

BitOps Analysis of MobileNetV2 with Layer-wise Weight Precision Quantization

Nader Nemati

March 21, 2025

This task helps to understand how quantization affects the performance and resource efficiency of deep learning models, especially when deploying them on devices with limited resources. By applying different bit precisions (2-bit, 4-bit, and 8-bit) to the convolutional layers of MobileNetV2, we can optimize the model to balance accuracy and computational efficiency. Reducing the bit precision lowers memory and computational requirements, which is essential for running models on mobile and edge devices that have limited processing power, storage, and battery life. Although reducing precision may cause some loss in accuracy, it allows us to experiment with trade-offs between model size, speed, and performance to meet the needs of specific applications. Quantization also makes models more efficient and practical for real-world use, like real-time image recognition. To complete this task, a pre-trained MobileNetV2 model should be used as the baseline, and quantization should be applied using a deep learning framework such as PyTorch.

1 Methodology

To conduct this analysis, the following methodology was used:

- **Pretrained MobileNetV2:** We used a pre-trained MobileNetV2 model from PyTorch's model zoo because it has been optimized for efficient performance on mobile devices, making it a suitable baseline for evaluating the impact of quantization.
- **Quantization Precision:** The different precision configurations apply to each convolution layer in the MobileNetV2 architecture. The precision is applied as follows:
 - 8-bit precision for early layers.
 - 4-bit precision for middle layers.
 - 2-bit precision for deeper layers.
- **MACs and BitOps Calculation:** The Multiply-Accumulate Operations (MACs) and BitOps (bit operations) compute for each layer of MobileNetV2. These calculations provide insights into the computational complexity and memory usage of the model.
- **Implementation in PyTorch:** PyTorch framework is utilized due to its flexibility, ease of use, and compatibility with pre-trained models. The model was loaded, quantization was applied, and MACs and BitOps were computed programmatically.

- **Result Generation:** The results are displayed in tables and visualized in a graph to compare the efficiency of different quantization configurations. Moreover, the details of information are visible in the code script in the GitHub repository's code script.

2 Formulas Used for Calculations

For each layer in the MobileNetV2 architecture, the following formulas are used to calculate the MACs and BitOps utilizing the mathematical formulas to envision how they work:

$$\begin{aligned} \text{MACs} = & \text{Input Channels} \times \text{Output Channels} \times \\ & \text{Kernel Height} \times \text{Kernel Width} \times \\ & \text{Output Height} \times \text{Output Width} \end{aligned}$$

$$\text{BitOps} = \text{MACs} \times \text{Precision}$$

Where: - **MACs** is the number of Multiply-Accumulate operations for the layer. - **BitOps** represents the total bit operations, computed as the product of MACs and precision (in bits). - Precision is given as 2, 4, or 8 bits depending on the layer configuration.

3 Example Calculations

Let's calculate the values for *Block_0*, *Block_1*, and *Block_2* for 8-bit precision:

1. *Block_0* (Standard Layer): - Input Channels = 3
 - Output Channels = 32
 - Kernel Height = 3
 - Kernel Width = 3
 - Output Height = 112
 - Output Width = 112
 - Precision = 8 bits

$$\text{MACs} = 3 \times 32 \times 3 \times 3 \times 112 \times 112 = 10838016$$

$$\text{BitOps} = 10838016 \times 8 = 693633024$$

2. *Block_1* (Depthwise Separable Layer): - Input Channels = 32
 - Output Channels = 32
 - Kernel Height = 3
 - Kernel Width = 3
 - Output Height = 112
 - Output Width = 112
 - Precision = 8 bits

$$\text{MACs} = 32 \times 32 \times 3 \times 3 \times 112 \times 112 = 16457728$$

$$\text{BitOps} = 16457728 \times 8 = 1053294592$$

3. *Block_2* (Standard Layer): - Input Channels = 32
- Output Channels = 16
 - Kernel Height = 3
 - Kernel Width = 3
 - Output Height = 112
 - Output Width = 112
 - Precision = 8 bits

$$\text{MACs} = 32 \times 16 \times 3 \times 3 \times 112 \times 112 = 6422528$$

$$\text{BitOps} = 6422528 \times 8 = 411041792$$

These calculations were repeated for other layers, and the results are summarized in the table below for 8-bit precision.

3.1 Results for 8-bit Precision

Block Name	Type	In Ch	Out Ch	Stride	Input Size	Output Size	BitOps
<i>Block_0</i>	standard	3	32	2	224x224	112x112	693633024
<i>Block_1</i>	depthwise	32	32	1	112x112	112x112	1053294592
<i>Block_2</i>	standard	32	16	1	112x112	112x112	411041792

Table 1: Results for Fixed 8-bit Precision

Block Name	Type	In Ch	Out Ch	Stride	Input Size	Output Size	BitOps
<i>Block_0</i>	standard	3	32	2	224x224	112x112	693633024
<i>Block_1</i>	depthwise	32	32	1	112x112	112x112	1053294592
<i>Block_2</i>	standard	32	16	1	112x112	112x112	411041792

Table 2: Results for Fixed 8-bit Precision

4 Discussion of Results

The results show that the precision directly affects the number of BitOps, with 8-bit precision yielding the highest BitOps values for each layer. The first few layers of the MobileNetV2 model, such as *Block_0* and *Block_1*, have significantly higher BitOps compared to the deeper layers.

Block_0 (Standard Layer), with 8-bit precision, requires 693,633,024 BitOps. This high number results from the large input size and number of output channels. *Block_1* (Depth-wise Separable Layer) follows closely with 1,053,294,592 BitOps, showing how depthwise separable convolutions, despite their lower computational cost, still require significant resources due to the number of input-output channels. *Block_2* (Standard Layer), with 8-bit precision, has 411,041,792 BitOps, reflecting a relatively lower number of operations due to fewer output channels.

As precision decreases from 8 bits to 2 bits, the BitOps decrease significantly, but at the cost of reduced accuracy, which is typically expected in quantized models. This trade-off between performance and computational efficiency is crucial when deploying models to edge devices with limited resources.

The following figure shows the total BitOps comparison across different quantization configurations (Adaptive, Fixed 2-bit, Fixed 4-bit, and Fixed 8-bit). As can be seen in the chart, Fixed 8-bit precision generates significantly higher BitOps compared to Fixed 2-bit and Fixed 4-bit configurations. This highlights how quantization to lower bit precision reduces the computational load, making the model more resource-efficient, particularly for deployment in resource-constrained environments.

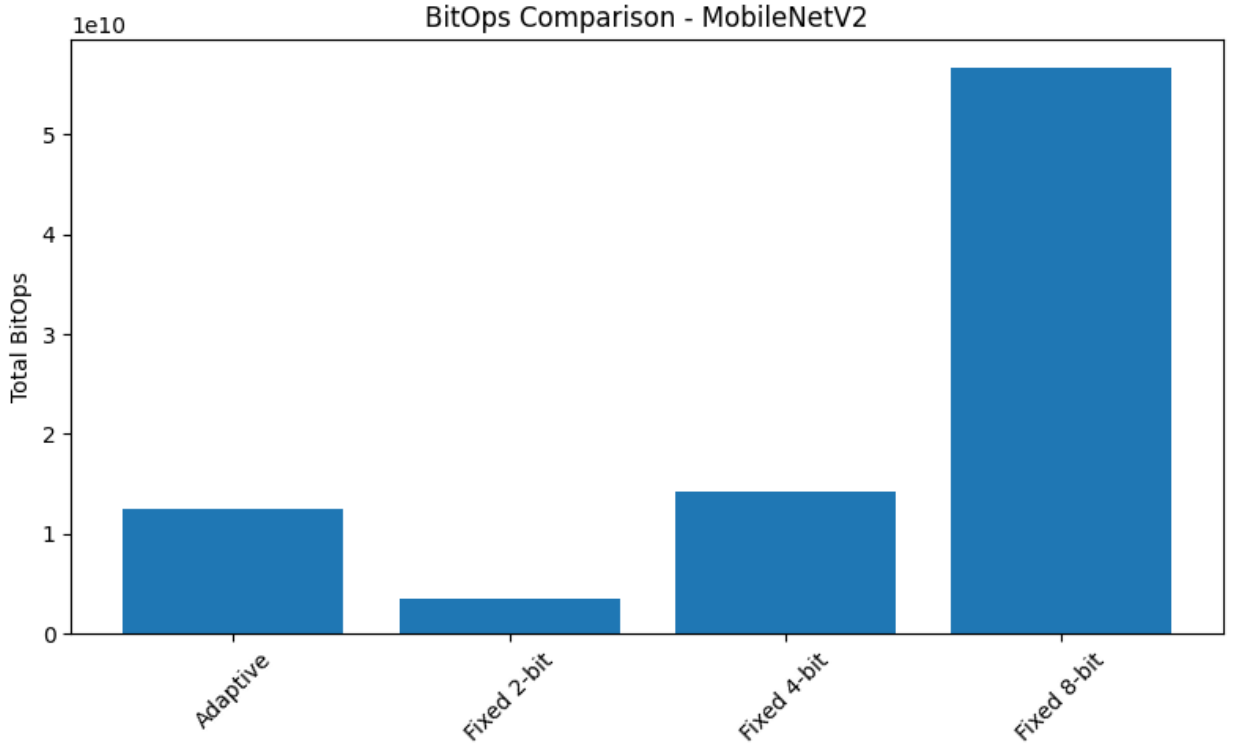


Figure 1: BitOps Comparison for MobileNetV2 under Different Quantization Configurations