

Comparative Analysis of Lightweight Gesture Recognition Based on Raspberry Pi Devices

Zhangyang Chen

*School of Artificial Intelligence, Ning Bo University of Finance & Economics, Ningbo, China
chenzhangyang1@outlook.com*

Abstract. With the rapid development of edge computing and the Internet of Things. Deploying deep learning models on resource-constrained devices such as Raspberry Pi has become a research hotspot. Gesture recognition is a natural human-computer interaction method. It has broad application prospects in fields such as smart homes and virtual reality. However, balancing real-time performance and accuracy on edge devices such as Raspberry Pi remains a challenge. This paper uses MobileNetV2 and ResNet50 models as research methods. By performing post-training integer quantization processing, an analysis of the balance between model training performance and speed on edge devices such as Raspberry Pi. Experimental results show that MobileNetV2 outperforms ResNet50 in both accuracy and inference speed in the original state. After quantization, the accuracy of MobileNetV2 improved to 94.62%. The inference speed has been increased to 39.63 FPS, meeting the requirements for real-time recognition. After ResNet50 quantization, the accuracy dropped to 63.17%, while the speed only increased to 4.53 FPS. In summary, MobileNetV2 has a greater advantage on edge devices, and quantization technology can significantly improve its deployment performance. This article provides a reference for selecting gesture recognition models for edge devices such as Raspberry Pi.

Keywords: Raspberry Pi device, Gesture recognition, Post-training quantization, Model comparison

1. Introduction

With the rise of the Internet of Things and edge computing. Moving artificial intelligence applications from the cloud to terminal devices has become an important trend [1-3]. Gesture recognition is based on images or videos of a hand obtained by a camera. Deep Learning Models for Automatic Recognition of Gestures. It is a natural and intuitive way to have people communicate with computers, and thus can replace traditional physical buttons, touchscreens or voice commands. It has many applications in smart home management, barrier-free interaction systems and virtual reality, etc. Raspberry Pi is a small single-board computer about the size of a credit card. The equipment has a low cost and low power consumption, so it is suitable for deploying such applications [4, 5]. It has a relatively small amount of computing power and is not suitable for running complex deep learning models directly. The goal of real-time inference at the edge is to retain the accuracy of models.

Lightweight neural networks, such as MobileNet, ResNet and EfficientNet, have been developed recently to reduce the computational cost of these models through depthwise separable convolutions and residual connections. It has achieved good results in the task of mobile image classification [6, 7]. The actual performance of the models on the edge devices is not uniform. Deployment directly will still have issues of memory consumption and inference speed. Pruning, knowledge distillation and quantisation are model compression techniques that have also been used. Among the above methods, Quantization reduces the bit-width of the model's weights from 32 bits to 8 bits and, thereby, substantially lowers both storage and computation costs with a relatively small drop in accuracy. Most of the previous research has focused on improving single models and has not systematically compared several lightweight models for the Raspberry Pi platform, nor has any quantitative analysis been performed to quantify the impact.

Given the above shortcomings, this paper will explore the performance variations of different lightweight models in their original and quantized forms on edge devices, such as Raspberry Pi. Which Model is more suitable for deployment? Two typical models were selected for this study, MobileNetV2 and ResNet50, and their gesture recognition accuracy and inference speed were systematically compared. A dual-metric evaluation method based on accuracy and FPS provides a basis for model selection in practical deployment.

2. Research methods

2.1. Overall process

This experimental procedure is divided into three stages (Figure 1): data preparation, model processing, and quantitative evaluation and comparative analysis. First, in the data preparation stage, the Rock Paper Scissors gesture dataset was loaded, and all images were scaled to 224×224 pixels. Then, normalization processing was performed on MobileNetV2 and ResNet50 to adapt their respective pre-trained weights. The former was mapped to $[-1, 1]$, and the latter was first scaled to $[0, 1]$ and then ImageNet channel normalization was performed. Next, in the model processing stage, the same classification head (global average pooling, Dropout 0.2, and three-class fully connected layer) was added to both models, the convolutional base was frozen, the classification head was trained for 5 epochs, the original model was saved, and its accuracy and inference speed were evaluated. Finally, in the quantitative evaluation and comparative analysis phase, both models were trained with integer quantization, the quantized models were saved, and evaluated again. Evaluate the performance difference between the original model and the quantized model, and analyze the strengths and weaknesses of both models for edge deployment.

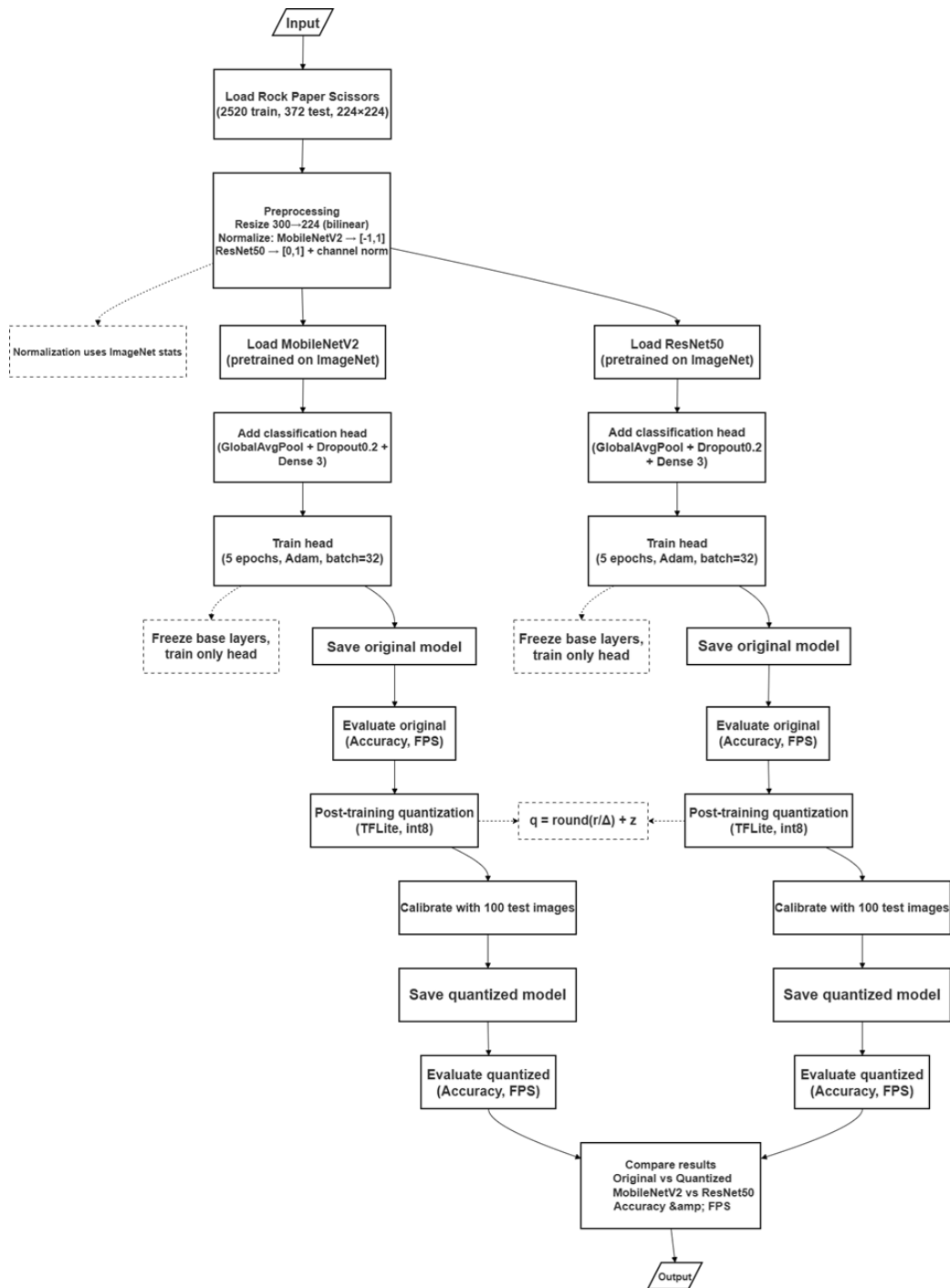


Figure 1. Flowchart (picture credit: original)

2.2. Dataset

The Rock-Paper-Scissors dataset in TensorFlow Datasets is used here. The pictures are from people of different races, ages and genders who made rock-paper-scissors gestures and were labelled. A total of 2520 training images and 372 test images are available at a resolution of 224×224. The data in the experiment were first scaled and normalised through preprocessing, and then all original images of 300×300 pixels were uniformly reduced to 224×224 pixels to meet the input size

requirements of both models. Normalisation for different models is not the same. MobileNetV2 is normalised to the $[-1, 1]$ range, and ResNet50 is normalised to the $[0, 1]$ range.

2.3. Model

MobileNetV2 is a relatively light-weight model for mobile and embedded devices that was developed by Google. Reduce both the computation and memory requirements of the model significantly, maintain a high recognition accuracy, and thus be suitable for real-time operation on resource-constrained edge devices, such as the Raspberry Pi. MobileNetV2 uses depthwise separable convolution for its lightweight realisation and divides a general convolution into two parts. First, it conducts depthwise convolution to spatially convolve each channel of the input independently. Then, it performs pointwise convolution to combine the channels of the depthwise convolution output with a 1×1 kernel. Therefore, the number of parameters and floating-point operations are relatively small. MobileNetV2 is also an inverse residual structure and a linear bottleneck layer that enhance feature extraction efficiency further. Model Size (FP32): Approximately 13.3MB.

ResNet50 is an early, well-known deep residual network that introduced the idea of residual learning [8, 9]. Traditional deep networks are likely to have gradient vanishing or exploding issues as the number of layers increases. ResNet50, on the other hand, adds a skip connection to learn the "residual" of the input and output at the network level, rather than learning the entire output mapping directly. Traditional deep networks are subject to the problems of gradient vanishing or explosion with a large number of layers. However, ResNet50 is used to introduce skip connections; that is, the network layers learn the "residual" between the input and the output rather than the full output mapping directly. Therefore, it is possible to construct deep neural networks with many layers, and at the same time, enhance both their feature learning and classification accuracy significantly. ResNet50 has a bottleneck design; each residual block contains three convolutional layers and is relatively computationally lightweight. Model Size (FP32): approximately 98MB.

2.4. Post-training quantization

TensorFlow Lite with post-training integer quantization is used in this experiment, and 100 images from the test set are selected as the representative dataset for calibration. The final 8-bit integer model has been quantized. After quantisation, the model size is reduced to one-quarter of the original size and integer operations can speed up inference. The Formula for Quantification is as follows:

$$q = \text{round}(r/\Delta) + z \quad (1)$$

r is a floating-point number, Δ is a scaling factor, and z is the zero point. Quantisation has been performed to reduce the model size by about 4 times and accelerate integer operations.

3. Experimental results and analysis

3.1. Experimental setup

The model training environment for this experiment was conducted on the Google Colab platform with 12GB of memory, simulating the low-computing-power environment of edge devices such as Raspberry Pi. The operating system is Linux (Ubuntu), and the Python version is 3.10. Two pre-trained models, MobileNetV2 and ResNet50, were selected. The original classification heads were removed, and global average pooling layers, Dropout (0.2), and three fully connected layers were added. Freeze the convolutional base parameters, train a new classification layer for 5 epochs, use Adam as the optimizer, and set the learning rate to 0.001.

3.2. Comparative experiment

Table 1. Model performance comparison

Model	Version	Accuracy (%)	Inference speed (FPS)
MobileNetV2	original	90.59	3.42
MobileNetV2	Quantification	94.62	39.63
ResNet50	original	70.43	2.30
ResNet50	Quantification	63.17	4.53

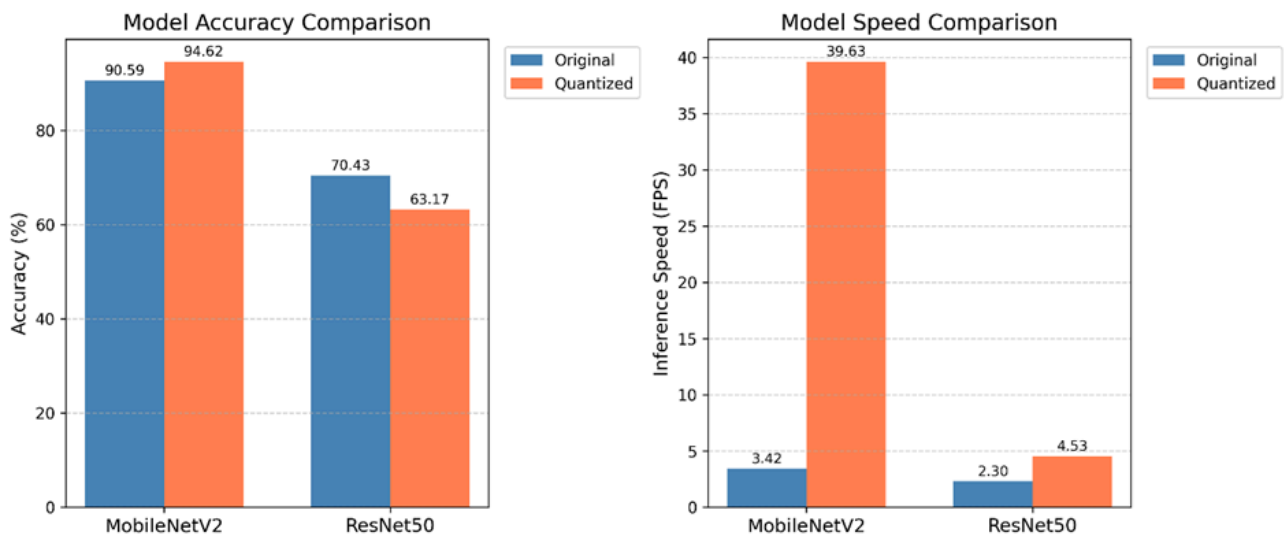


Figure 2. Comparison of model accuracy and inference speed (picture credit: original)

The accuracy and inference speed (FPS) of the original model on the test set are shown in Table 1. MobileNetV2 achieved a significantly higher accuracy (90.59%) than ResNet50 (70.43%), and was also slightly faster (3.42 FPS vs 2.30 FPS), indicating that MobileNetV2 already possesses better potential for edge deployment even without optimization.

After quantization, MobileNetV2's accuracy improved to 94.62%, and its speed increased to 39.63 FPS, achieving an acceleration of approximately 11.6 times and reaching the real-time recognition standard (>30 FPS). The accuracy of ResNet50 dropped to 63.17%, and the speed

increased to 4.53 FPS (approximately doubled), but it still couldn't meet real-time requirements. Quantization has a more significant acceleration effect on MobileNetV2, and the accuracy increases instead of decreasing. Wang Yuyang et al. also observed a significant acceleration effect brought about by quantization in an offline gesture recognition system based on SSDLite-MobileNetV2, which may be due to the regularization effect introduced by quantization [10].

Figure 2 visually illustrates the comparison of accuracy and speed between the two models in the original and quantized versions. After quantization, MobileNetV2 has achieved real-time recognition, while ResNet50 still suffers from significant accuracy loss despite the speed improvement. MobileNetV2 performs better than ResNet50 in its original and quantized states, and is therefore more suitable for gesture recognition on resource-constrained devices such as Raspberry Pi.

4. Conclusions

Compare the gesture recognition results of the two lightweight models, MobileNetV2 and ResNet50, in their original and quantized states, and systematically study the effect of model structure on edge deployment. MobileNetV2 is better than ResNet50 in both accuracy and speed from the experimental results in its original form. Quantisation Methods can speed up the inference process, but the corresponding increase in speed differs significantly among different models. MobileNetV2 can be used for real-time recognition and high-accuracy performance after quantization at the edge. ResNet50 is not much faster or more accurate than others after quantization. Therefore, quantized MobileNetV2 is selected as the gesture recognition model for Raspberry Pi-like edge devices. This experiment also has some limitations and shortcomings, such as only using post-training integer quantization and not exploring quantization-aware training or mixed-precision quantization, which may prevent the full potential of quantization from being realized. The experiment only compared two models, and the dataset only included static gestures, failing to verify generalization ability under dynamic gestures or complex backgrounds. Future will adopt more advanced quantization techniques to explore quantization-aware training, try lower bit quantization, and further compress the model. The significance of this paper lies in its first systematic comparison of the performance differences between depthwise separable convolutions and residual structures before and after quantization, providing empirical evidence for lightweight model design.

References

- [1] Wang, Y., & Zhou, P. (2026). Research on multimodal artificial intelligence perception method in edge computing environment. *Intelligent Internet of Things Technology*, 58(3), 85–88. <https://doi.org/10.26921/j.cnki.2096-6059.2026.03.018>
- [2] Zheng, C., Guo, R., Wang, C., et al. (2026). Design and implementation of intelligent nursing bed based on Raspberry Pi and Arduino. *Internet of Things Technology*, 16(4), 79–83. <https://doi.org/10.16667/j.issn.2095-1302.2026.04.017>
- [3] Li, Z., Li, H., Yuan, X., et al. (2022). Optimization design based on bionic robotic hand gesture recognition. *Light Industry Science and Technology*, 38(4), 34–36.
- [4] Guo, J. (2024). *Research and application of gesture recognition based on deep learning* (Doctoral dissertation, Inner Mongolia University). <https://doi.org/10.27224/d.cnki.gnmdu.2024.000165>
- [5] Zhou, Q., & Zhang, S. (2023). Design of MIDI somatosensory interaction system based on hand posture recognition. *Journal of Hubei University of Technology*, 38(2), 27–32.
- [6] Gong, X., Zhang, D., & Xie, S. (2024). Gesture recognition based on lightweight neural network in virtual reality. *Journal of Nanjing University of Science and Technology*, 48(3), 367–373. <https://doi.org/10.14177/j.cnki.32->

1397n.2024.48.03.015

- [7] Li, C. (2025). Design and implementation of image classification optimization algorithm based on lightweight neural network model. *Computer Knowledge and Technology*, 21(35), 25–27. <https://doi.org/10.14004/j.cnki.ckt.2025.1772>
- [8] Niu, Q., Shi, L., Jia, K., et al. (2024). Gesture recognition based on surface electromyography signals of improved ResNet50. *Foreign Electronic Measurement Technology*, 43(4), 181–189. <https://doi.org/10.19652/j.cnki.femt.2305650>
- [9] Wu, J., Tian, Q., & Yue, J. (2022). Static gesture recognition based on residual dual attention and cross-level feature fusion module. *Computer Systems Applications*, 31(11), 111–119. <https://doi.org/10.15888/j.cnki.csa.008770>
- [10] Wang, Y., Gu, Y., Xiao, Z., et al. (2024). Research on intelligent interactive system based on static gesture recognition. *Technology and Market*, 31(1), 20–24.