<u>MobileNet</u>

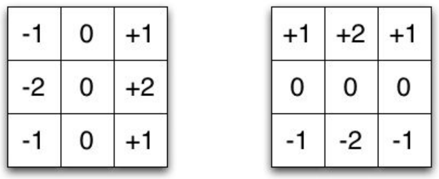
MobileNet is an efficient model for mobile and embedded vision applications. MobileNet is based on a streamlined architecture to build light weight deep neural networks, it also introduces two hyper-parameters which efficiently tradeoff between latency and accuracy. MobileNet uses less variables and less computation power than popular deep models with a slight drop in accuracy which makes it suitable for devices such as mobile phones which have less computation power and also require outputs very fast. This is supported by a wide variety of experiments for various applications.

<u>Architecture</u>

MobileNet architecture is built using depth wise separable filters which comprises of most of the architecture's layers.

<u>Separable Convolutions</u>

In a separable convolution, we can split the kernel operation into multiple steps such that we have the same output using less parameters. For example, suppose we have 2 matrices x and y.



-1	0	+1
-2	0	+2
-1	0	+1

x filter

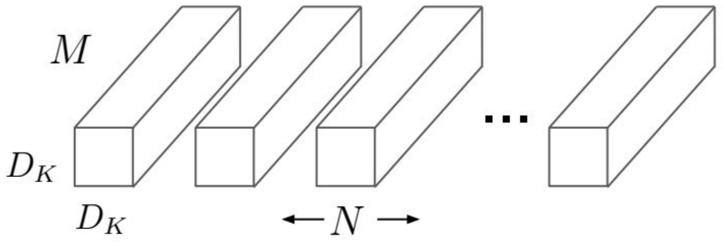
+1	+2	+1
0	0	0
-1	-2	-1

y filter

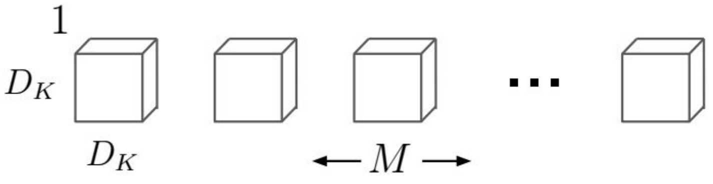
Now suppose we have to perform a kerel operation (element wise multiplication and then summation) using x and y. We'll get the output to be 0. Now we can get the same output by performing kernel operation on [-1, 0, 1] and [1, 0, -1]. So we have divided a 2D convolution into a 1D convolution and also we have reduced the number of multiplication operations to 3 from 9 and the number of parameters to 6 from 9. This is a spatial separable convolution.

<u>Depthwise separable convolution</u>

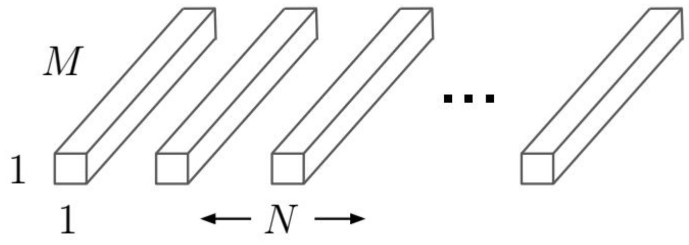
Depthwise separable convolutions are a form of factorised convolutions which factorize a convolution into a depthwise convolution and then a 1x1 convolution called a pointwise convolution. Suppose we have an **Input Image I** with dimensions Di x Di x M where Di is the spatial width and M is the number of input channels, and the **output image O** having dimensions Di x Di x N where N is number of output channels and a **Kernel K** with dimensions Dk x Dk x M x N.
A depthwise convolution applies a single filter to each input channel and so it takes an input with dimension Di x Di x M and outputs Di x Di x M, the filter size is Di x Di x 1 and there are M such filters.
Now, pointwise convolution applies 1 x 1 convolution to combine the outputs of depthwise convolution, so it takes Di x Di x M and outputs Di x Di x N. So, we have M such filters having dimensions 1 x 1 x N.
So the total computation cost is : **Dk x Dk x M x Di x Di + M x N x Di x Di** whereas for a standard convolutional layer the cost is : **Di x Di x M x N x Dk x Dk**.



(a) Standard Convolution Filters



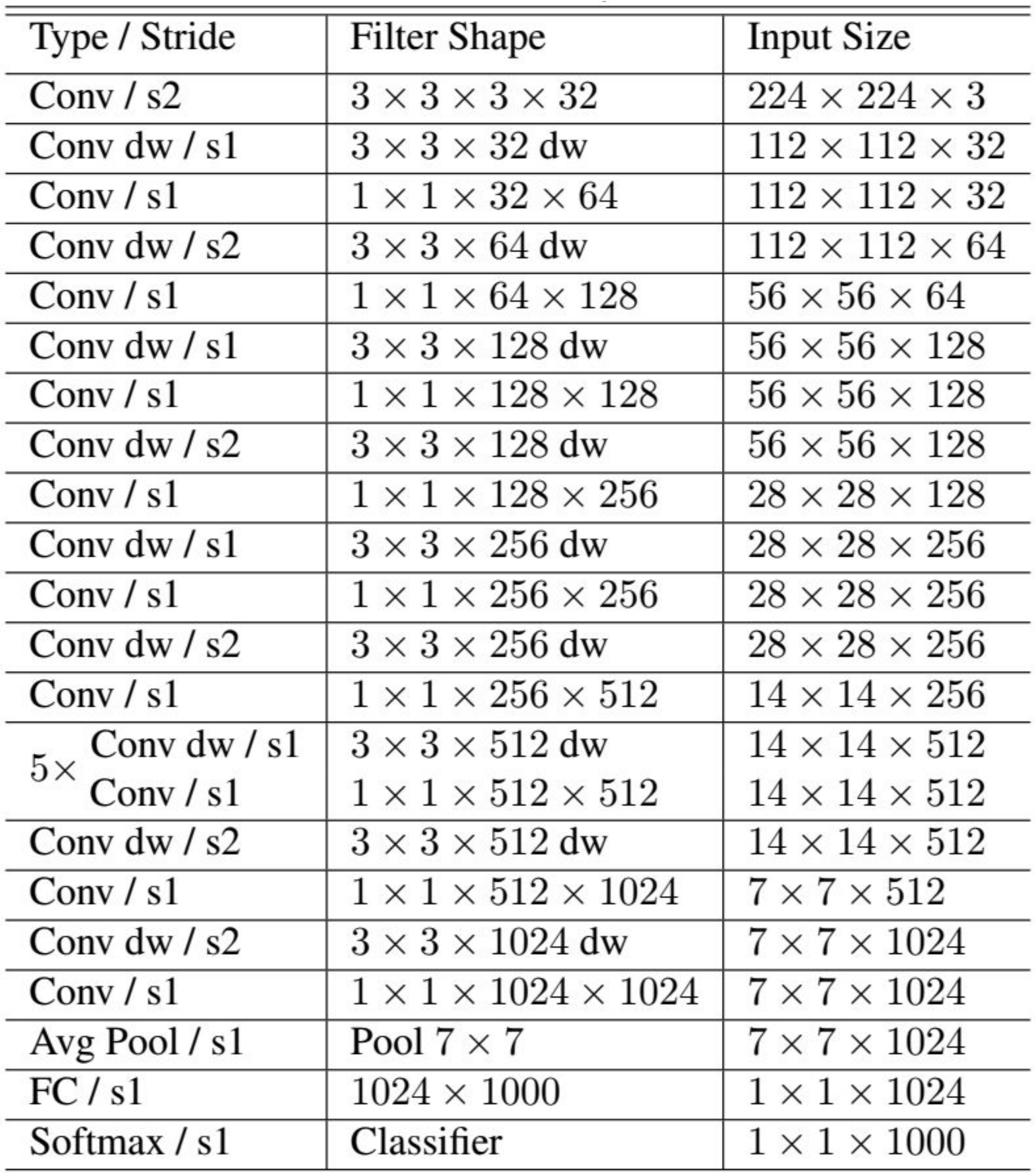
(b) Depthwise Convolutional Filters



(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

Architecture

After every depth wise separable layer batchnorm and ReLU on-linearity are applied. There is also a standard convolutional layer at the beginning which is also used for Downsampling and the last layer is a fully connected layer. Downsampling is also handled with strided convolution in the depthwise convolution (i.e. using stride > 1). Also just before the last fully connected layer average pool is used for reducing spatial resolution to 1.



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	$14 \times 14 \times 512$
5× 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	$1 \times 1 \times 1000$

MobileNet has a total of 28 layers. The optimizer used is RMSprop with asynchronous gradient descent. MobileNet spends 95% of it's computation time in 1 x 1 layer which has 75% of the model's parameters.

Additional Hyper-Parameters

2 additional hyper parameters are introduced to make the model less computationally expensive with a slight drop in accuracy. So, these hyper-parameters can be set according to the latency and computation time and power required.

Width Multiplier (α)

α makes the network even thinner by reducing the number of channels uniformly at each layer. So, α changes the input channels to α x M and output channels into α x N. The total cost for a layer becomes Dk x Dk x α x M x Di x Di + α x M x α x N x Di x Di. Where $\alpha \in (0, 1]$ with typical setting of 0.75, 0.5, 0.25. $\alpha = 1$ is baseline model and $\alpha < 1$ are reduced MobileNets. Width multiplier has the effect of reducing computational cost and the number of parameters quadratically by roughly α^2 .

Resolution Multiplier (ρ)

ρ makes the network less computationally expensive by reducing the resolution of input image and the internal representation of every layer at each layer. So, ρ changes the input resolution to α x Di and output resolution to α x Di.
The Cost for a layer becomes Dk x Dk x M x ρ x Di x ρ x Di + M x N x ρ x Di x ρ x Di. Where $\rho \in (0, 1]$. $\rho = 1$ is baseline model and $\rho < 1$ are reduced MobileNets. Width multiplier has the effect of reducing computational cost and the number of parameters quadratically by roughly ρ^2 .

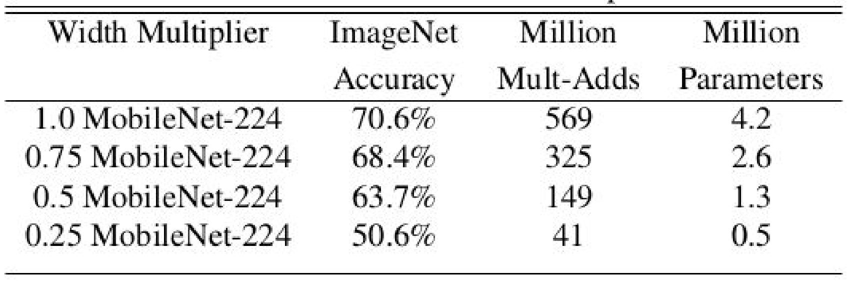
So, **Total Cost = Dk x Dk x α x M x ρ x Di x ρ x Di + α x M x α x N x ρ x Di x ρ x Di.**

Experiments

Hyperparameters

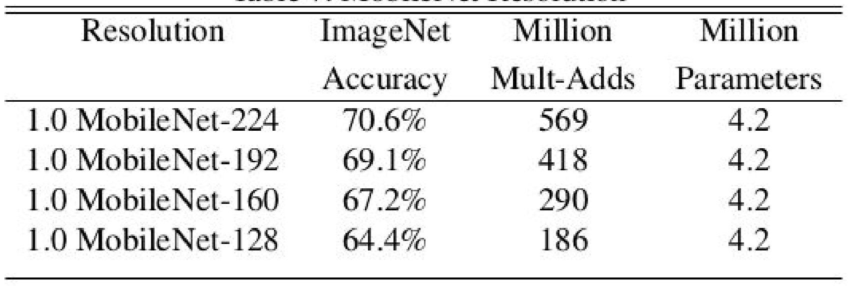
Testing different combinations of these hyper-parameters on ImageNet dataset.

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

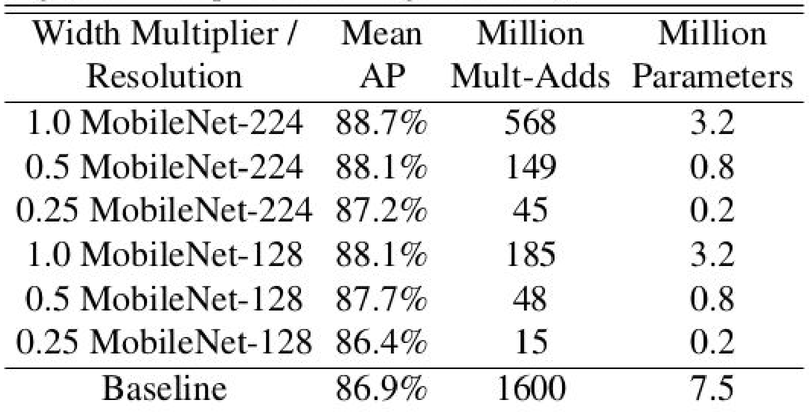
Table 7. MobileNet Resolution



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

Testing different combinations of hyper-parameters for face-attribute classifier.

Table 12. Face attribute classification using the MobileNet architecture. Each row corresponds to a different hyper-parameter setting (width multiplier α and image resolution).



Width Multiplier / Resolution	Mean AP	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	88.7%	568	3.2
0.5 MobileNet-224	88.1%	149	0.8
0.25 MobileNet-224	87.2%	45	0.2
1.0 MobileNet-128	88.1%	185	3.2
0.5 MobileNet-128	87.7%	48	0.8
0.25 MobileNet-128	86.4%	15	0.2
Baseline	86.9%	1600	7.5

<u>Comparison with different models</u>

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

Table 10. MobileNet for Stanford Dogs

Model	Top-1 Accuracy	Million Mult-Adds	Million Parameters
Inception V3 [18]	84%	5000	23.2
1.0 MobileNet-224	83.3%	569	3.3
0.75 MobileNet-224	81.9%	325	1.9
1.0 MobileNet-192	81.9%	418	3.3
0.75 MobileNet-192	80.5%	239	1.9

Table 11. Performance of PlaNet using the MobileNet architecture. Percentages are the fraction of the Im2GPS test dataset that were localized within a certain distance from the ground truth. The numbers for the original PlaNet model are based on an updated version that has an improved architecture and training dataset.

Scale	Im2GPS [7]	PlaNet [35]	PlaNet MobileNet
Continent (2500 km)	51.9%	77.6%	79.3%
Country (750 km)	35.4%	64.0%	60.3%
Region (200 km)	32.1%	51.1%	45.2%
City (25 km)	21.9%	31.7%	31.7%
Street (1 km)	2.5%	11.0%	11.4%

Conclusion

MobileNet is an architecture suitable for devices with limited computational power. It has a slightly lower accuracy than the state of the art models like inception v3 but it’s also very cheap in terms of computational expense and it outperforms the older state of the art models.