MobileNet V1 (2017)

서론

AlexNet, VGG, GoogleNet, ResNet 같은 모델들은 점점 깊고 복잡한 구조를 만들어 정확도를 높임.

하지만 이런 모델은 **매우 크고, 계산량이 많아** 강력한 GPU가 필수적임. 로봇, 자율주행 자동차, 증강현실(AR) 등 실제 세상에서 사용되는 기술들은 제한된 컴퓨팅 자원을 가진 디바이스에서 **Real time으로 작동해야함**.

따라서 MobileNet이 등장함. **정확도를 적절히 유지하면서도, 모델의 크기와 연산 속도(latency)를 획기적으로 줄인 효율적인 아키텍처** 제안.

메인 아이디어(Main Idea)

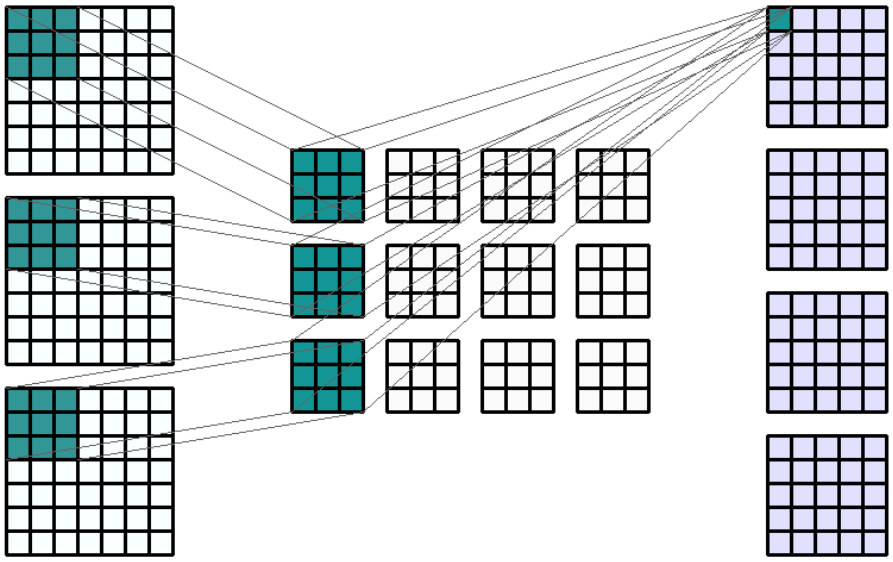
Convolution 대신 Depthwise Separable Convolution을 적용하자.

변수	설명
M	입력 데이터 채널(깊이)
D_K	Kernel(필터)의 크기 (넓이X, 길이O)
N	출력 데이터 채널(깊이)
D_F	Feature Map(출력 데이터)의 크기 (넓이X, 길이O)

Standard Convolution 연산량

$$Cost_{Standard} = M \cdot D_K \cdot D_K \cdot N \cdot D_F \cdot D_F$$

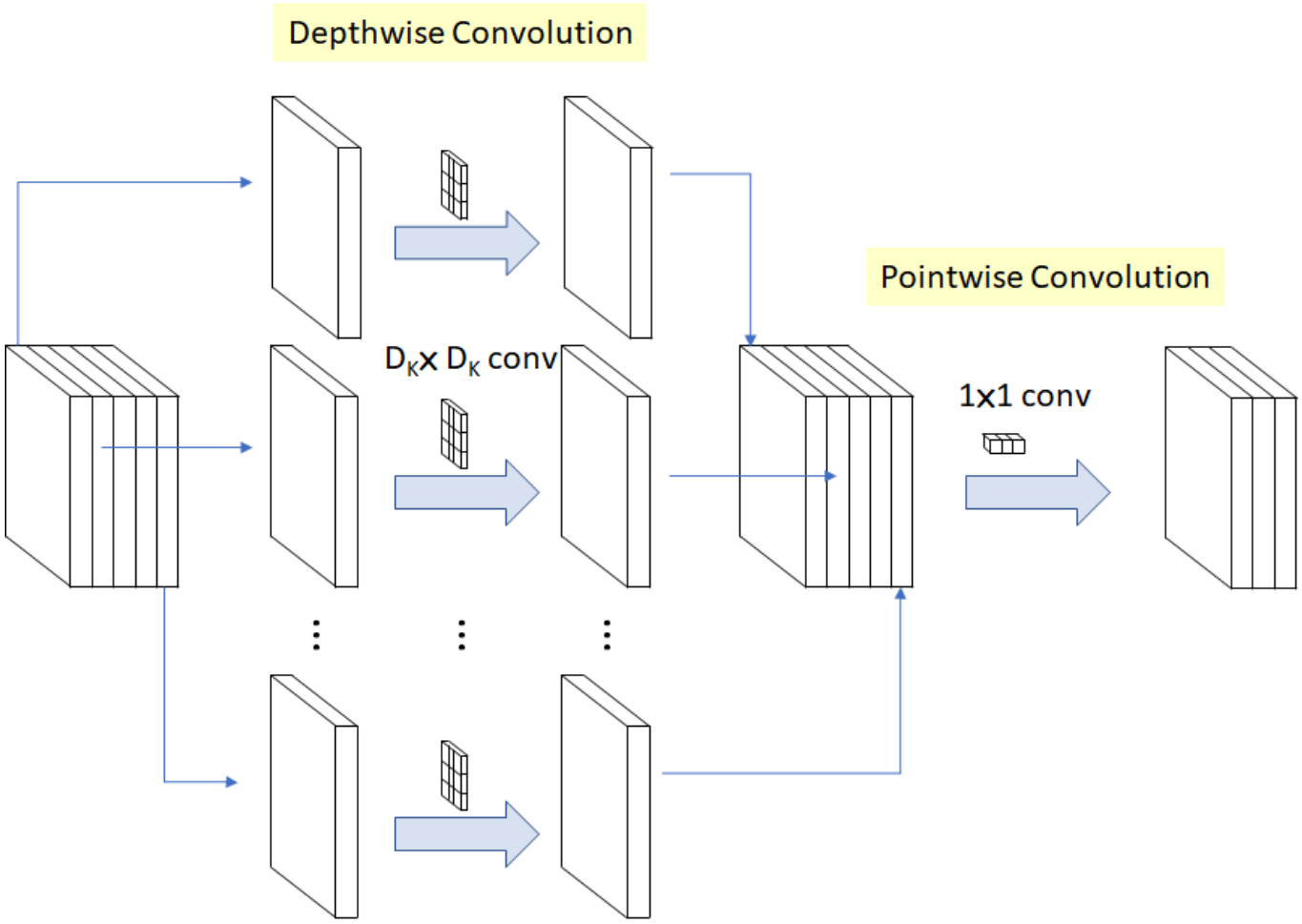
Key Point
하나의 출력 채널을 만드는데 $M \cdot D_K \cdot D_K \cdot D_F \cdot D_F$ 만큼의 연산을 하게 된다. **즉, 하나의 채널을 만드는데 항상 모든 입력 데이터를 채널을 다시 Convolution 해야한다는 것이다.** 따라서 **공간(Spatial)와 채널(Channel) 정보를 한 번에 조합**하여 하나의 채널을 만든다.



Convolution 연산에 대한 자세한 연산 링크

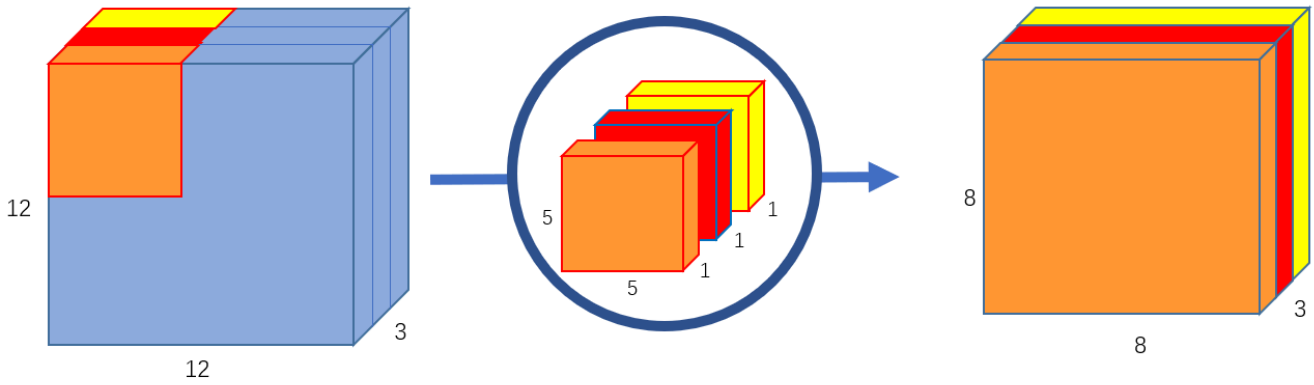
Depthwise Separable Convolution

`Depthwise Separable Convolution` 은 `Depthwise Convolution + Pointwise Convolution` 이다.



Depthwise Convolution

채널을 결합하지 않고, 각 입력 채널에 대해 **독립적으로** 필터링을 한다.



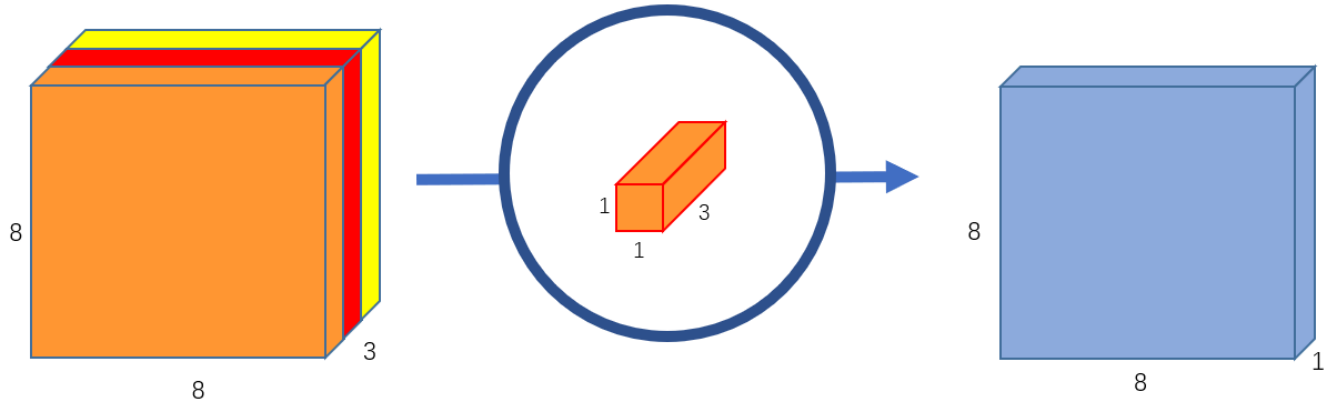
연산량

$$Cost_{Depthwise} = M \cdot D_K \cdot D_K \cdot D_F \cdot D_F$$

Key Point
항상 출력 채널은 입력 채널과 같으므로, 하나의 출력 채널을 만드는데 $D_K \cdot D_K \cdot D_F \cdot D_F$ 만큼의 연산을 하게 된다. **즉, 하나의 채널을 만들 때 입력 채널 하나만 Convolution하게된다.** 이는 모든 출력 데이터를 만만드는데 **입력 채널 하나마다 한번의 Convolution 밖에 진행하지 않는다는 것**이다. **공간(Spatial) 정보만 추출한다.**

Pointwise Convolution

1x1 Convolution이다. 여러 입력 채널(M)을 섞어 하나의 출력 채널을 만드는데, 이를 출력 채널의 수(N)만큼 생성한다.



연산량

$$Cost_{Pointwise} = M \cdot N \cdot D_F \cdot D_F$$

Key Point

하나의 출력 채널을 만들 때 이번엔 모든 입력 채널을 섞으며 $M \cdot D_F \cdot D_F$ 만큼의 연산을 하게 된다. **즉, 하나의 채널을 만들 때 이번엔 입력 채널 모두 사용하여 Convolution하게된다.** 이는 출력 채널별로 입력 채널을 어떻게 조합하여 채널을 만들지 계산하는 연산이다. **채널(Channel) 정보를 조합한다.**

총 연산량

`Depthwise Convolution+Pointwise Convolution` 한 연산량, 즉 `Depthwise Separable Convolution` 의 연산량은 다음과 같다.

$$Cost_{Total} = (M \cdot D_K \cdot D_K \cdot D_F \cdot D_F) + (M \cdot N \cdot D_F \cdot D_F)$$

`Standard Convolution` 과 비교하여 연산량 감소율을 다음과 같다.

$$\frac{Cost_{Total}}{Cost_{Standard}} = \frac{(M \cdot D_K \cdot D_K \cdot D_F \cdot D_F) + (M \cdot N \cdot D_F \cdot D_F)}{M \cdot D_K \cdot D_K \cdot N \cdot D_F \cdot D_F} = \frac{1}{N} + \frac{1}{D_K^2}$$

3x3 커널 사이즈의 경우($D_K = 3$)의 경우, 연산량을 약 **8~9배**나 줄일 수 있다.

Key Point

`Standard Convolution` 은 **공간(Spatial)와 채널(Channel) 정보를 한 번에 조합**하여 하나의 채널을 만들고, **하나의 출력 채널을 만들기 위해서 매번 모든 입력 채널에 대해 Convolution 진행한다. 반면** `Depthwise Separable Convolution` **은 Depthwise로 입력 채널별로 각각 한번의 Convolution만 진행하여 공간(Spatial) 정보만 추출한 후, Pointwise Convolution으로 각 채널(Channel) 정보를 조합하는 것이다.**

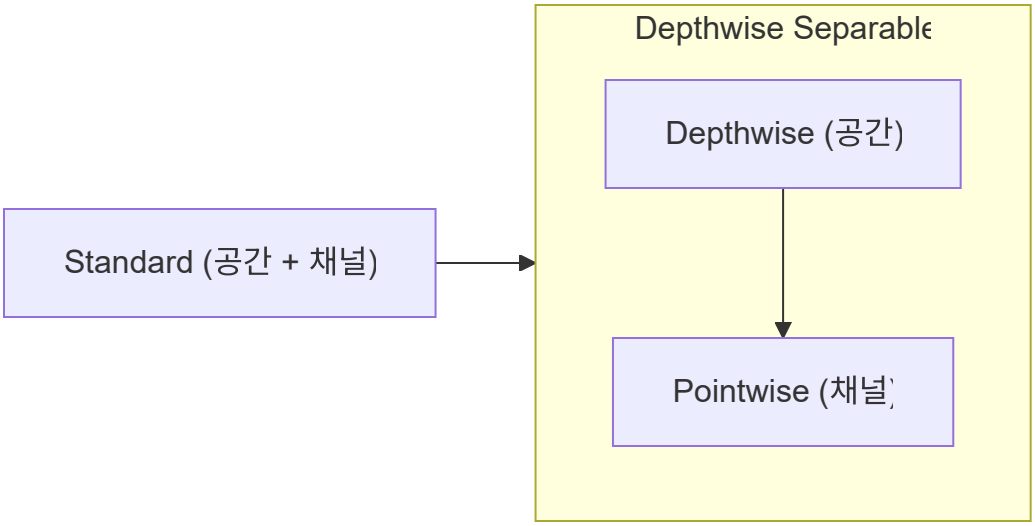


Table 4. Depthwise Separable vs Full Convolution MobileNet

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
Conv MobileNet	71.7%	4866	29.3
MobileNet	70.6%	569	4.2

연산량이 **8.55배** 줄어든 모습을 볼 수 있다.

$$\frac{4866}{569} = 8.551845342706503$$

또한 파라미터수는 약 **7배** 감소하였다.

$$\frac{29.3}{4.2} = 6.976190476190476$$

일반 모델과 비교하여 정확도는 **1.1%만** 하락하였다.

제안된 하이퍼파라미터

개발자가 애플리케이션 요구사항(속도, 크기 등)에 맞춰서 모델을 조절할 수 있도록 **Width Multiplier(α)와 Resolution Multiplier(ρ)를 제안하였다.**

Width Multiplier(α)

역할: 네트워크의 각 레이어에 있는 채널 수를 조절하여 모델을 더 얇게 만들도록 조절
작동 방식: $\alpha \in (0, 1]$ 이 되도록 값을 갖고, 입력 채널 M을 αM으로, 출력 채널 수 N을 αN으로 줄인다.
효과: 연산량과 파라미터 수를 약 α^2 비율로 감소시킨다.
(ex. $\alpha = 0.5$로 설정하면 연산량이 약 $1/4$으로 감소)

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

위 이미지에서 α를 1에서 0.25로 줄여가며 정확도, 연산량, 파라미터 수가 점차 감소하는 것을 확인할 수 있다.

Resolution Multiplier(ρ)

역할: 입력 이미지의 해상도를 조절하며 전체적인 연산량을 감소
작동 방식: $\rho \in (0, 1]$ 이 되도록 값을 갖고, 입력 이미지 해상도를 낮추는 방식
효과: 연산량을 ρ^2 비율로 감소시킨다.
(ex. $224 \times 224 \rightarrow \rho 224 \times \rho 224$).

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

해상도를 224에서 128로 줄여가며 정확도, 연산량이 점점 감소하는 것을 확인할 수 있다.

추가적으로 배울 점 (논문 직접 언급X)

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 Conv / s1	$3 \times 3 \times 512$ dw $1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$ $14 \times 14 \times 512$
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	$1 \times 1 \times 1000$

맨 첫번째 layer에서 `Standard Convolution`을 사용하는 이유

1. 풍부한 초기 특징
   맨 첫 번째 layer는 3채널(RGB)의 원본 이미지로부터 가장 기본적인 특징을 추출하는 매우 중요한 역할을 함. Standard Convolution은 **공간(Spatial)과 채널(Channel) 정보를 한번에 조합**하기 때문에, 초기 3채널의 상호 관계를 풍부하게 반영한 고차원 특징을 만들 수 있음.

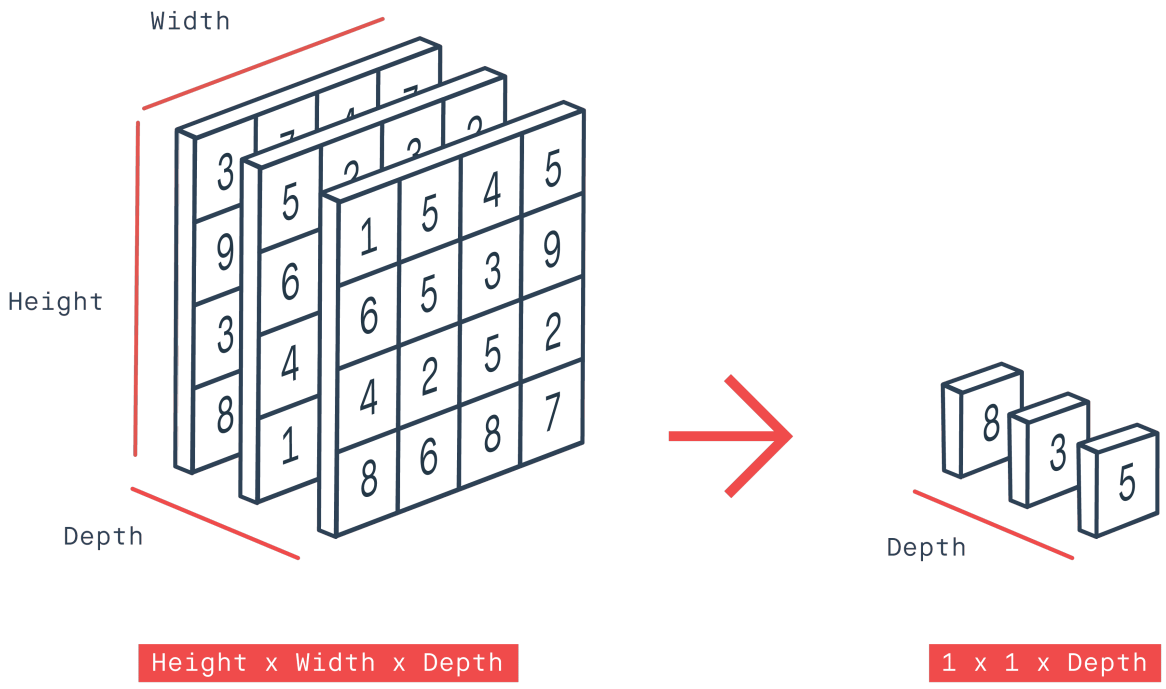
2. 적은 입력 채널로 인한 효율성 저하
   Depthwise Separable Convolution의 효율은 입력 채널(M)이 많은 때 극대화 된다. But 첫번째 layer의 입력 채널은 3개 뿐이라 연산량 감소 효과가 미미. 오히려 채널을 분리해서 처리하는 방식이 RGB 채널 간의 중요한 초기 연관성을 놓치게 만들어 성능에 안 좋을 수 있음.
3. 전체 연산량에서 차지하는 비중이 작음
   모델의 전체 연산량은 대부분 채널 수가 많은 중간 및 후반부 레이어에서 발생. 첫 layer는 연산량이 매우 적게 차지하므로, 이 부분은 약간 비용이 더 드는 Standard Convolution을 사용해 **안정적인 특징 기반을 다지는 것**이 모델 성능에 유리.

풀링 대신 `Convolution with stride`

모델을 보면 풀링은 Convolution에서 FC로 넘어가는 GAP(Global Average Pooling) 뿐이다. Max 풀링 대신 3x3 필터 크기를 갖는(1x1에서는 X, 안정적인 receptive field 갖도록) Depthwise Convolution에서 Stride를 2로 설정하여 이미지 해상도를 낮춘다. **이와 같은 방식이 성능이 더 좋다고 알려짐.**
*STRIVING FOR SIMPLICITY: THE ALL CONVOLUTIONAL NET*에서 모든 Pooling을 `Convolution with Stride`로 변경 시 성능 상승 효과가 있는 것을 확인함.

더욱 자세한 정보

Flatten 대신 `GAP`



Convolution 맨 마지막 차원의 크기를 보면 $7 \times 7 \times 1024$ 임을 확인할 수 있다. 이를 Flatten으로 펼치게 되면 $50,176$ 이된다. 이를 차원을 줄여가며 맨 마지막으로 1000개의 Class로 분류를 하게 하기 위해

서는 FC Layer라 하나라하는 가정하에도 $50,176 \times 1,000 = 50,176,000$ 개의 가중치가 필요하고, Bias까지 추가면 $50,177,000$ 이 된다. **파라미터 수가 너무 많아지며 over fitting이 발생할 수도 있다.** 하지만 `GAP(Global Average Pooling)` 을 적용하면 7×7의 resolution을 하나의 스칼라 1×1 로 바꾸기 때문에 FC로 넘어가기 전의 벡터들이 $1,024$개만 존재하게 되고, 이도 마찬가지로 하나의 FC Layer만 있다는 가정하에 Bias도 계산하여 하이퍼 파라메터 수를 계산하면 $1,024 \times 1000 + 1000 = 1,025,000$이으로 약 50배 줄어듦을 확인할 수 있다. 또 대부분의 CNN 모델에서 가중치들이 FC Layer들에서만 있었는데 GAP을 활용하여 이를 획기적으로 해결.

- 많은 파라미터(가중치)로 인한 over fitting 예방
- 계산량 감소에 따른 속도 증가 및 모델 경량화.