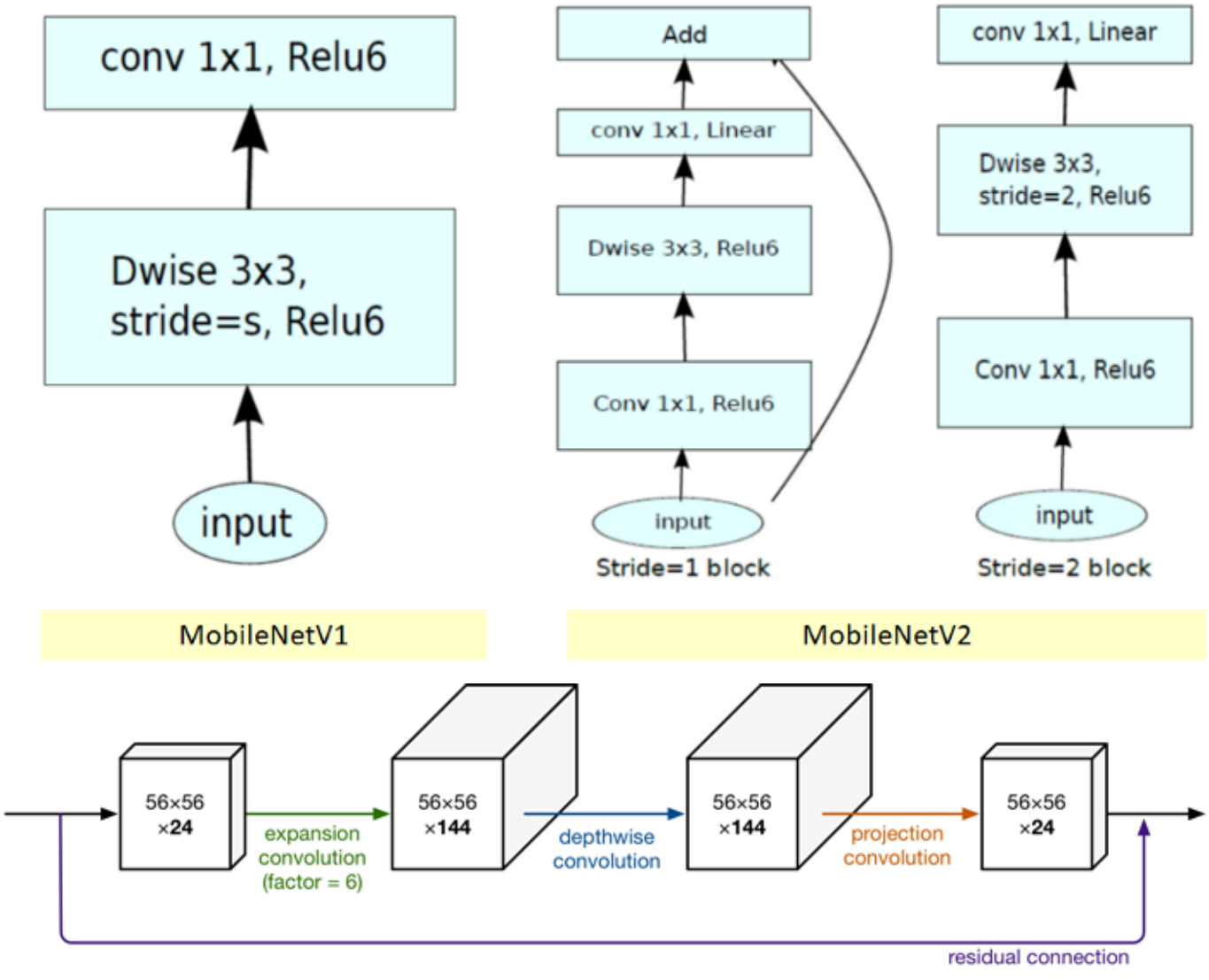
MobileNet V2 (2019)

서론

MobileNet V1과 비슷한 사유를 가지고 있음. V1에 비해 더욱 개선된 아키텍처 제안.

V1 vs V2



? V1에서의 ReLU6?

V2논문 내의 이미지에서 V1이 ReLU6를 사용한걸로 나온다. MobileNet V1논문에서는 ReLU6를 사용했다는 언급이 없고, 단순 ReLU만을 언급. 실제로 V1에서는 ReLU6를 사용한 것으로 보임. 왜 ReLU6?

주요 차이점

- Depthwise 전에 Pointwise 추가
  이를 `expansion convolution`으로 명명, 차원을 늘리는 역할.
  추가적으로 Depthwise 이후의 1x1 convolution은 `projection convolution`으로 불림. 차원을 다시 축소하는 역할. **Bottleneck**을 만듦.
- 마지막 layer의 1x1 convolution, `projection convolution`에서의 activation 제거
- stride별로 2개의 block
  `stride=1` 일 때 residual block, `stride=2` 일 때는 down sizeing을 위한 block
  추가로 `stride=2`에서는 skip connection이 없는데, resolution이 줄어들게되어 skip connection 또한 줄어든 크기에 맞춰야하는 문제가 있어 skip connection이 적용하지 않은 듯함.
- `expansion factor` 도입
  block 내에서 `expansion convolution`을 통과하여 채널(차원) 수를 늘릴 때 사용되는 상수

Input	Operator	Output
$h \times w \times k$	1x1 conv2d , ReLU6	$h \times w \times (tk)$
$h \times w \times tk$	3x3 dwise s=s, ReLU6	$\frac{h}{s} \times \frac{w}{s} \times (tk)$
$\frac{h}{s} \times \frac{w}{s} \times tk$	linear 1x1 conv2d	$\frac{h}{s} \times \frac{w}{s} \times k'$

  위 이미지를 보면 입출력 크기에 상수 `t`가 보이는데 이가 `expansion factor`로 논문에서 모두 6으로 정함. (ex. 입력 채널이 24개이고 `t=6` 일 때 하나의 block 내에서 채널은 $144 = 24 \times t(=6)$)

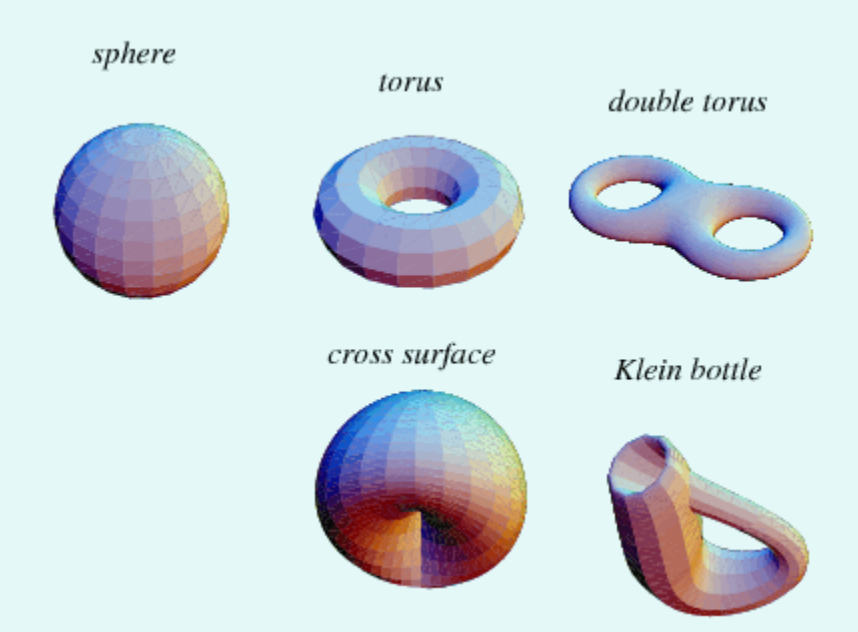
메인 아이디어(Main Idea)

MobileNet V1에서의 `Depthwise Seperable Convolution` 아이디어에서, 추가적으로 `Linear Bottlenecks`이 적용된 `Inverted Residuals`를 사용하자.

사전 배경지식 >

Manifold

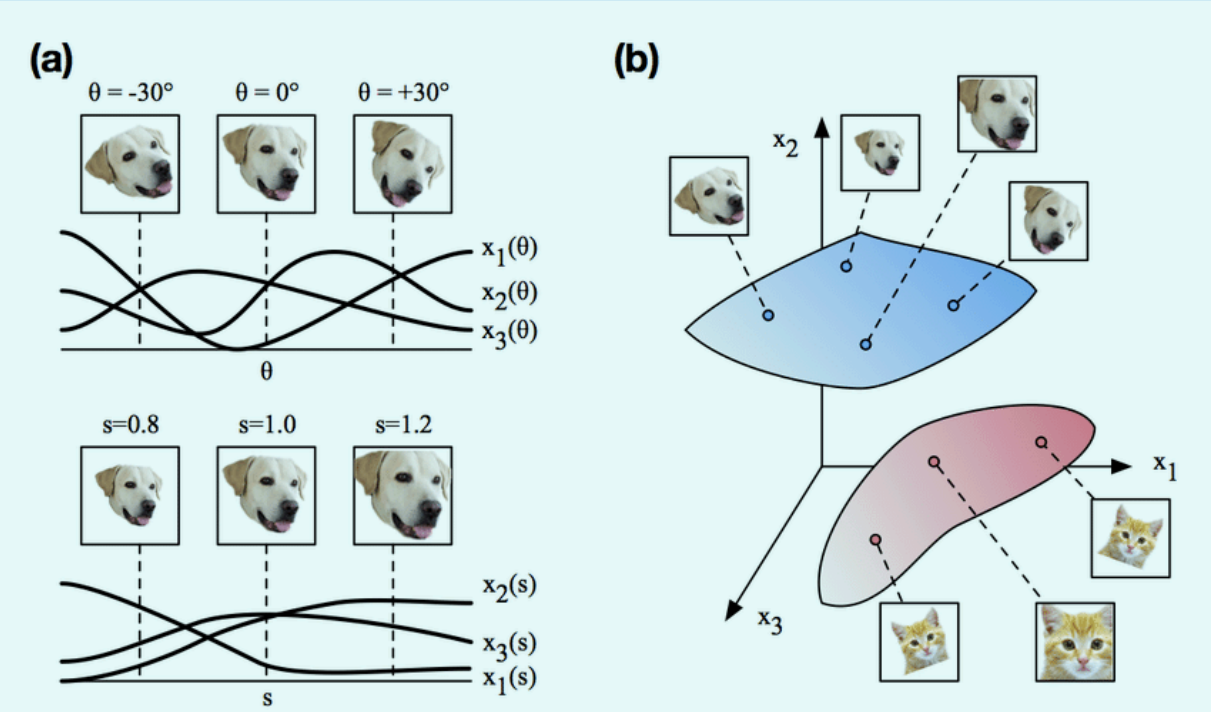
data science에서 n차원에서 표면의 smooth한 geometric surface를 말함. 아래 그림들이 3차원에서는 `manifold` 이다.



ⓘ Manifold of interest

아래 첨부된 왼쪽 이미지들(고차원의 데이터)을 오른쪽 이미지(저차원의 데이터)와 같이 매핑하는 과정에서 생기는 `manifold` 들이 있을 것인데, 이를 `manifold of interest` 라고 한다.
즉, **고차원의 데이터 공간**에서, **의미있는 정보들이 밀집**되어 있는 더 **낮은 차원**의 `manifold`

라고 할 수 있다.



Linear Bottlenecks

차원을 축소하는 `projection convolution`(bottleneck 구조)에서 비선형(non-linear, ex. ReLU) 활성화 함수를 제거하자는 아이디어

저자의 가설

1. 만약 ReLU 변환을 거친 후에도 정보가 손실되지 않고 온전한 형태(non-zero volume)를 유지 했다면, 이는 입력값이 음수가 아니기에 0이 되지 않은 상태라는 것이고, 그렇다면 그냥 ReLU는 `Linear Transformation`을 거친 것이라고 말 할 수 있다. (identity matrix를 곱한것이라고 말 할 수 있다.)
2. ReLU가 입력 데이터의 정보를 손실 없이 완벽하게 보존하는 것이 불가능하지 않다. 하지만 이는 입력 `manifold`가 전체 activation space보다 훨씬 낮은 차원의 subspace에 존재할 때, 즉 입력 `manifold` **주변에 충분한 여유 공간이 있을 때 가능하다는 것이다.**

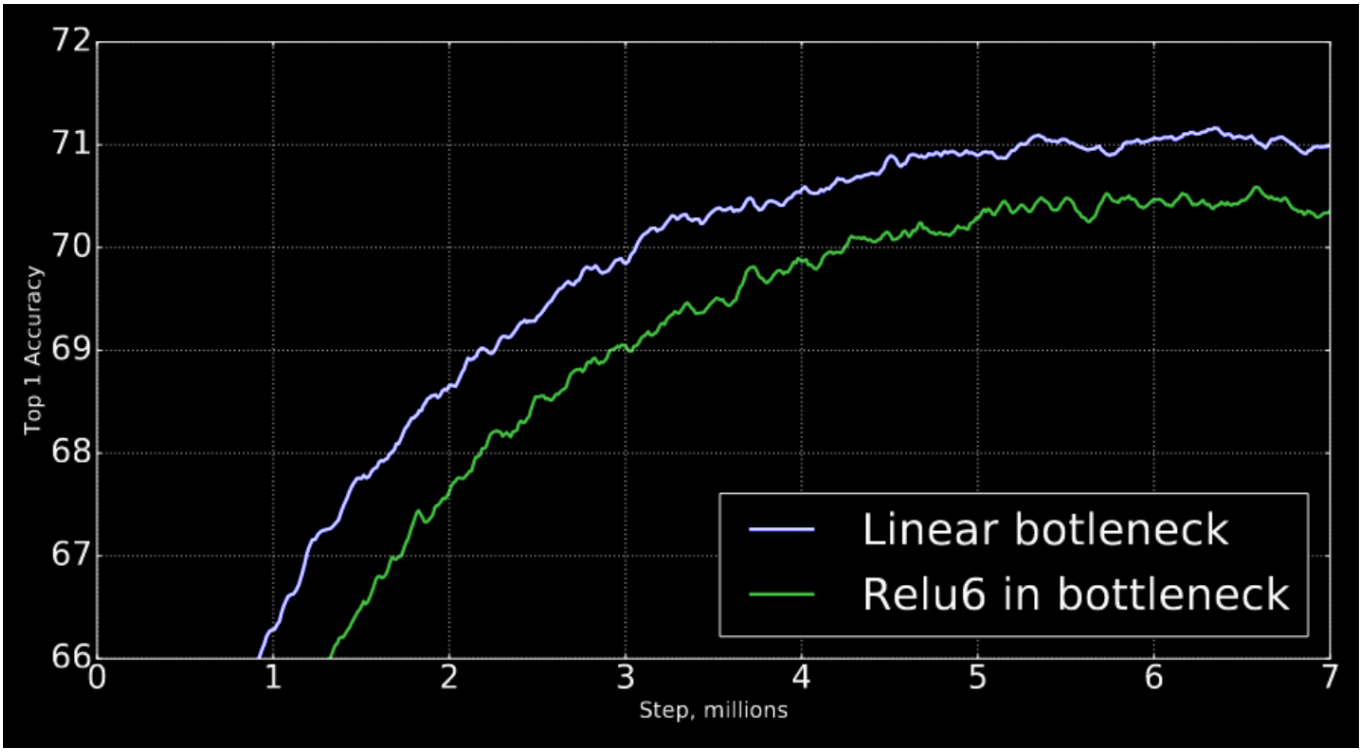
가설1은 ReLU가 비선형성을 부여하는 과정(ReLU가 제대로 사용되는 과정)에서 **어느 정보의 정보 손실이 필연적일 수 있다.**
가설2는 ReLU를 안전하게 사용하기 위해서는 `manifold`를 **고차원으로 Expansion하여 여유 공간을 만들어 주면 된다는 것을 시사한다.**

가설이 알려주는 문제점

문제 상황:
효율성을 위해 네트워크 채널 수를 줄여 bottleneck을 만들면, 그 안의 `manifold of interest` 는 더 이상 가설2에서 제안한 낮은 차원의 subspace에 존재하지 않음.(`manifold of interest` 의 주변 여유 공간이 사라진다.)
결과:
여유공간 없이 정보로 꽉 찬 bottleneck space에 ReLU를 적용하면, ReLU는 필연적으로 정보의 일부를 0으로 만들며 파괴하게 되고, 이는 압축된 정보 표현력을 손상시킨다.

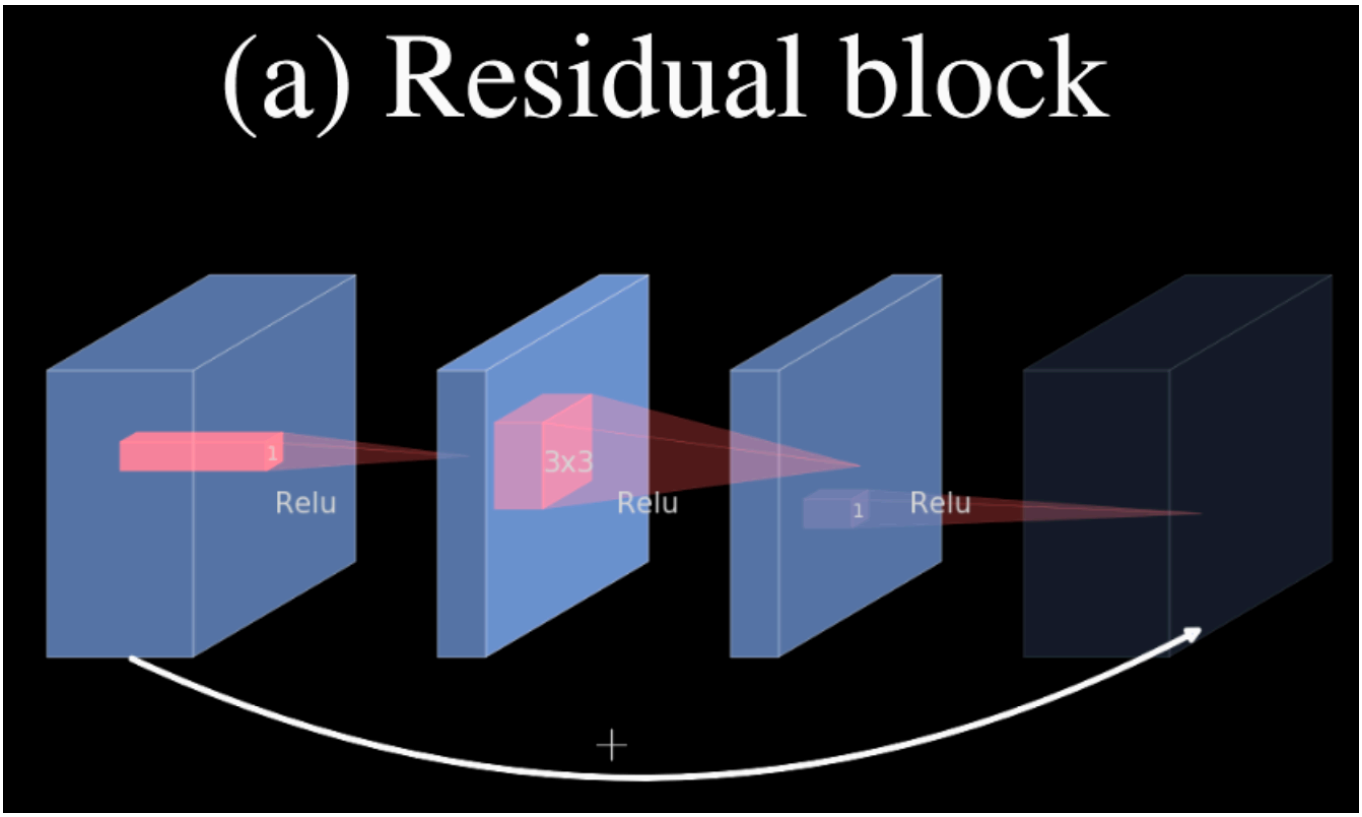
해결방안

단순히 Bottleneck에서 ReLU가 정보를 파괴하니, ReLU를 제거하자.

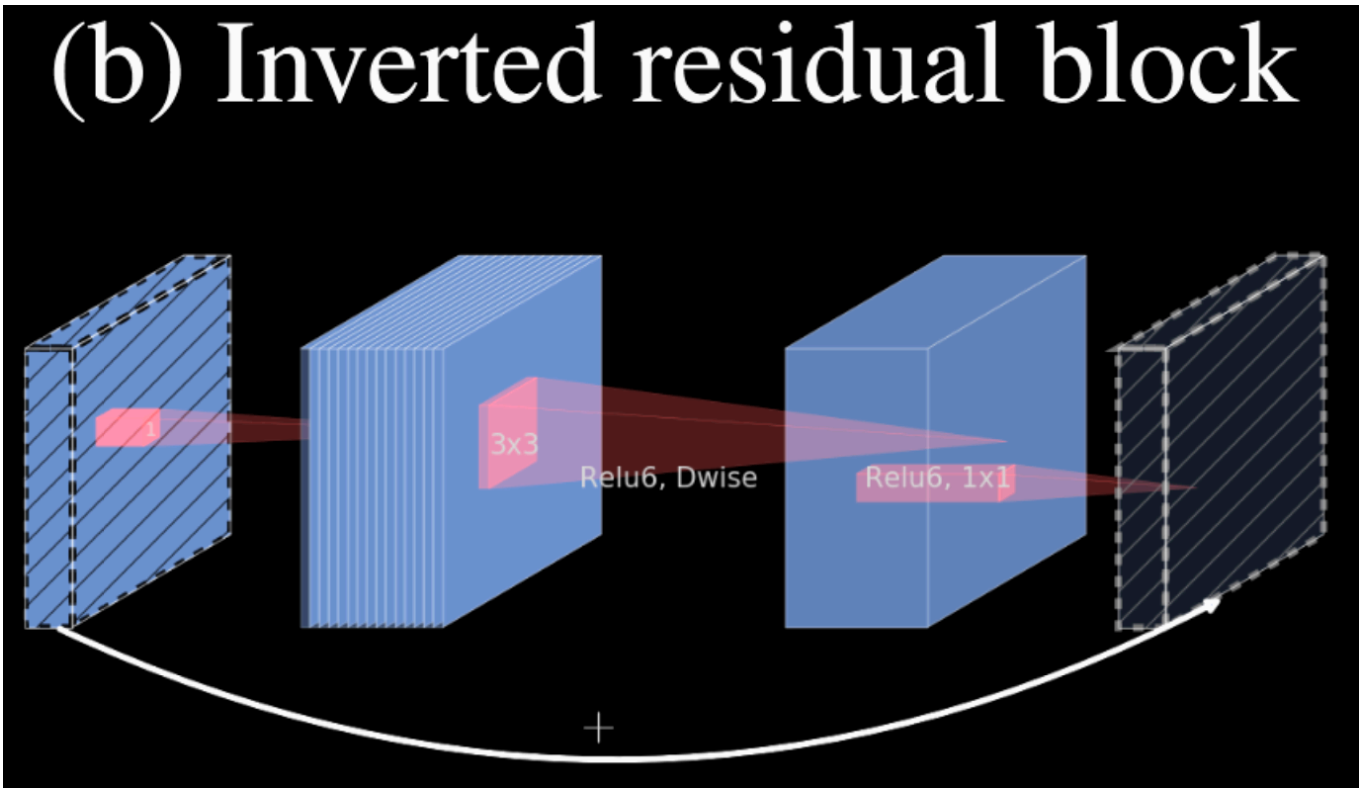


실제로 Bottleneck에서 ReLU를 제거하니 모델 성능이 향상되었음.

Inverted Residuals



위와 같은 일반적인 Residual block은 **wide > narrow > wide** 형태로 가운데에서 bottleneck(**narrow**)을 만들게 되고, 이후 skip connection과 합쳐지기 위해 1x1 convolution으로 처음 입력 사이즈로 복원됨.



- Inverted residual block에서는 **narrow > wide > narrow** 형태로, Inverted의 뜻 '반전된'이라는 이름에 맞게 residual block과 반대로 진행된다.

- 이와 같이 시도한 이유는 **narrow**에 해당하는 저차원 layer에는 **실용적인 정보만 압축**된다는 가정을 하였기 때문이다.
- **narrow** layer에서 다시 **narrow** layer로 shortcut으로 연결
- 앞서 설명한 Linear Bottlenecks를 적용하여 **wide > narrow**될 때 ReLU를 거치지 않는다.

하드웨어 부분 장점

최대 메모리 사용량(peek memory usage)를 획기적으로 줄여준다.

block의 입력으로 들어온 **narrow** layer를 skip connection으로 사용하기 때문에 메모리 사용량을 줄일 수 있다. (Wide를 skip connection으로 하면 narrow에 비하여 보다 고차원의 tensor를 메모리 상에 놓아야하기 때문)

Size	MobileNetV1	MobileNetV2	ShuffleNet (2x,g=3)
112x112	64/1600	16/400	32/800
56x56	128/800	32/200	48/300
28x28	256/400	64/100	400/600K
14x14	512/200	160/62	800/310
7x7	1024/199	320/32	1600/156
1x1	1024/2	1280/2	1600/3
max	1600K	**400K**	600K

위 테이블은 각 input size에 따른 `채널 수/ 해당 채널을 저장하는 데 필요한 메모리(Kb 단위)`의 최대 크기를 나타냄. **MobileNet V2가 최대 메모리가 가장 작음을 알 수 있다.**

모델 구조

변수	설명
t	expansion factor
c	output channel 수
n	반복횟수

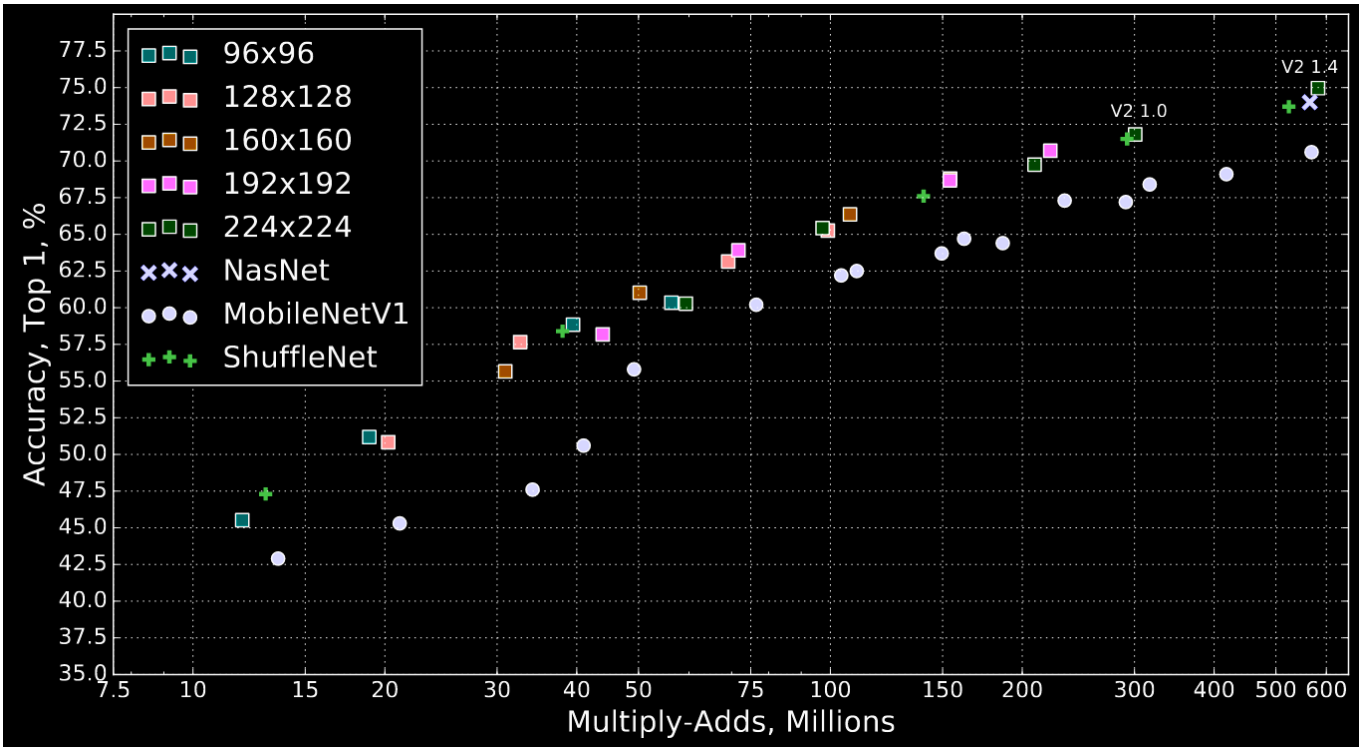
변수	설명
s	stride

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

모델 성능

Network	Top 1	Params	MAdds	CPU
MobileNetV1	70.6	4.2M	575M	113ms
ShuffleNet (1.5)	71.5	**3.4M**	292M	-
ShuffleNet (x2)	73.7	5.4M	524M	-
NasNet-A	74.0	5.3M	564M	183ms
MobileNetV2	**72.0**	**3.4M**	**300M**	**75ms**
MobileNetV2 (1.4)	**74.7**	6.9M	585M	**143ms**

위 이미지를 보면 당시 V1보다 연산량이 $300/575 = 0.5217391304347826$ **약 48% 줄었음에도 정확도는 1.4% 가량 높았다.**
맨 하단에 보이는 V2(1.4)는 V1에서 제안한 Width Multiplier(α)를 1.4로 설정하여 모델의 크기를 키운 것이다. (이렇게 하면 채널 수가 기존대비 1.4배로 증가)

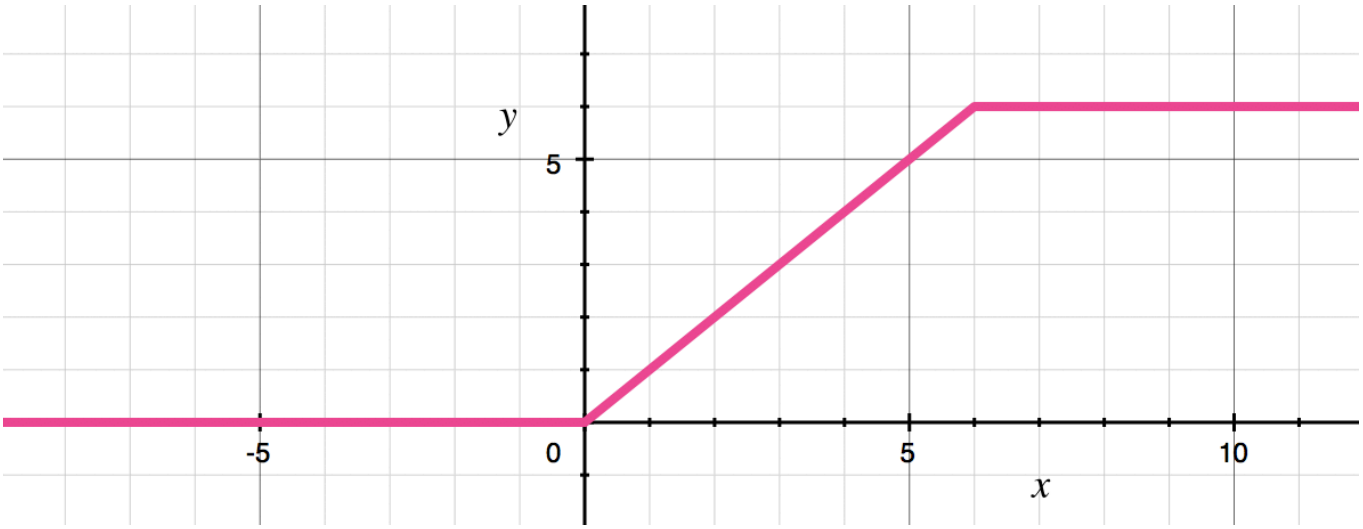


위 그래프는 연산량 대비 정확도 곡선이다. MobileNet V2가 꾸준히 최상단에 있음을 확인할 수 있다. 가장 오른쪽 위는 $\alpha = 1.4$ 로 키운 모델, 왼쪽 V2 1.0이 기본모델

추가적으로 배울 점

ReLU 대신 ReLU6를 사용한 이유

$ReLU(x) = max(0, x)$
위는 익숙한 ReLU 함수의 식이다.
아래는 ReLU6 함수와 그 그래프이다.
$ReLU6(x) = min(max(0, x), 6)$



ReLU6는 upper bound가 존재(6으로 너무 커짐을 제한).

- **저정밀도**(bit수가 적은) **환경**에서 **ReLU**는 활성화 값의 범위가 $[0, \infty)$로 너무 넓어서 정밀도 손실이 발생하기 쉽다. (ex. 0~1000까지의 값을 256개의 정수 level로 표현하려면 각 레벨이 나타내는 범위가 넓어져 세밀한 표현 불가능)
- 반면, **ReLU6**는 활성화 값의 범위가 $[0, 6]$으로 제한되어 있어 양자화를 하더라도 정밀도 손실을 최소화 할 수 있음.
  결론적으로 ReLU6는 모델을 더 가볍고 빠르게 만들기 위한 양자화 과정에서 성능 저하를 막아주기 때문에 ReLU6를 활용. (워크플로우가 다음과 같기 때문.)

1. 학습 - 서버/GPU
   서버/GPU환경에서 ImageNet을 사용하여 학습(16개의 GPU를 비동기 방식으로 학습했다고 논문 명시). 모델의 파라미터가 일반적으로 float32와 같은 **고정밀도(High-Precision)** 표현되어 있음. 미세한 변화까지 반영하여 모델의 정확도를 최대한으로 끌어올리기 위함.
2. 배포 - 모바일
   학습이 완료된 모델을 모바일 기기에서 실행하기 위해 양자화 과정을 거침. float32 -> int8와 같은 **저정밀도(low-precision)** 형식으로 변하여 모델의 크기를 줄이고, 정수연산으로 하드웨어에서 속도 가속.

1x1 Convolution for Classification

모델 구조를 보면 GAP이후에 FC가 없는 것을 확인할 수 있다. 이 대신 $1 \times 1 \times 1280$의 입력을 **1x1 convolution을 사용하여 `k`개의 클래스 만큼의 채널을 만들어 classification**하는 것을 확인할 수 있다.

이는 GAP 이후 Flatten + FC랑 완전히 똑같은 연산이다.

현대 네트워크에서 `1x1 Conv` 라는 용어를 더 선호하는 이유는 전체 네트워크를 "Fully Convolution"하다고 개념적으로 일관성 있게 설명할 수 있고, 다양한 입력 이미지 크기에 더 유연하게 대응할 수 있기 때문이다.