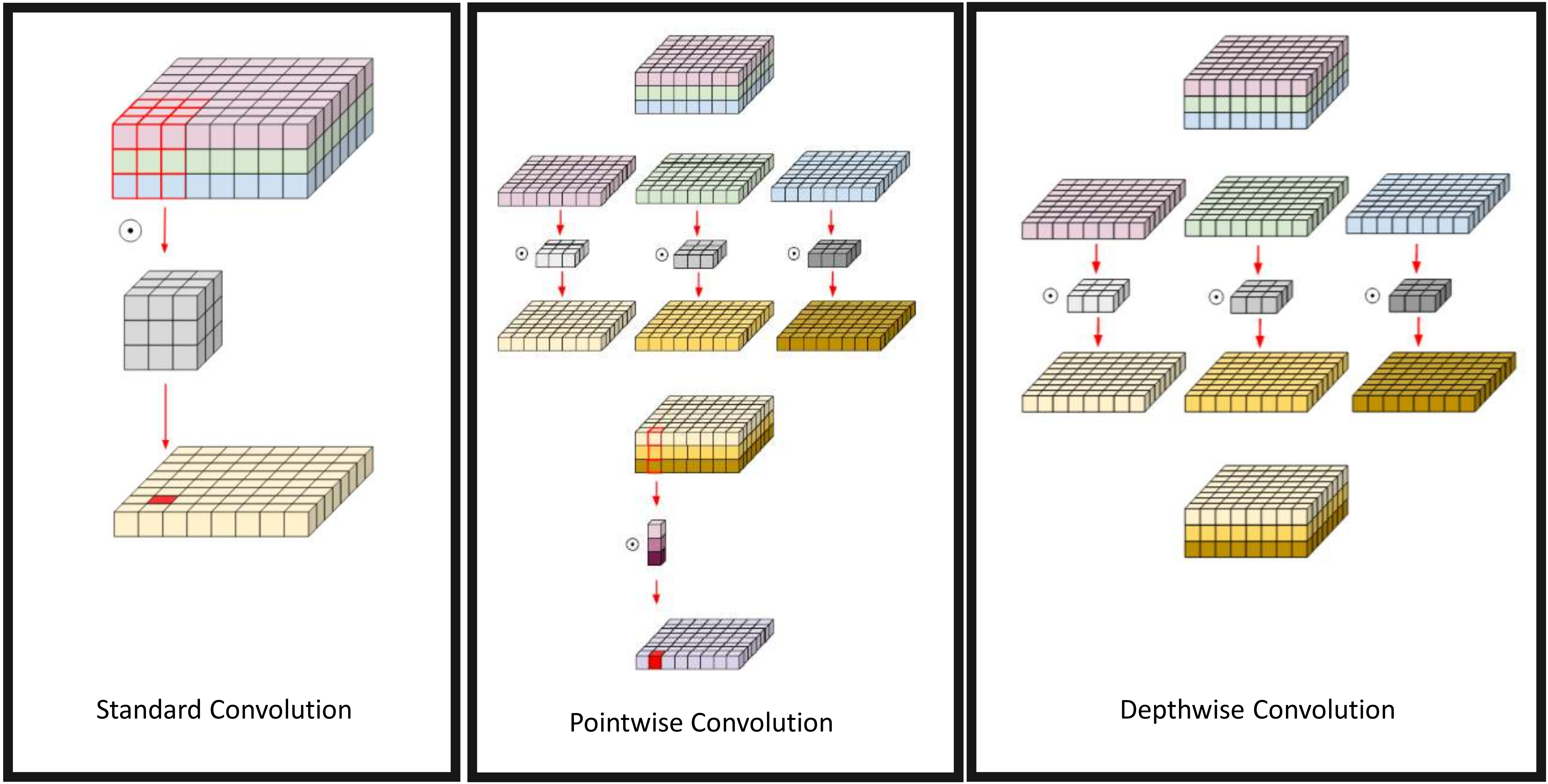
MobileNet: A Deep Dive into Efficiency

- The importance of MobileNet for efficient computations on mobile and embedded devices.
- Key building blocks:
  1. **Standard Convolution** – The traditional approach.
  2. **Depthwise Convolution** – Reducing spatial computation.
  3. **Pointwise Convolution** – Channel-wise computation optimization.
- Focus on how these techniques reduce computational complexity and parameters while maintaining performance.

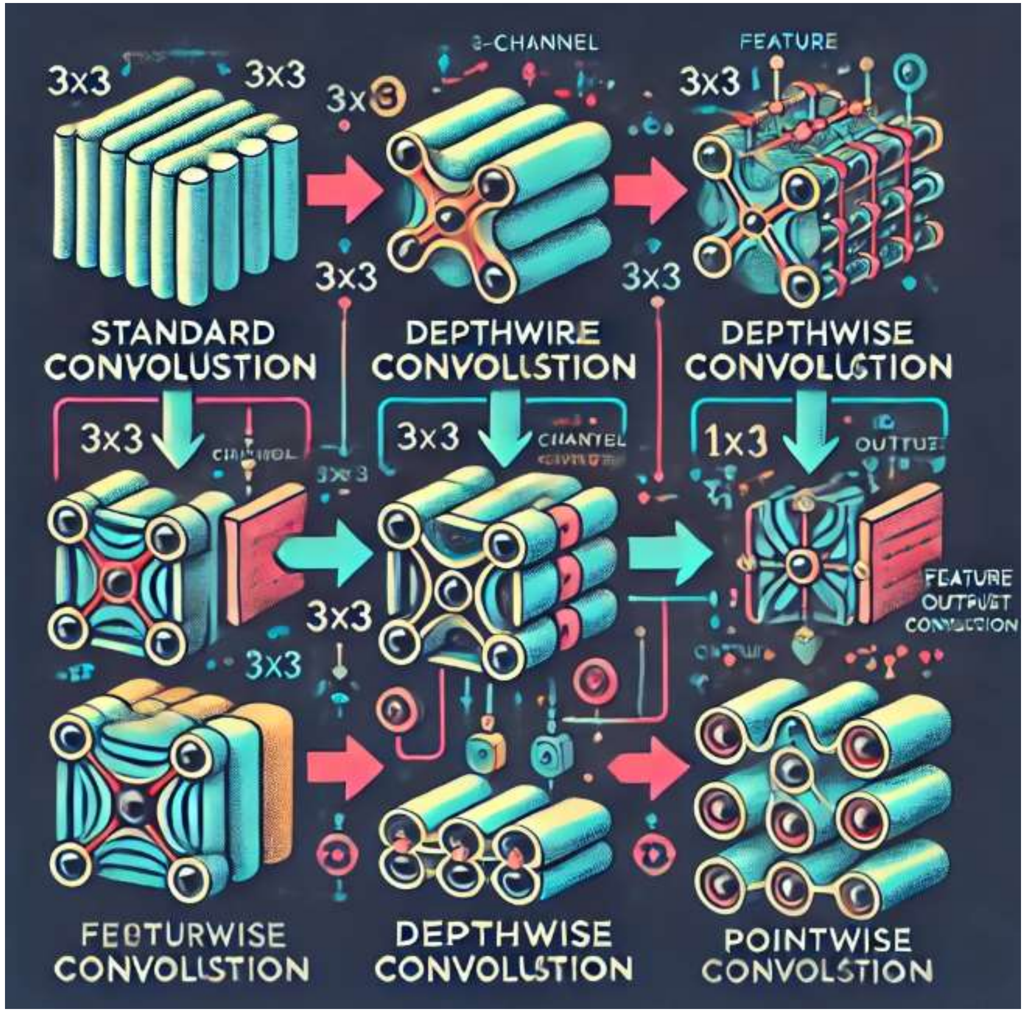
Objective:
To understand how MobileNet achieves high efficiency and accuracy for real-world applications.


Standard Convolution
Pointwise Convolution
Depthwise Convolution

Standard Convolution:
Computationally heavy, combines spatial and channel info in one step.

Depthwise Convolution:
Applies spatial filtering to each channel separately.

Pointwise Convolution:
Combines depth information across channels.



MobileNet allows to operate on devices with limited computational power while maintaining reasonable accuracy

In MobileNet

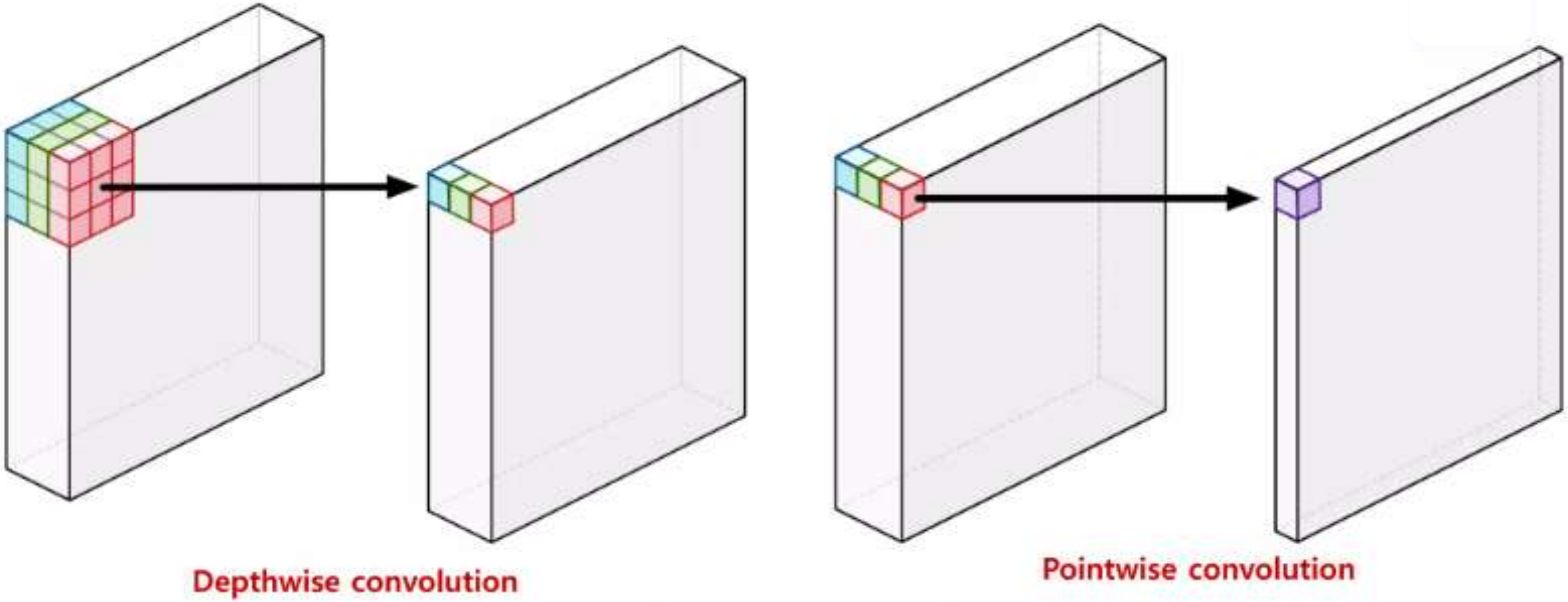
standard convolution computationally expensive

depthwise convolution & **pointwise convolution**

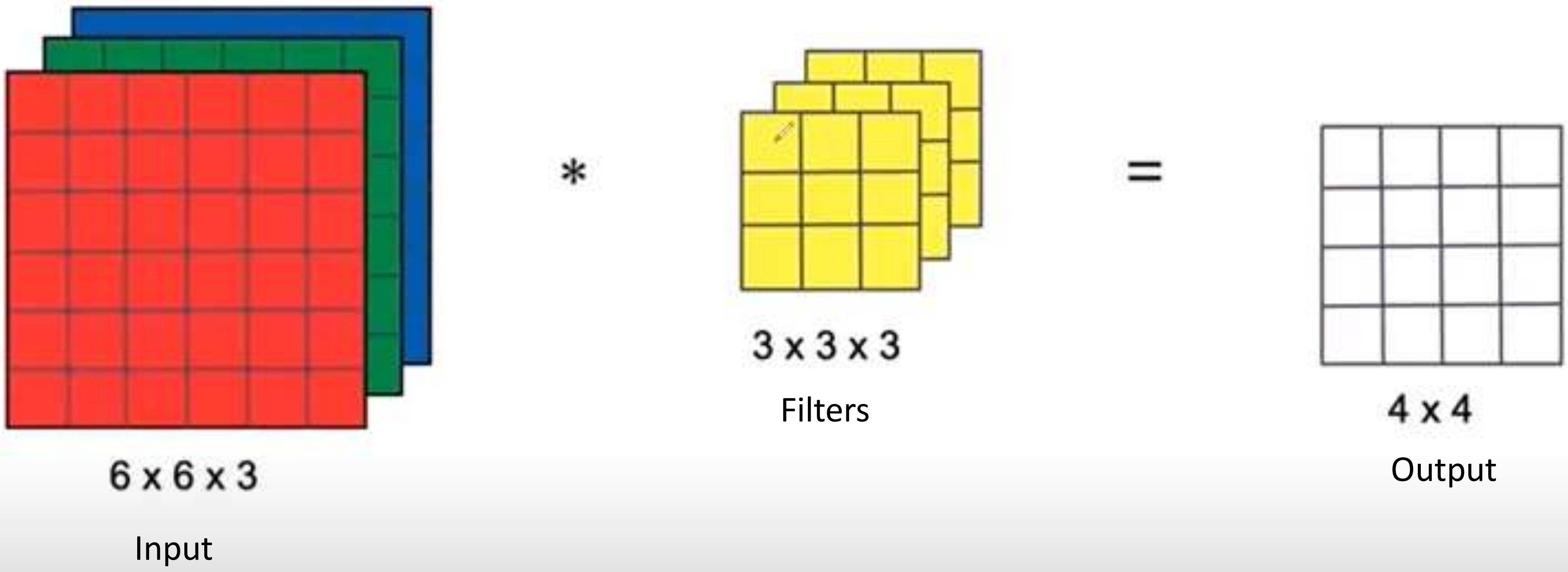
These two methods together reduce the number of parameters and computational complexity, making MobileNet more efficient while maintaining good performance for image classification tasks.

Depthwise Separable Convolution

- Depthwise Convolution + Pointwise Convolution(1x1 convolution)

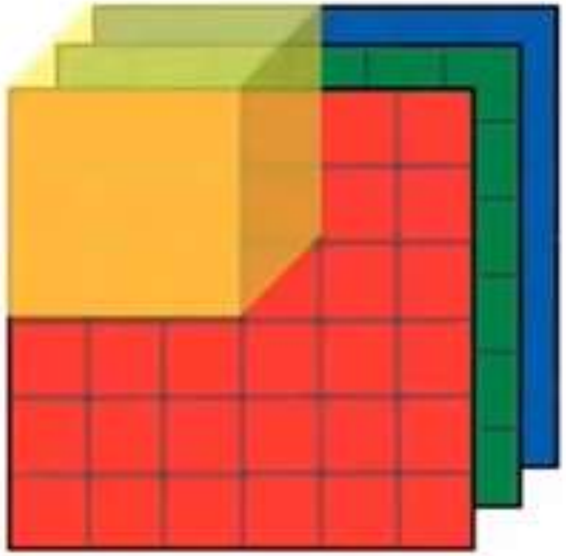


Normal Convolution



Normal Convolution

Number of Filters = **5** (you want 5 output channels).



6 x 6 x 3

$n \times n \times n_c$

Input:
Size = **6 × 6 × 3**
Width & Height = **6 × 6**
Depth/Channels = **3**

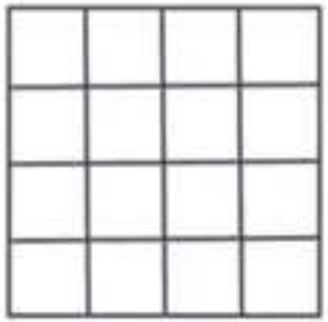
*

3 x 3 x 3

$f \times f \times n_c$

Filters:
Size = **3 × 3 × 3**:
Width & height = **3 × 3**:
Channels = **3**

=



4 x 4

Output:
Output size = **4 × 4 × 5**
Width & height = **4 × 4**
Channels = **5**

Step 1: Compute Output Dimensions

The formula for the output spatial dimensions is:

$$\text{Output Size} = \frac{\text{Input Size} - \text{Filter Size}}{\text{Stride}} + 1$$

- Input size = 6×6
- Filter size = 3×3
- Stride = 1 (filter moves 1 step at a time).
- Padding = 0 (no extra padding added).

$$\text{Output Size (per side)} = \frac{6 - 3}{1} + 1 = 4$$

So, the spatial output size is **4 × 4**.

With 5 filters, the full output size becomes **4 × 4 × 5**.

Step 2: Number of Parameters per Filter

Each filter is a cube of size $3 \times 3 \times 3$:

$$\text{Parameters per Filter} = 3 \times 3 \times 3 = 27$$

There are **5 filters**, so the total number of filter parameters is:

$$\text{Total Parameters} = 27 \times 5 = 135$$

Step 3: Number of Filter Positions

The filter moves across the input at each position to calculate the output. The number of filter positions is based on the output size:

$$\text{Filter Positions} = \text{Output Width} \times \text{Output Height} = 4 \times 4 = 16$$

Step 4: Total Computational Cost

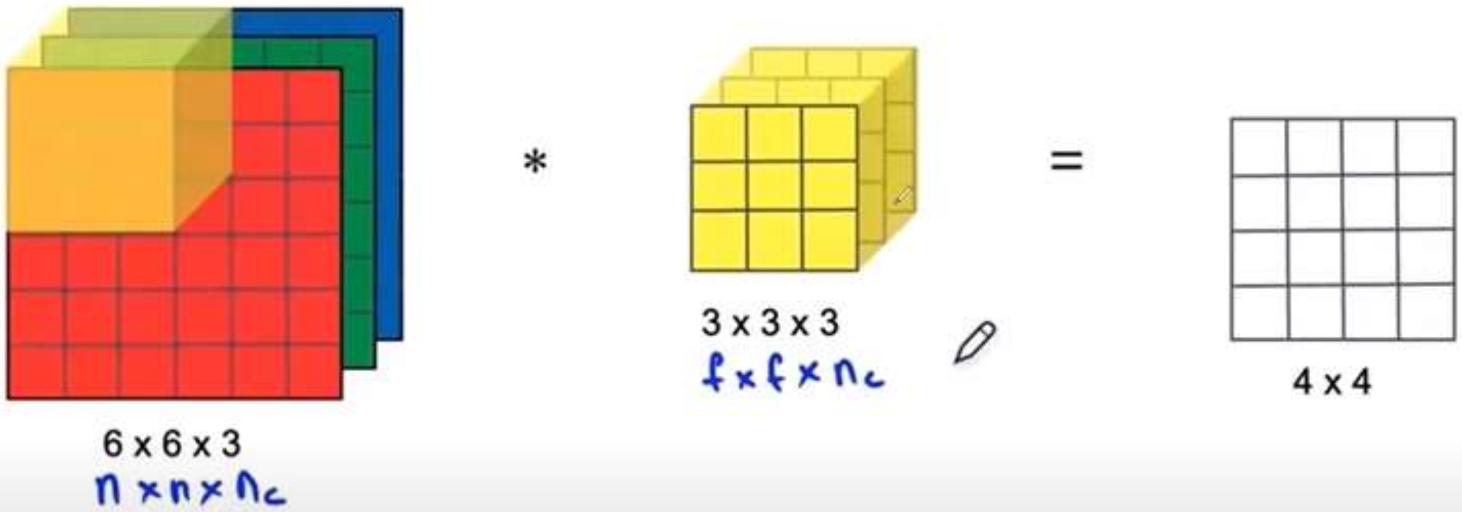
At each position, for each filter, we perform the following:

- Multiply and add for all the filter parameters.
- Each filter has **27 parameters**, and there are **16 positions** per filter.
- For **5 filters**, the total computation is:

$$\text{Computational Cost} = \text{Parameters per Filter} \times \text{Filter Positions} \times \text{Number of Filters}$$

$$\text{Computational Cost} = 27 \times 16 \times 5 = 2160$$

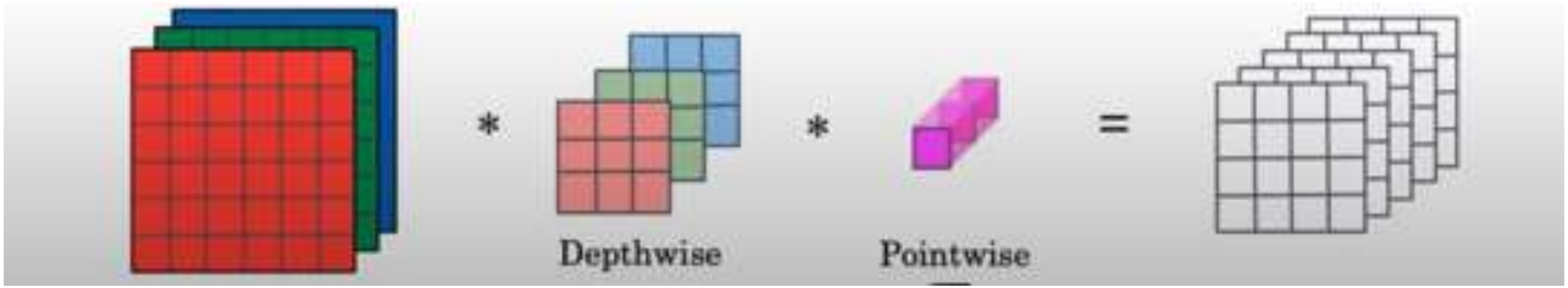
Normal Convolution



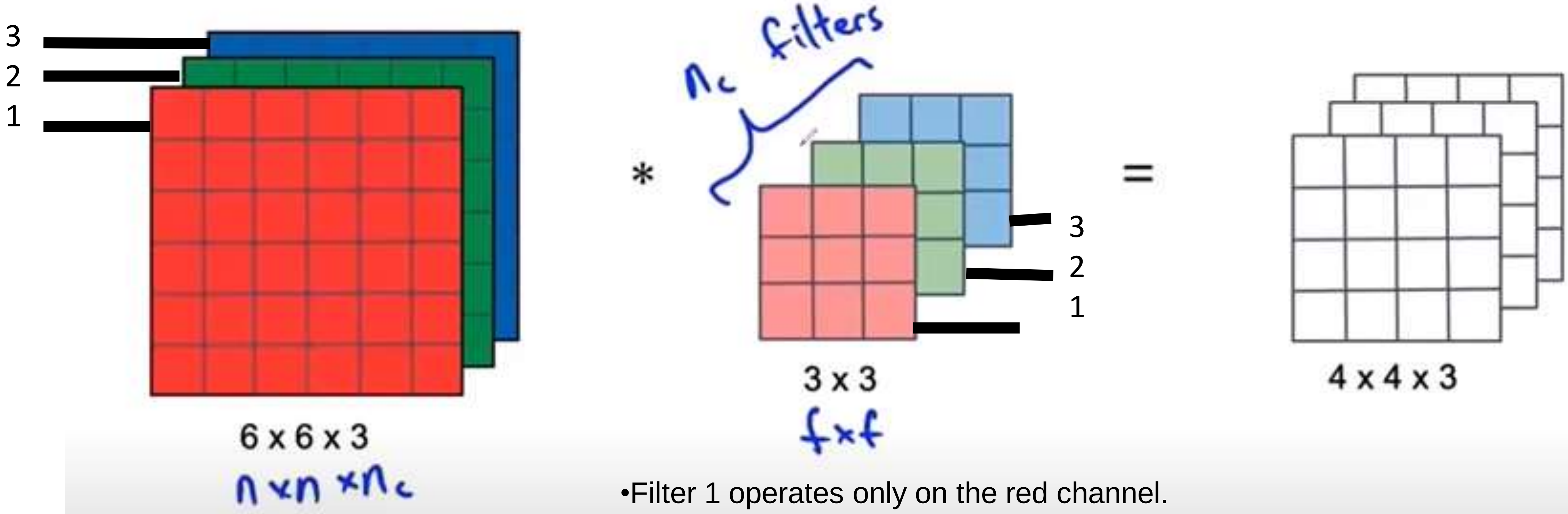
Summary of Results:

1. **Output Size:** $4 \times 4 \times 5$ (Width × Height × Channels).
2. **Filter Parameters:** $27 \times 5 = 135$.
3. **Computational Cost:** 2160.

Depthwise Separable Convolution



Depthwise Convolution



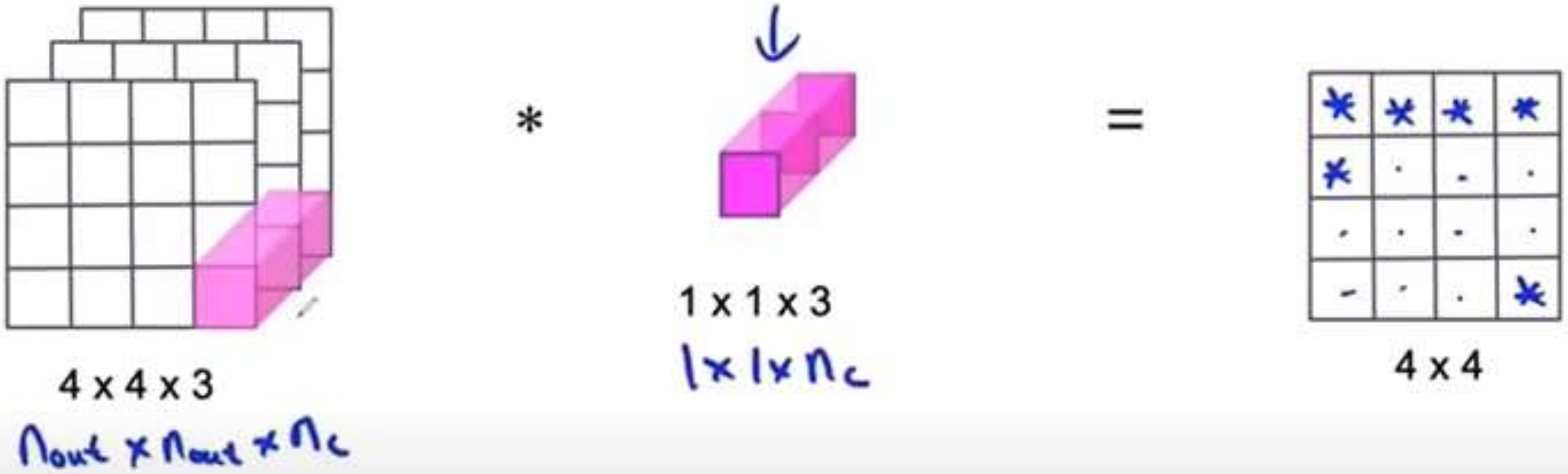
- Filter 1 operates only on the red channel.
- Filter 2 operates only on the green channel.
- Filter 3 operates only on the blue channel.

Computational Cost:

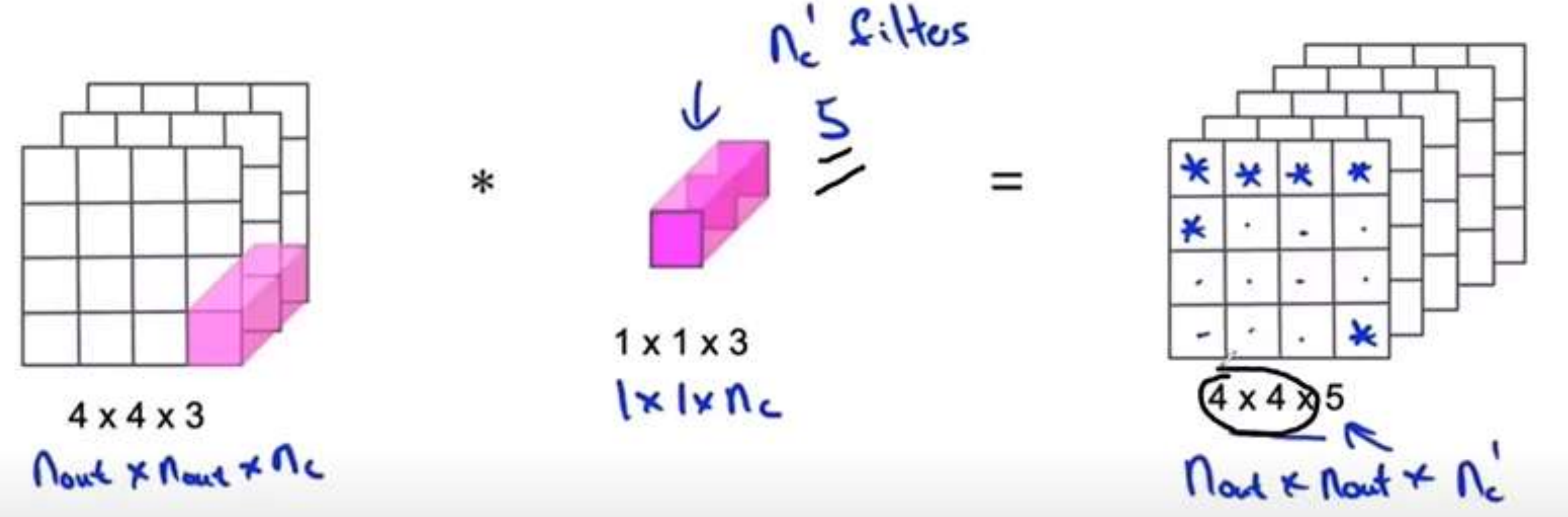
Depthwise convolution drastically reduces the computation compared to standard convolution.

- **Number of Filter Parameters:**
  - Each 3×3 filter has $3 \times 3 = 9$ parameters.
  - With 3 filters, the total is $9 \times 3 = 27$ parameters.
- **Number of Filter Positions**:
  - The filter slides across $4 \times 4 = 16$ positions for each channel.
  - Total positions = $16 \times 3 = 48$.
- **Total Multiplications**:
  - 27 filter parameters are applied across 48 positions: $27 \times 48 = 432$ multiplications.

Pointwise Convolution



Pointwise Convolution



- **Input Tensor**: 4×4×34
- **Filters**: 5 filters, each of size
- **Output Tensor**: After applying 5 filters, the output will have 4×4×5
- Here, 5 (output channels) corresponds to the number of filters.

Computational Cost:

Step-by-Step Explanation:

1. **Filter Parameters:**

   Each filter is of size **1x1x3**, so there are $1 \cdot 1 \cdot 3 = 3$ parameters per filter.

2. **Filter Positions:**

   The filter moves across every spatial location in the input, which is $4 \cdot 4 = 16$ positions.

3. **Number of Filters:**

   There are 5 filters, corresponding to N', the number of desired output channels.

Total Computational Cost:

$$\text{Cost} = 3 \cdot 16 \cdot 5 = 240$$

Comparison of computational costs

Type	Input Dimensions	Filters Used	Output Dimensions	Computational Cost (Operations)
Standard Convolution	$6 \times 6 \times 3$	$5 \times (3 \times 3 \times 3)$	$4 \times 4 \times 5$	$2,160$
Depthwise Convolution	$6 \times 6 \times 3$	$3 \times (3 \times 3)$	$4 \times 4 \times 3$	423
Pointwise Convolution	$4 \times 4 \times 3$	$5 \times (1 \times 1 \times 3)$	$4 \times 4 \times 5$	240

Key Takeaways:

- Depthwise and Pointwise Convolutions significantly reduce computational complexity.
- MobileNet achieves efficiency without compromising accuracy, making it suitable for devices with limited resources.

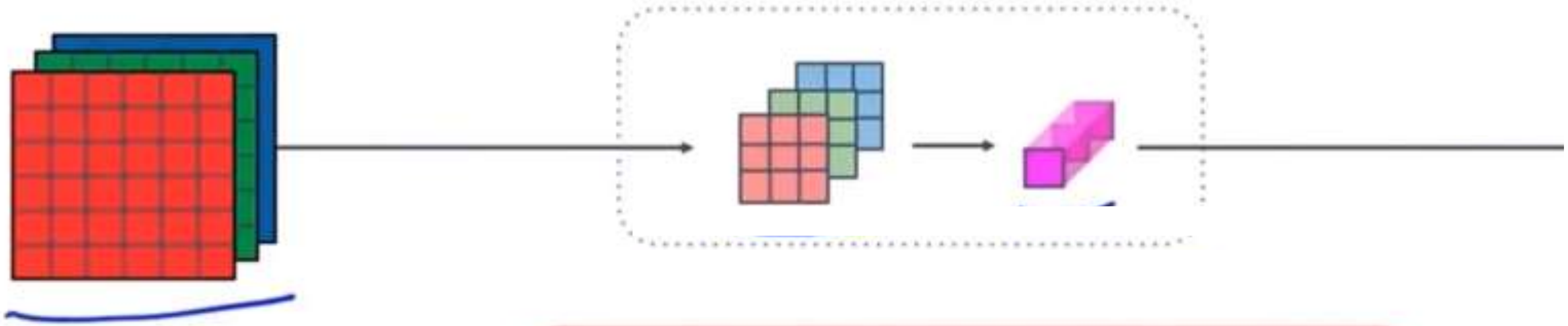
Real-World Applications:

- Widely used in **object detection**, **face recognition**, and **image classification** on mobile devices and edge platforms.

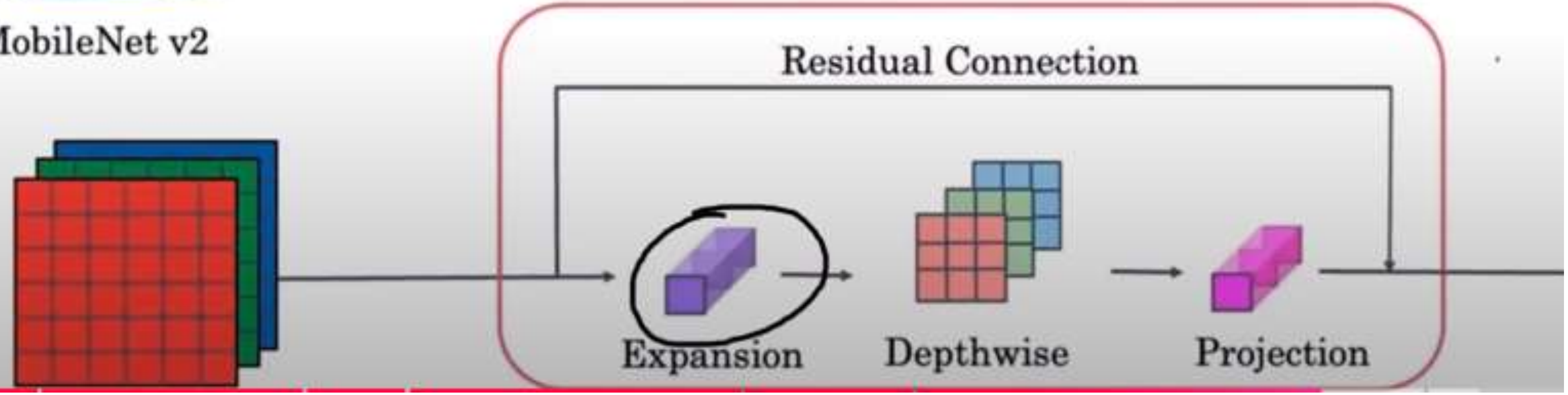
Mobile
Net

MobileNet

MobileNet v1



MobileNet v2



MobileNet v1:

- It uses **depthwise separable convolutions**, which break down the image into smaller parts and process them independently.
- It processes the image step by step for **13 layers**, extracting features like edges, colors, and textures.
- However, it doesn’t reuse information, so it’s slower and less efficient.

MobileNet v2:

1. **Expansion:**

   Temporarily increases the data size (like zooming in on details) to capture richer information.

2. **Depthwise Convolution:**

   Processes only the important parts of the data, reducing unnecessary computations.

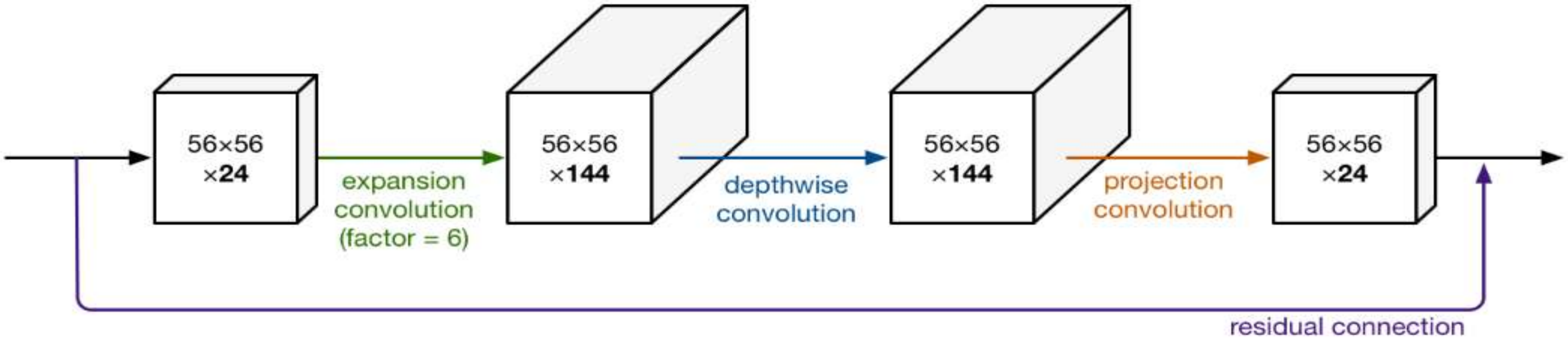
3. **Projection:**

   Shrinks the data back to its original size to save memory.

4. **Residual Connection:**

   Skips some layers when the important features (like a clear edge or shape) are already captured.

This process happens **17 times**, making it more accurate and faster while using fewer resources.



- **MobileNet v1**: Processes an image layer by layer (step by step). It's simple but uses more resources and time.
- **MobileNet v2**: Improves the process by:
    1. **Expanding:** Temporarily making the data bigger to capture more details.
    2. **Depthwise Convolution:** Efficiently processing the data to reduce computations.
    3. **Compressing:** Shrinking the data back to save memory.
    4. **Shortcut (Residual Connection):** Skipping layers when the data is already good, making the process faster.

Result: **MobileNet v2 is faster, uses less power, and is more accurate than v1.**