terms of storage and memory size should be similar after INT8 quantization, a comparison of Figs. 7 and 8 reveals that this is not the case. This is because that INT8 quantization usually consumes more memory due to additional quantization, dequantization and filter process. This phenomenon is also more intuitively reflected in the inference time.

5.4 Power Consumption

According to Fig. 9, peak power consumption is reduced by 10.3% with INT8 quantization, while it is reduced by 8.7% with pruning. However, this does not mean that quantization contributes more to overall energy consumption than pruning, as the advantage of pruning in inference time might lead to less overall energy consumption. The overall energy consumption is defined in Eq. (7),

$$E = \sum_{0}^{\max(i)} P(i)T(i)[T(i) \to 0] \tag{7}$$

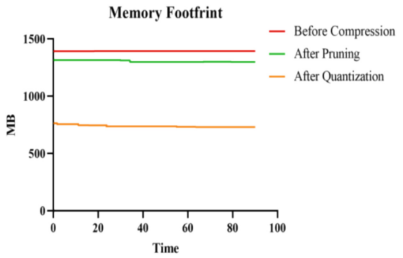


Fig. 7. Effect of two compressed methods on the model memory usage

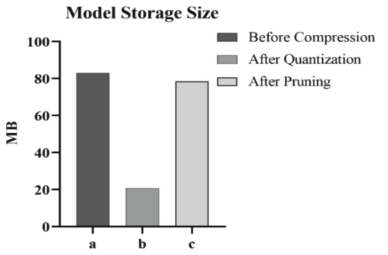


Fig. 8. Effect of two compressed methods on the model storage size

where E represents Energy, P for Energy Consumption, and T for Time. During the experiment, the model's input and output are limited to 30 frames per second, which is less than the platform's processing capacity. This means that quantization and pruning have the same workload. Based on the overall energy consumption calculated using Eq. (7) for processing every 18,000 images on the computing platform, the pruned model consumes 3058.72J, while the INT8 quantized model consumes 3177.05J. It is shown that the difference between the two is not large, and the pruned model has relatively lower total energy consumption.

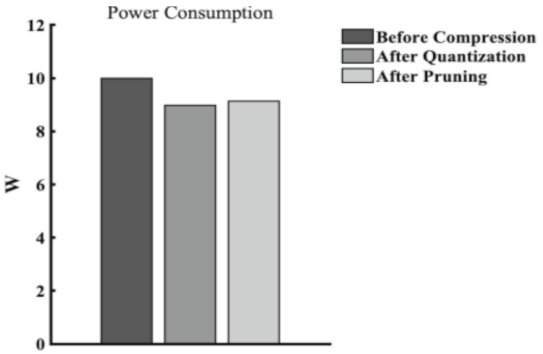


Fig. 9. The effect of two compressed methods on the energy consumption

5.5 Prediction Accuracy

Average offset rate is used to assess the prediction accuracy changes by the two proposed compressed model, and the larger offset rate indicates the worse accuracy. According to the data from Fig. 10, the model after quantization has an average offset rate of 8.95% compared to the original model, while the average offset rate after pruning is 6.712%. This indicates that pruning has a precision advantage over quantization. Moreover, the prediction precision can affect the tracking time. The traceability principle of the test platform determines that it is sensitive to abnormal data and the accuracy of human

orientation output by the model. Figure 11 clearly demonstrates the advantage of pruning in terms of tracking time. The average time taken to track a human body is reduced by 24.01% after pruning, whereas INT8 quantization results in a 21.55% increase. The experimental results show that although each tracking instance on the test platform is random due to environmental factors, pruning's performance can still be confirmed as relatively superior, this is because the model after quantization required filtering algorithms are more complex and less effective.

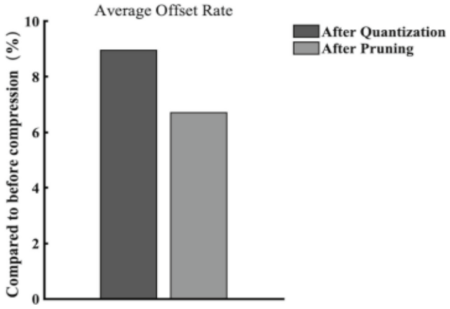


Fig. 10. The effect of two compressed methods offset rate

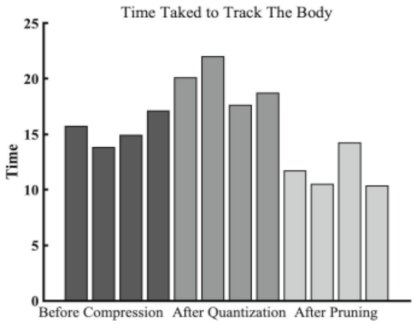


Fig. 11. Time comparison of platform implementation tracking

6 Conclusions

This paper proposes two compressed Pose ResNet18 Body model for embedded platform, and explicit performances are analyzed. In conclusion, INT8 quantization often has a better ability to reduce model memory pressure and storage pressure than pruning. However, it comes with more computational pressure and accuracy loss. It will lead to an increase in the dispersion degree of the inference time, which will adversely affect the automatic control, filtering algorithm and other fields. In contrast, pruning is more computationally efficient and can obtain a more stable output. In terms of the ability to reduce energy consumption, the INT8 quantization operation has lower peak energy consumption, which is more suitable for the model that is sensitive to the total running time but not sensitive to the total amount of computation, while the pruning operation is more suitable for the model that requires the overall amount of computation but is not sensitive to the total running time. On the problem of accuracy reduction, both can have a good performance on the test equipment, and the degree of accuracy reduction is acceptable.

Acknowledgement. This work is supported by the National College Students' Innovation and Entrepreneurship Training Project (202210212129) and Basic Scientific Research Project of Heilongjiang Province (2020-KYYWF-1003); Associate Professor Yue Li from the School of Electronic Engineering of Heilongjiang University provided guidance for this paper. Jingxuan Zhang designed and completed the research of this paper; Mingze Zhao from the College of Journalism and Communications of Heilongjiang University; Jianxun Wen from the College of Computer Science of Nanjing University, Yuan Xu from the College of Life Science of Northeast Agricultural

University, and Kun Tian from the College of Electronic Engineering of Heilongjiang University collected part of the data and draw some charts.

References

1. Cao, K., Liu, Y., Meng, G., Sun, Q.: An overview on edge computing research. IEEE Access **8**, 8571485728 (2020)
2. Canziani, A., Paszke, A., Culurciello, E.: An analysis of deep neural network models for practical applications. arXiv preprint arXiv:1605.07678 (2016)
3. Zhu, M., Gupta, S.: To prune, or not to prune: exploring the efficacy of pruning for model compression. arXiv preprint arXiv:1710.01878 (2017)
4. Gupta, S., Agrawal, A., Gopalakrishnan, K.: Deep learning with limited numerical precision. In: International Conference on Machine Learning, PMLR, vol. 37, pp. 1737–1746 (2015)
5. Han, S., Mao, H., Dally, W. J.: Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. arXiv preprint arXiv:1510.00149 (2015)
6. Cheng, Y., Wang, D., Zhou, P.: A survey of model compression and acceleration for deep neural networks. arXiv preprint arXiv:1710.09282 (2017)
7. Capra, M., Peloso, R., Masera, G.: Edge computing: a survey on the hardware requirements in the internet of things world. Future Internet **11**(4), 100 (2019)
8. Rahman, M.A., Hamada, M.: Lossless text compression using GPT-2 language model and Huffman coding. In: SHS Web of Conferences, EDP Sciences, vol. 102, p. 04013 (2021)
9. Han, S., Pool, J., Tran, J.: Learning both weights and connections for efficient neural network. In: Advances in Neural Information Processing Systems, Montreal, p. 28 (2015)
10. Mehra, R.: On the identification of variances and adaptive Kalman filtering. IEEE Trans. Autom. Control **15**(2), 175–184 (1970)
11. Li, Q., Li, R., Ji, K.: Kalman filter and its application. In: 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS), IEEE, Tianjin, pp. 74–77 (2015)
12. Agarwal, P., Alam, M.: A lightweight deep learning model for human activity recognition on edge devices. Procedia Comput. Sci. **167**, 2364–2373 (2020)
13. Chen, G., Cheng, R., Lin, X.: LMDFS: a lightweight model for detecting forest fire smoke in UAV images based on YOLOv7. Remote Sens. **15**(15), 3790 (2023)
14. Gai, K., Qiu, M., Zhao, H.: Energy aware task assignment for mobile cyber enabled applications in heterogeneous cloud computing. J. Parallel Distrib. Comput. **111**, 126–135 (2018)
15. Deng, L., Li, G., Han, S.: Model compression and hardware acceleration for neural networks: a comprehensive survey. Proc. IEEE **108**(4), 485–532 (2020)
16. Sipola, T., Alatalo, J., Kokkonen, T.: Artificial intelligence in the IoT era: a review of edge AI hardware and software. In: 31st Conference of Open Innovations Association (FRUCT), IEEE, Helsinki, pp. 320–331 (2022)
17. O'Donovan, P., Gallagher, C., Leahy, K.: A comparison of fog and cloud computing cyber-physical interfaces for Industry 4.0 real time embedded machine learning engineering applications. Comput. Ind. **110**, 12–35 (2019)
18. Buciluǎ, C., Caruana, R., Niculescu Mizil, A.: Model compression. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Barcelona, pp. 535–541 (2006)
19. Hallmans, D., Sandström, K., Nolte, T.: Challenges and opportunities when introducing cloud computing into embedded systems. In: 13th International Conference on Industrial Informatics (INDIN), IEEE, Cambridge, pp. 454–459 (2015)