

You have **3** free member-only stories left this month. [Upgrade for unlimited access.](#)



Arjun Sarkar

Follow

Jun 25, 2021 · 5 min read · ✨ Member-only · 🎧 Listen



Understanding Depthwise Separable Convolutions and the efficiency of MobileNets

Explanation of MobileNets and Depthwise Separable Convolutions

Introduction:

In convolutional neural networks (CNN), 2D convolutions are the most frequently used convolutional layer. MobileNet is a CNN architecture that is much faster as well as a smaller model that makes use of a new kind of convolutional layer, known as Depthwise Separable convolution. Because of the small size of the model, these models are considered very useful to be implemented on mobile and embedded devices. Hence the name MobileNet.

The original paper on MobileNets is available here — [MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications](#)

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
GoogleNet	69.8%	1550	6.8
VGG 16	71.5%	15300	138

Table 1. MobileNet parameter and accuracy comparison against GoogleNet and VGG 16 (Source: Table from the original paper)

Table 1 clearly shows how even the deepest MobileNet architectures have fewer parameters than the famous CNN architectures. The smaller MobileNets have as few as 1.3 million parameters. Also, the size of the models is significantly less. While a VGG16 model can take up to 500 MB of disk space, MobileNet just needs 16–18MB. This makes it ideal to be loaded on mobile devices.



Figure 1. MobileNets can be used for Object Detection, Classification, Attribute detection, or even Landmark recognition on mobile devices (Source: Image from the original paper)

Depthwise Convolutions:

Differences —

The main difference between 2D convolutions and Depthwise Convolution is that 2D convolutions are performed over all/multiple input channels, whereas in Depthwise convolution, each channel is kept separate.

Approach —

1. Input tensor of 3 dimensions is split into separate channels
2. For each channel, the input is convolved with a filter (2D)
3. The output of each channel is then stacked together to get the output on the entire 3D tensor

Graphical Description —

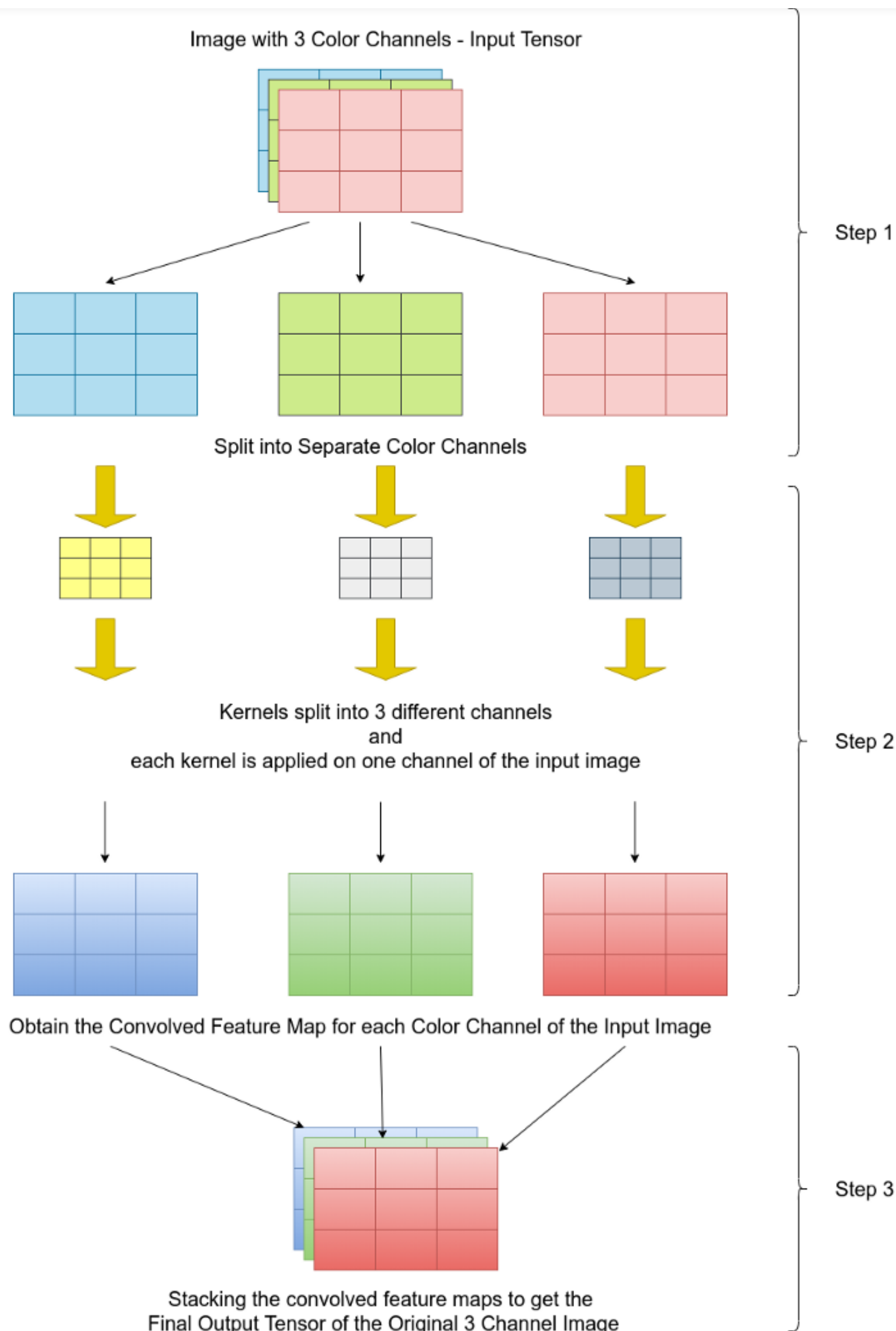


Figure 2. Diagrammatic explanation of Depthwise Convolutions (Source: Image created by author)

Depthwise Separable Convolutions:

Depthwise convolutions are generally applied in combination with another step — Depthwise Separable Convolution. This contains two parts— 1. Filtering (all the previous steps) and 2. Combining (combining the 3 color channels to form 'n' number of channels, as desired — in the example below we see how the 3 channel can be combined to form a 1 channel output).

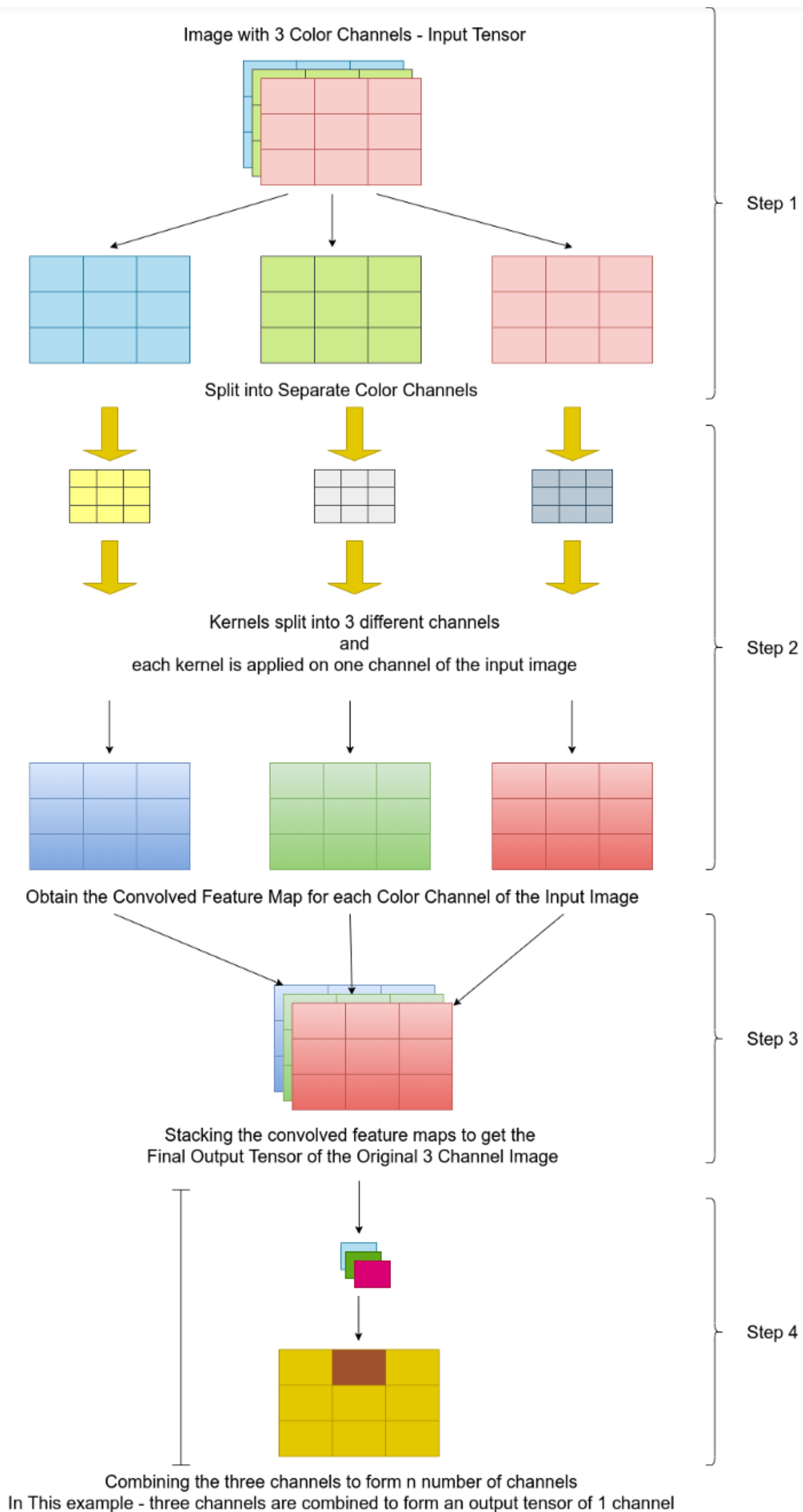


Figure 3. Diagrammatic explanation of Depthwise Separable Convolutions (Source: image created by author)



Depthwise Convolution is -1x1 convolutions across all channels

Let's assume that we have an input tensor of size — $8 \times 8 \times 3$,

And the desired output tensor is of size — $8 \times 8 \times 256$

In 2D Convolutions —

Number of multiplications required — $(8 \times 8) \times (5 \times 5 \times 3) \times (256) = 1,228,800$

In Depthwise Separable Convolutions —

The number of multiplications required:

a. **Filtering** — Split into single channels, so $5 \times 5 \times 1$ filter is required in place of $5 \times 5 \times 3$, and since there are three channels, so the total number of $5 \times 5 \times 1$ filters required is 3, so,

$$(8 \times 8) \times (5 \times 5 \times 1) \times (3) = 3,800$$

b. **Combining** — Total number of channels required is 256, so,

$$(8 \times 8) \times (1 \times 1 \times 3) \times (256) = 49,152$$

$$\text{Total number of multiplications} = 3,800 + 49,152 = 53,952$$

So a 2D convolution will require 1,228,800 multiplications, while a Depthwise Separable convolution will require only 53,952 multiplications to reach the same output.

Finally,

$$1,228,800 / 53,952 = 23 \times \text{less multiplications required}$$

Hence the efficiency of Depthwise Separable convolutions is so high. These are the layers implemented in the MobileNet architecture, to decrease the number of computations and making them less power-hungry, so that they can be run on mobile / embedded devices which do not have powerful graphical processing units in them.

2. Using two *Shrinking Hyperparameters*:

a. *Width multiplier* which adjusts the number of channels

Width Multiplier	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
0.75 MobileNet-224	68.4%	325	2.6
0.5 MobileNet-224	63.7%	149	1.3
0.25 MobileNet-224	50.6%	41	0.5

Table 2. Width multiplier in MobileNet (Source: Table from the original paper)

b. *Resolution multiplier* which adjusts the spatial dimensions of the feature maps and the input image

Resolution	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
1.0 MobileNet-192	69.1%	418	4.2
1.0 MobileNet-160	67.2%	290	4.2
1.0 MobileNet-128	64.4%	186	4.2

Table 3. Resolution multiplier in MobileNet (Source: Table from the original paper)

Architecture — The first layer of the MobileNet is a full convolution, while all following layers are Depthwise Separable Convolutional layers. All the layers are followed by batch normalization and ReLU activations. The final classification layer has a softmax activation. The full architecture is shown in Table 4.

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
5×	Conv dw / s1	$3 \times 3 \times 512$ dw
	Conv / s1	$1 \times 1 \times 512 \times 512$
	Conv dw / s2	$3 \times 3 \times 512$ dw
	Conv / s1	$1 \times 1 \times 512 \times 1024$
	Conv dw / s2	$3 \times 3 \times 1024$ dw
	Conv / s1	$1 \times 1 \times 1024 \times 1024$
	Avg Pool / s1	Pool 7×7
	FC / s1	1024×1000
	Softmax / s1	Classifier

Table 4. MobileNet architecture (Source: table from the original paper)

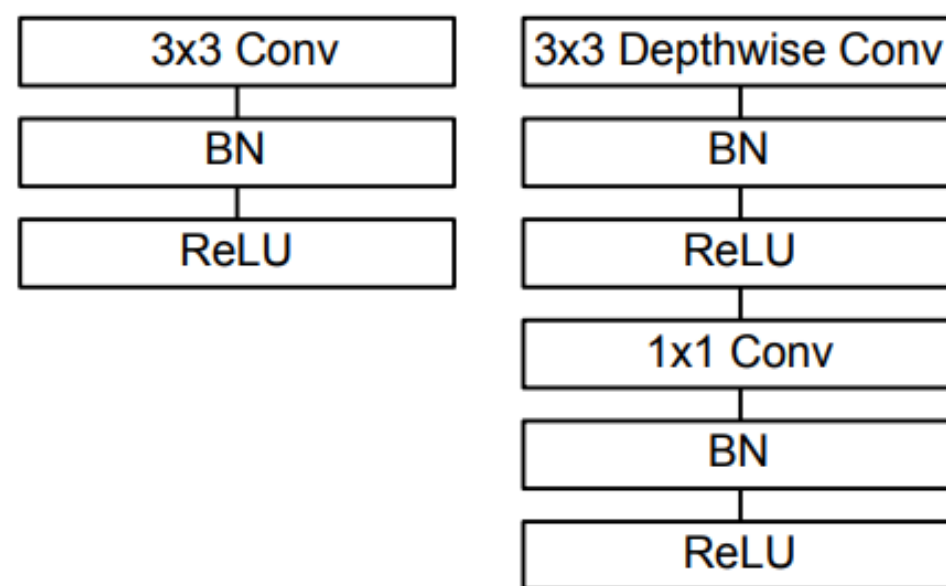


Figure 4. Left: Standard Convolutional layer, Right: Depthwise Separable Convolutional layers in MobileNet (Source: image from the original paper)

Figure 4 shows the difference in architecture flow between normal CNN models vs MobileNets. On the left of the image, we see a 3x3 Convolutional layer followed by batch normalization and ReLU, while, on the right, we see the Depthwise Separable Convolutional layer — consisting of a 3x3 Depthwise Convolution with a batch norm and ReLU followed by a 1x1 pointwise convolution followed by a batch norm and ReLU.

Conclusion:



when compared with the larger fully convolutional architectures, but that is very minute. For example, on the Stanford dogs dataset, while the Inception V3 model gets an accuracy of 84%, the largest MobileNet gets an accuracy of 83.3%. But if we look at the number of parameters of each model architecture, while the Inception V3 has 23.2 million parameters, the MobileNet has only 3.3 million parameters. Also, it is possible to make even smaller and faster MobileNet versions just by using a width multiplier or a resolution multiplier. This makes MobileNets a highly sought-after deep learning model on mobile and embedded devices.

Next, I intend to show how to create a MobileNet architecture from scratch in python, using TensorFlow.

References:


1. Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. & Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications (cite arxiv:1704.04861)

Sign up for The Variable

By Towards Data Science

Every Thursday, the Variable delivers the very best of Towards Data Science: from hands-on tutorials and cutting-edge research to original features you don't want to miss. [Take a look.](#)

Emails will be sent to caoyixue@usc.edu. [Not you?](#)

 [Get this newsletter](#)