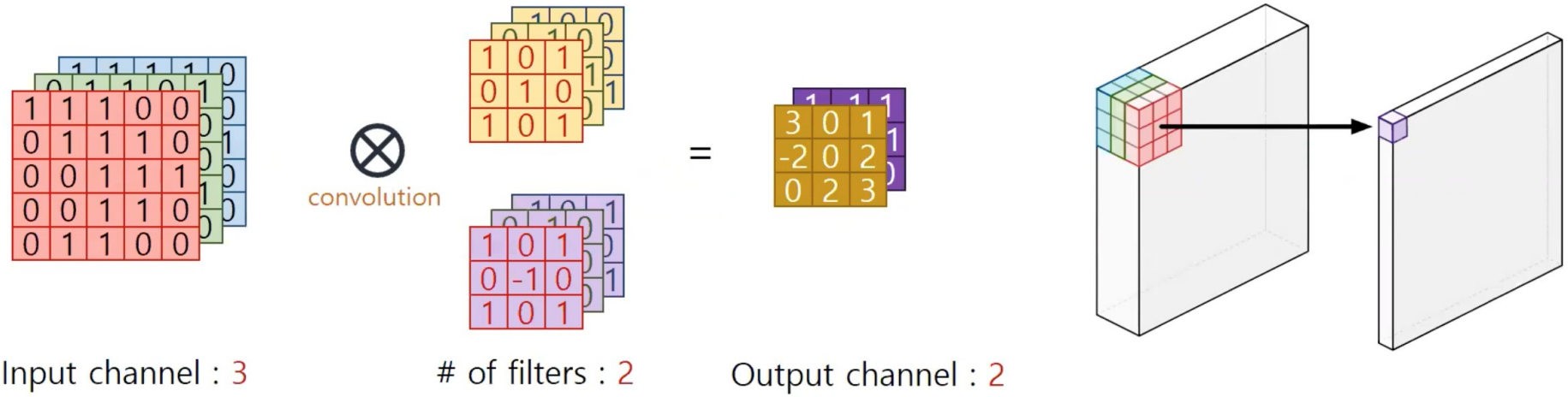
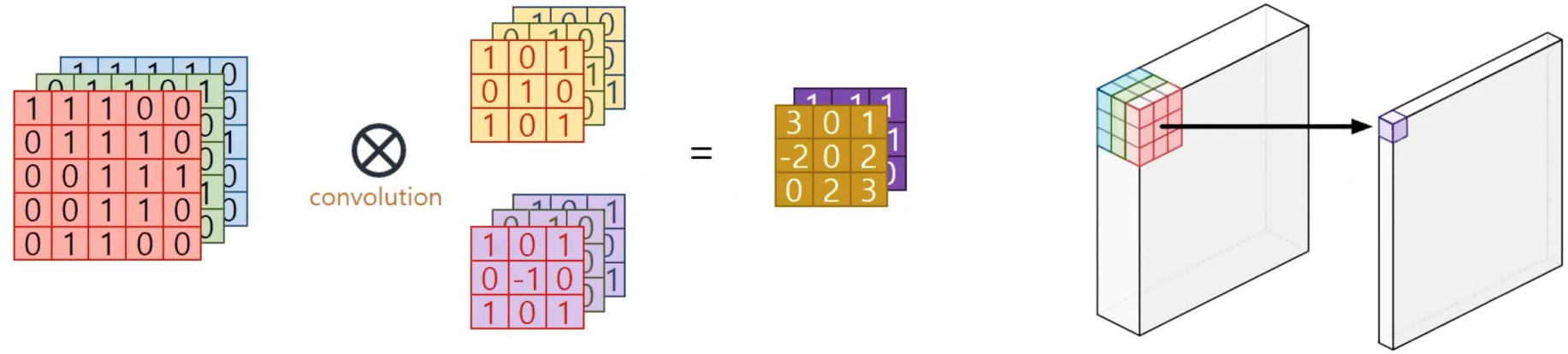
Mobilenets: Depthwise Separable Convolutions

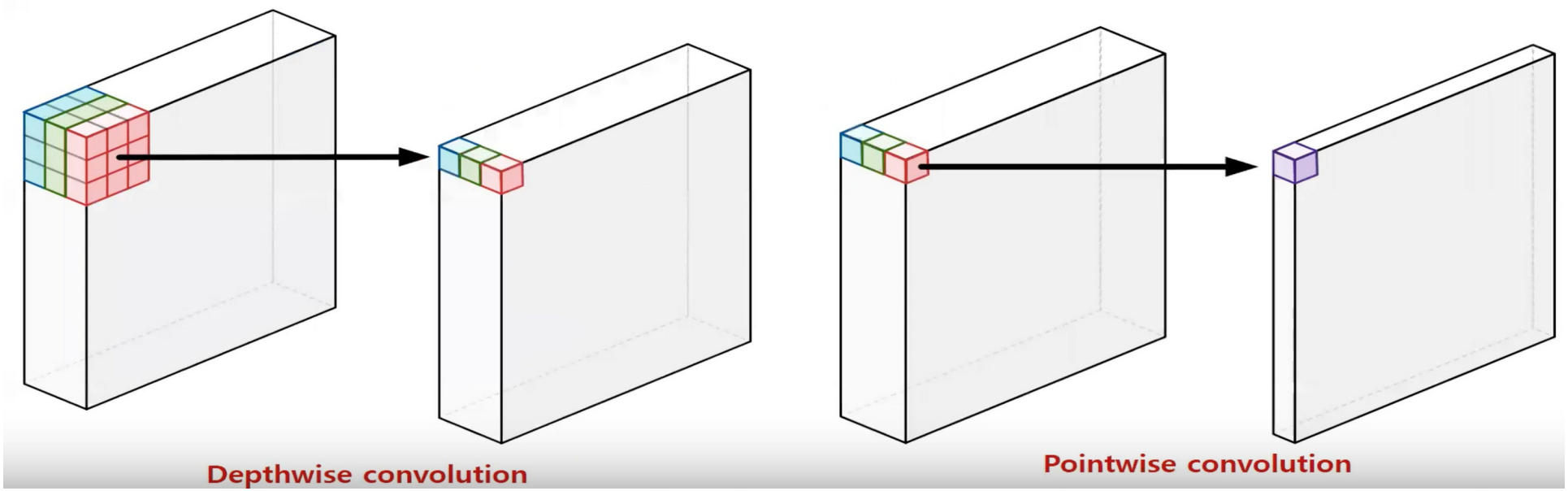
Standard Convolution



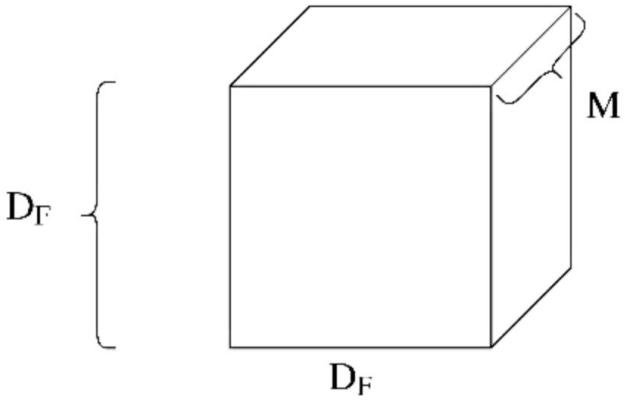
Standard Convolution



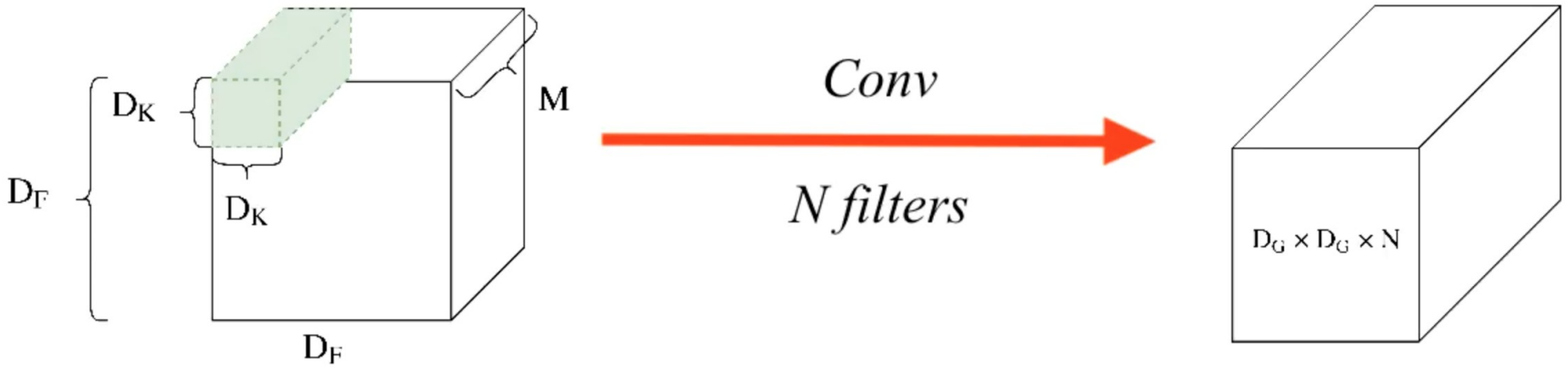
Depthwise Separable Convolution

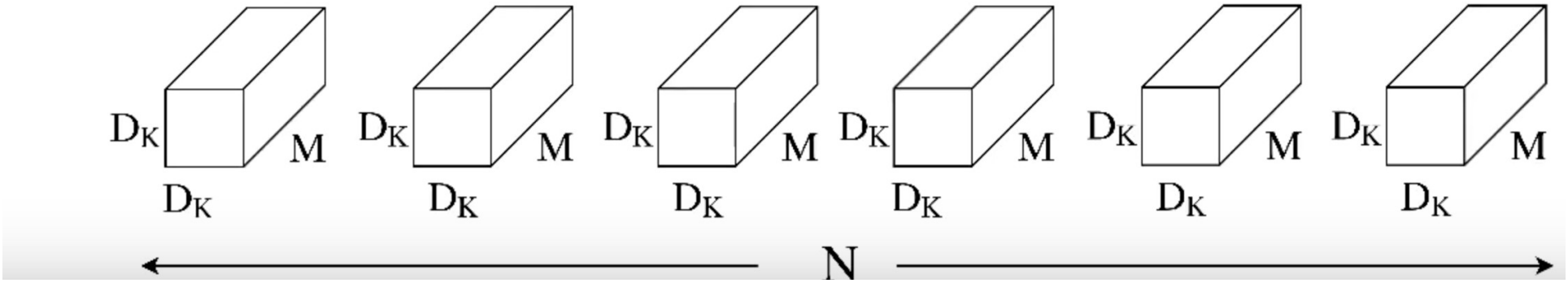


Standard Convolution

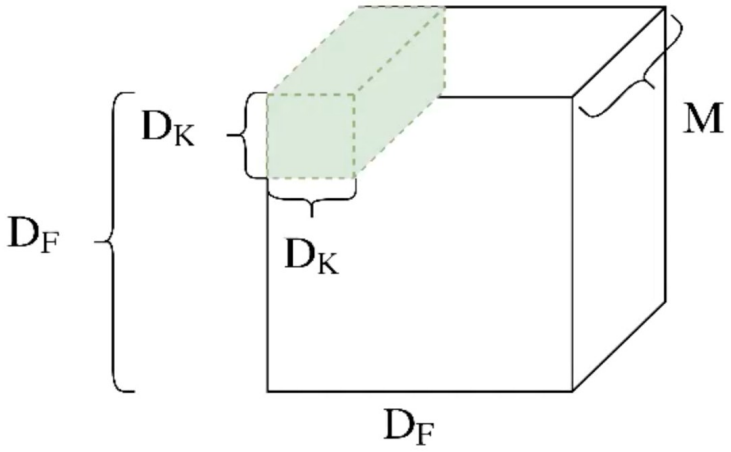


Standard Convolution



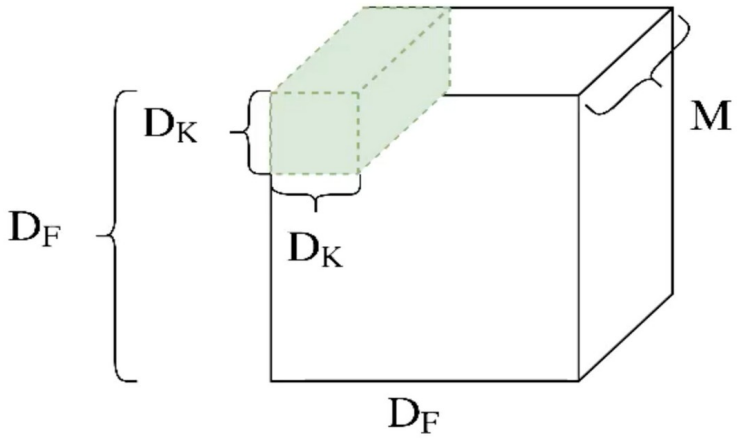


Standard Convolution



Mults once = $D_K^2 \times M$

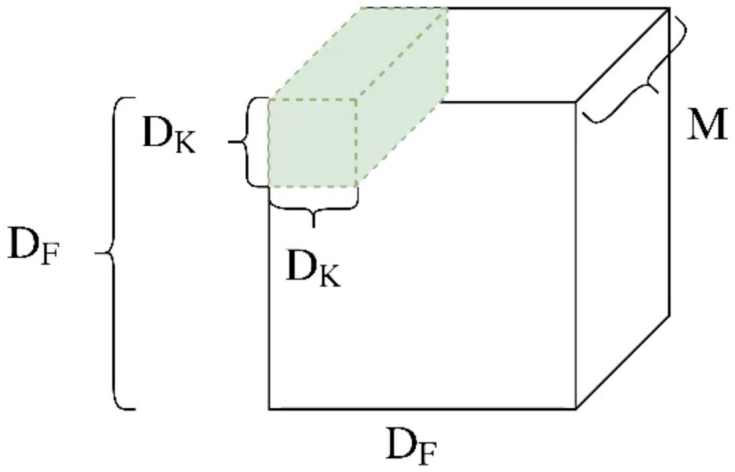
Standard Convolution



$$\text{Mults once} = D_K^2 \times M$$

$$\text{Mults per Kernel} = D_G^2 \times D_K^2 \times M$$

Standard Convolution



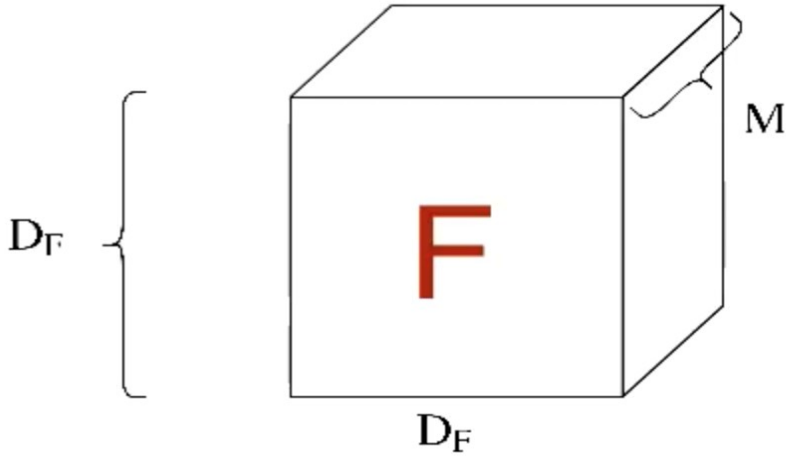
$$\text{Mults once} = D_K^2 \times M$$

$$\text{Mults per Kernel} = D_G^2 \times D_K^2 \times M$$

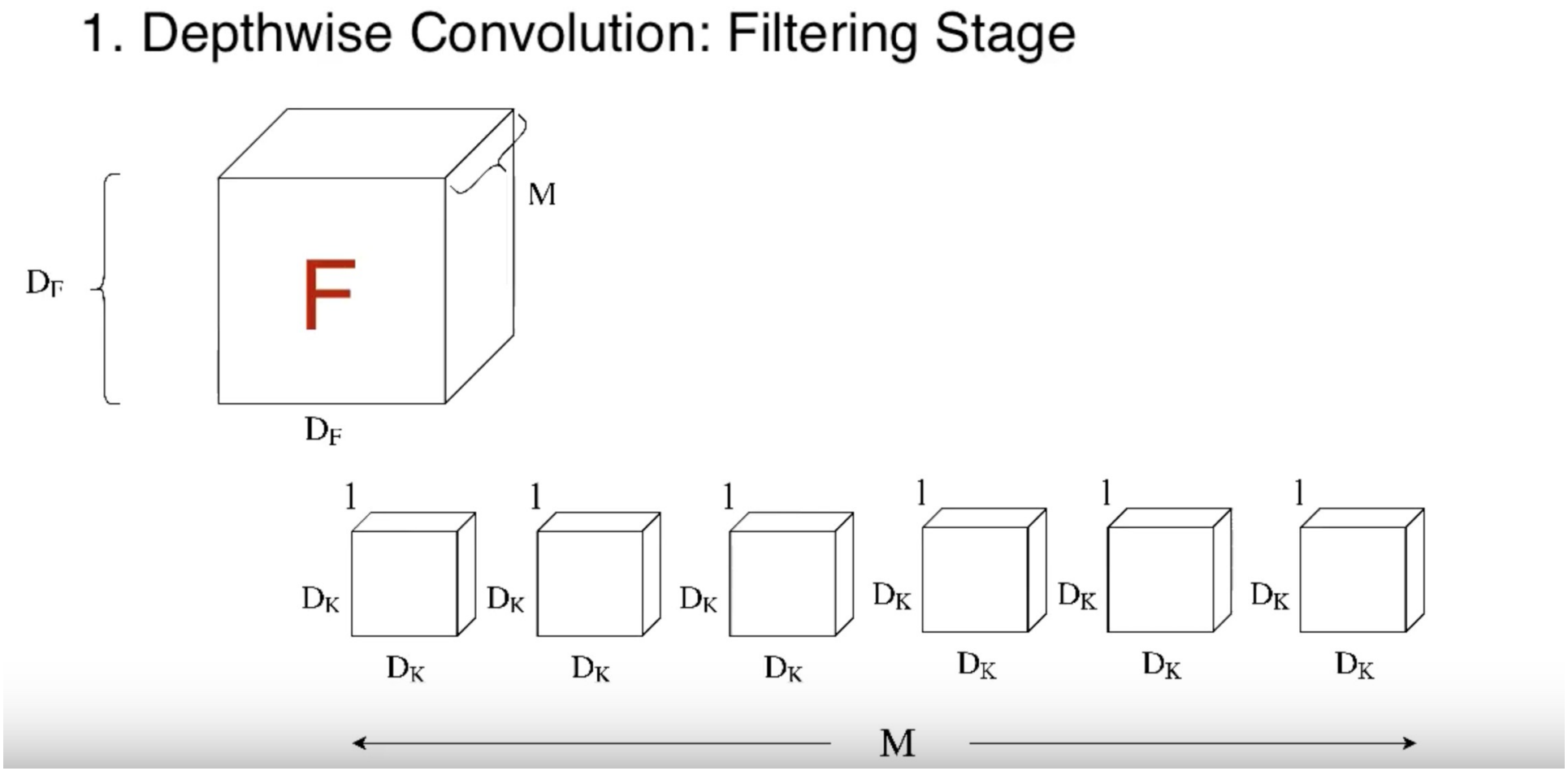
$$\text{Mults N Kernels} = N \times D_G^2 \times D_K^2 \times M$$

Depthwise Separable Convolution

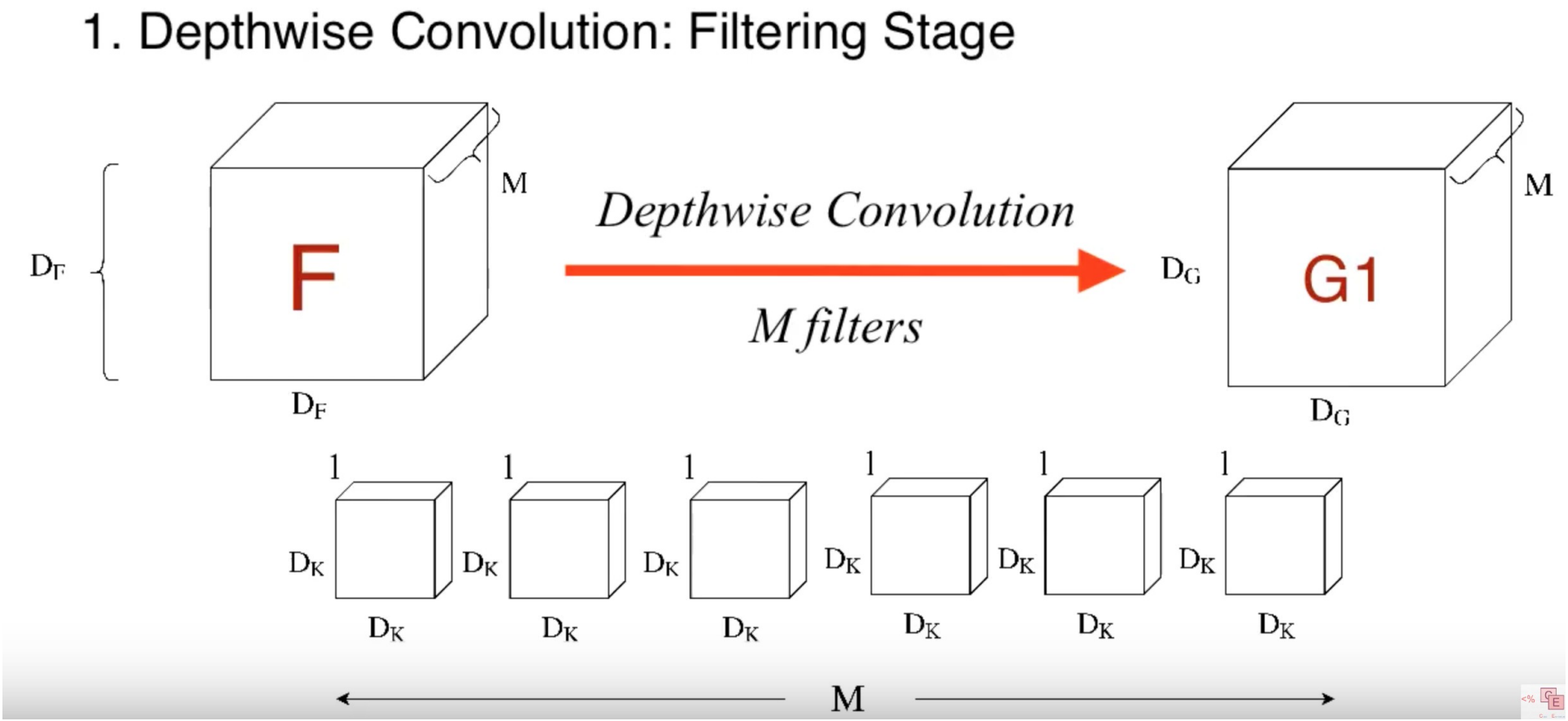
1. Depthwise Convolution: Filtering Stage



Depthwise Separable Convolution

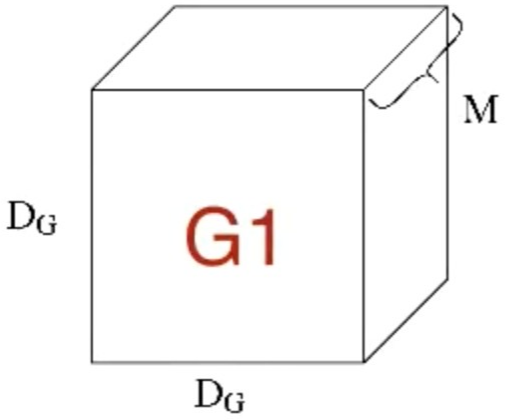


Depthwise Separable Convolution



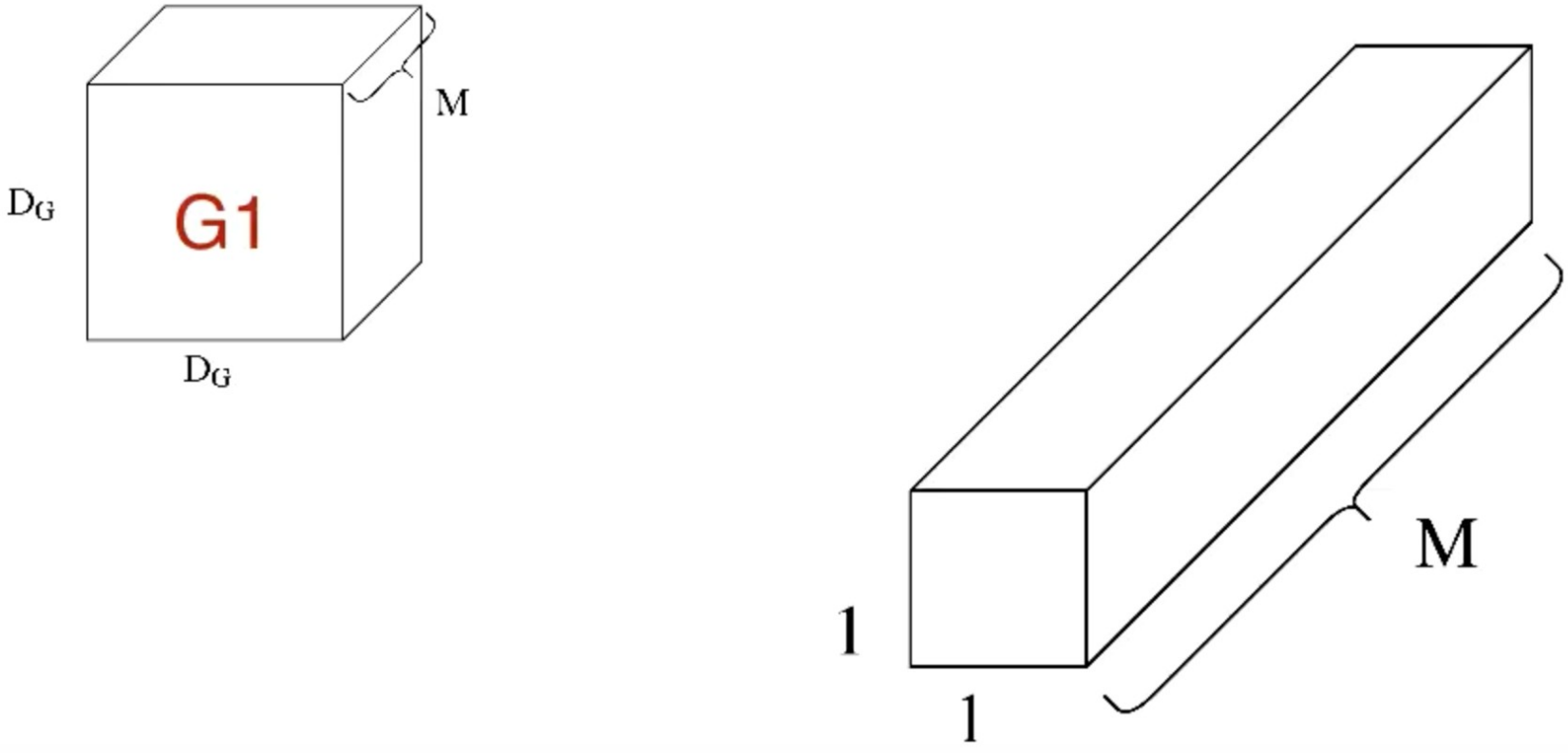
Depthwise Separable Convolution

2. Pointwise Convolution: Filtering Stage

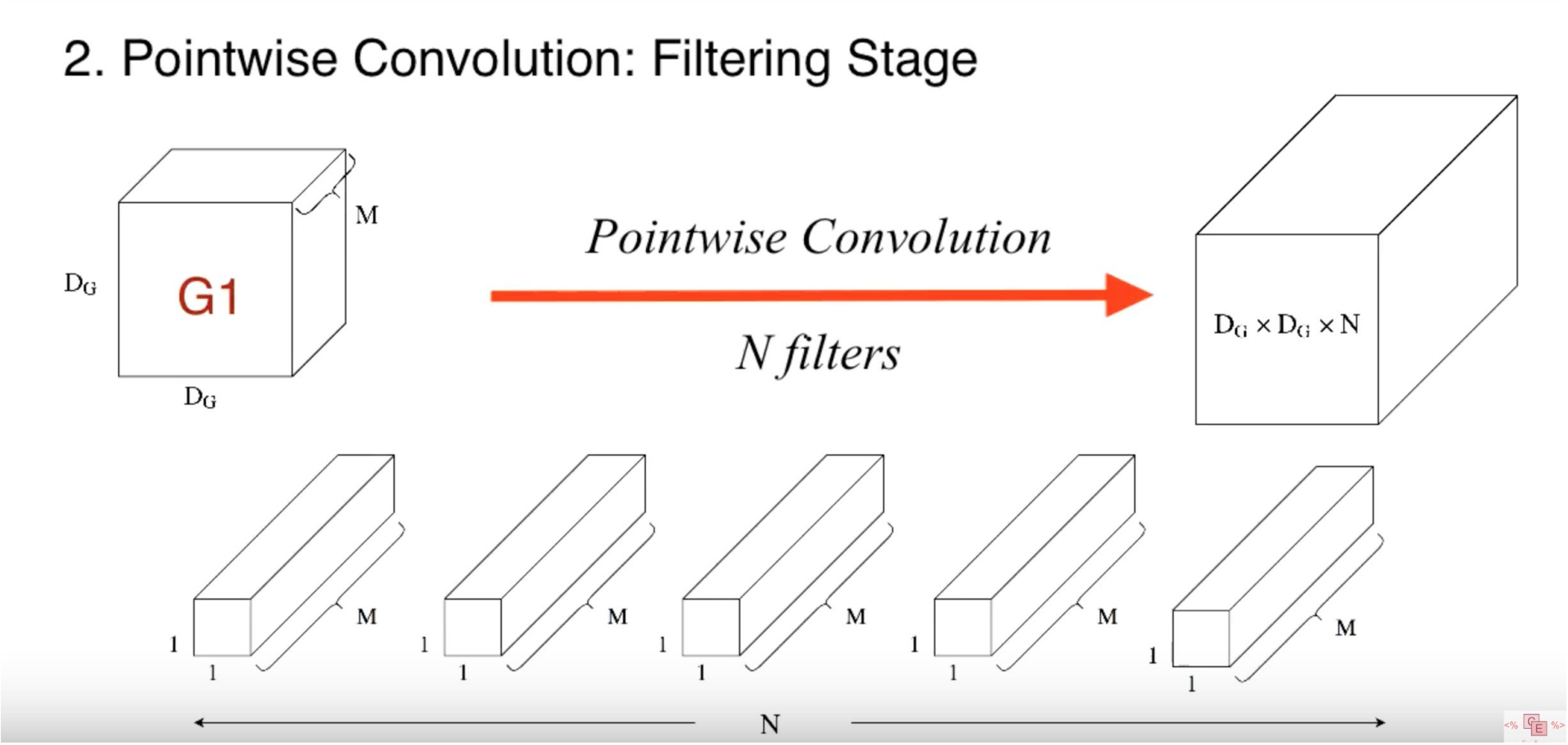


Depthwise Separable Convolution

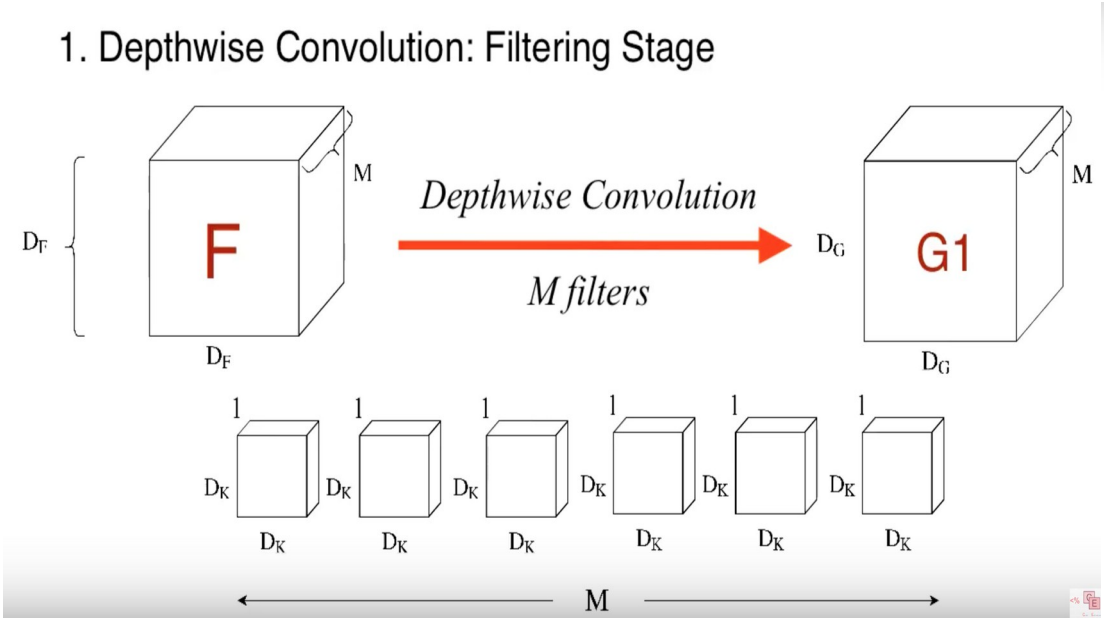
2. Pointwise Convolution: Filtering Stage



Depthwise Separable Convolution



Computations

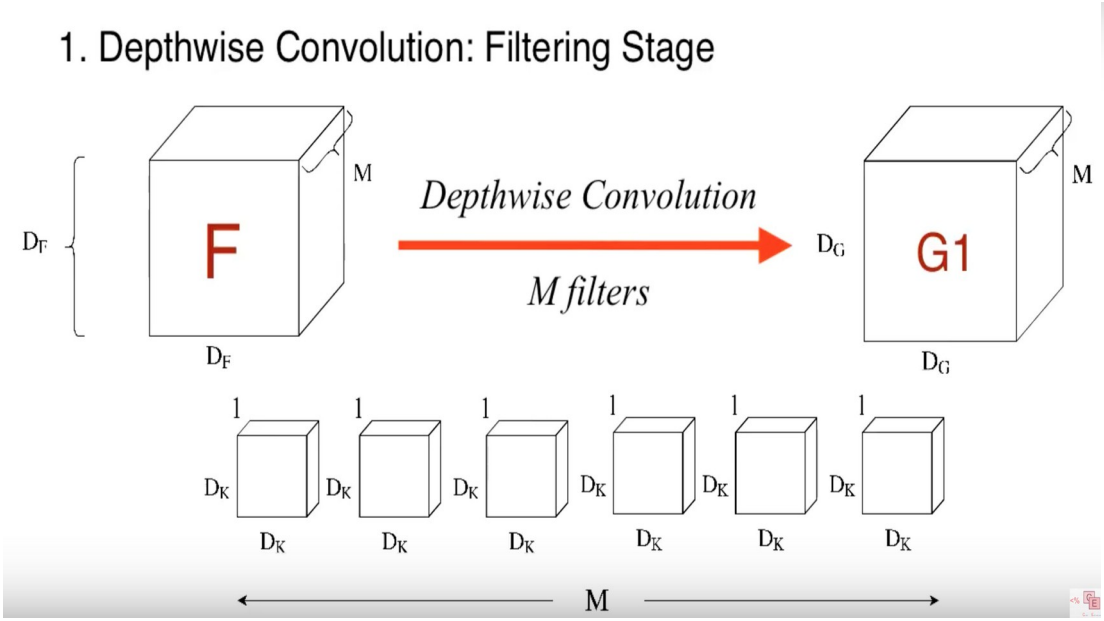


Mults once = D_K^2

Mults 1 Channel = $D_G^2 \times D_K^2$

DC Mults = $M \times D_G^2 \times D_K^2$

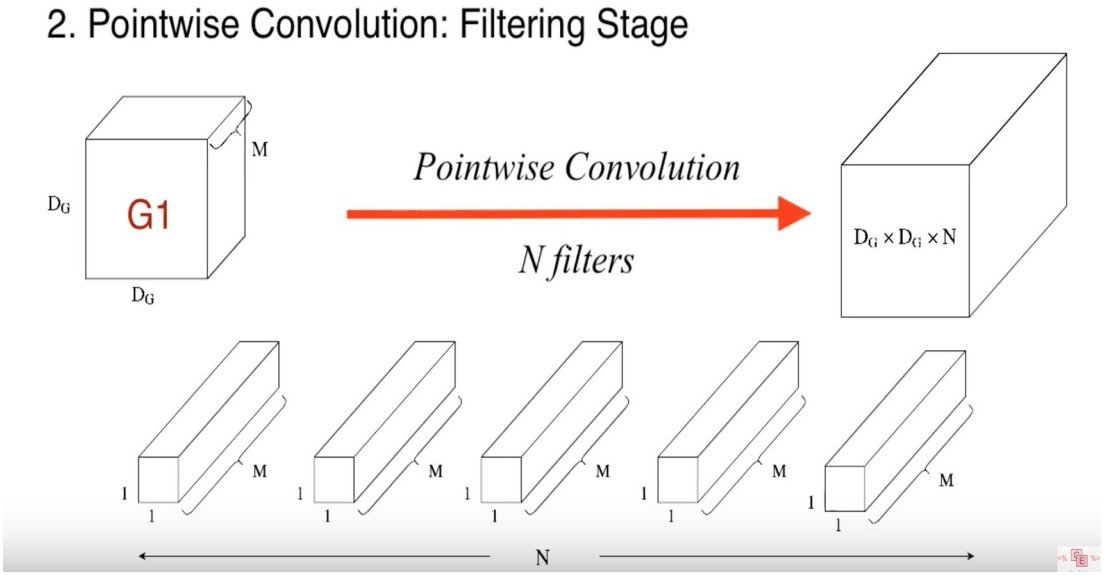
Computations



Mults once = D_K^2

Mults 1 Channel = $D_G^2 \times D_K^2$

DC Mults = $M \times D_G^2 \times D_K^2$

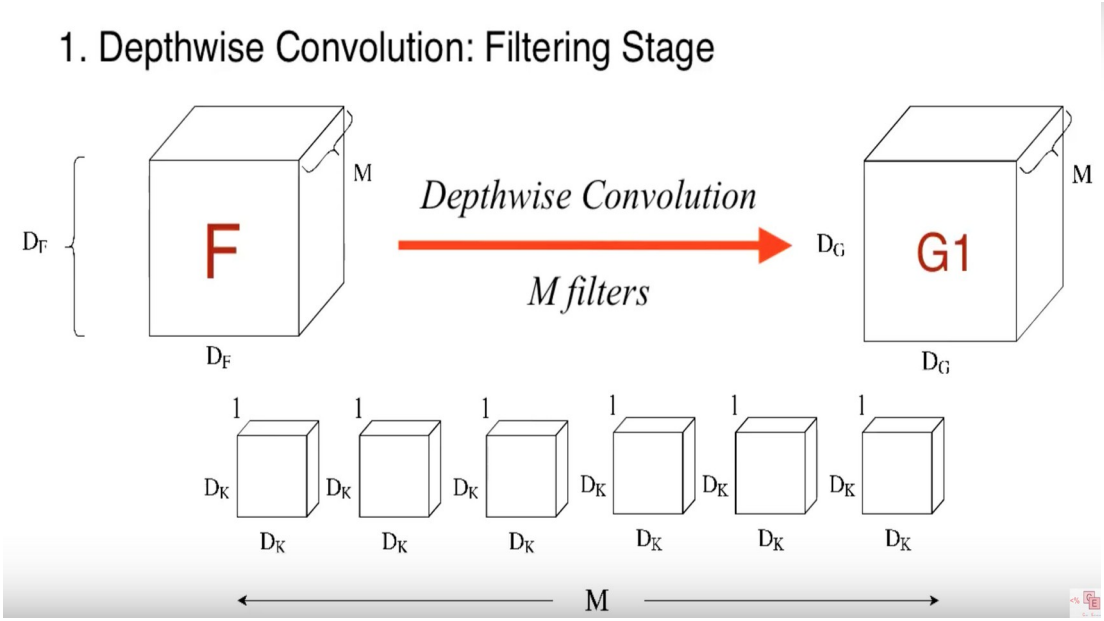


Mults once = M

Mults 1 Kernel = $D_G \times D_G \times M$

PC Mults = $N \times D_G \times D_G \times M$

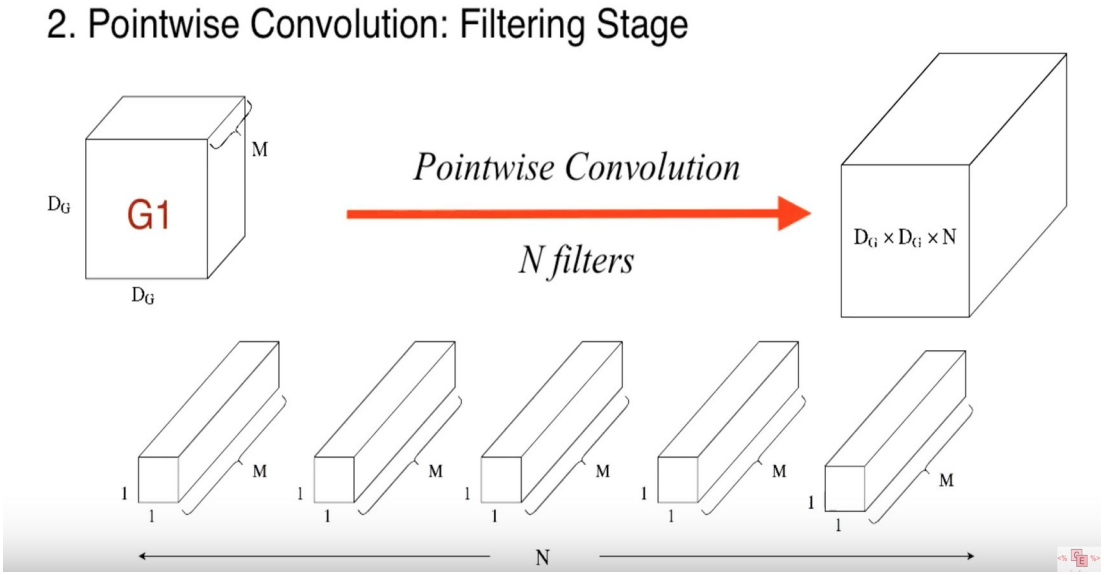
Computations



Mults once = D_K^2

Mults 1 Channel = $D_G^2 \times D_K^2$

DC Mults = $M \times D_G^2 \times D_K^2$



Mults once = M

Mults 1 Kernel = $D_G \times D_G \times M$

PC Mults = $N \times D_G \times D_G \times M$

Total = DC Mults + PC Mults

Computations

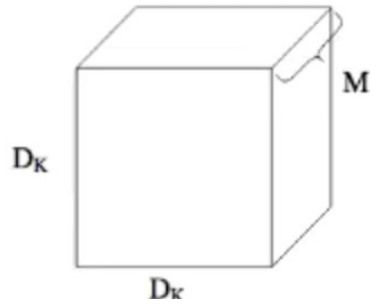
$$\frac{No.Mults\ in\ Depthwise\ Separable\ Conv}{No.Mults\ in\ Standard\ Conv} = \frac{M \times D_G^2\ (D_K^2 + N)}{N \times D_G \times D_G \times D_K \times D_K \times M}$$

$$\frac{No.\ Mults\ in\ Depthwise\ Separable\ Conv}{No.\ Mults\ in\ Standard\ Conv} = \frac{D_K^2 + N}{(D_K^2 \times N)} = \frac{1}{N} + \frac{1}{D_K^2}$$

$$N = 1{,}024 \qquad D_K = 3$$

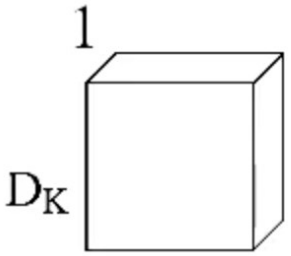
$$\frac{No.\ Mults\ in\ Depthwise\ Separable\ Conv}{No.\ Mults\ in\ Standard\ Conv} = \frac{1}{1024} + \frac{1}{3^2} = 0.112$$

Parameters



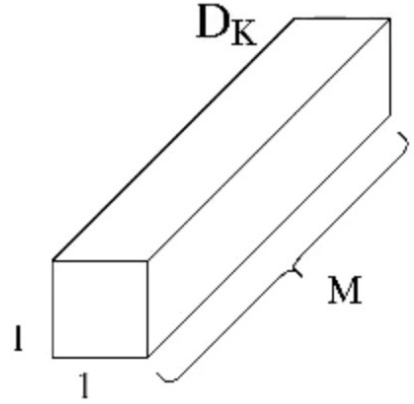
Param 1 Kernel = $D_K^2 \times M$

Param N Kernels = $N \times M \times D_K^2$



Param 1 Kernel = D_K^2

Param M Kernels = $M \times D_K^2$



Param 1 Kernel = M

Param N Kernels = $N \times M$

$M(D_K^2 + N)$

$N = 1{,}024 \quad D_K = 3$

$$\frac{No.params\ in\ Depthwise\ Separable\ Conv}{No.params\ in\ Standard\ Conv} : \frac{1}{N} + \frac{1}{D_K^2}$$

Overall Network

Table 1. MobileNet Body Architecture

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	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Table 2. Resource Per Layer Type

Type	Mult-Adds	Parameters
Conv 1×1	94.86%	74.59%
Conv DW 3×3	3.06%	1.06%
Conv 3×3	1.19%	0.02%
Fully Connected	0.18%	24.33%

More Non-linearities

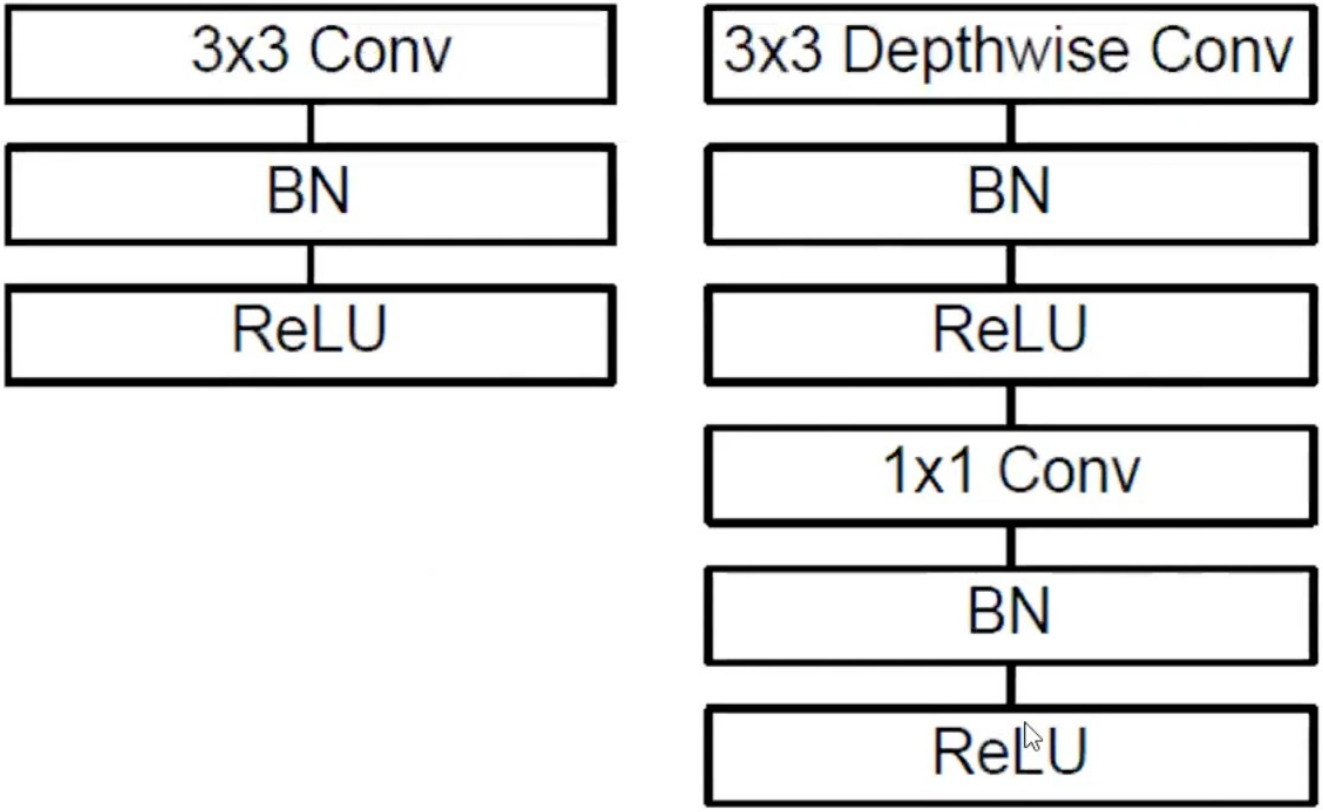


Figure 3. Left: Standard convolutional layer with batchnorm and ReLU. Right: Depthwise Separable convolutions with Depthwise and Pointwise layers followed by batchnorm and ReLU.

Making Mobilenets Even Smaller

- Width Multiplier – Thinner Models
  - For a given layer and width multiplier α, the number of input channels M becomes αM and the number of output channels N becomes αN – where α with typical settings of 1, 0.75, 0.6 and 0.25
- Resolution Multiplier – Reduced Representation
  - The second hyper-parameter to reduce the computational cost of a neural network is a resolution multiplier ρ
  - $0<\rho\leq 1$, which is typically set of implicitly so that input resolution of network is 224, 192, 160 or 128($\rho = 1,\ 0.857, 0.714, 0.571$)
- Computational cost:

$$D_K \times D_K \times \alpha M \times \rho D_F \times \rho D_F + \alpha M \times \alpha N \times \rho D_F \times \rho D_F$$

Code Example

Tensorflow Implementation

```
def _depthwise_separable_conv(inputs,
                              num_pwc_filters,
                              width_multiplier,
                              sc,
                              downsample=False):
  """ Helper function to build the depth-wise separable convolution layer.
  """
  num_pwc_filters = round(num_pwc_filters * width_multiplier)
  _stride = 2 if downsample else 1

  # skip pointwise by setting num_outputs=None
  depthwise_conv = slim.separable_convolution2d(inputs,
                                                num_outputs=None,
                                                stride=_stride,
                                                depth_multiplier=1,
                                                kernel_size=[3, 3],
                                                scope=sc+'/depthwise_conv')

  bn = slim.batch_norm(depthwise_conv, scope=sc+'/dw_batch_norm')
  pointwise_conv = slim.convolution2d(bn,
                                      num_pwc_filters,
                                      kernel_size=[1, 1],
                                      scope=sc+'/pointwise_conv')
  bn = slim.batch_norm(pointwise_conv, scope=sc+'/pw_batch_norm')
  return bn
```

```
with tf.variable_scope(scope) as sc:
  end_points_collection = sc.name + '_end_points'
  with slim.arg_scope([slim.convolution2d, slim.separable_convolution2d],
                      activation_fn=None,
                      outputs_collections=[end_points_collection]):
    with slim.arg_scope([slim.batch_norm],
                        is_training=is_training,
                        activation_fn=tf.nn.relu):
      net = slim.convolution2d(inputs, round(32 * width_multiplier), [3, 3], stride=2, padding='SAME', scope='conv_1')
      net = slim.batch_norm(net, scope='conv_1/batch_norm')
      net = _depthwise_separable_conv(net, 64, width_multiplier, sc='conv_ds_2')
      net = _depthwise_separable_conv(net, 128, width_multiplier, downsample=True, sc='conv_ds_3')
      net = _depthwise_separable_conv(net, 128, width_multiplier, sc='conv_ds_4')
      net = _depthwise_separable_conv(net, 256, width_multiplier, downsample=True, sc='conv_ds_5')
      net = _depthwise_separable_conv(net, 256, width_multiplier, sc='conv_ds_6')
      net = _depthwise_separable_conv(net, 512, width_multiplier, downsample=True, sc='conv_ds_7')

      net = _depthwise_separable_conv(net, 512, width_multiplier, sc='conv_ds_8')
      net = _depthwise_separable_conv(net, 512, width_multiplier, sc='conv_ds_9')
      net = _depthwise_separable_conv(net, 512, width_multiplier, sc='conv_ds_10')
      net = _depthwise_separable_conv(net, 512, width_multiplier, sc='conv_ds_11')
      net = _depthwise_separable_conv(net, 512, width_multiplier, sc='conv_ds_12')

      net = _depthwise_separable_conv(net, 1024, width_multiplier, downsample=True, sc='conv_ds_13')
      net = _depthwise_separable_conv(net, 1024, width_multiplier, sc='conv_ds_14')
      net = slim.avg_pool2d(net, [7, 7], scope='avg_pool_15')

  end_points = slim.utils.convert_collection_to_dict(end_points_collection)
  net = tf.squeeze(net, [1, 2], name='SpatialSqueeze')
  end_points['squeeze'] = net
  logits = slim.fully_connected(net, num_classes, activation_fn=None, scope='fc_16')
  predictions = slim.softmax(logits, scope='Predictions')

  end_points['Logits'] = logits
```

Results

Table 4. Depthwise Separable vs Full Convolution MobileNet

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
Conv MobileNet	71.7%	4866	29.3
MobileNet	70.6%	569	4.2

Table 5. Narrow vs Shallow MobileNet

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
0.75 MobileNet	68.4%	325	2.6
Shallow MobileNet	65.3%	307	2.9

Table 6. MobileNet Width Multiplier

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 7. MobileNet Resolution

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 8. MobileNet Comparison to Popular Models

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 9. Smaller MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
0.50 MobileNet-160	60.2%	76	1.32
Squeezenet	57.5%	1700	1.25
AlexNet	57.2%	720	60