MobileNet

0. Abstract

1. Depthwise seperable convolutions
   (깊이별 분리 합성곱)

1. Introduction

1. latency ↓, acc ↑ → 2개의 hyper-parameter ┌ Width Multiplier (α) 너비 배수
   (resource) └ Resolution Multiplier (ρ) 해상도 배수
   but, trad-off 관계

2. Prior work

1. dataset에 맞게 fine-tuning net 많이 or 작은 네트워크
   (resource)
   → "speed" 관점 필요

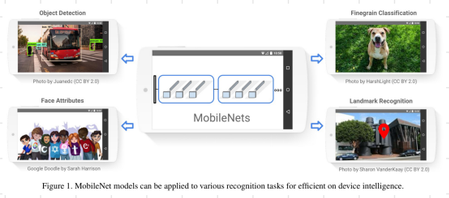


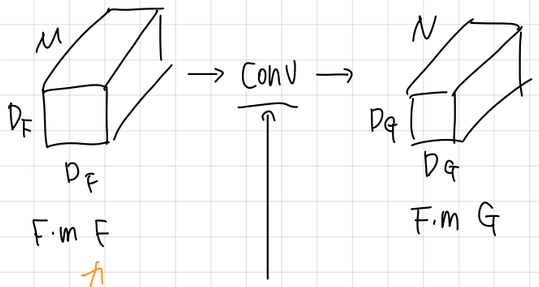
Figure 1. MobileNet models can be applied to various recognition tasks for efficient on device intelligence.

3. MobilNet Architecture

3.1 Depthwise Seperable Convolution

1. Depthwise seperable convolutions (factorized conv)
   → Standard Conv ⟶ (seperate) depthwise conv + pointwise conv
   각 입력 채널 마다 1개 필터 (필터링) / 1×1 conv (결합)

2.



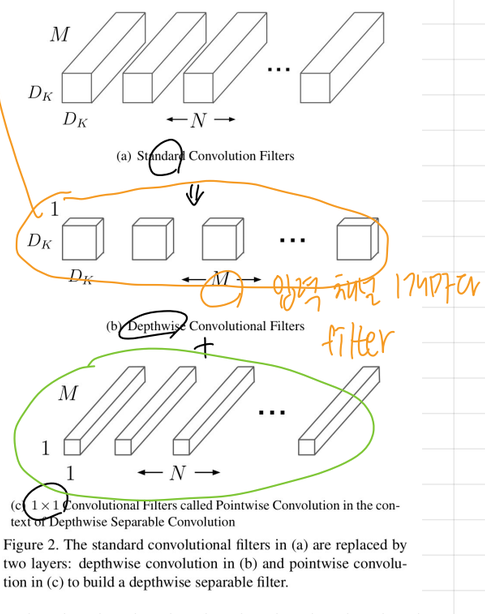


Figure 2. The standard convolutional filters in (a) are replaced by two layers: depthwise convolution in (b) and pointwise convolution in (c) to build a depthwise separable filter.

필터 f·m

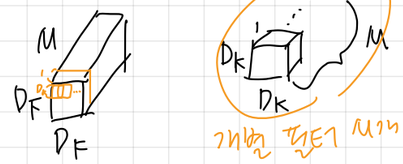
(a) $\mathbf{G}_{k,l,m} = \sum_{i,j,m} \mathbf{K}_{i,j,m,n} \cdot \mathbf{F}_{k+i-1,l+j-1,m}$

필터 f·m

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

Chin M개
마다의 Ch의 filter
적용

depth-wise filter



$= D_G \times D_G \times M$

$\frac{D_F - D_K + 2P}{S} + 1$

conv Computational cost

$D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F$

⇒ "depth wise seperable"

↓ "계산량 감소기법"

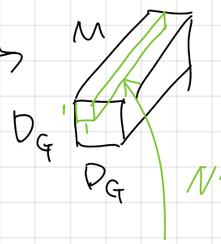
$D_K \cdot D_K \cdot M \cdot D_F \cdot D_F$

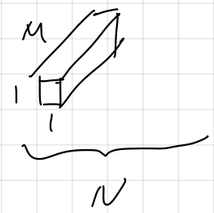
↓ "곱셈"

(c) $D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F$

$\Rightarrow \frac{D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F}{D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F}$

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



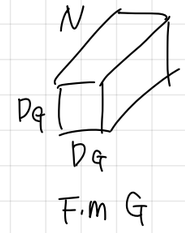


: 연산후 깊이 다른 N개의 1×1×M의 filter 적용후 채널간 쌓음

(1개 필터당 1×1×1 ⟶ 1×1×N)

N개 Concat

=



3.2 Network Structure and Training

1.

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	$1 \times 1 \times 1000$

→ 첫 층만 보편적 Conv

↳ 나머지는 Depth-wise seperable conv

→ HxW줄 때 1로

2.

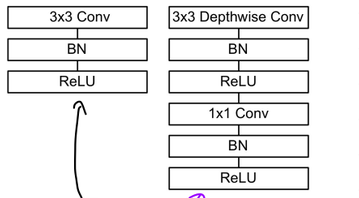


Figure 3. Left: Standard convolutional layer with batchnorm and ReLU. Right: Depthwise Separable convolutions with Depthwise and Pointwise layers followed by batchnorm and ReLU.

3.

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%

↳ 1x1 conv가 메모리 계산 시간, Params

4. depthwise 필터에는 메개변수가 적으므로 weight-decay 아주 적게 안 주는게 포인트

3.3 Width Multiplier: Thinner Models

1. 더 낮은 계산 비용

2. α = width multiplier

   ↳ 각 layer에서 net을 균일하게 thin하도록

   ↳ 입력 Ch $M \rightarrow \alpha M$
   출력 Ch $N \rightarrow \alpha N$ ($\alpha \in [0,1]$, generally $\underbrace{1}_{\text{base model}}, 0.75, 0.5, \underbrace{0.25}_{<1\,:\,\text{reduced}}$)

   ↳ $D_K \cdot D_K \cdot \alpha M \cdot D_F \cdot D_F + \alpha M \cdot \alpha N \cdot D_F \cdot D_F$

   ↳ α^2에 비례해 계산 비용 · 파라미터 수 ↓

3.4 Resolution Multiplier: Reduced Representation

1. ρ = resolution multiplier

   ↳ 해상도 바꾸고 ($\rho \in [0,1]$, 224/192/160/128 입력에 $\rho=1$ (base), $\rho<1$: reduced)

2. 최종 α, ρ 적용 계산 비용

   ∴ $D_K \cdot D_K \cdot \alpha M \cdot \rho D_F \cdot \rho D_F + \alpha M \cdot \alpha N \cdot \rho D_F \cdot \rho D_F$

   ↳ ρ^2에 비례 계산 ↓

3. 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 $D_K = 3$, $M = 512$, $N = 512$, $D_F = 14$. → 입력 f.m: 14×14×512 (standard: 3×3×512)

Layer/Modification	Million Mult-Adds	Million Parameters
(standard) fully – Convolution	462	2.36
Depthwise Separable Conv	52.3	0.27
$\alpha = 0.75$	29.6	0.15
$\rho = 0.714$	15.1	0.15

4. Experiments

4.1 Model choices / 4.2 Model Shrinking Hyperparameters

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

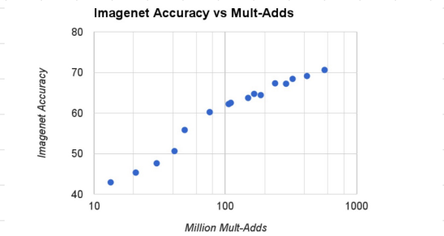


Figure 4. This figure shows the trade off between computation (Mult-Adds) and accuracy on the ImageNet benchmark. Note the log linear dependence between accuracy and computation.

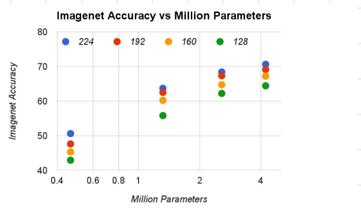


Figure 5. This figure shows the trade off between the number of parameters and accuracy on the ImageNet benchmark. The colors encode input resolutions. The number of parameters do not vary based on the input resolution.

3.

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%

4.3 ~ 4.7 참고

Table 14. MobileNet Distilled from FaceNet

Model	1e-4 Accuracy	Million Mult-Adds	Million Parameters
FaceNet [25]	83%	1600	7.5
1.0 MobileNet-160	79.4%	286	4.9
1.0 MobileNet-128	78.3%	185	5.5
0.75 MobileNet-128	75.2%	166	3.4
0.75 MobileNet-128	72.5%	108	3.8

5. Conclusions

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

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)

Framework Resolution	Model	mAP	Billion Mult-Adds	Million Parameters
SSD 300	deeplab-VGG	21.1%	34.9	33.1
	Inception V2	22.0%	3.8	13.7
	MobileNet	19.3%	1.2	6.8
Faster-RCNN 300	VGG	22.9%	64.3	138.5
	Inception V2	15.4%	118.2	13.3
	MobileNet	16.4%	25.2	6.1
Faster-RCNN 600	VGG	25.7%	149.6	138.5
	Inception V2	21.9%	129.6	13.3
	Mobilenet	19.8%	30.5	6.1

Figure 6. Example objection detection results using MobileNet SSD.