MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications

Andrew G. Howard Menglong Zhu Bo Chen Dmitry Kalenichenko
Weijun Wang Tobias Weyand Marco Andreetto Hartwig Adam

Google Inc.

{howarda,menglong,bochen,dkalenichenko,weijunw,weyand,anm,hadam}@google.com

Abstract

We present a class of efficient models called MobileNets for mobile and embedded vision applications. MobileNets are based on a streamlined architecture that uses depthwise separable convolutions to build light weight deep neural networks. We introduce two simple global hyperparameters that efficiently trade off between latency and accuracy. These hyper-parameters allow the model builder to choose the right sized model for their application based on the constraints of the problem. We present extensive experiments on resource and accuracy tradeoffs and show strong performance compared to other popular models on ImageNet classification. We then demonstrate the effectiveness of MobileNets across a wide range of applications and use cases including object detection, finegrain classification, face attributes and large scale geo-localization.

1. Introduction

Convolutional neural networks have become ubiquitous in computer vision ever since AlexNet [19] popularized deep convolutional neural networks by winning the ImageNet Challenge: ILSVRC 2012 [24]. The general trend has been to make deeper and more complicated networks in order to achieve higher accuracy [27, 31, 29, 8]. However, these advances to improve accuracy are not necessarily making networks more efficient with respect to size and speed. 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.

This paper describes an efficient network architecture and a set of two hyper-parameters in order to build very small, low latency models that can be easily matched to the design requirements for mobile and embedded vision applications. Section 2 reviews prior work in building small models. Section 3 describes the MobileNet architecture and two hyper-parameters width multiplier and resolution multiplier to define smaller and more efficient MobileNets. Section 4 describes experiments on ImageNet as well a variety of different applications and use cases. Section 5 closes with a summary and conclusion.

2. Prior Work

There has been rising interest in building small and efficient neural networks in the recent literature, e.g. [16, 34, 12, 36, 22]. Many different approaches can be generally categorized into either compressing pretrained networks or training small networks directly. 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. MobileNets primarily focus on optimizing for latency but also yield small networks. Many papers on small networks focus only on size but do not consider speed.

MobileNets are built primarily from depthwise separable convolutions initially introduced in [26] and subsequently used in Inception models [13] to reduce the computation in the first few layers. Flattened networks [16] build a network out of fully factorized convolutions and showed the potential of extremely factorized networks. Independent of this current paper, Factorized Networks[34] introduces a similar factorized convolution as well as the use of topological connections. Subsequently, the Xception network [3] demonstrated how to scale up depthwise separable filters to out perform Inception V3 networks. Another small network is Squeezenet [12] which uses a bottleneck approach to design a very small network. Other reduced computation networks include structured transform networks [28] and deep fried convnets [37].

A different approach for obtaining small networks is shrinking, factorizing or compressing pretrained networks. Compression based on product quantization [36], hashing

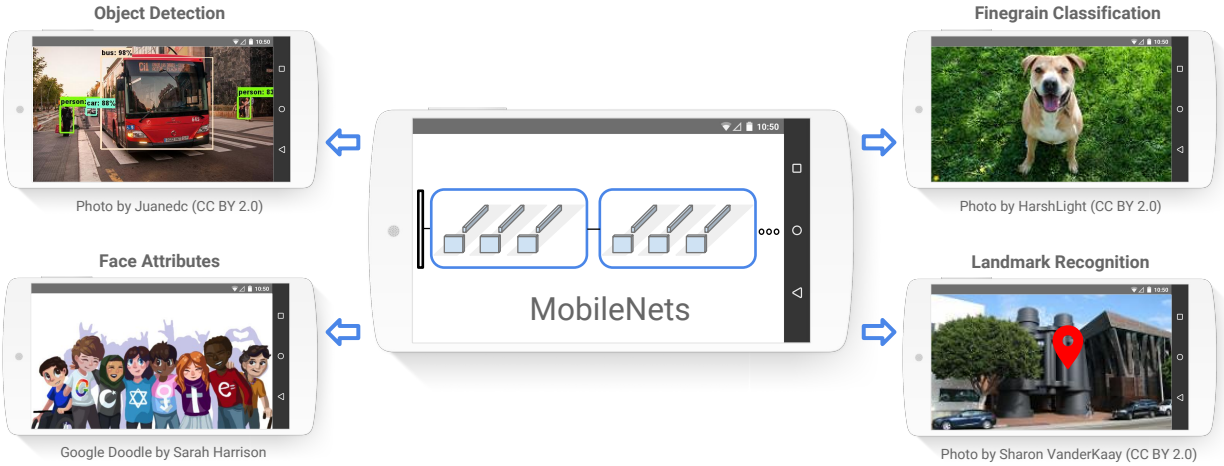


Figure 1. MobileNet models can be applied to various recognition tasks for efficient on device intelligence.

[2], and pruning, vector quantization and Huffman coding [5] have been proposed in the literature. Additionally various factorizations have been proposed to speed up pretrained networks [14, 20]. Another method for training small networks is distillation [9] which uses a larger network to teach a smaller network. It is complementary to our approach and is covered in some of our use cases in section 4. Another emerging approach is low bit networks [4, 22, 11].

3. MobileNet Architecture

In this section we first describe the core layers that MobileNet is built on which are depthwise separable filters. We then describe the MobileNet network structure and conclude with descriptions of the two model shrinking hyperparameters width multiplier and resolution multiplier.

3.1. Depthwise Separable Convolution

The MobileNet model is based on depthwise separable convolutions which is a form of factorized convolutions which factorize a standard convolution into a depthwise convolution and a 1×1 convolution called a pointwise convolution. For MobileNets the depthwise convolution applies a single filter to each input channel. The pointwise convolution then applies a 1×1 convolution to combine the outputs the depthwise convolution. A standard convolution both filters and combines inputs into a new set of outputs in one step. The depthwise separable convolution splits this into two layers, a separate layer for filtering and a separate layer for combining. This factorization has the effect of drastically reducing computation and model size. Figure 2 shows how a standard convolution 2(a) is factorized into a depthwise convolution 2(b) and a 1×1 pointwise convolution 2(c).

A standard convolutional layer takes as input a $D_F \times D_F \times M$ feature map $\mathbf{F}$ and produces a $D_F \times D_F \times N$ feature map $\mathbf{G}$ where D_F is the spatial width and height of a square input feature map[1], M is the number of input channels (input depth), D_G is the spatial width and height of a square output feature map and N is the number of output channel (output depth).

The standard convolutional layer is parameterized by convolution kernel $\mathbf{K}$ of size $D_K \times D_K \times M \times N$ where D_K is the spatial dimension of the kernel assumed to be square and M is number of input channels and N is the number of output channels as defined previously.

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

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

Standard convolutions have the computational cost of:

$$D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F \tag{2}$$

where the computational cost depends multiplicatively on the number of input channels M, the number of output channels N the kernel size $D_k \times D_k$ and the feature map size $D_F \times D_F$. MobileNet models address each of these terms and their interactions. First it uses depthwise separable convolutions to break the interaction between the number of output channels and the size of the kernel.

The standard convolution operation has the effect of filtering features based on the convolutional kernels and combining features in order to produce a new representation. The filtering and combination steps can be split into two steps via the use of factorized convolutions called depthwise

[1]We assume that the output feature map has the same spatial dimensions as the input and both feature maps are square. Our model shrinking results generalize to feature maps with arbitrary sizes and aspect ratios.

separable convolutions for substantial reduction in computational cost.

Depthwise separable convolution are made up of two layers: depthwise convolutions and pointwise convolutions. We use depthwise convolutions to apply a single filter per each input channel (input depth). Pointwise convolution, a simple 1×1 convolution, is then used to create a linear combination of the output of the depthwise layer. MobileNets use both batchnorm and ReLU nonlinearities for both layers.

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

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

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:

$$D_K \cdot D_K \cdot M \cdot D_F \cdot D_F \tag{4}$$

Depthwise convolution is extremely efficient relative to standard convolution. However it only filters input channels, it does not combine them to create new features. So an additional layer that computes a linear combination of the output of depthwise convolution via 1×1 convolution is needed in order to generate these new features.

The combination of depthwise convolution and 1×1 (pointwise) convolution is called depthwise separable convolution which was originally introduced in [26].

Depthwise separable convolutions cost:

$$D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F \tag{5}$$

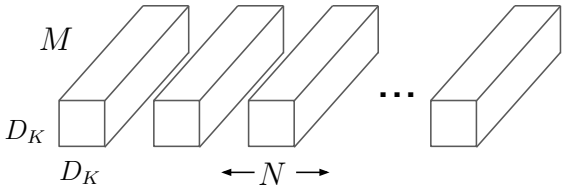
which is the sum of the depthwise and 1×1 pointwise convolutions.

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

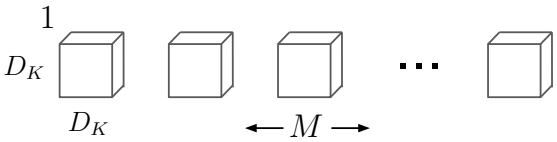
$$\begin{aligned} & \frac{D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F}{D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F} \\ = \quad & \frac{1}{N} + \frac{1}{D_K^2} \end{aligned}$$

MobileNet uses 3×3 depthwise separable convolutions which uses between 8 to 9 times less computation than standard convolutions at only a small reduction in accuracy as seen in Section 4.

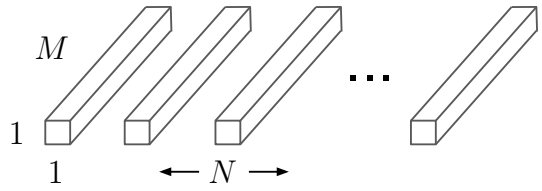
Additional factorization in spatial dimension such as in [16, 31] does not save much additional computation as very little computation is spent in depthwise convolutions.



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



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

Figure 2. The standard convolutional filters in (a) are replaced by two layers: depthwise convolution in (b) and pointwise convolution in (c) to build a depthwise separable filter.

3.2. Network Structure and Training

The MobileNet structure is built on depthwise separable convolutions as mentioned in the previous section except for the first layer which is a full convolution. By defining the network in such simple terms we are able to easily explore network topologies to find a good network. The MobileNet architecture is defined in Table 1. All layers are followed by a batchnorm [13] and ReLU nonlinearity with the exception of the final fully connected layer which has no nonlinearity and feeds into a softmax layer for classification. Figure 3 contrasts a layer with regular convolutions, batchnorm and ReLU nonlinearity to the factorized layer with depthwise convolution, 1×1 pointwise convolution as well as batchnorm and ReLU after each convolutional layer. Down sampling is handled with strided convolution in the depthwise convolutions as well as in the first layer. A final average pooling reduces the spatial resolution to 1 before the fully connected layer. Counting depthwise and pointwise convolutions as separate layers, MobileNet has 28 layers.

It is not enough to simply define networks in terms of a small number of Mult-Adds. It is also important to make sure these operations can be efficiently implementable. For