MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications

Date of Publication: 17th April, 2017

Journal: IEEE Transactions on Mobile Computing

Abstract

1. MobileNets are a class of efficient models for mobile and embedded vision applications.
2. MobileNets are based on a streamlined architecture that uses depthwise separable convolutions to build light weight deep neural networks.
3. Since 2017, MobileNets have been used across a wide range of applications which will be discussed later.
4. This paper introduces two simple global hyper-parameters that efficiently trade off between latency and accuracy and allow the model builder to choose the right sized model for their application based on the constraints of the problem.

Introduction

1. The general trend has been to make deeper and more complicated networks in order to achieve higher accuracy.
2. In many real world applications such as robotics, self-driving car and augmented reality, the recognition tasks need to be carried out in a timely fashion on a computationally limited platform.
3. This paper describes an efficient network architecture in order to build models that can be easily matched to the design requirements for mobile and embedded vision applications.
4. Two hyper-parameters ***width multiplier*** and *resolution multiplier* are introduced to define smaller and more efficient MobileNets.

How to Improve the efficiency of the neural networks?

Earlier Efforts......Compressing pretrained networks or training small networks directly.

Efforts Made by the Paper.....

But, this paper proposes a class of network architectures that allows a model developer to specifically choose a small network that matches the resource restrictions (latency, size) for their application.

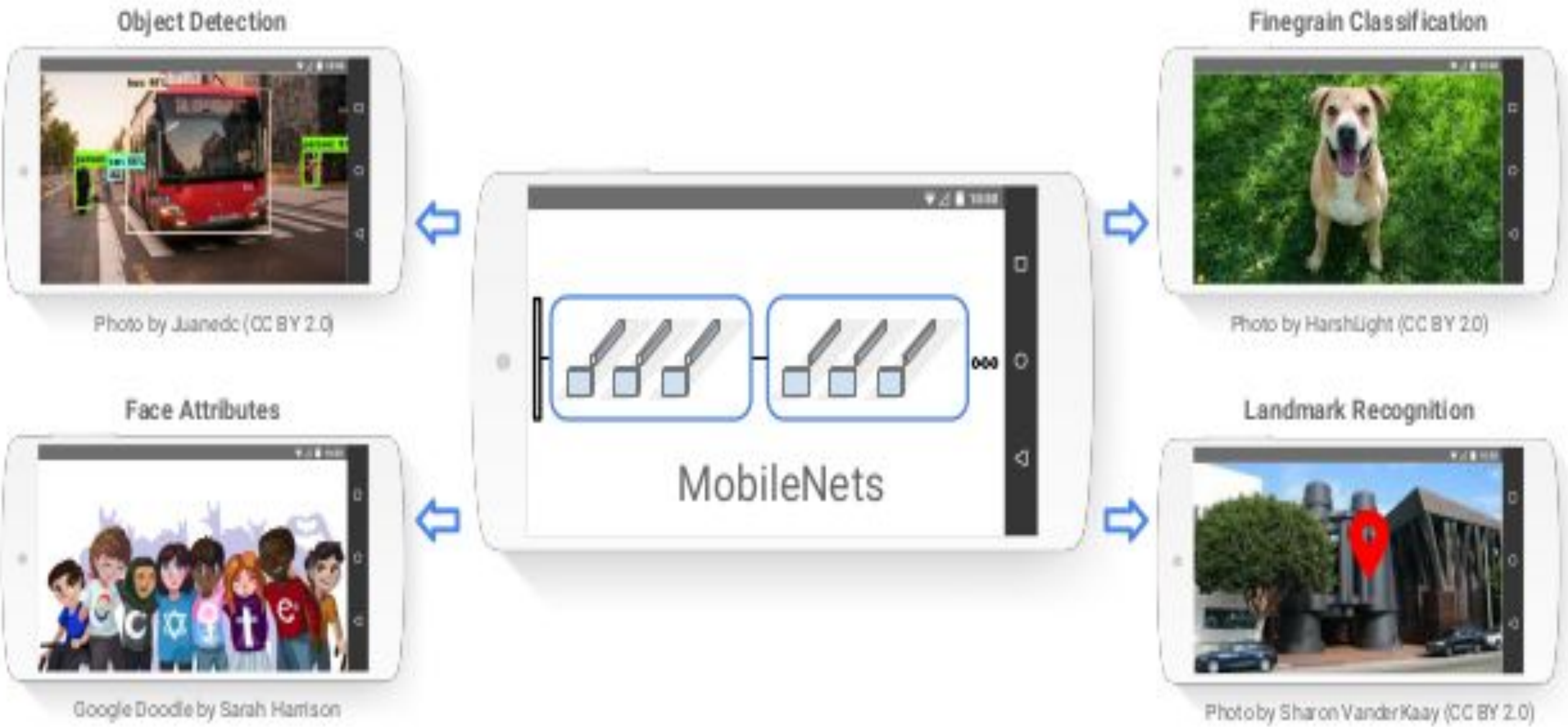
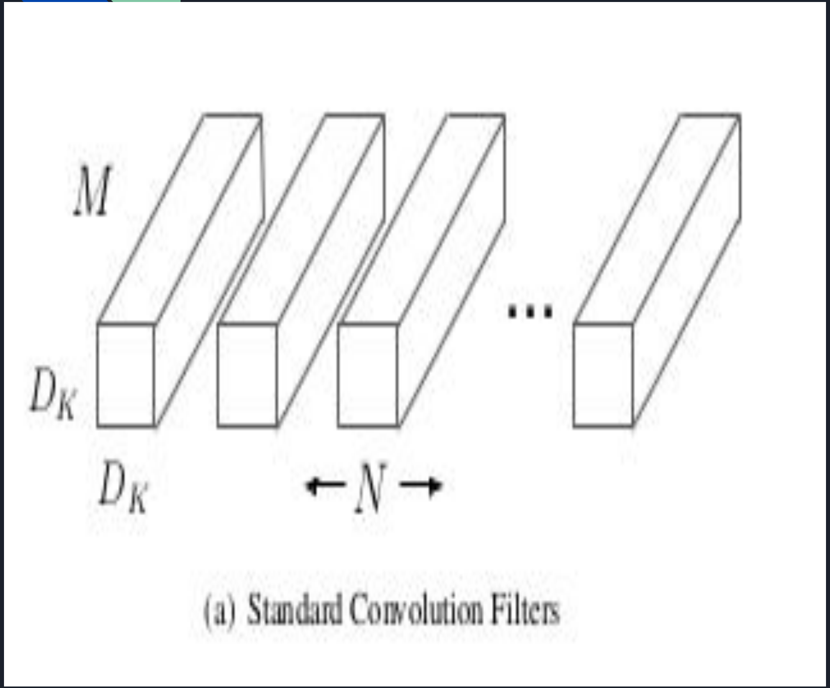

Object Detection
Finegrain Classification
Photo by Juanedc (CC BY 2.0)
Photo by HarshLight (CC BY 2.0)
MobileNets
Face Attributes
Landmark Recognition
Google Doodle by Sarah Harrison
Photo by Sharon VanderKaay (CC BY 2.0)

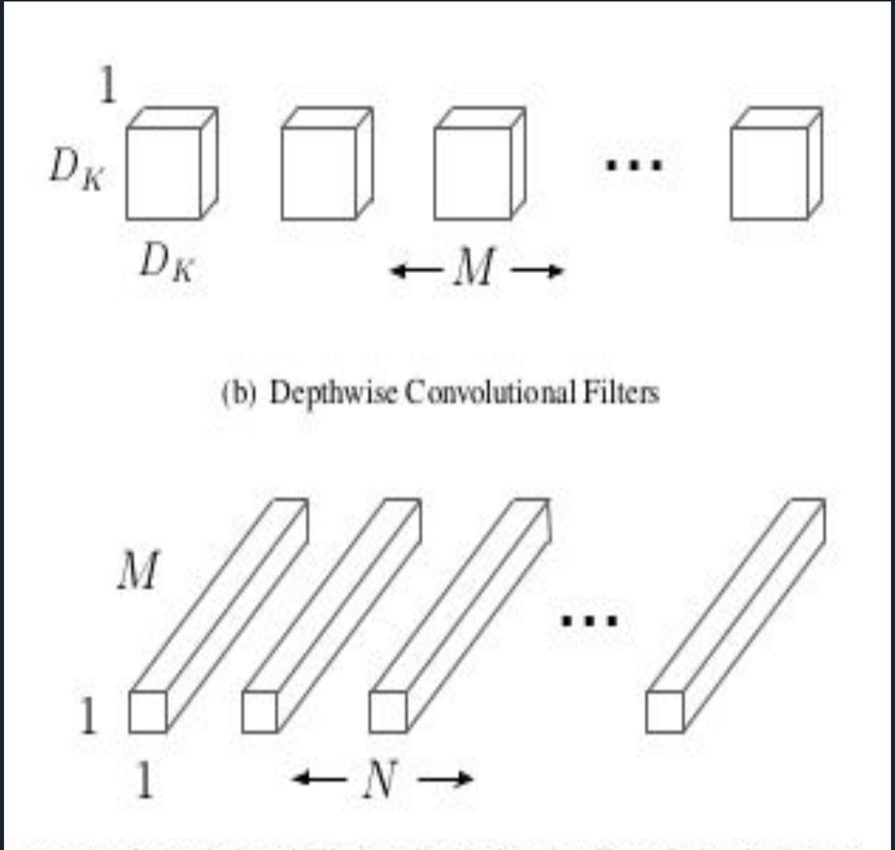
Figure 1. MobileNet models can be applied to various recognition tasks for efficient on device intelligence.

Mobile Net Architecture

1. Uses depthwise separable convolutions, form of factorized convolutions which factorize a standard convolution into a depthwise convolution and a 1×1 convolution called a pointwise convolution for substantial reduction in computational cost.
2. the depthwise convolution applies a single filter to each input channel and the pointwise convolution then applies a 1×1 convolution to combine the outputs from the depthwise convolution



(a) Standard Convolution Filters



1
D_K
D_K
$\leftarrow M \rightarrow$
...
(b) Depthwise Convolutional Filters
M
1
1
$\leftarrow N \rightarrow$
...

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

Where :

F: Input Feature Map

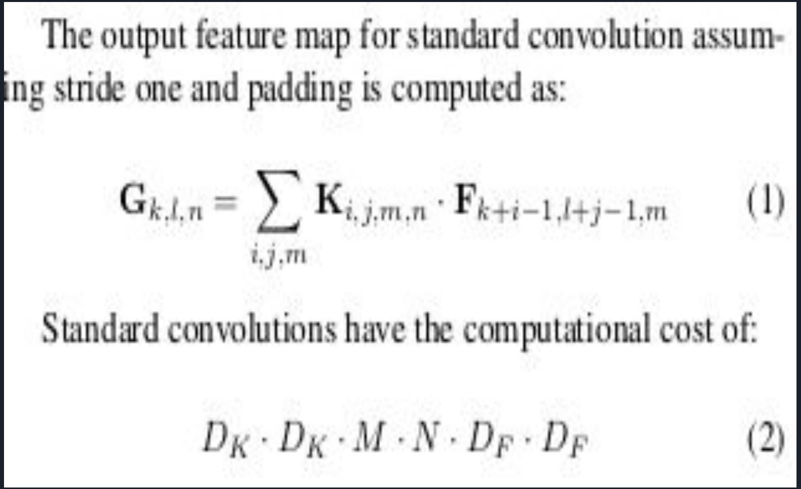
G: Output Feature Map

K: Kernel

Dk : spatial width/height of the kernel K

Df: spatial width and height of the feature maps

M, N: Number of Input and Output Channels respectively



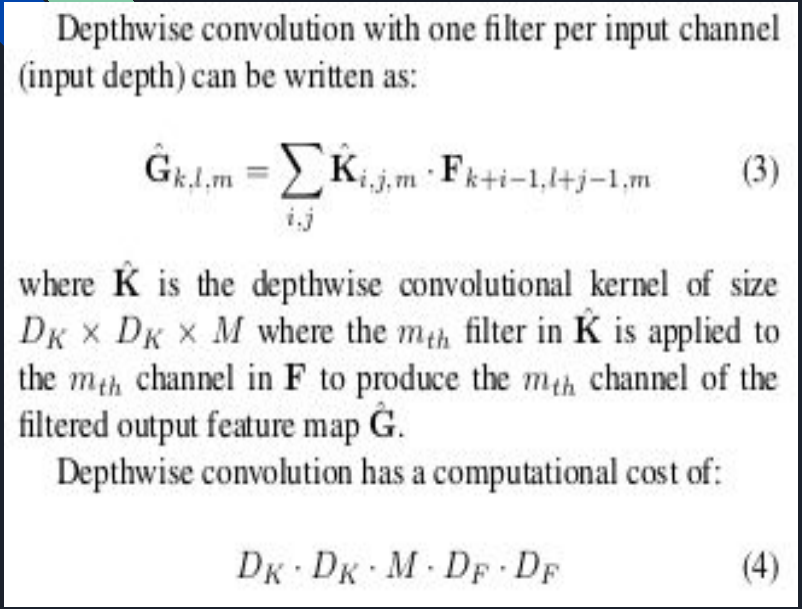
The output feature map for standard convolution assuming stride one and padding is computed as:

$$\mathbf{G}_{k,l,n} = \sum_{i,j,m} \mathbf{K}_{i,j,m,n} \cdot \mathbf{F}_{k+i-1,l+j-1,m} \qquad (1)$$

Standard convolutions have the computational cost of:

$$D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F \qquad (2)$$

Cost Computation.....



Depthwise convolution with one filter per input channel (input depth) can be written as:

$$\hat{\mathbf{G}}_{k,l,m} = \sum_{i,j} \hat{\mathbf{K}}_{i,j,m} \cdot \mathbf{F}_{k+i-1,l+j-1,m} \quad (3)$$

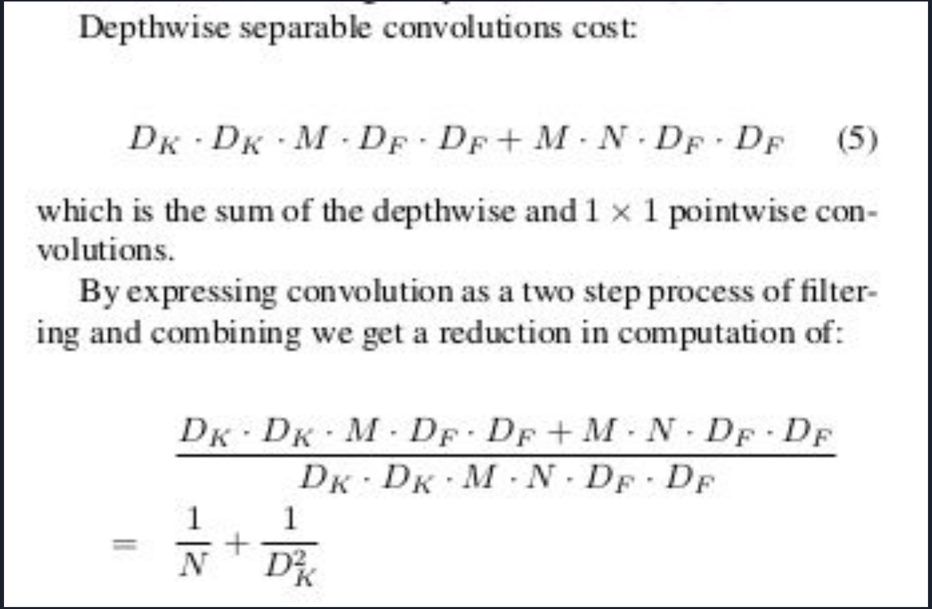
where $\hat{\mathbf{K}}$ is the depthwise convolutional kernel of size $D_K \times D_K \times M$ where the m_{th} filter in $\hat{\mathbf{K}}$ is applied to the m_{th} channel in $\mathbf{F}$ to produce the m_{th} channel of the filtered output feature map $\hat{\mathbf{G}}$.

Depthwise convolution has a computational cost of:

$$D_K \cdot D_K \cdot M \cdot D_F \cdot D_F \quad (4)$$

But depthwise convolutions do not produce new features, therefore 1x1 convolutions are needed to linearly combine it's results.

Hence, the final cost and the corresponding reduction in the cost:



Depthwise separable convolutions cost:

$$D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F \quad (5)$$

which is the sum of the depthwise and 1×1 pointwise convolutions.

By expressing convolution as a two step process of filtering and combining we get a reduction in computation of:

$$\frac{D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F}{D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F}$$
$$= \frac{1}{N} + \frac{1}{D_K^2}$$

MobileNet uses 3 × 3 depthwise separable convolutions which uses between 8 to 9 times less computation than standard convolutions.

Network Structure

1. All layers are followed by a batchnorm and ReLU nonlinearity with the exception of the final fully connected layer which has no nonlinearity and feeds into a softmax layer for classification.
2. Down sampling is handled with strided convolution in the depthwise convolutions as well as in the first layer.
3. Average pooling reduces the spatial resolution to 1 before the fully connected layer. MbileNet has 28 layers.

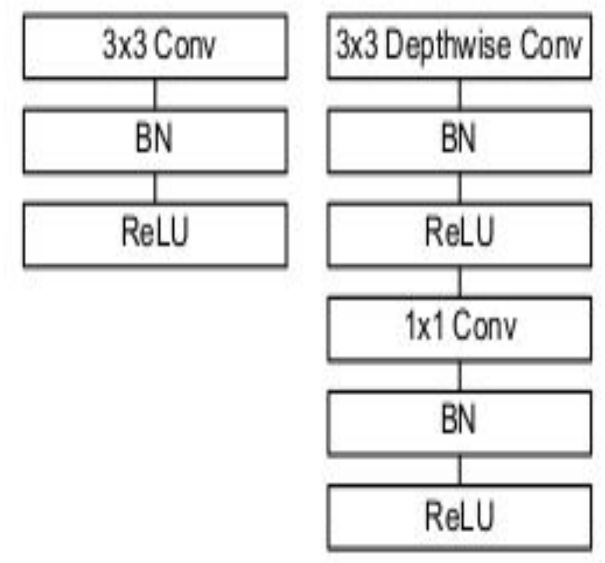


Figure 3. Left: Standard convolutional layer with batchnorm and ReLU. Right: Depthwise Separable convolutions with Depthwise and Pointwise layers followed by batchnorm and ReLU.

Network Structure

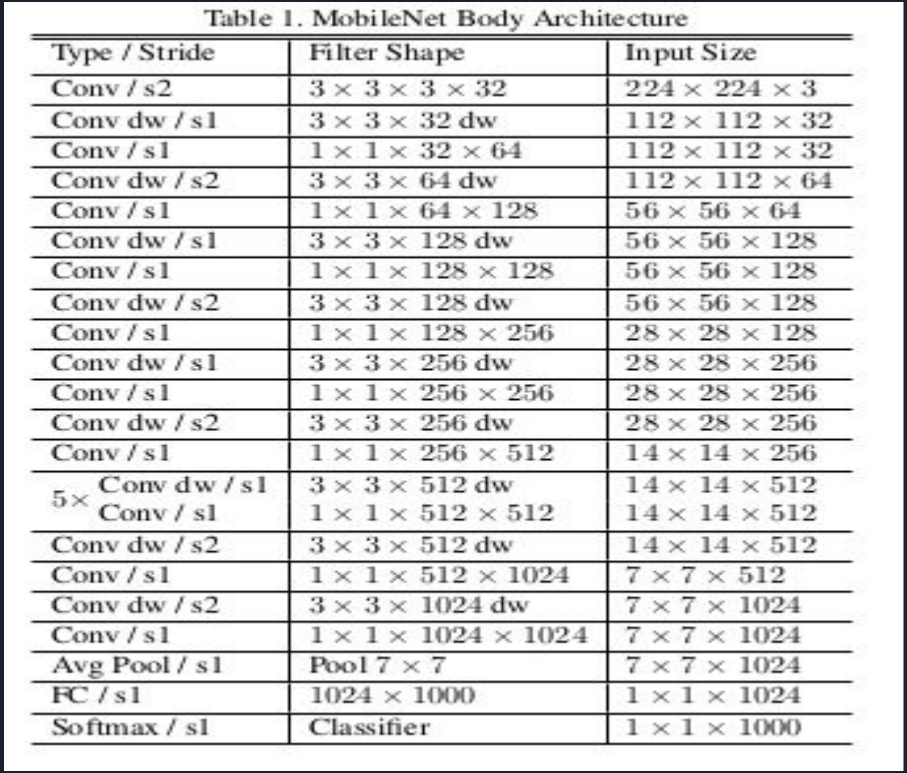
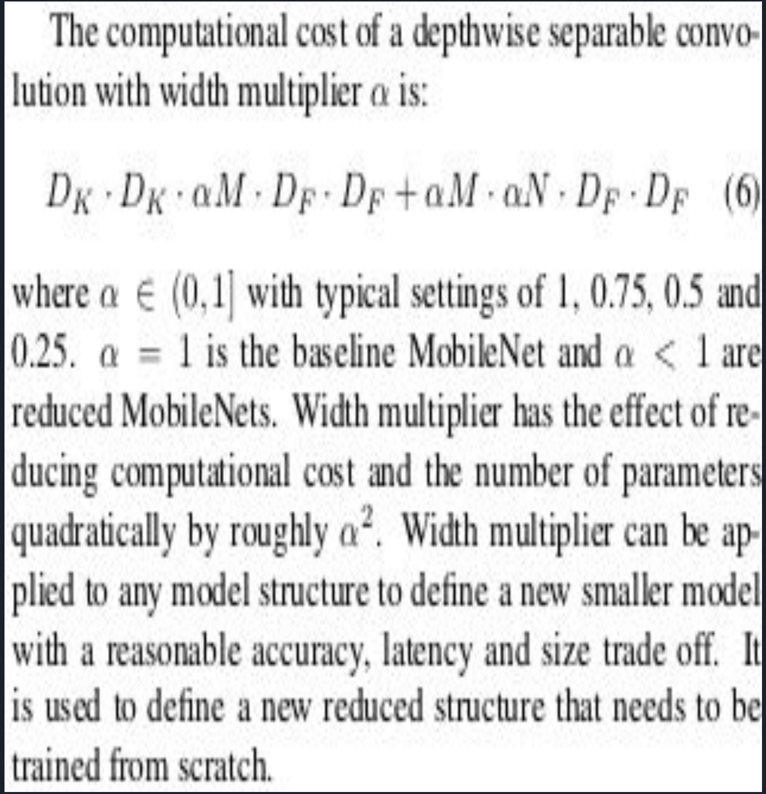


Table 1. MobileNet Body Architecture

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

Width Multiplier:Thinner Models

1. MobileNet architecture is already small and has low latency, many times a specific use case may require the model to be smaller and faster.
2. a very simple parameter α called width multiplier, is introduced.
3. The role of the width multiplier α is to thin a network uniformly at each layer.
4. 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 .



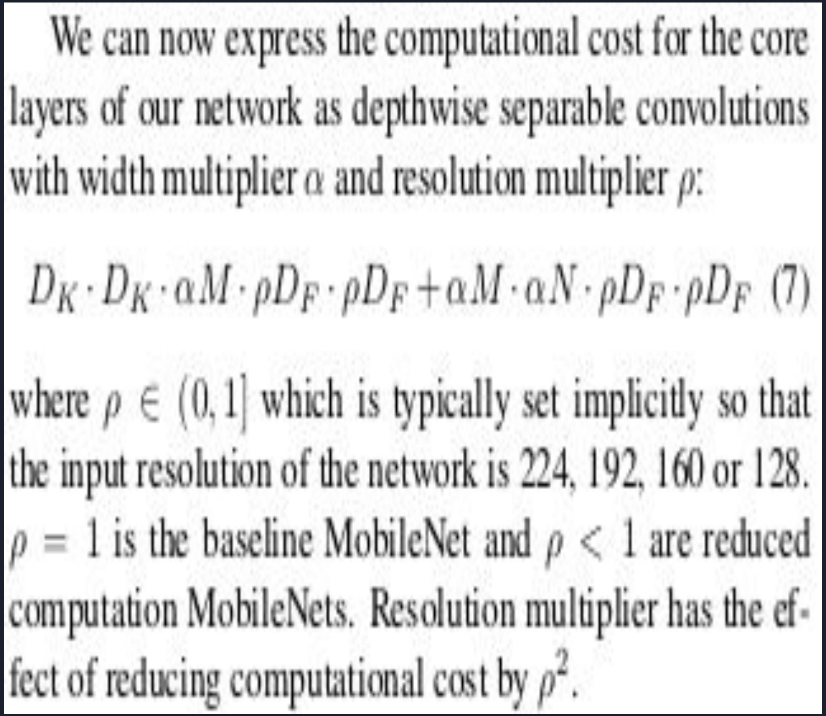
The computational cost of a depthwise separable convolution with width multiplier α is:

$$D_K \cdot D_K \cdot \alpha M \cdot D_F \cdot D_F + \alpha M \cdot \alpha N \cdot D_F \cdot D_F \quad (6)$$

where $\alpha \in (0,1]$ with typical settings of 1, 0.75, 0.5 and 0.25. $\alpha = 1$ is the baseline MobileNet 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. Width multiplier can be applied to any model structure to define a new smaller model with a reasonable accuracy, latency and size trade off. It is used to define a new reduced structure that needs to be trained from scratch.

Resolution Multiplier: Reduced Representation

1. The second hyper-parameter to reduce the computational cost of a neural network is a resolution multiplier ρ.
2. It is applied this to the input image and the internal representation of every layer is subsequently reduced by the same multiplier.



We can now express the computational cost for the core layers of our network as depthwise separable convolutions with width multiplier α and resolution multiplier ρ:

$$D_K \cdot D_K \cdot \alpha M \cdot \rho D_F \cdot \rho D_F + \alpha M \cdot \alpha N \cdot \rho D_F \cdot \rho D_F \quad (7)$$

where $\rho \in (0, 1]$ which is typically set implicitly so that the input resolution of the network is 224, 192, 160 or 128. $\rho = 1$ is the baseline MobileNet and $\rho < 1$ are reduced computation MobileNets. Resolution multiplier has the effect of reducing computational cost by ρ^2.

Depthwise Separable vs Full MobileNet

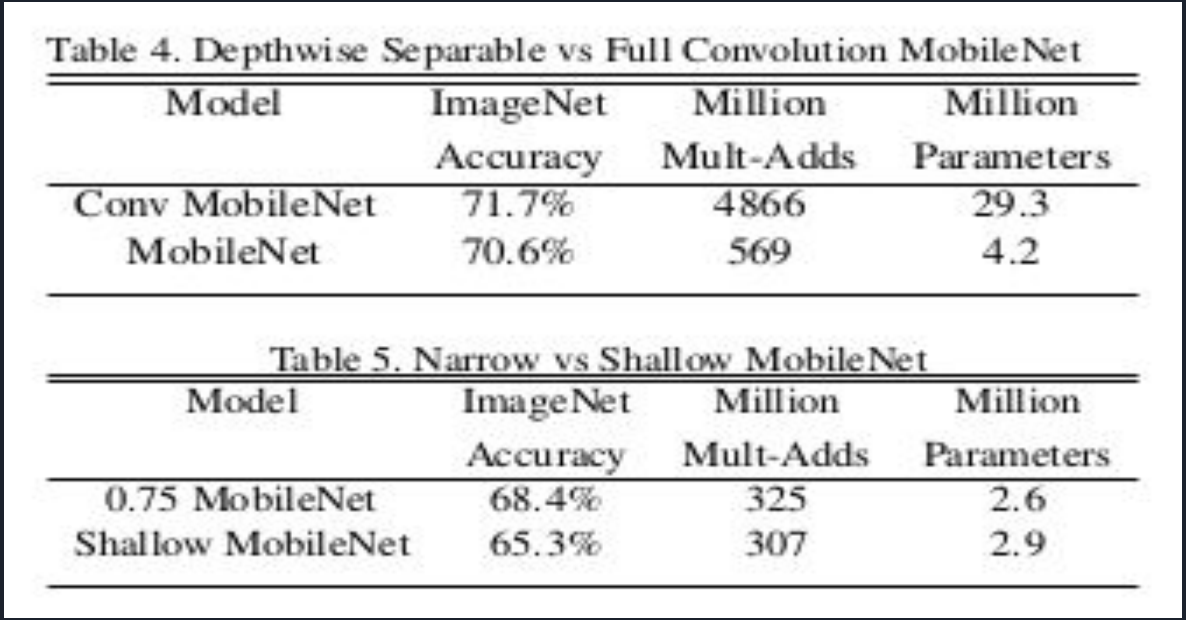


Table 4. Depthwise Separable vs Full Convolution MobileNet

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

Results

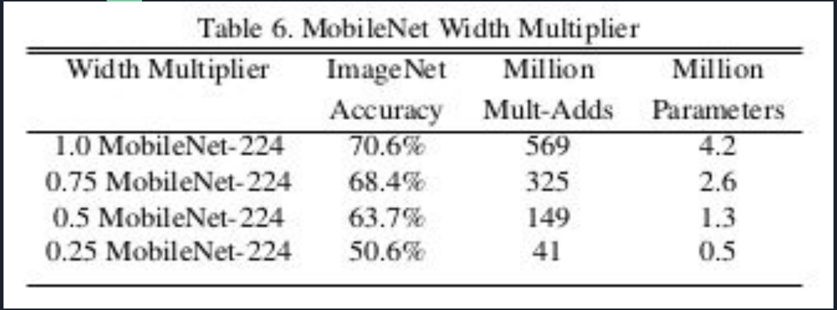


Table 6. MobileNet Width Multiplier

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

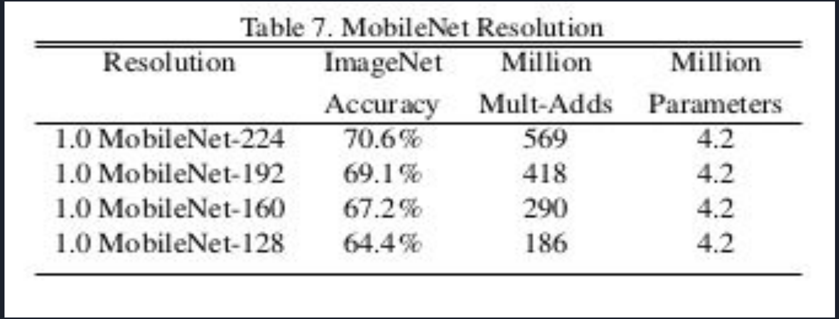


Table 7. MobileNet Resolution

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 6 shows the accuracy, computation and size trade offs of shrinking the MobileNet architecture with the width multiplier α. Accuracy drops off smoothly until the architecture is made too small at α = 0.25.

Table 7 shows the accuracy, computation and size trade-offs for different resolution multipliers by training MobileNets with reduced input resolutions. Accuracy drops off smoothly across resolution.

Results: Fine-Grained Recognition

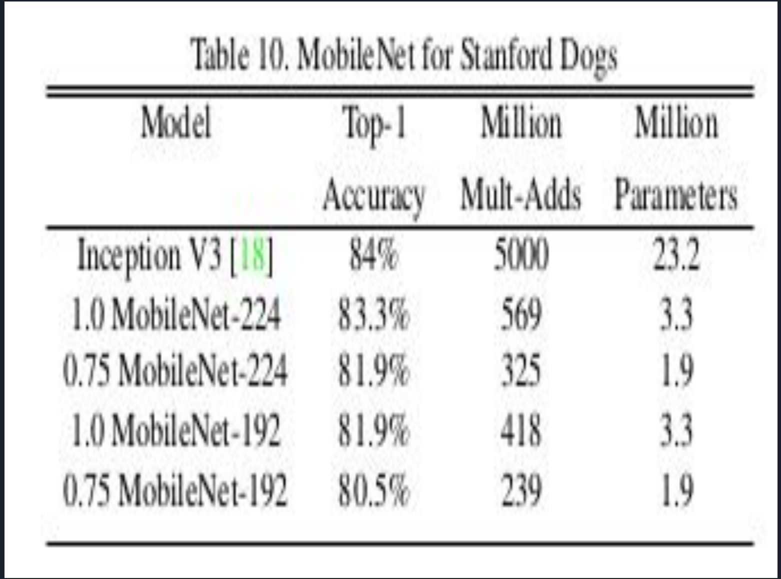


Table 10. MobileNet for Stanford Dogs

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

We train MobileNet for fine grained recognition on the Stanford Dogs dataset The noisy web data is used to pretrain a fine grained dog recognition model and then fine tune the model on the Stanford Dogs training set. Results on Stanford Dogstest set are in Table 10. MobileNet can almost achieve the state of the art results

Results: Large Scale GeoLocalizations

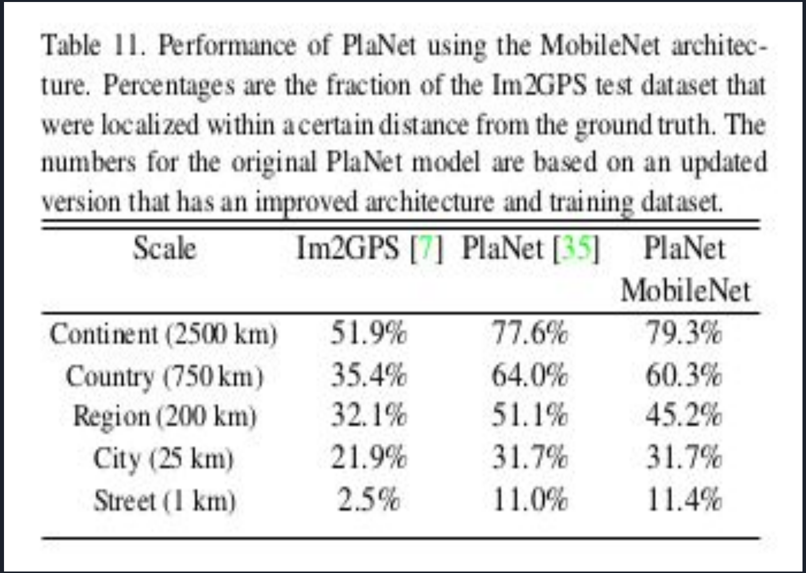


Table 11. Performance of PlaNet using the MobileNet architecture. Percentages are the fraction of the Im2GPS test dataset that were localized within a certain distance from the ground truth. The numbers for the original PlaNet model are based on an updated version that has an improved architecture and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

PlaNet casts the task of determining where on earth a photo was taken as a classification problem. The approach divides the earth into a grid of geographic cells that serve as the target classes and trains a convolutional neural network geo-tagged photos. PlaNet is retrained using the MobileNet architecture on the same data. As shown in Tab. 11, the MobileNet version delivers only slightly decreased performance compared to PlaNet despite being much more compact.

Conclusion

1. MobileNets reduce computation cost of the neural network training and thus is breaking the stereotype of using large networks for better accuracy.
2. Some of the important design decisions were discussed leading to an efficient model.
3. Smaller and faster MobileNets using width multiplier and resolution multiplier by trading off a reasonable amount of accuracy to reduce size and latency were developed.

Thank You

Roll Number :
1601CS56(Arunika Yadav)