**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

**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.

우리는 모바일 및 임베디드 비전 애플리케이션을 위한 MobileNets라는 효율적인 모델의 클래스를 제공합니다. MobileNets는 depthwise separable convolutions를 사용하여 경량 심층 신경 네트워크를 구축하는 간소화된 아키텍처를 기반으로 합니다. 대기 시간과 정확성을 효율적으로 상호 조정하는 두 가지 간단한 글로벌 하이퍼파라미터를 소개합니다. 이러한 하이퍼파라미터를 통해 모델 작성자는 문제의 제약 조건에 따라 애플리케이션에 적합한 크기의 모델을 선택할 수 있습니다. 리소스 및 정확도 절충에 대한 광범위한 실험을 제시하고 ImageNet 분류에서 다른 인기 모델에 비해 우수한 성능을 보여줍니다. 그런 다음 객체 감지, 세분화 분류, 얼굴 속성 및 대규모 지리적 지역화 등 광범위한 애플리케이션과 사용 사례에서 MobileNet의 효과를 시연합니다.

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.

Convolutional Neural Network는 ImageNet Challenge에서 우승한 AlexNet[19] 이후 컴퓨터 비전에 보편화되었습니다. ILSVRC 2012 [24]. 일반적으로 더 높은 정확도를 얻기 위해 더 깊고 복잡한 네트워크를 만드는 경향이 있었습니다 [27, 31, 29, 8]. 그러나 정확성을 높이기 위한 이러한 발전으로 인해 크기와 속도 측면에서 네트워크의 효율성이 반드시 향상되는 것은 아닙니다. 로봇, 자율주행차, 증강현실 등 많은 현실에서 인식 작업은 계산적으로 제한된 플랫폼에서 적시에 수행해야 한다.

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.

이 문서에서는 모바일 및 임베디드 비전 애플리케이션의 설계 요구사항에 쉽게 부합할 수 있는 매우 작고 짧은 대기 시간 모델을 구축하기 위한 효율적인 네트워크 아키텍처와 두 가지 하이퍼 파라미터 세트에 대해 설명합니다. 섹션 2는 소형 모델 구축에 대한 이전 작업을 검토합니다. 섹션 3에서는 MobileNet 아키텍처와 더 작고 효율적인 MobileNet을 정의하기 위한 2개의 하이퍼 파라미터 폭 승수 및 해상도 승수를 설명합니다. 섹션 4는 ImageNet에서의 실험뿐만 아니라 다양한 응용 프로그램 및 사용 사례에 대해 설명합니다. 섹션 5는 요약과 결론으로 마무리됩니다.

1. **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.

최근 문헌에서는 작고 효율적인 신경망을 구축하는 것에 대한 관심이 높아지고 있습니다. [16, 34, 12, 36, 22]. 다양한 접근법은 일반적으로 사전 훈련된 네트워크를 압축하거나 소규모 네트워크를 직접 교육하는 것으로 분류할 수 있습니다. 본 문서는 모델 개발자가 자신의 애플리케이션에 대한 리소스 제한(대기 시간, 크기)과 일치하는 소규모 네트워크를 구체적으로 선택할 수 있도록 하는 네트워크 아키텍처 클래스를 제안합니다. MobileNets는 주로 대기 시간에 대한 최적화에 중점을 두지만 소규모 네트워크도 제공합니다. 소규모 네트워크에 관한 많은 논문들은 크기에만 초점을 맞추고 속도를 고려하지 않는다.

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].

모바일넷은 [26]에서 처음 소개된 깊이별 depthwise separable convolutions로부터 구축되며, 최초 몇 개의 계층에서 연산을 줄이기 위해 인셉션 모델[13]에서 후속적으로 사용된다. 플랫 네트워크[16]는 완전히 인수화된 컨버전스로 네트워크를 구축하고, 극도로 인수화된 네트워크의 잠재력을 보여주었습니다. 본 논문과는 별개로, Factorized Networks[34]는 위상 연결의 사용뿐만 아니라 유사한 인수화 컨볼루션을 도입합니다. 이어서, Xception 네트워크[3]는 depthwise separable filter를 확장하여 Inception V3 네트워크를 능가하는 방법을 시연했습니다. 또 다른 소형 네트워크는 Squeezenet[12]로, 병목 현상 접근 방식을 사용하여 매우 작은 네트워크를 설계합니다. 축소된 다른 계산 네트워크에는 구조화된 변환 네트워크[28]와 딥 프리드 컨벳[37]이 포함됩니다.

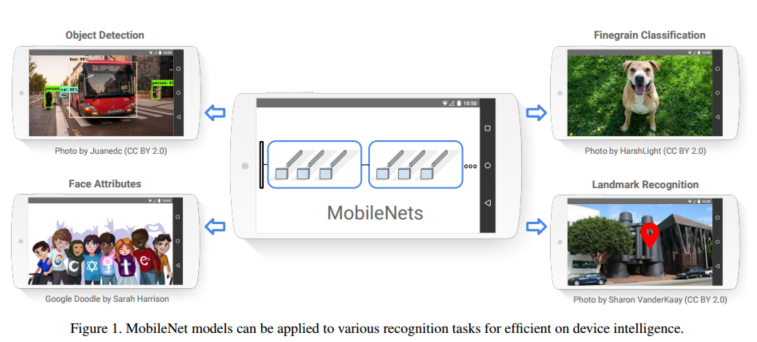


Figure 1. MobileNet models can be applied to various recognition tasks for efficient on device intelligence.

그림 1 MobileNet 모델은 다양한 인식 작업에 적용되어 기기 인텔리전스에 효율적입니다.

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

소규모 네트워크를 얻기 위한 또 다른 접근 방식은 사전 훈련된 네트워크를 축소, 인수화 또는 압축하는 것입니다. 제품 양자화[36], 해싱 [2]에 기초한 압축과 가지치기, 벡터 양자화 및 허프만 부호화[5]가 문헌에서 제안되었다. 또한 사전 훈련된 네트워크 속도를 높이기 위해 다양한 요소화가 제안되었습니다 [14, 20]. 소규모 네트워크를 교육하는 또 다른 방법은 더 큰 네트워크를 사용하여 더 작은 네트워크를 가르치는 증류[9]입니다. 이 내용은 당사의 접근 방식을 보완하며 섹션 4의 일부 사용 사례에서 다룹니다. 또 다른 새로운 접근 방식은 저비트 네트워크입니다 [4, 22, 11].

1. **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.

이 섹션에서는 먼저 depwise separable filter인 MobileNet의 핵심 계층에 대해 설명합니다. 그런 다음 MobileNet 네트워크 구조에 대해 설명하고 축소 하이퍼 파라미터 폭 승수와 해상도 승수에 대한 설명으로 결론을 내립니다.

**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).

MobileNet 모델은 depthwise separable convolutions을 기반으로 합니다. 이는 표준 컨볼루션을 depthwise 컨볼루션으로 인수화하여 1×1 컨볼루션을 pointwise 컨볼루션이라고 합니다. MobileNets의 경우 깊이별 변환은 각 입력 채널에 단일 필터를 적용합니다. 그런 다음 점별 컨볼루션은 1×1 컨볼루션을 적용하여 출력을 깊이별 컨볼루션을 결합합니다. 표준 컨볼루션은 입력을 필터링하고 새로운 출력 세트로 한 번에 결합합니다. 깊이별로 분리 가능한 컨볼루션은 이 두 레이어, 즉 필터링을 위한 별도의 레이어, 결합을 위한 별도의 레이어로 나눕니다. 이러한 요인화는 계산과 모델 크기를 획기적으로 줄이는 효과가 있습니다. 그림 2는 표준 콘볼루션 2(a)가 깊이 콘볼루션 2(b)와 1 × 1 점 콘볼루션 2(c)로 인수 분해되는 방법을 보여줍니다.

A standard convolutional layer takes as input a DF × DF × M feature map F and produces a DF × DF × N feature map G where DF is the spatial width and height of a square input feature map 1, M is the number of input channels (input depth), DG is the spatial width and height of a square output feature map and N is the number of output channel (output depth).

표준 컨볼루션 레이어는 DF × DF × M 피쳐 맵 F를 입력으로 사용하고 DF × DF × N 피쳐 맵 G를 생성합니다. 여기서 DF는 사각 입력 피쳐 맵의 공간 폭과 높이1, M은 입력 채널의 수(입력 깊이), DG는 사각 출력 피쳐 맵의 공간 폭과 높이, N은 출력 채널의 수(출력 디)입니다.pth).

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.

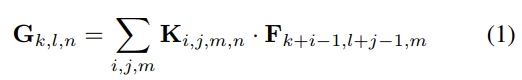
출력 피쳐 맵은 입력과 공간 치수가 동일하고 두 피쳐 맵이 모두 사각형이라고 가정합니다. 당사의 모델 축소 결과는 임의의 크기와 가로 세로 비율을 가진 피쳐 맵으로 일반화됩니다.

The standard convolutional layer is parameterized by convolution kernel K of size DK ×DK ×M×N where DK 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:

표준 콘볼루션 레이어는 크기가 DK × DK × M × N인 컨볼루션 커널 K에 의해 파라미터화됩니다. 여기서 DK는 정사각형으로 가정된 커널의 공간 치수이고 M은 입력 채널의 수이고 N은 이전에 정의된 출력 채널의 수입니다.

1단계와 패딩을 가정하여 표준 컨볼루션에 대한 출력 피쳐 맵은 다음과 같이 계산됩니다.



Standard convolutions have the computational cost of:

표준 변환의 계산 비용은 다음과 같습니다.



where the computational cost depends multiplicatively on the number of input channels M, the number of output channels N the kernel size Dk × Dk and the feature map size DF × DF . 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.

여기서, 계산 비용은 입력 채널 M의 수, 출력 채널 N의 수, 커널 크기 Dk × Dk, 피쳐 맵 크기 DF × DF에 곱하여 좌우됩니다. MobileNet 모델은 이러한 각 용어와 상호작용을 다룹니다. 먼저 깊이별 구분 가능한 변환을 사용하여 출력 채널 수와 커널 크기 간의 상호 작용을 중단합니다.

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 separable convolutions for substantial reduction in computational cost.

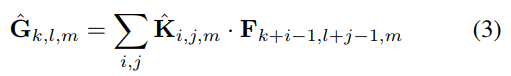
표준 컨볼루션 연산은 새로운 표현을 생성하기 위해 컨볼루션 커널을 기반으로 피쳐를 필터링하고 피쳐를 결합하는 효과가 있습니다. 필터링 및 결합 단계는 계산 비용의 상당한 절감을 위해 깊이 분리 가능 컨버전스라고 하는 인수 분해된 컨볼루션을 사용하여 두 단계로 분할할 수 있습니다.

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:

깊이별 분리 가능한 컨볼루션은 깊이별 컨볼루션과 점별 컨볼루션의 두 레이어로 구성됩니다. 깊이별 변환을 사용하여 각 입력 채널당 하나의 필터(입력 깊이)를 적용합니다. 그런 다음 깊이 층의 출력에 대한 선형 조합을 만드는 데 단순한 1×1 변환인 점별 변환이 사용됩니다. MobileNet은 두 레이어에 배치 표준과 ReLU 비선형성을 모두 사용합니다.

입력 채널당 하나의 필터(입력 깊이)를 사용하는 깊이별 컨볼루션은 다음과 같이 쓸 수 있습니다.



where Kˆ is the depthwise convolutional kernel of size DK × DK × M where the mth filter in Kˆ is applied to the mth channel in F to produce the mth channel of the filtered output feature map Gˆ.

Depthwise convolution has a computational cost of:

여기서 KΩ은 크기 DK × DK × M의 깊이 방향 컨볼루션 커널이며, 여기서 KΩ의 m번째 필터가 F의 m번째 채널에 적용되어 필터링된 출력 피쳐 맵 GΩ의 m번째 채널을 생성합니다.

깊이 변환의 계산 비용은 다음과 같습니다.



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.

깊이별 컨볼루션은 표준 컨볼루션에 비해 매우 효율적입니다. 그러나 입력 채널을 필터링할 뿐 결합해서 새 기능을 만들지는 않습니다. 따라서 이러한 새로운 특징을 생성하기 위해서는 1 × 1 컨볼루션을 통해 깊이 변환의 출력의 선형 조합을 계산하는 추가 계층이 필요합니다.

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:

깊이 방향 컨볼루션과 1 × 1 (점 방향) 컨볼루션의 조합을 [26]에서 처음 소개된 깊이 방향 분리 가능 컨볼루션이라고 부릅니다.

깊이별 분리형 컨버전스 비용:

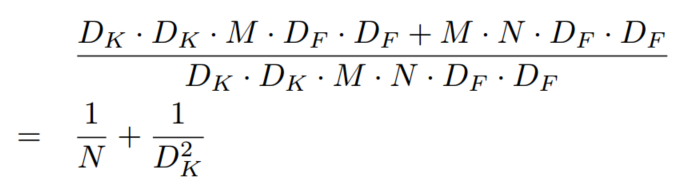


which is the sum of the depthwise and 1 × 1 pointwise convolutions.

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

즉, 깊이 방향과 점 방향의 1 × 1 변환의 합입니다.

컨볼루션을 필터링 및 결합의 2단계 프로세스로 표현하면 다음 작업을 줄일 수 있습니다.



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.

MobileNet은 3 × 3 깊이 구분 가능한 변환을 사용합니다. 이 변환은 섹션 4에서 볼 수 있는 정확도의 작은 감소로 표준 변환보다 8배에서 9배 더 적은 계산을 사용합니다.

[16, 31]과 같은 공간 차원의 추가 인자화는 깊이별 변환에서 사용되는 계산이 거의 없기 때문에 추가 계산을 많이 절약하지 않는다.

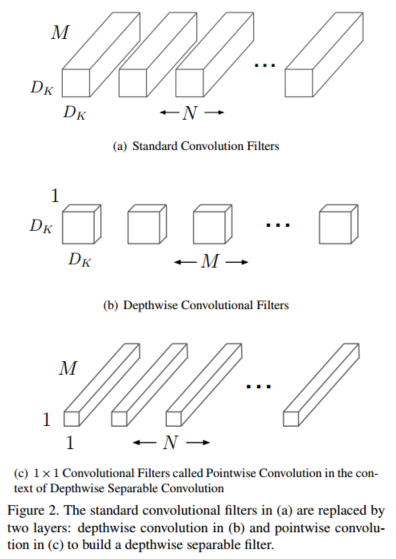
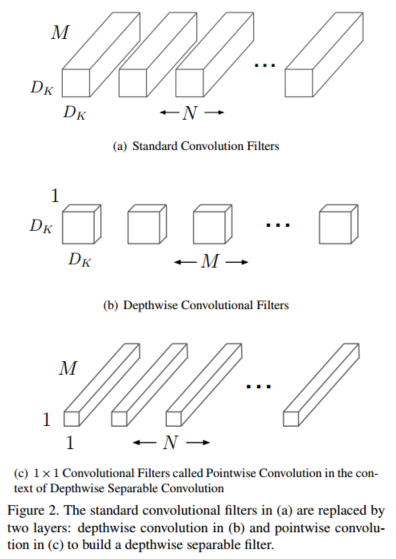


Figure 2. The standard convolutional filters in (a) are replaced by two layers: depthwise convolution in (b) and pointwise convolution in (c) to build a depthwise separable filter.

그림 2 (a)의 표준 컨볼루션 필터는 (b)의 깊이 방향 컨볼루션과 (c)의 점 방향 컨볼루션의 두 레이어로 대체되어 깊이 방향 분리 가능한 필터를 구축한다.

**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.

MobileNet 구조는 완전한 컨볼루션인 첫 번째 레이어를 제외하고 이전 섹션에서 언급한 것처럼 깊이별로 분리 가능한 컨볼루션을 기반으로 구축됩니다. 네트워크를 이렇게 간단한 용어로 정의하면 네트워크 토폴로지를 쉽게 탐색하여 좋은 네트워크를 찾을 수 있습니다. MobileNet 아키텍처는 표 1에 정의되어 있습니다. 비선형성이 없고 분류를 위해 부드러운 최대 레이어로 공급되는 최종 완전 연결 레이어를 제외하고 모든 레이어에는 배치 표준[13] 및 ReLU 비선형성이 뒤따릅니다. 그림 3은 규칙적인 컨볼루션, 배치 노름 및 ReLU 비선형성을 가진 레이어를 깊이별 컨볼루션, 1 × 1 점별 컨볼루션뿐만 아니라 각 컨볼루션 레이어 이후의 배치 노름 및 ReLU와 대비시킵니다. 다운 샘플링은 첫 번째 레이어뿐만 아니라 깊이별 컨볼루션에서도 스트라이드 컨볼루션으로 처리됩니다. 최종 평균 풀링은 공간 분해능을 완전히 연결된 도면층 이전의 1로 줄입니다. 깊이별 및 점별 변환을 별도의 계층으로 계산하면 MobileNet에는 28개의 계층이 있습니다.

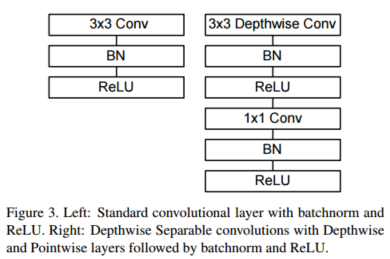


Figure 3. Left: Standard convolutional layer with batchnorm and ReLU. Right: Depthwise Separable convolutions with Depthwise and Pointwise layers followed by batchnorm and ReLU.

그림 3 왼쪽: 배트놈과 ReLU를 사용하는 표준 컨볼루션 레이어.오른쪽: 깊이와이즈 레이어와 포인트와이즈 레이어를 사용하는 깊이 분리 가능 컨볼루션, 그 다음에 배트놈과 ReLU.

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 instance unstructured sparse matrix operations are not typically faster than dense matrix operations until a very high level of sparsity. Our model structure puts nearly all of the computation into dense 1 × 1 convolutions. This can be implemented with highly optimized general matrix multiply (GEMM) functions. Often convolutions are implemented by a GEMM but require an initial reordering in memory called im2col in order to map it to a GEMM. For instance, this approach is used in the popular Caffe package [15]. 1×1 convolutions do not require this reordering in memory and can be implemented directly with GEMM which is one of the most optimized numerical linear algebra algorithms. MobileNet spends 95% of it’s computation time in 1 × 1 convolutions which also has 75% of the parameters as can be seen in Table 2. Nearly all of the additional parameters are in the fully connected layer.

단순히 네트워크를 소수의 Multi-Adds로 정의하는 것만으로는 충분하지 않습니다. 또한 이러한 작업을 효율적으로 구현할 수 있도록 하는 것도 중요합니다. 예를 들어, 구조화되지 않은 스파스 매트릭스 연산은 일반적으로 매우 높은 스파스 레벨이 될 때까지 고밀도 매트릭스 연산보다 빠르지 않습니다. 우리의 모델 구조는 거의 모든 계산을 밀도 있는 1 × 1 컨볼루션으로 만듭니다. 고도로 최적화된 GEMM(General Matrix Multiply) 기능으로 구현할 수 있다. 종종 GEMM에 의해 컨볼루션이 구현되지만, GEMM에 매핑하기 위해 im2col이라는 메모리에서 초기 재주문이 필요합니다. 예를 들어, 이 접근법은 널리 사용되는 카페 패키지[15]에서 사용됩니다. 1×1 변환은 메모리에서 이러한 순서를 변경할 필요가 없으며 가장 최적화된 수치 선형 대수 알고리즘 중 하나인 GEMM을 사용하여 직접 구현할 수 있습니다. MobileNet은 계산 시간의 95%를 1 × 1 컨볼루션에 소비하며, 표 2에서 볼 수 있는 바와 같이 파라미터의 75%도 가지고 있습니다. 거의 모든 추가 매개변수가 완전히 연결된 도면층에 있습니다.

MobileNet models were trained in TensorFlow [1] using RMSprop [33] with asynchronous gradient descent similar to Inception V3 [31]. However, contrary to training large models we use less regularization and data augmentation techniques because small models have less trouble with overfitting. When training MobileNets we do not use side heads or label smoothing and additionally reduce the amount image of distortions by limiting the size of small crops that are used in large Inception training [31]. Additionally, we found that it was important to put very little or no weight decay (l2 regularization) on the depthwise filters since their are so few parameters in them. For the ImageNet benchmarks in the next section all models were trained with same training parameters regardless of the size of the model.

MobileNet 모델은 Inception V3[31]과 유사한 비동기 구배 강하를 가진 RMSprop[33]을 사용하여 TensorFlow [1]에서 교육되었다. 그러나 대형 모델 교육과는 달리 소형 모델은 과대 맞춤에 대한 문제가 적기 때문에 정규화 및 데이터 확대 기술을 덜 사용합니다. MobileNets를 교육할 때 우리는 측면 헤드나 라벨 평활을 사용하지 않으며 대규모 인셉션 교육에 사용되는 소규모 작물의 크기를 제한하여 왜곡의 양을 추가로 줄입니다 [31]. 또한 깊이 필터에는 파라미터가 거의 없기 때문에 무게 감소(l2 정규화)를 거의 또는 전혀 가하지 않는 것이 중요하다는 것을 발견했습니다. 다음 섹션의 ImageNet 벤치마크의 경우 모델 크기에 관계없이 모든 모델이 동일한 교육 매개변수로 교육되었습니다.

**3-3. Width Multiplier: Thinner Models**

Although the base MobileNet architecture is already small and low latency, many times a specific use case or application may require the model to be smaller and faster. In order to construct these smaller and less computationally expensive models we introduce a very simple parameter α called width multiplier. The role of the width multiplier α is to thin a network uniformly at each layer. 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.

기본 MobileNet 아키텍처는 이미 작고 대기 시간이 짧지만, 여러 번 특정 사용 사례 또는 애플리케이션이 모델을 더 작고 빠르게 만들어야 할 수도 있습니다. 이러한 더 작고 계산적으로 덜 비싼 모델을 만들기 위해 우리는 폭 승수라고 불리는 매우 간단한 매개변수 α를 도입합니다. 폭 승수 α의 역할은 각 계층에서 네트워크를 균일하게 얇게 하는 것입니다. 주어진 계층과 폭 승수 α에 대하여, 입력 채널 M의 수는 αM이 되고 출력 채널 N의 수는 αN이 된다.

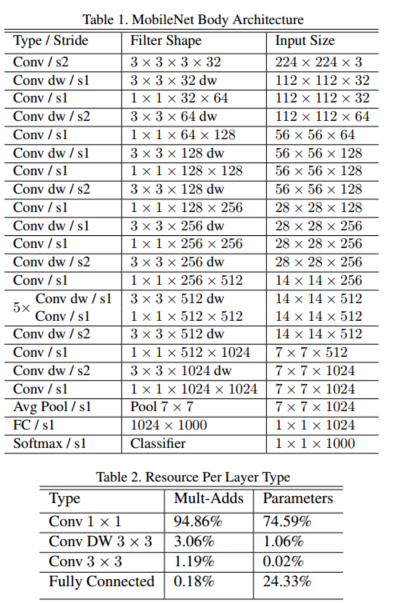


Table 1. MobileNet Body Architecture

표 1.. MobileNet 바디 아키텍처

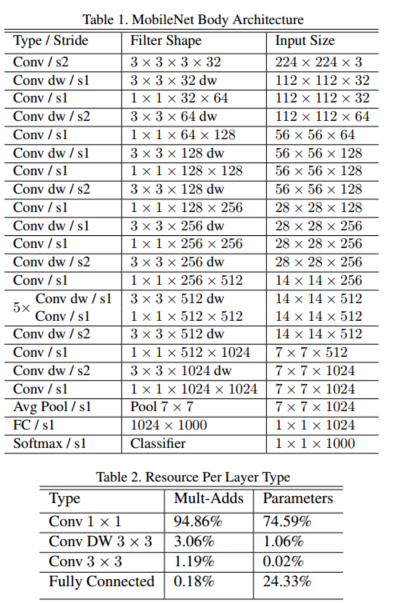


Table 2. Resource Per Layer Type

표 2 계층 유형별 리소스

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

폭 승수 α를 갖는 깊이별 분리 가능 합성곱의 계산 비용은 다음과 같다.



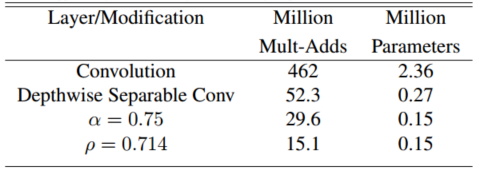
where α ∈ (0, 1] with typical settings of 1, 0.75, 0.5 and 0.25. α = 1 is the baseline MobileNet and α < 1 are reduced MobileNets. Width multiplier has the effect of reducing computational cost and the number of parameters quadratically by roughly α2. Width multiplier can be applied to any model structure 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.

여기서 α θ(0, 1)의 일반적인 설정은 1, 0.75, 0.5, 0.25입니다. α = 1은 기본 MobileNet이고 α < 1은 감소된 MobileNet입니다. 폭 승수는 계산 비용과 매개변수 수를 대략 α2만큼 이차적으로 줄이는 효과가 있습니다. 폭 승수를 모든 모델 구조에 적용하여 합리적인 정확도, 지연 시간 및 크기 균형을 가진 새로운 소형 모델을 정의할 수 있습니다. 처음부터 훈련해야 하는 새로운 축소된 구조를 정의하는 데 사용됩니다.

**3-4. Resolution Multiplier: Reduced Representation**

The second hyper-parameter to reduce the computational cost of a neural network is a resolution multiplier ρ. We ap-Table 3. Resource usage for modifications to standard convolution. Note that each row is a cumulative effect adding on top of the previous row. This example is for an internal MobileNet layer with DK = 3, M = 512, N = 512, DF = 14.

신경 네트워크의 계산 비용을 줄이기 위한 두 번째 하이퍼 파라미터는 분해능 승수 θ입니다. 무기-표 3 표준 컨볼루션을 수정하기 위한 리소스 사용량입니다. 각 행은 이전 행 위에 추가된 누적 효과입니다. 이 예는 DK = 3, M = 512, N = 512, DF = 14인 내부 MobileNet 계층에 대한 것입니다.



ply this to the input image and the internal representation of every layer is subsequently reduced by the same multiplier. In practice we implicitly set ρ by setting the input resolution.

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:

이 값을 입력 이미지에 적용하면 이후 모든 레이어의 내부 표현이 동일한 승수로 감소합니다. 실제로 우리는 입력 분해능을 설정하여 암묵적으로 by를 설정합니다.

이제 네트워크의 핵심 계층에 대한 계산 비용을 폭 승수 α와 분해능 승수 α를 사용하여 깊이별 분리 가능한 변환으로 표현할 수 있습니다.



where ρ ∈ (0, 1] which is typically set implicitly so that the input resolution of the network is 224, 192, 160 or 128. ρ = 1 is the baseline MobileNet and ρ < 1 are reduced computation MobileNets. Resolution multiplier has the effect of reducing computational cost by ρ2.

그래서 네트워크의 입력 해상도는 224,192개국의 160또는 128여야 어디 ρ ∈(0,1-RSB-. 그것은 일반적으로 암시적으로 설정됩니다.ρ=1베이스 라인 MobileNet과 ρ < 1이 줄어든 계산 MobileNets.결의 승수 ρ2에 의해 계산 비용을 대폭 줄이고 효과를 가지고 있다.

As an example we can look at a typical layer in MobileNet and see how depthwise separable convolutions, width multiplier and resolution multiplier reduce the cost and parameters. Table 3 shows the computation and number of parameters for a layer as architecture shrinking methods are sequentially applied to the layer. The first row shows the Mult-Adds and parameters for a full convolutional layer with an input feature map of size 14 × 14 × 512 with a kernel K of size 3 × 3 × 512 × 512. We will look in detail in the next section at the trade offs between resources and accuracy.

예를 들어 MobileNet의 일반적인 계층을 살펴보고 깊이 구분 가능한 변환, 폭 승수 및 해상도 승수를 통해 비용과 매개변수를 얼마나 절감할 수 있는지 확인할 수 있습니다. 표 3은 계층에 순차적으로 적용되는 아키텍처 축소 방법에 따른 계층에 대한 계산과 매개변수 수를 보여줍니다. 첫 번째 행은 크기가 14 × 14 × 512인 입력 기능 맵과 크기가 3 × 512 × 512인 커널 K를 가진 전체 컨볼루션 레이어에 대한 다중 추가 및 매개변수를 보여줍니다. 다음 섹션에서 리소스와 정확성 간의 절충에 대해 자세히 알아보겠습니다.

**4. Experiments**

In this section we first investigate the effects of depthwise convolutions as well as the choice of shrinking by reducing the width of the network rather than the number of layers. We then show the trade offs of reducing the network based on the two hyper-parameters: width multiplier and resolution multiplier and compare results to a number of popular models. We then investigate MobileNets applied to a number of different applications.

이 섹션에서는 먼저 깊이별 변환의 영향과 네트워크의 수를 줄이는 것에 의한 축소 선택에 대해 조사합니다. 그런 다음 폭 승수와 해상도 승수의 두 가지 하이퍼 매개변수를 기반으로 네트워크를 축소하는 경우의 단점을 보여주고 여러 인기 모델과 결과를 비교합니다. 그런 다음 다양한 애플리케이션에 적용되는 MobileNets에 대해 조사합니다.

**4.1. Model Choices**

First we show results for MobileNet with depthwise separable convolutions compared to a model built with full convolutions. In Table 4 we see that using depthwise separable convolutions compared to full convolutions only reduces accuracy by 1% on ImageNet was saving tremendously on mult-adds and parameters.

먼저 완전 컨버전스로 구축된 모델에 비해 깊이별로 분리 가능한 컨버전스로 MobileNet에 대한 결과를 보여줍니다. 표 4에서 우리는 완전 변환에 비해 깊이 구분 가능한 변환을 사용하면 ImageNet에서 정확도가 1%만 감소하여 다중 추가 및 매개변수를 크게 절약할 수 있음을 알 수 있습니다.

We next show results comparing thinner models with width multiplier to shallower models using less layers. To make MobileNet shallower, the 5 layers of separable filters with feature size 14 × 14 × 512 in Table 1 are removed. Table 5 shows that at similar computation and number of parameters, that making MobileNets thinner is 3% better than making them shallower.

다음으로 폭 승수를 사용하는 얇은 모델과 더 적은 레이어를 사용하는 얕은 모델을 비교한 결과를 보여줍니다. MobileNet을 낮게 만들기 위해, 표 1에서 피쳐 크기 14 × 14 × 512의 분리 가능한 필터의 5개 레이어를 제거합니다. 표 5는 유사한 연산 및 매개변수 수에서 MobileNets를 얇게 만드는 것이 얕게 만드는 것보다 3% 더 낫다는 것을 보여줍니다.

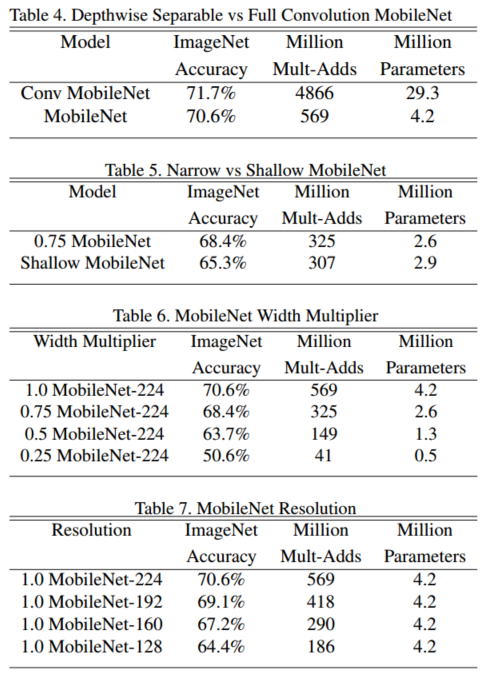


Table 4. Depthwise Separable vs Full Convolution MobileNet표 4 깊이 분리 가능과 풀 컨볼루션 MobileNet

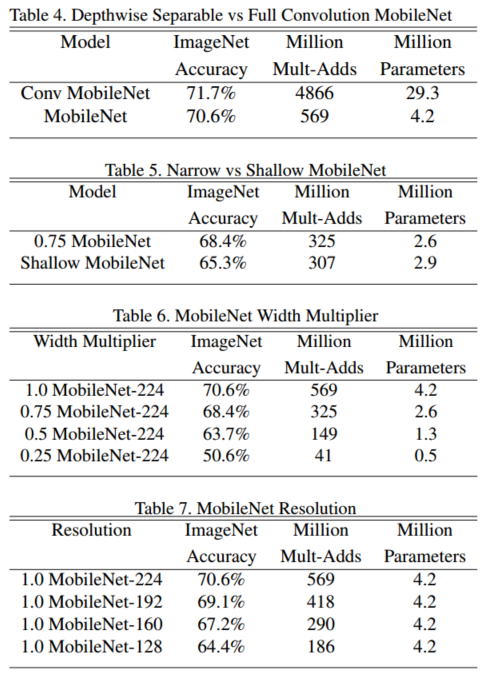


Table 5. Narrow vs Shallow MobileNet표 5 좁은 모바일 네트워크와 얕은 모바일 네트워크

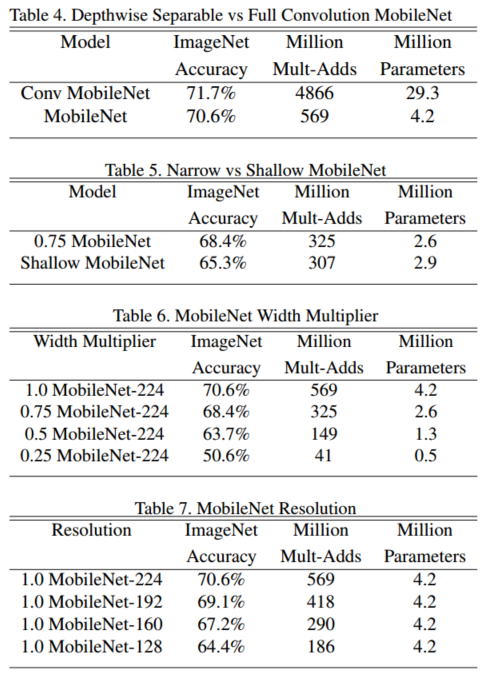


Table 6. MobileNet Width Multiplie

표 6 MobileNet 폭의 배수

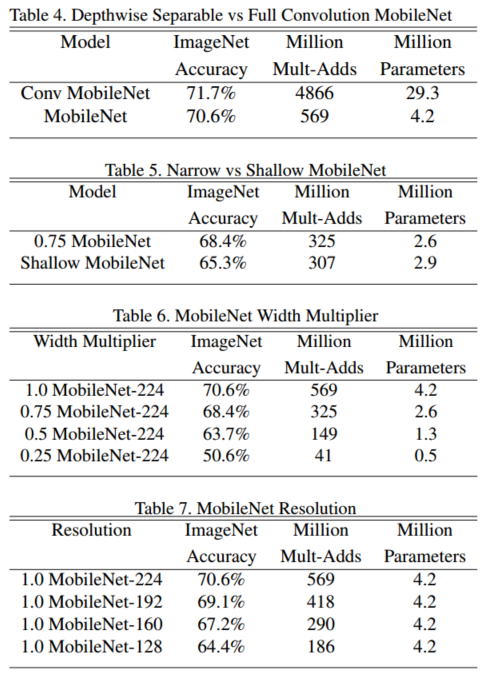


Table 7. MobileNet Resolution

표 7 MobileNet 해상도

**4.2. Model Shrinking Hyperparameters**

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.

Figure 4 shows the trade off between ImageNet Accuracy and computation for the 16 models made from the cross product of width multiplier α ∈ {1, 0.75, 0.5, 0.25} and resolutions {224, 192, 160, 128}. Results are log linear with a jump when models get very small at α = 0.25.

표 6은 폭 승수 α로 MobileNet 아키텍처를 축소하는 정확성, 계산 및 크기 균형을 보여줍니다. 정확도는 아키텍처가 α = 0.25로 너무 작게 만들어질 때까지 완만하게 떨어집니다.

표 7은 입력 분해능을 줄인 MobileNets를 교육하여 다양한 해상도 승수에 대한 정확도, 계산 및 크기 균형을 보여줍니다. 해상도에 걸쳐 정확도가 부드럽게 떨어집니다.

그림 4는 폭 승수 α α {1, 0.75, 0.5, 0.25} 및 해상도 {224, 192, 160, 128}의 교차곱으로부터 만들어진 16개 모델의 ImageNet Accuracy와 계산 간의 절충점을 보여줍니다. 모델이 α = 0.25로 매우 작아지면 결과는 점프로 로그 선형입니다.

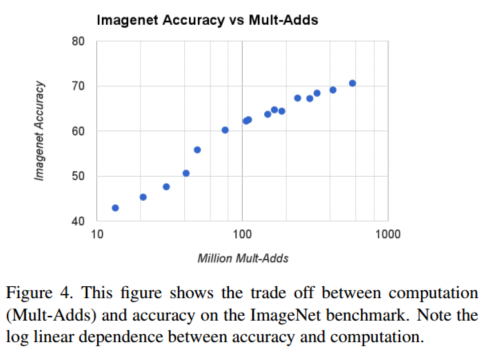


Figure 4. This figure shows the trade off between computation (Mult-Adds) and accuracy on the ImageNet benchmark. Note the log linear dependence between accuracy and computation.

그림 4 이 그림은 ImageNet 벤치마크에서의 계산(멀티애드)과 정확도의 균형을 나타내고 있습니다. 정확도와 계산 사이의 로그 선형 종속성에 유의하십시오.

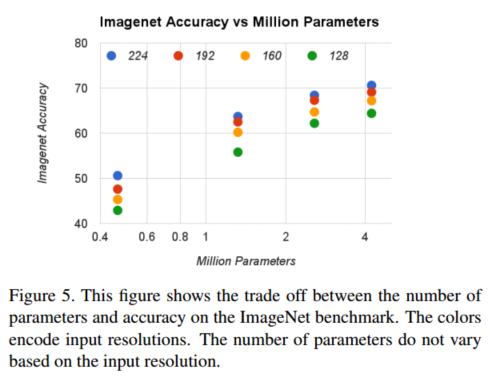


Figure 5. This figure shows the trade off between the number of parameters and accuracy on the ImageNet benchmark. The colors encode input resolutions. The number of parameters do not vary based on the input resolution.

그림 5 이 그림은 파라미터의 수와 ImageNet 벤치마크의 정확도 간의 균형을 나타내고 있습니다. 색상은 입력 분해능을 인코딩합니다. 파라미터의 수는 입력 분해능에 따라 달라지지 않습니다.

Figure 5 shows the trade off between ImageNet Accuracy and number of parameters for the 16 models made from the cross product of width multiplier α ∈{1, 0.75, 0.5, 0.25} and resolutions {224, 192, 160, 128}.

Table 8 compares full MobileNet to the original GoogleNet [30] and VGG16 [27]. MobileNet is nearly as accurate as VGG16 while being 32 times smaller and 27 times less compute intensive. It is more accurate than GoogleNet while being smaller and more than 2.5 times less computation.

Table 9 compares a reduced MobileNet with width multiplier α = 0.5 and reduced resolution 160 × 160. Reduced MobileNet is 4% better than AlexNet [19] while being 45×smaller and 9.4× less compute than AlexNet. It is also 4% better than Squeezenet [12] at about the same size and 22×less computation.

그림 5는 폭 승수 α {1, 0.75, 0.5, 0.25} 및 해상도 {224, 192, 160, 128}의 교차곱으로부터 만들어진 16개 모델에 대한 ImageNet Accuracy와 매개 변수 수 사이의 절충점을 보여줍니다.

표 8은 전체 MobileNet을 원본 Google Net [30] 및 VGG16 [27]과 비교합니다. MobileNet은 VGG16만큼 정확하면서도 크기는 32배, 컴퓨팅 집약도는 27배 낮습니다. 구글넷보다 더 정확하면서도 크기는 작고 연산은 2.5배 이상 적다.

표 9는 축소된 MobileNet과 폭 승수 α = 0.5 및 축소된 해상도 160 × 160을 비교합니다. 축소된 MobileNet은 AlexNet[19]보다 4% 더 나은 반면, AlexNet보다 45배 작고 9.4배 적은 컴퓨팅입니다. 또한 크기가 거의 같고 계산이 22배 적은 Squeezenet[12]보다 4% 더 좋습니다.

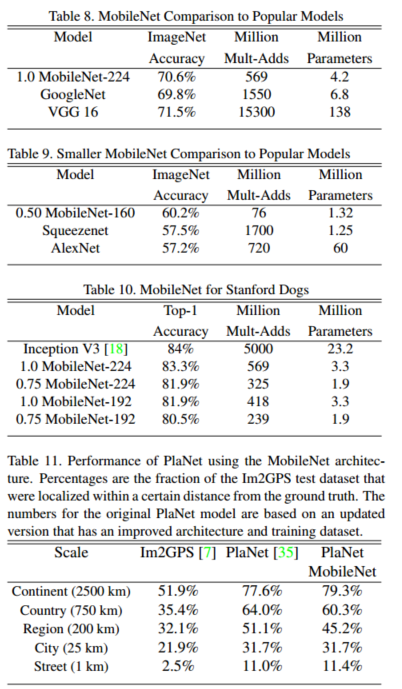


Table 8. MobileNet Comparison to Popular Models

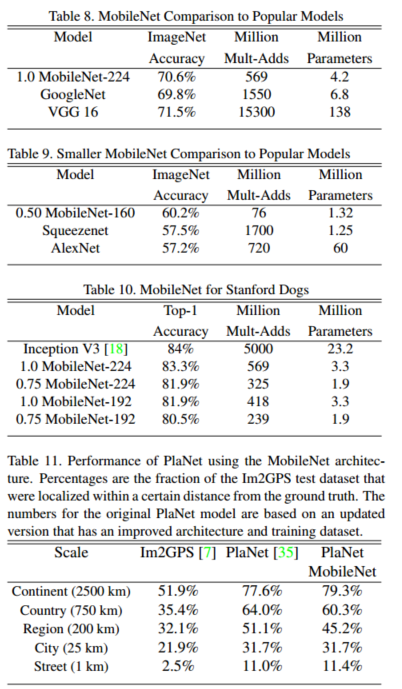
표 14 FaceNet에서 증류된 MobileNet

Table 9. Smaller MobileNet Comparison to Popular Models

표 9 일반 모델과의 소형 MobileNet 비교

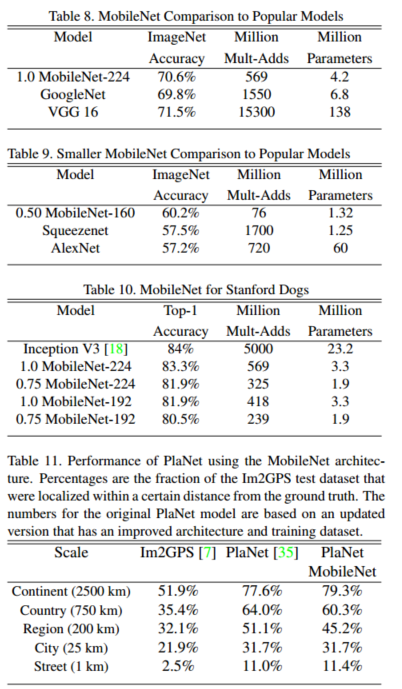


Table 10. MobileNet for Stanford Dogs

표 10 스탠포드 애견용 모바일 네트워크

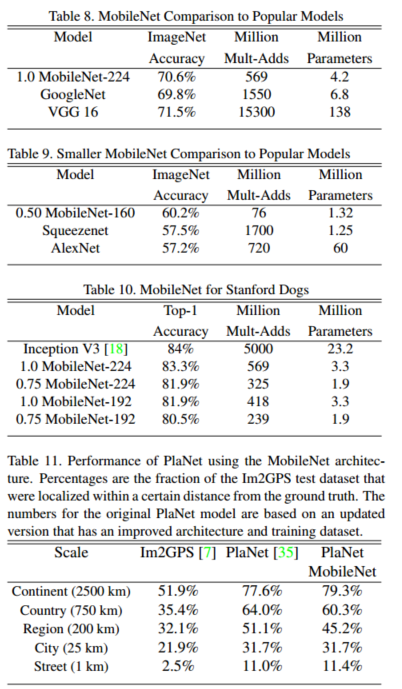


Table 11. Performance of PlaNet using the MobileNet architecture. Percentages are the fraction of the Im2GPS test dataset that were localized within a certain distance from the ground truth. The numbers for the original PlaNet model are based on an updated version that has an improved architecture and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

표 11 MobileNet 아키텍처를 사용한 PlaNet의 퍼포먼스. 퍼센티지는 Im2GPS 테스트 데이터 세트의 일부이며, 실제 데이터로부터 일정 거리 내에 현지화되어 있습니다. 원래 PlaNet 모델의 숫자는 향상된 아키텍처 및 교육 데이터 세트를 가진 업데이트된 버전을 기반으로 합니다.

**4.3. Fine Grained Recognition**

We train MobileNet for fine grained recognition on the Stanford Dogs dataset [17]. We extend the approach of [18] and collect an even larger but noisy training set than [18] from the web. We use the noisy web data to pretrain a fine grained dog recognition model and then fine tune the model on the Stanford Dogs training set. Results on Stanford Dogs test set are in Table 10. MobileNet can almost achieve the state of the art results from [18] at greatly reduced computation and size.

Stanford Dogs 데이터셋에서 MobileNet을 세밀하게 인식하도록 교육합니다 [17]. [18]의 접근 방식을 확장하고 [18]보다 더 크지만 소음이 심한 교육 세트를 웹에서 수집합니다. 노이즈가 많은 웹 데이터를 사용하여 미세한 입자 인식 모델을 사전 훈련한 다음 스탠포드 도그 훈련 세트에 모델을 미세 조정합니다. Stanford Dogs 테스트 세트에 대한 결과는 표 10에 나와 있습니다. MobileNet은 [18]의 연산 및 크기를 크게 줄이면 최첨단 결과를 얻을 수 있습니다.

**4.4. Large Scale Geolocalization**

PlaNet [35] 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 networkon millions of geo-tagged photos. PlaNet has been shown to successfully localize a large variety of photos and to outperform Im2GPS [6, 7] that addresses the same task.

PlaNet[35]은 분류 문제로 사진이 촬영된 장소를 파악하는 작업을 수행합니다. 이 접근 방식은 지구를 표적 등급의 역할을 하는 지리 세포의 격자로 나누고 수백만 개의 지리 태그가 붙은 사진에 대한 컨볼루션 신경망을 훈련시킵니다. PlaNet은 다양한 사진을 성공적으로 현지화하고 동일한 작업을 처리하는 Im2GPS[6, 7]보다 우수한 성능을 보이는 것으로 나타났습니다.

We re-train PlaNet using the MobileNet architecture on the same data. While the full PlaNet model based on the Inception V3 architecture [31] has 52 million parameters and 5.74 billion mult-adds. The MobileNet model has only 13 million parameters with the usual 3 million for the body and 10 million for the final layer and 0.58 Million mult-adds. As shown in Tab. 11, the MobileNet version delivers only slightly decreased performance compared to PlaNet despite being much more compact. Moreover, it still outperforms Im2GPS by a large margin.

동일한 데이터에 대해 MobileNet 아키텍처를 사용하여 PlaNet을 재교육합니다. Inception V3 아키텍처를 기반으로 하는 전체 PlaNet 모델 [31]은 5천 2백만 개의 파라미터와 57억 4천만 개의 멀티 애드입니다. 모바일넷 모델에는 1300만 개의 파라미터가 있으며, 보통 차체는 300만 개, 최종 계층은 1000만 개, 멀티 추가는 158만 개에 불과합니다. 표 11에서 볼 수 있듯이, 모바일넷 버전은 훨씬 소형임에도 불구하고 PlaNet에 비해 약간의 성능 저하만을 제공합니다. 또한 여전히 Im2GPS보다 큰 차이로 앞서고 있습니다.

**4.5. Face Attributes**

Another use-case for MobileNet is compressing large systems with unknown or esoteric training procedures. In a face attribute classification task, we demonstrate a synergistic relationship between MobileNet and distillation [9], a knowledge transfer technique for deep networks. We seek to reduce a large face attribute classifier with 75 million parameters and 1600 million Mult-Adds. The classifier is trained on a multi-attribute dataset similar to YFCC100M [32].

MobileNet의 또 다른 사용 사례는 알 수 없거나 난해한 교육 절차로 대형 시스템을 압축하는 것입니다. 얼굴 속성 분류 과제에서는 심층 네트워크를 위한 지식 전달 기법인 MobileNet과 증류[9]의 시너지 관계를 보여줍니다. 7500만 개의 파라미터와 1억 6000만 개의 멀티 애드(Mult-Adds)를 가진 대형 얼굴 속성 분류기를 줄이려고 합니다. 분류자는 YFCC100M[32]과 유사한 다중 속성 데이터 집합에 대해 교육됩니다.

We distill a face attribute classifier using the MobileNet architecture. Distillation [9] works by training the classifier to emulate the outputs of a larger model2 instead of the ground-truth labels, hence enabling training from large (and potentially infinite) unlabeled datasets. Marrying the scalability of distillation training and the parsimonious parameterization of MobileNet, the end system not only requires no regularization (e.g. weight-decay and early-stopping), but also demonstrates enhanced performances. It is evident from Tab. 12 that the MobileNet-based classifier is resilient to aggressive model shrinking: it achieves a similar mean average precision across attributes (mean AP) as the in-house while consuming only 1% the Multi-Adds.

우리는 MobileNet 아키텍처를 사용하여 얼굴 속성 분류기를 증류합니다. 증류[9]는 분류자가 지면-진실 라벨 대신 더 큰 모델 2의 출력을 에뮬레이트하도록 훈련함으로써 작동하며, 따라서 라벨이 부착되지 않은 대규모(및 잠재적으로 무한) 데이터 집합에서 훈련을 가능하게 한다. 증류 훈련의 확장성과 MobileNet의 인색한 매개변수화와 결합하면, 최종 시스템은 정규화(예: 중량 감소 및 조기 중지)가 필요하지 않을 뿐만 아니라 향상된 성능을 보여줍니다. MobileNet 기반 분류기는 Multi-Adds 1%만 소비하면서 사내 속성과 유사한 평균 정밀도(평균 AP)를 달성하면서 공격적인 모델 축소에 탄력적으로 대처한다는 것을 표 12에서 확인할 수 있습니다.

The emulation quality is measured by averaging the per-attribute cross-entropy over all attributes.

에뮬레이션 품질은 모든 속성에 대한 속성별 교차 엔트로피의 평균을 구하여 측정됩니다.

**4.6. Object Detection**

MobileNet can also be deployed as an effective base network in modern object detection systems. We report results for MobileNet trained for object detection on COCO data based on the recent work that won the 2016 COCO challenge [10]. In table 13, MobileNet is compared to VGG and Inception V2 [13] under both Faster-RCNN [23] and SSD [21] framework. In our experiments, SSD is evaluated with 300 input resolution (SSD 300) and Faster-RCNN is compared with both 300 and 600 input resolution (FasterRCNN 300, Faster-RCNN 600). The Faster-RCNN model evaluates 300 RPN proposal boxes per image. The models are trained on COCO train+val excluding 8k minival images and evaluated on minival. For both frameworks, MobileNet achieves comparable results to other networks with only a fraction of computational complexity and model size.

MobileNet은 최신 객체 감지 시스템에서 효과적인 기본 네트워크로 구현될 수도 있습니다. 우리는 2016년 COCO 도전 과제에서 우승한 최근 작업을 바탕으로 COCO 데이터에 대한 객체 감지 교육을 받은 MobileNet의 결과를 보고합니다[10]. 표 13에서 MobileNet은 Faster-RCNN [23] 및 SSD [21] 프레임워크 모두에서 VGG 및 Inception V2[13]와 비교됩니다. 본 실험에서 SSD는 300 입력 해상도(SSD 300)로 평가되며, Faster-RCNN은 300 및 600 입력 해상도(Fast RCN 300, Faster-RCNN 600)와 비교됩니다. Faster-RCNN 모델은 이미지당 300개의 RPN 제안 상자를 평가합니다. 모델들은 8k 미니발 이미지를 제외한 COCO 트레인+발에 대해 교육을 받고 미니발에 대해 평가됩니다. 두 프레임워크 모두에서 MobileNet은 계산 복잡성과 모델 크기의 극히 일부만으로 다른 네트워크와 비슷한 결과를 달성합니다.

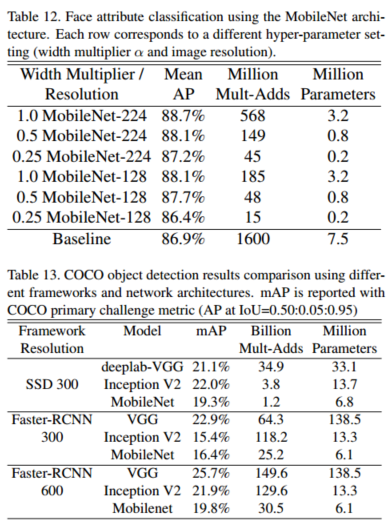


Table 12. Face attribute classification using the MobileNet architecture. Each row corresponds to a different hyper-parameter setting (width multiplier α and image resolution).

표 12 MobileNet 아키텍처를 사용한 얼굴 속성 분류. 각 행은 다른 하이퍼 파라미터 설정(폭 곱셈기α 및 화상 해상도)에 대응합니다.

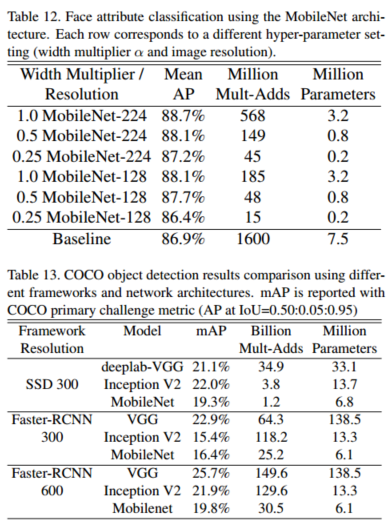


Table 13. COCO object detection results comparison using different frameworks and network architectures. mAP is reported with COCO primary challenge metric (AP at IoU=0.50:0.05:0.95)

표 13. 다른 프레임워크와 네트워크 아키텍처를 사용한 COCO 객체 감지 결과 비교. mAP는 COCO 1차 도전 메트릭(IoU=0.50:0.05:0.95)으로 보고된다.

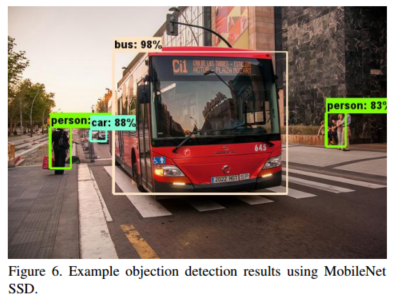


Figure 6. Example objection detection results using MobileNet SSD.

그림 6 MobileNet SSD를 사용한 이의 검출 결과 예시.

**4.7. Face Embeddings**

The FaceNet model is a state of the art face recognition model [25]. It builds face embeddings based on the triplet loss. To build a mobile FaceNet model we use distillation to train by minimizing the squared differences of the output of FaceNet and MobileNet on the training data. Results for very small MobileNet models can be found in table 14.

FaceNet 모델은 최첨단 얼굴 인식 모델입니다 [25]. 3중 손실을 기준으로 페이스 임베딩을 구축합니다. 모바일 FaceNet 모델을 구축하기 위해 교육 데이터에 대한 FaceNet과 MobileNet의 출력의 제곱 차이를 최소화하여 증류를 사용하여 교육합니다. 매우 작은 MobileNet 모델에 대한 결과는 표 14에서 확인할 수 있습니다.

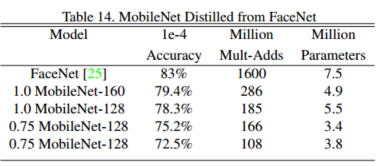


Table 14. MobileNet Distilled from FaceNet

표 14 FaceNet에서 증류된 MobileNet

**5. Conclusion**

We proposed a new model architecture called MobileNets based on depthwise separable convolutions. We investigated some of the important design decisions leading to an efficient model. We then demonstrated how to build smaller and faster MobileNets using width multiplier and resolution multiplier by trading off a reasonable amount of accuracy to reduce size and latency. We then compared different MobileNets to popular models demonstrating superior size, speed and accuracy characteristics. We concluded by demonstrating MobileNet’s effectiveness when applied to a wide variety of tasks. As a next step to help adoption and exploration of MobileNets, we plan on releasing models in Tensor Flow.

깊이 있게 분리할 수 있는 컨버전스를 기반으로 MobileNets라는 새로운 모델 아키텍처를 제안했습니다. 효율적인 모델로 이어지는 몇 가지 중요한 디자인 결정을 조사했습니다. 그런 다음 크기와 대기 시간을 줄이기 위해 적절한 정확도를 절충하여 폭 승수와 해상도 승수를 사용하여 더 작고 빠른 MobileNet을 구축하는 방법을 시연했습니다. 그런 다음 서로 다른 MobileNets를 우수한 크기, 속도 및 정확성을 보여주는 인기 모델과 비교했습니다. Mobile 시연으로 마무리했습니다.넷은 다양한 작업에 적용했을 때 효과적입니다. MobileNets의 채택과 탐색을 돕기 위한 다음 단계로, Tensor Flow에서 모델을 출시할 계획입니다.