

SENet: Squeeze and Excitation Networks

Squeeze and Excitation Networks (CVPR 2018)

Jie Hu, Li Shen, Gang Sun

[Squeeze and Excitation Networks \(CVPR 2018\)](#)

[Abstract](#)

[1. Introduction](#)

[2. Related Work](#)

[Deep architectures.](#)

[Attention and gating mechanisms](#)

[3. Squeeze-and-Excitation Blocks](#)

[3-1. Squeeze : Global Information Embedding](#)

[3-2. Excitation : Adaptive Recalibration](#)

[3-3. Exemplars : SE-Inception and SE-ResNet](#)

[4. Model and Computational Complexity](#)

[5. Implementation](#)

[6. Experiments](#)

[6-1. ImageNet Classification](#)

[Network depth](#)

[Mobile setting](#)

[Integration with modern architectures](#)

[6-2. Scene Classification](#)

[6-3. Object Detection on COCO](#)

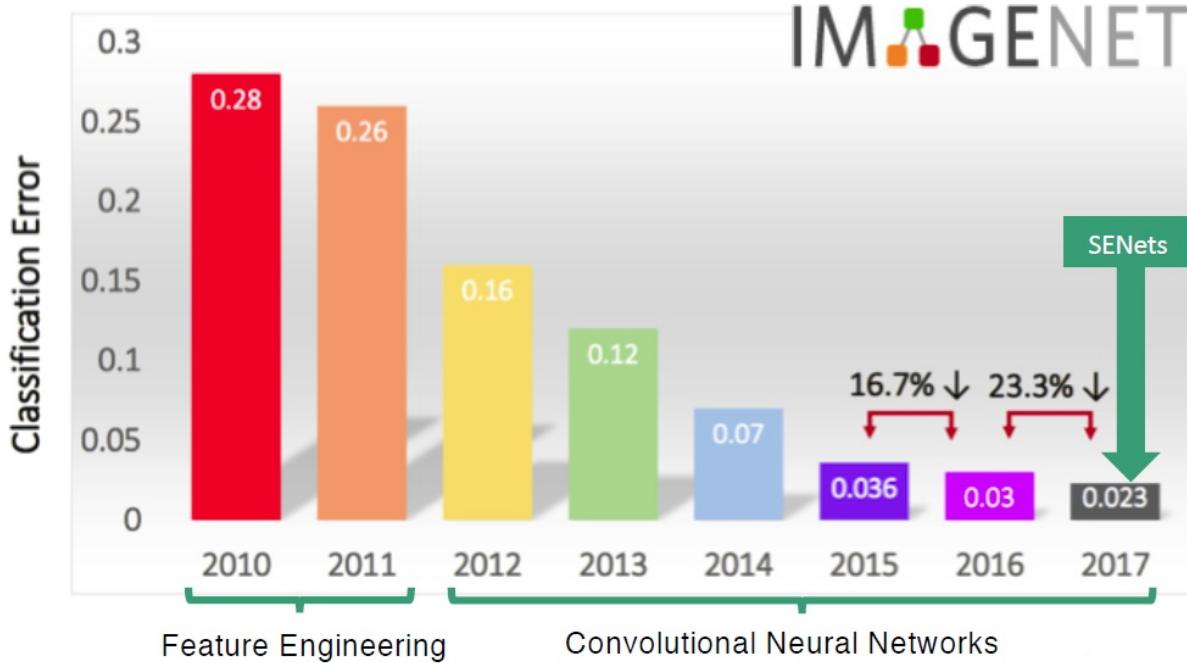
[6-4. Analysis and Interpretation](#)

[Reduction ratio](#)

[Role of Excitation](#)

[7. Conclusion](#)

Abstract



SENet은 2017 ILSVRC에서 1등을 차지한 CNN 구조

Classification에서 top-5 error 2.251%로 전년도 대비 25% 나아진 결과를 보이며 SOTA를 갱신함

보통 Spatial information에 집중하는 다른 연구와는 다르게 SENet은 채널 간의 상호작용을 학습한 뒤, 그 정보를 사용해 채널 단위로 새로운 가중치를 주어 **최소한의 추가 계산 비용으로 상당한 성능 향상을** 이끌어냄

1. Introduction

CNN의 각 conv layer의 filter들은 입력 채널을 따라 인근의 **spatial connectivity pattern**을 나타냄

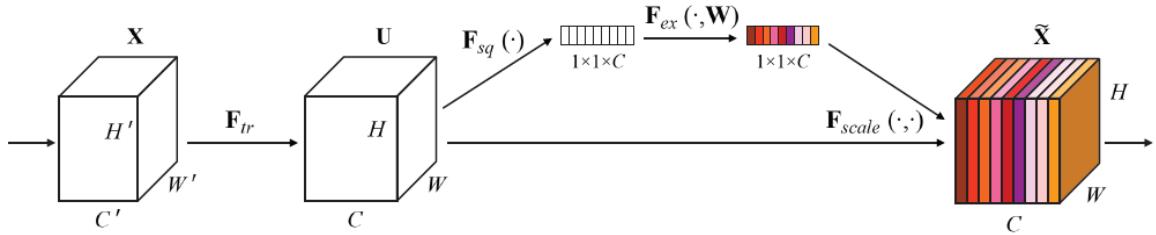
CNN은 일련의 conv layer에 non-linear activation(활성함수)과 downsampling operator(풀링)를 끼워 넣음으로써 **image representation**을 생성

Computer vision 연구의 중심 주제는 주어진 작업에 가장 중요한 이미지의 속성을만을 캡쳐하고 성능을 향상시킬 수 있는 **powerful representation**을 찾는 것!

최근 연구는 feature간 spatial correlation을 캡처하는데 도움이 되는 네트워크를 통합

→ SENet에서는 feature recalibration을 수행할 수 있는 메커니즘으로 **channel간 relationship조사**

→ 이 메커니즘은 global information으로부터 유익한 feature를 선택적으로 강조하면서 덜 유용한 feature를 억제함



F_{tr} 은 입력 X 는 feature map U 로 mapping되고, U 는 다음의 두 operation을 순서대로 통과



Squeeze($F_{sq}(\cdot)$)

- Spatial dimension(HxW)에 걸친 feature map을 aggregation함으로써 channel descriptor생성
- 이 descriptor는 channel-wise feature response의 global distribution을 임베딩하여, 네트워크의 모든 layer에서 information을 사용할 수 있도록 함



Excitation($F_{ex}(\cdot, W)$, $F_{scale}(\cdot, \cdot)$)

- 임베딩을 input으로 하여 per-channel modulation weight를 생성하는 simple-self-gating 메커니즘 형태
- 이 weight들은 feature map U 에 적용되며, 이렇게 생성된 SE block의 출력은 후속 layer에 바로 사용될 수 있음

이러한 **SE block**의 stack으로 **SENet**을 구성

SE block은 다양한 depth의 네트워크 아키텍쳐에서 original block의 대체로도 사용

SE block의 구조는 간단하면서도 성능을 효과적으로 향상시킬 수 있으며 기존의 **SOTA architecture**에서 SE block에 대응되는 component만 교체하는 것으로 직접적인 사용이 가능

또한 계산적으로도 가볍기 때문에 SE block을 사용할 시에도 model complexity와 computational cost를 약간만 증가시킴

2. Related Work

Deep architectures.

- VGGNet, Inception 네트워크의 depth를 늘리면 학습할 수 있는 representation의 품질을 크게 높일 수 있음을 보여줌
- ResNet) batch-normalization, identity-based skip connection을 사용해 상당히 깊고 강력한 네트워크를 학습시킴
- 그 외) 네트워크 내에 포함된 computational unit을 개선하거나 학습된 transformation의 cardinality를 증가시킴



반대로 SENet에서는 **global information**을 채널 간 **non-linear dependency**의 모델링을 명시적으로 해주는 메커니즘의 unit에 제공

학습 프로세스를 용이하게 하고 네트워크의 representational power(네트워크 표현력)를 크게 향상시킬 수 있다고 주장

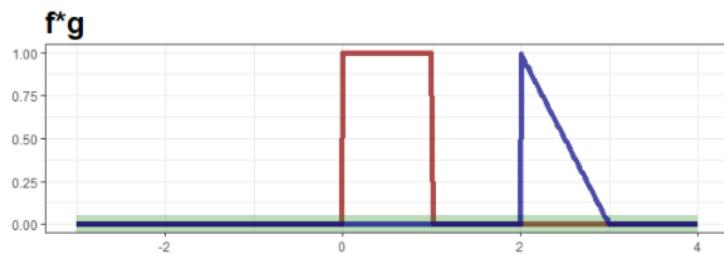
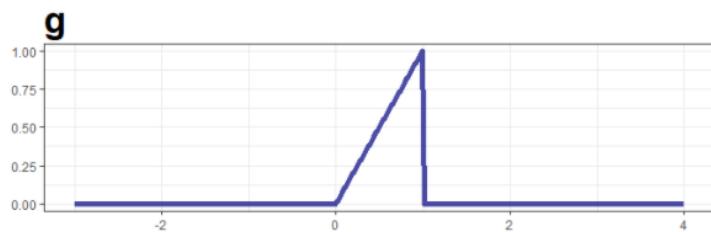
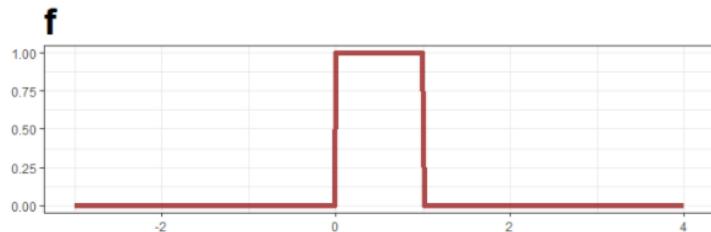
▼ Cross Correlation이란?

Convolution은 두 함수 (f, g) 를 이용해서 한 함수(f)의 모양이 나머지 함수(g)에 의해 모양이 수정된 제3의 함수 $(f * g)$ 를 생성해주는 연산자

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t - \tau)d\tau$$

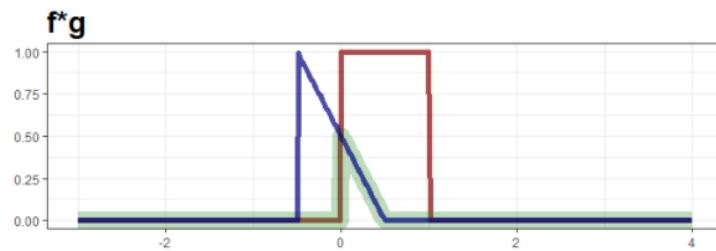
함수 f 와 y축 기준 좌우가 반전된 함수 g 를 우측으로 t만큼 이동한 함수를 곱한 함수의 적분

$$f(t) = I(t \in [0, 1]), g(t) = tI(t \in [0, 1])$$

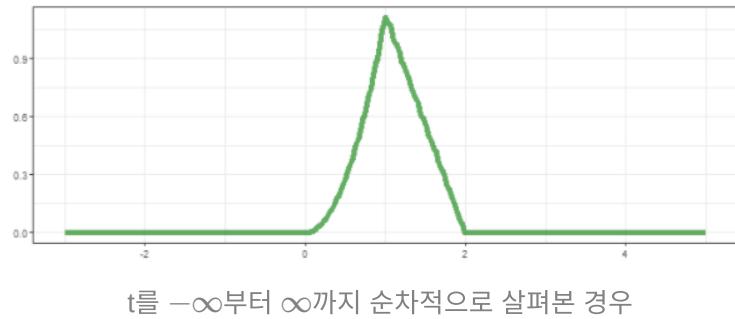


ex. $t = 3$ 인 경우

빨간 선 f 는 고정된 것과 파란선 g 를 y 축 대칭시킨 뒤 우측으로 3만큼 이동
두 함수를 곱한 것이 적분의 대상이 되며 초록띠와 대응됨
초록띠와 x 축으로 둘러 쌓인 면적이 0이므로 convolution은 0



ex. $t = 0.5$ 인 경우



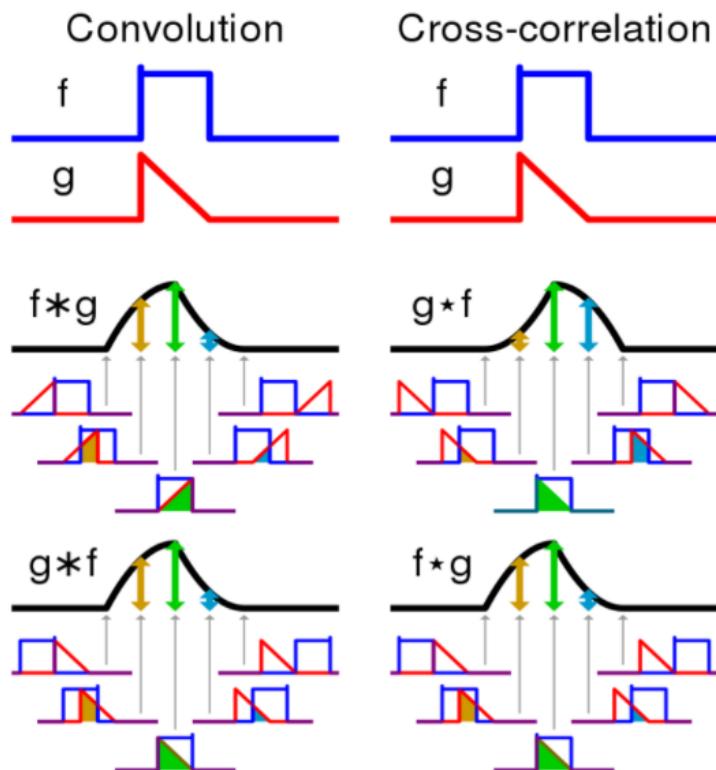
Cross-correlation(교차상관)

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t + \tau)d\tau$$

g 에서 y축 반전을 거치지 않는 것이 convolution과의 차이

Convolutional Layer의 연산법은 정확하게는 cross-correlation

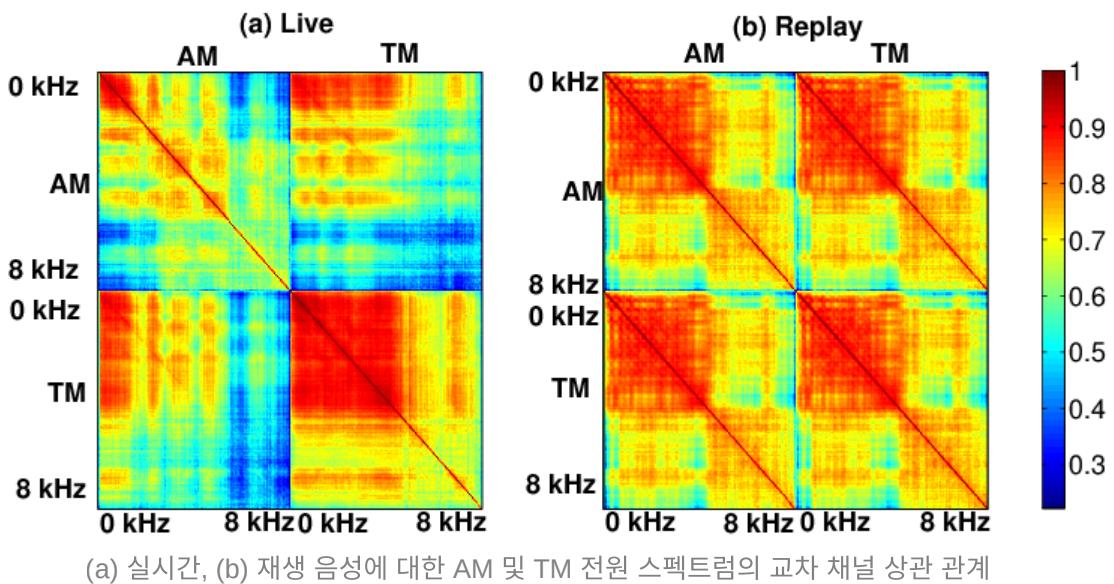
하지만 CNN에서는 가중치를 학습하기 때문에 convolution과 cross-correlation을 정확히 구분할 필요가 없음



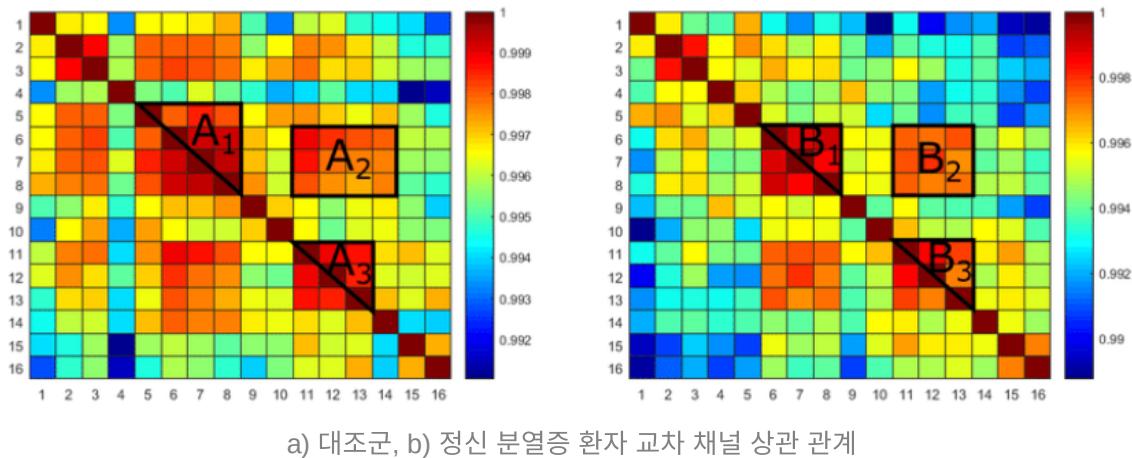
▼ Cross Channel Correlation이란?

일반적으로 spatial structure와 독립적으로 새로운 기능의 조합으로 맵핑

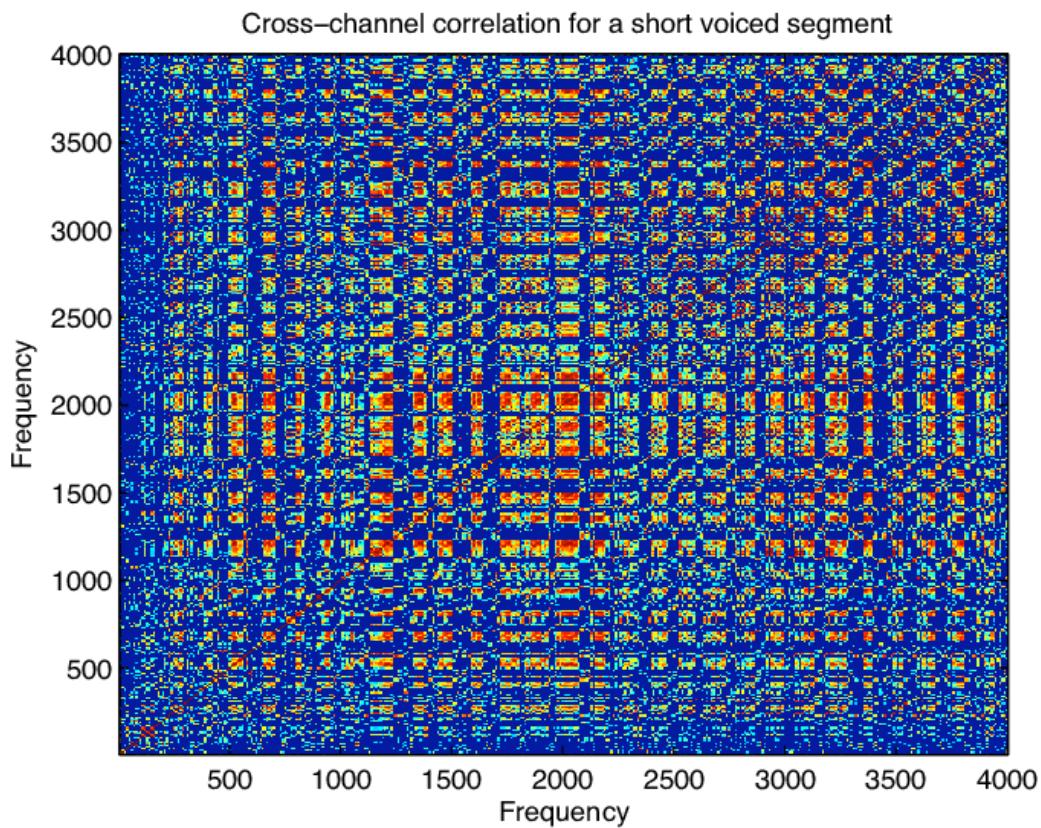
1×1 의 컨볼루션으로 표준 컨볼루션 필터를 사용하여 매핑



(a) 실시간, (b) 재생 음성에 대한 AM 및 TM 전원 스펙트럼의 교차 채널 상관 관계



a) 대조군, b) 정신 분열증 환자 교차 채널 상관 관계



주파수 135.5Hz에서 짧은 유성 세그먼트에 대한 모든 주파수 빈의 교차 채널 상관 관계

Attention and gating mechanisms

Attention은 이용 가능한 computational resource를 signal 중 가장 유익한 요소에 할당하도록 유도

Sequence Learning, Localization and understanding in image, Image captioning, Lip reading 분야를 포함한 많은 분야에서 유용성을 입증

Deep residual network의 내부에 삽입되는 hourglass module에 기반한, **강력한 trunk-and-mask attention 매커니즘을 도입**



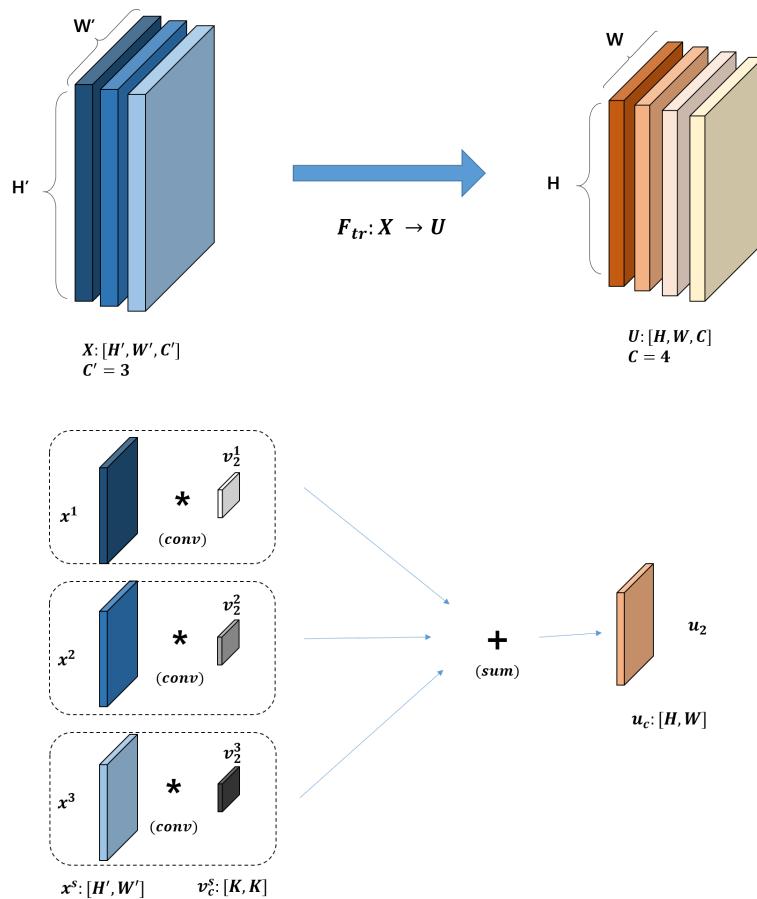
반대로, SE block은 **computationally efficient**한 **channel-wise relationship**을 모델링하여, 네트워크의 representational power를 향상시키는 것에 중점을 둔 **lightweight gating** 메커니즘으로 구성

3. Squeeze-and-Excitation Blocks

Squeeze-and-Excitation block은 입력 X 를 feature-map U 에 mapping하는 transformation F_{tr} 위에 구축될 수 있는 computational unit

$$u_c = v_c * X = \sum_{s=1}^{C'} v_c^s * x^s$$

- $*$ 은 convolutional operation
- $X = [x^1, x^2, \dots, x^{C'}]$, $V = [v^1, v^2, \dots, v^{C'}]$, $U = [u^1, u^2, \dots, u^{C'}]$
- $v_c = [v_c^1, v_c^2, \dots, v_c^{C'}]$
 v_c 는 c-th filter의 parameter
- v_c^s 는 v_c 의 s-th channel에 해당하는 2D spatial filter
- $u_c \in R^{H \times W}$





SE block의 두 가지 목적

1. Global Information에 대한 access 제공 (GAP)
2. Next transformation 이전에, squeeze와 excitation으로 이루어진 2-step operation을 통한 filter response의 recalibration

3-1. Squeeze : Global Information Embedding

이전) 학습된 filter들은 local receptive field에서 동작하기 때문에 transformation output인 U 의 각 unit들은 해당 영역 외의 contextual information을 이용할 수 없음

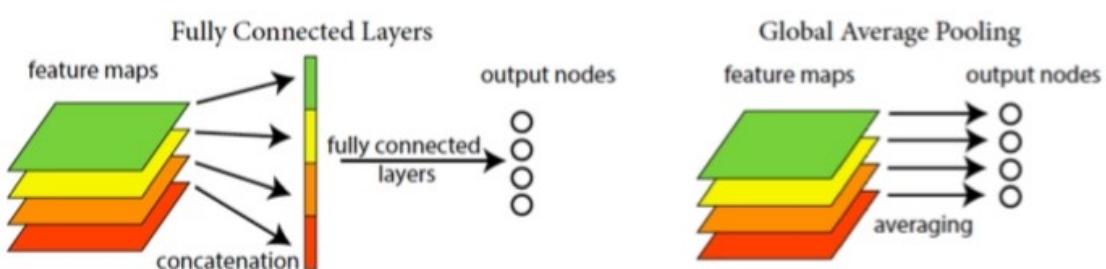
→ channel descriptor로 global spatial information을 squeeze

$$z_c = F_{sq}(u_c) = \frac{1}{H \times W} \sum_{i=1}^H \sum_{j=1}^W u_c(i, j)$$

Transformation의 출력인 U 는, 전체 이미지에 대한 local descriptor의 collection
(=이전의 feature engineering task)

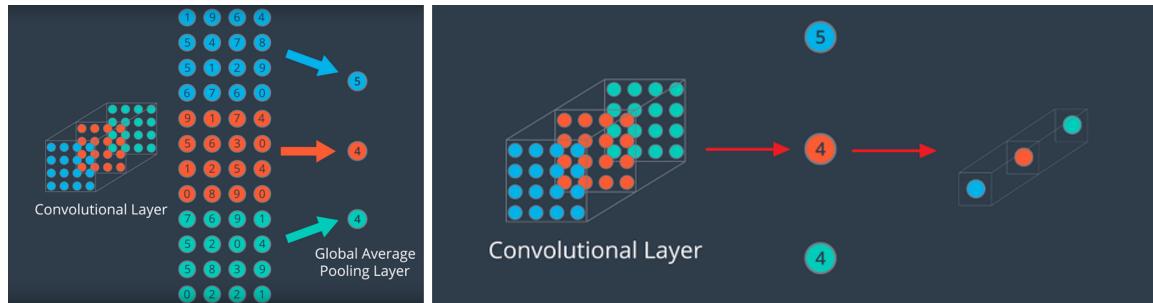
▼ GAP(Global Averaging Pooling)이란?

- Max(Average) Pooling보다 더 급격하게 feature의 수를 줄임
- GAP의 목적은 feature를 1차원 벡터로 만들기 위함
(height, width, channel) → (channel,)



같은 채널의 feature들을 모두 평균을 낸 다음에 채널의 개수만큼의 원소를 가지는 벡터로 만듦

→ 어떤 크기의 feature라도 같은 채널의 값들을 하나의 평균 값으로 대체하기 때문에 벡터가 되며 최종 출력에 FC Layer 대신 사용 가능



FC Layer를 classifier로 하는 경우 파라미터의 수가 많이 증가하고 feature 전체를 matrix 연산하기 때문에 위치에 대한 정보도 사라지며 input의 크기 또한 제한

cf. GAP은 파라미터가 추가되지 않으므로 학습 측면에서 유리하며 FC Layer에서 파라미터의 개수가 폭발적으로 추가되지 않아 overfitting 측면에서도 유리

3-2. Excitation : Adaptive Recalibration

Squeeze에서 얻은 aggregated information을 사용하기 위해, channel-wise dependency를 모두 캡처하는 두 번째 operation



두 가지 기준

- **Flexible**해야 함
특히, channel 간의 non-linear interaction을 학습할 수 있어야 함
- 여러 channel을 강조할 수 있도록, **non-mutually-exclusive relationship**을 학습해야 함
(mutually-exclusive relationship은 one-hot activation)

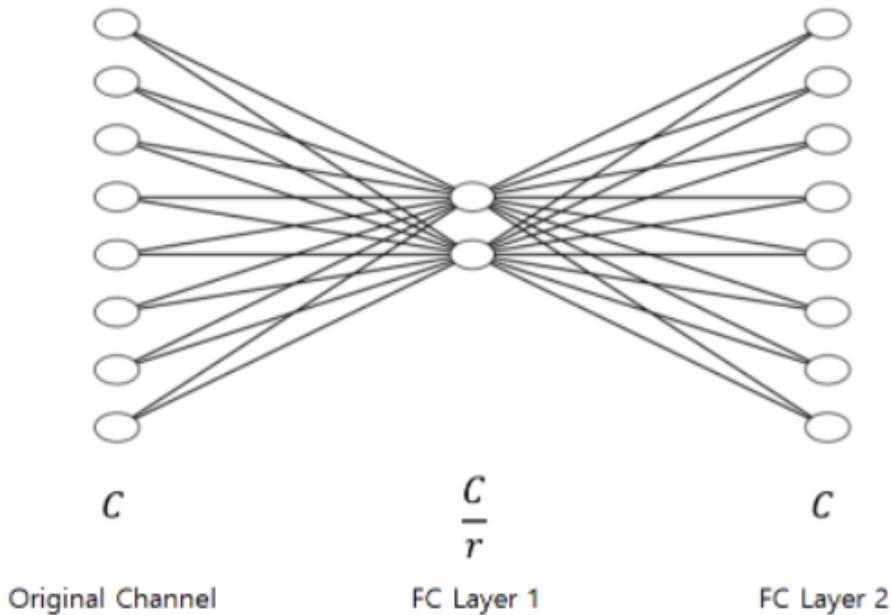
→ 이런 기준들을 충족시키기 위해, **sigmoid activation**을 사용하는 간단한 gating mechanism 채택

$$s = F_{ex}(z, W) = \sigma(g(z, W)) = \sigma(W_2 \delta(W_1 z))$$

- δ 는 ReLU
- σ 는 sigmoid
- $W_1 \in R^{\frac{C}{r} \times C}$
- $W_2 \in R^{C \times \frac{C}{r}}$

※ dimensionality-reduction layer(bottleneck) > relu > dimensionality-increasing layer > sigmoid

model complexity를 제한하면서 일반화를 둡기 위해, non-linearity 주위에 2개의 FC layer로 bottleneck을 형성



$$\tilde{x}_c = F_{scale}(u_c, s_c) = s_c u_c$$

SE block의 최종 출력은 **activation s** 로 U 를 **rescaling**하여 얻음

- $\tilde{X} = [\tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_c]$
- $F_{scale}(u_c, s_c)$ 는 channel-wise multiplication



Excitation에서 차원을 축소해 가중치를 뽑아주는 것의 의미

Squeeze 층에서 GAP를 한것을 Excitation에서 바로 sigmoid 연산을 해주면 단순히 featuremap에서 **channel별 평균값이 큰 것이 큰 가중치를 가지게 됨(X)**

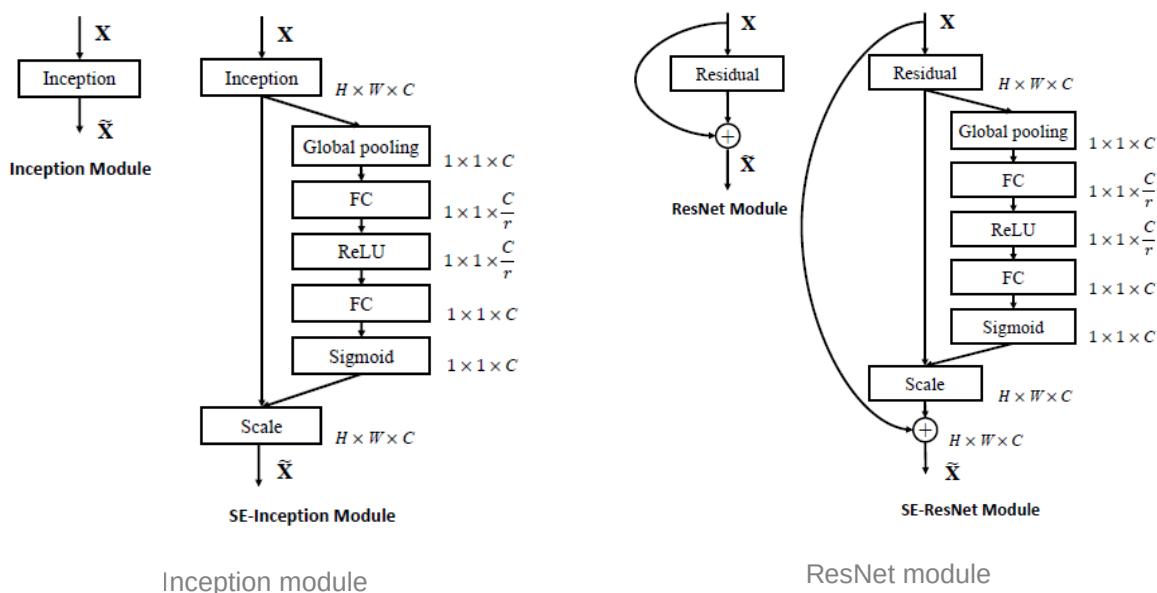
(ex. channel이 256인 경우 GAP하고 FC를 거치게 되면 $256 \times 256 + 256$ 으로 너무 커짐)

channel별 가중치를 조절하기 위해서는 추가적인 FC Layer를 원래의 차원 그대로 넣어주면 parameter가 너무 크게 증가해 버리니까 차원을 조금 줄여주는 것!

(FC Layer에서 일반화를 해줌으로써 조금 더 일반적인 가중치를 뽑을 수 있게 됨)

3-3. Exemplars : SE-Inception and SE-ResNet

SE block은 각 convolution에 따라오는 non-linearity 뒤에 삽입하는 방식으로 AlexNet, VGGNet과 같은 기존의 architecture에 통합 (flexibility)



Squeeze와 Excitation 모두 identity branch와 summation이 되기 전에 수행됨

ResNeXt, Inception-ResNet, MobileNet, ShuffleNet에 SE block을 통합한 버전도 비슷하게 구성

Output size	ResNet-50	SE-ResNet-50	SE-ResNeXt-50 ($32 \times 4d$)
112×112		conv, 7×7 , 64, stride 2 max pool, 3×3 , stride 2	
56×56	$\begin{bmatrix} \text{conv, } 1 \times 1, 64 \\ \text{conv, } 3 \times 3, 64 \\ \text{conv, } 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} \text{conv, } 1 \times 1, 64 \\ \text{conv, } 3 \times 3, 64 \\ \text{conv, } 1 \times 1, 256 \\ fc, [16, 256] \end{bmatrix} \times 3$	$\begin{bmatrix} \text{conv, } 1 \times 1, 128 \\ \text{conv, } 3 \times 3, 128 \\ C = 32 \\ \text{conv, } 1 \times 1, 256 \\ fc, [16, 256] \end{bmatrix} \times 3$
28×28	$\begin{bmatrix} \text{conv, } 1 \times 1, 128 \\ \text{conv, } 3 \times 3, 128 \\ \text{conv, } 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} \text{conv, } 1 \times 1, 128 \\ \text{conv, } 3 \times 3, 128 \\ \text{conv, } 1 \times 1, 512 \\ fc, [32, 512] \end{bmatrix} \times 4$	$\begin{bmatrix} \text{conv, } 1 \times 1, 256 \\ \text{conv, } 3 \times 3, 256 \\ C = 32 \\ \text{conv, } 1 \times 1, 512 \\ fc, [32, 512] \end{bmatrix} \times 4$
14×14	$\begin{bmatrix} \text{conv, } 1 \times 1, 256 \\ \text{conv, } 3 \times 3, 256 \\ \text{conv, } 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} \text{conv, } 1 \times 1, 256 \\ \text{conv, } 3 \times 3, 256 \\ \text{conv, } 1 \times 1, 1024 \\ fc, [64, 1024] \end{bmatrix} \times 6$	$\begin{bmatrix} \text{conv, } 1 \times 1, 512 \\ \text{conv, } 3 \times 3, 512 \\ C = 32 \\ \text{conv, } 1 \times 1, 1024 \\ fc, [64, 1024] \end{bmatrix} \times 6$
7×7	$\begin{bmatrix} \text{conv, } 1 \times 1, 512 \\ \text{conv, } 3 \times 3, 512 \\ \text{conv, } 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} \text{conv, } 1 \times 1, 512 \\ \text{conv, } 3 \times 3, 512 \\ \text{conv, } 1 \times 1, 2048 \\ fc, [128, 2048] \end{bmatrix} \times 3$	$\begin{bmatrix} \text{conv, } 1 \times 1, 1024 \\ \text{conv, } 3 \times 3, 1024 \\ C = 32 \\ \text{conv, } 1 \times 1, 2048 \\ fc, [128, 2048] \end{bmatrix} \times 3$
1×1		global average pool, 1000-d fc , softmax	

SE-ResNet-50과 SE-ResNeXt-50 구조

- f_c 는 excitation operation의 두 FC layer에 해당

4. Model and Computational Complexity

SE block이 실용화되려면, **performance(성능)**과 **model complexity(복잡도)**간의 trade-off를 고려

	original		re-implementation			SENet		
	top-1 err.	top-5 err.	top-1 err.	top-5 err.	GFLOPs	top-1 err.	top-5 err.	GFLOPs
ResNet-50 [13]	24.7	7.8	24.80	7.48	3.86	23.29 _(1.51)	6.62 _(0.86)	3.87
ResNet-101 [13]	23.6	7.1	23.17	6.52	7.58	22.38 _(0.79)	6.07 _(0.45)	7.60
ResNet-152 [13]	23.0	6.7	22.42	6.34	11.30	21.57 _(0.85)	5.73 _(0.61)	11.32
ResNeXt-50 [19]	22.2	-	22.11	5.90	4.24	21.10 _(1.01)	5.49 _(0.41)	4.25
ResNeXt-101 [19]	21.2	5.6	21.18	5.57	7.99	20.70 _(0.48)	5.01 _(0.56)	8.00
VGG-16 [11]	-	-	27.02	8.81	15.47	25.22 _(1.80)	7.70 _(1.11)	15.48
BN-Inception [6]	25.2	7.82	25.38	7.89	2.03	24.23 _(1.15)	7.14 _(0.75)	2.04
Inception-ResNet-v2 [21]	19.9 [†]	4.9 [†]	20.37	5.21	11.75	19.80 _(0.57)	4.79 _(0.42)	11.76

GFLOPs) 3.86 vs. 3.87 (0.26% 증가) / 정확도는 101과 비슷
single forward 및 backward) 190ms vs. 209ms(millisecond)

▼ GFLOPs(GPU FLoating point Operations Per Second) 이란?

컴퓨터의 성능을 수치로 나타낼 때 주로 사용되는 단위

초당 부동소수점 연산이라는 의미로 컴퓨터가 1초동안 수행할 수 있는 부동소수점 연산의 횟수

고정소수점(Fixed Point) vs. 부동소수점(Floating Point)

컴퓨터는 숫자를 표현할 때 2진수를 사용



ex. $13 = 8 + 4 + 1 \rightarrow 1101$

ex. $0.75 = 0.5 + 0.25 \rightarrow 0.11$

ex. $0.3 \Rightarrow 0.01001100110011\dots\dots(0011)$ 의 무한 반복

2진수로 표현하지 못하는 소수가 발생 → 근사치 값이 저장

근사 값을 저장하는 방법 고정소수점, 부동소수점

고정소수점

정수를 표현하는 비트 수와 소수를 표현하는 비트 수를 미리 정해놓고 해당 비트 만큼만 사용해서 숫자를 표현하는 방식

정수를 표현하는 bit를 늘리면 큰 숫자를 표현할 수 있지만 정밀한 숫자를 표현하기 힘듦

↔ 소수를 표현하는 bit를 늘릴 경우 정밀한 숫자를 표현할 수 있지만 큰 숫자를 표현하지 못함

부동소수점

부동 소수점을 표현하는 방식도 정하는 방식에 따라 다를 수 있지만 일반적으로 사용하고 있는 방식은 IEEE에서 표준으로 제안한 방식

IEEE 754 Standard for Floating-Point Arithmetic



263.3을 2진수 부동 소수점 방식으로 변환

100000111.010011001100110... 으로 표현되던 것을 맨 앞에 있는 1 바로 뒤로 소수점을 옮겨서 표현하도록 변환

→ 1.00000111010011001100110... * 2^8(2의 8승) 으로 표현

- 부호 비트(1 bit) : 0 (양수)
- 지수 비트(8 bit) : 10000111 (127 + 8 = 135)
- 가수 비트(23 bit) : 00000111010011001100110

SE block에 의해 발생하는 작은 계산 비용은 모델 성능에 대한 contribution에 의해 정당화

추가 parameter는 gating mechanism의 두 FC layer에서만 발생 (전체 capacity의 작은 일부)

$$\frac{2}{r} \sum_{s=1}^S N_s \cdot C_s^2$$

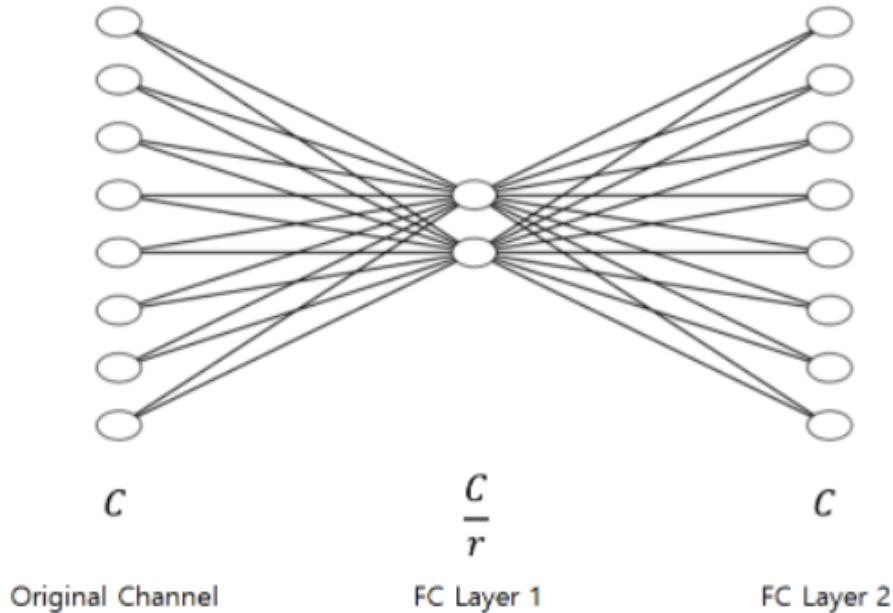
- r : reduction ratio

- S : stage

동일한 크기의 feature map에서 동작되는 block collection

56x56 feature map을 출력하는 3개의 block은 동일한 stage

- C_s : output channels의 차원
- N_s : stage s 에 대해 반복되는 block 수



각 층의 뉴런의 갯수는 채널의 갯수인 C 에서 시작해서 $\frac{C}{r}$ 로 감소했다가 다시 C 로 증가
 r 이 클수록 은닉 층의 뉴런 갯수가 더 적어지고, 복잡도도 더 낮아짐

하나의 SE block에 추가된 parameter수) $C \times \frac{C}{r} \times 2 = \frac{2}{r}C^2$

총 N개의 SE block에 추가된 parameter수) $\frac{2}{r}C^2 \times N = \frac{2}{r}NC^2$

25M parameter vs. 25M + 2.5M parameter (10% 증가)

final stage SE block을 제거하면 parameter증가율을 4%로 줄이면서 성능 손실 0.1% 미만



병목 구조 효과

- 파라미터의 개수를 많이 늘리지 않기 위함
- 일반화에 도움을 주기 위함

5. Implementation

동일한 Data Augmentation과 hyperparameter 적용

- Random cropping, Random horizontal flipping
- Synchronous SGD with momentum 0.9
- Minibatch size : 1024
- Initial learning rate : 0.6
- 30 epoch마다 learning rate를 10으로 나눔
- Weight initializer : He initialization
- Trained for 100 epochs

6. Experiments

6-1. ImageNet Classification

1.28M개의 training image, 50K개의 validation image로 구성된 ImageNet 2012 dataset에 대해 실험

Network depth

	original		re-implementation			SENet		
	top-1 err.	top-5 err.	top-1 err.	top-5 err.	GFLOPs	top-1 err.	top-5 err.	GFLOPs
ResNet-50 [13]	24.7	7.8	24.80	7.48	3.86	23.29 _(1.51)	6.62 _(0.86)	3.87
ResNet-101 [13]	23.6	7.1	23.17	6.52	7.58	22.38 _(0.79)	6.07 _(0.45)	7.60
ResNet-152 [13]	23.0	6.7	22.42	6.34	11.30	21.57 _(0.85)	5.73 _(0.61)	11.32
ResNeXt-50 [19]	22.2	-	22.11	5.90	4.24	21.10 _(1.01)	5.49 _(0.41)	4.25
ResNeXt-101 [19]	21.2	5.6	21.18	5.57	7.99	20.70 _(0.48)	5.01 _(0.56)	8.00
VGG-16 [11]	-	-	27.02	8.81	15.47	25.22 _(1.80)	7.70 _(1.11)	15.48
BN-Inception [6]	25.2	7.82	25.38	7.89	2.03	24.23 _(1.15)	7.14 _(0.75)	2.04
Inception-ResNet-v2 [21]	19.9 [†]	4.9 [†]	20.37	5.21	11.75	19.80 _(0.57)	4.79 _(0.42)	11.76

공정한 비교를 위해 baseline 모델을 re-implementation하여 학습한 결과와, 대응하는 SENet 버전의 결과를 모두 보여줌

괄호 안의 숫자는 re-implementation에 대한 성능 향상도

→ SE block은 계산량이 매우 작게 증가하면서 여러 depth에서 일관적으로 성능을 향상

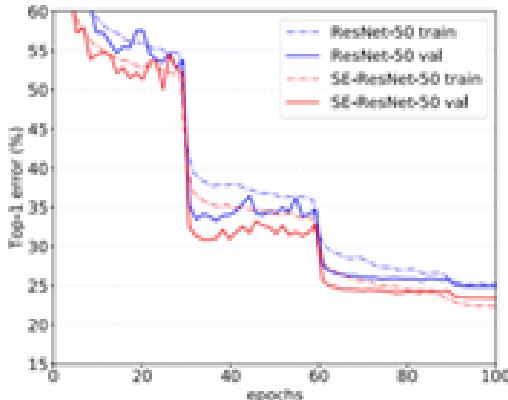


Figure 4: Training curves of ResNet-50 and SE-ResNet-50 on ImageNet.

Mobile setting

mobile-optimized network 중 대표적인 MobileNet, ShuffleNet 두 가지 아키텍처

- SGD with momentum 0.9
- Initial learning rate : 0.1
- Validation loss가 줄어들지 않을 때마다 10으로 나눔
- ~400 epochs

	original			re-implementation			SENet			
	top-1 err.	top-5 err.	top-1 err.	top-5 err.	MFLOPs	Params	top-1 err.	top-5 err.	MFLOPs	Params
MobileNet [64]	29.4	-	28.4	9.4	569	4.2M	25.3 _(3.1)	7.7 _(1.7)	572	4.7M
ShuffleNet [65]	32.6	-	32.6	12.5	140	1.8M	31.0 _(1.6)	11.1 _(1.4)	142	2.4M

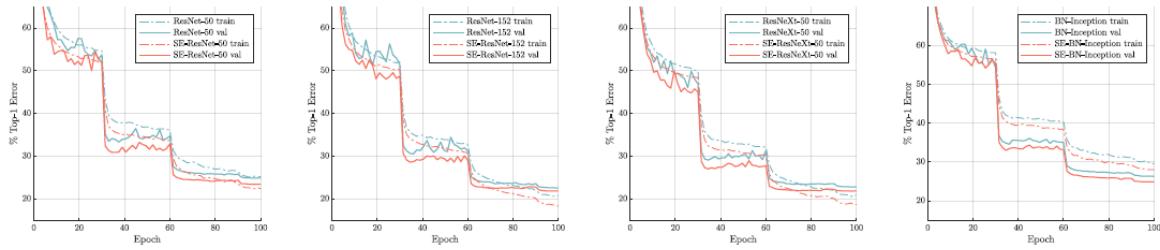
→ SE block이 계산 비용을 최소로 증가시키면서도 일관된 큰 성능 향상도

Integration with modern architectures

SE block을 두 개의 최신 아키텍처인 Inception-Resnet-v2와 ResNeXt(32x4d)에 통합

VGG-16과 BN-Inception 등 non-residual 네트워크에서 동작할 때의 SE block 영향 평가

→ Residual network에 대한 결과와 유사하게, SE block이 non-residual setting에서 성능을 개선



ResNet-50, ResNet-152, ResNeXt-50, BN-Inception

ILSVRC 1위 submission과 SE block을 수정한 ResNeXt와 통합하여 **SENet-154**를 구성

SENet-154의 성능은 224x224 및 320x320 pixel로 center crop하여 측정

	224 × 224		320 × 320 / 299 × 299	
	top-1 err.	top-5 err.	top-1 err.	top-5 err.
ResNet-152 [13]	23.0	6.7	21.3	5.5
ResNet-200 [14]	21.7	5.8	20.1	4.8
Inception-v3 [20]	-	-	21.2	5.6
Inception-v4 [21]	-	-	20.0	5.0
Inception-ResNet-v2 [21]	-	-	19.9	4.9
ResNeXt-101 (64 × 4d) [19]	20.4	5.3	19.1	4.4
DenseNet-264 [17]	22.15	6.12	-	-
Attention-92 [58]	-	-	19.5	4.8
PyramidNet-200 [77]	20.1	5.4	19.2	4.7
DPN-131 [16]	19.93	5.12	18.55	4.16
SENet-154	18.68	4.47	17.28	3.79

ImageNet validation set에 대한 성능 비교

▼ ResNeXt이란?

기존의 **ResNet**과 **Inception Model**의 주된 아이디어인 split-transform-merge 구조를 합친 것!

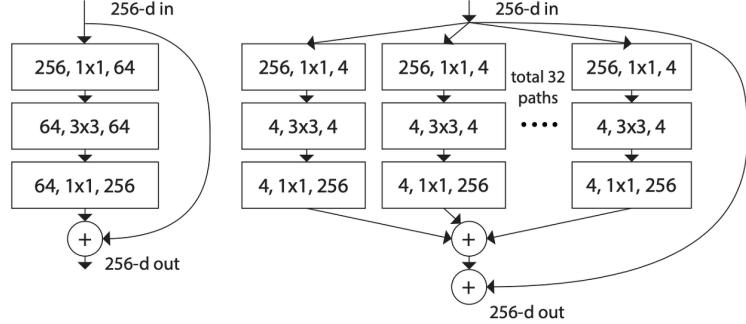


Figure 1. **Left:** A block of ResNet [14]. **Right:** A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).

같은 layer구성을 가지는데 이것을 grouped convolution이라고 함

ResNeXt는 다른 모델들 (ResNet-101/152/200, Inception-v3, Inception-ResNet-v2) 보다 더 간단한 구조를 가졌지만 더 나은 성능

stage	output	ResNet-50	ResNeXt-50 (32×4d)
conv1	112×112	7×7, 64, stride 2	7×7, 64, stride 2
		3×3 max pool, stride 2	3×3 max pool, stride 2
conv2	56×56	$\left[\begin{array}{c} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{array} \right] \times 3$	$\left[\begin{array}{c} 1\times1, 128 \\ 3\times3, 128, C=32 \\ 1\times1, 256 \end{array} \right] \times 3$
conv3	28×28	$\left[\begin{array}{c} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{array} \right] \times 4$	$\left[\begin{array}{c} 1\times1, 256 \\ 3\times3, 256, C=32 \\ 1\times1, 512 \end{array} \right] \times 4$
conv4	14×14	$\left[\begin{array}{c} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{array} \right] \times 6$	$\left[\begin{array}{c} 1\times1, 512 \\ 3\times3, 512, C=32 \\ 1\times1, 1024 \end{array} \right] \times 6$
conv5	7×7	$\left[\begin{array}{c} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{array} \right] \times 3$	$\left[\begin{array}{c} 1\times1, 1024 \\ 3\times3, 1024, C=32 \\ 1\times1, 2048 \end{array} \right] \times 3$
	1×1	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax
# params.		25.5×10^6	25.0×10^6
FLOPs		4.1×10^9	4.2×10^9

Table 1. **(Left)** ResNet-50. **(Right)** ResNeXt-50 with a 32×4d template (using the reformulation in Fig. 3(c)). Inside the brackets are the shape of a residual block, and outside the brackets is the number of stacked blocks on a stage. “C=32” suggests grouped convolutions [24] with 32 groups. *The numbers of parameters and FLOPs are similar between these two models.*

ResNet-50과 매우 비슷한 구조를 가짐

6-2. Scene Classification

Scene classification을 위한 Places365-Challenge dataset에 대한 실험



Buildings

Forest

Mountains



Glacier

Sea

Street

- 365-class
- 8M training set
- 36.5K validation set (1000K = 1 Million)

	top-1 err.	top-5 err.
Places-365-CNN [72]	41.07	11.48
ResNet-152 (ours)	41.15	11.61
SE-ResNet-152	40.37	11.01

Scene understanding 작업은 모델의 일반화 성능과 추상화 처리 능력에 대한 대안적인 평가를 제공

SE block의 효과를 평가하기 위한 baseline으로 ResNet-152를 채택

→ SE-ResNet-152의 top-5 error는 11.01%로 측정됐으며, 이는 ResNet-152의 11.61% 보다 낮음

6-3. Object Detection on COCO

COCO dataset을 사용하여 object detection 작업에 대한 SE block의 효과를 추가로 평가

- 80K training set
- 40K validation set
- Faster R-CNN base detection framework

	AP@IoU=0.5	AP
ResNet-50	57.9	38.0
SE-ResNet-50	61.0	40.4
ResNet-101	60.1	39.9
SE-ResNet-101	62.7	41.9

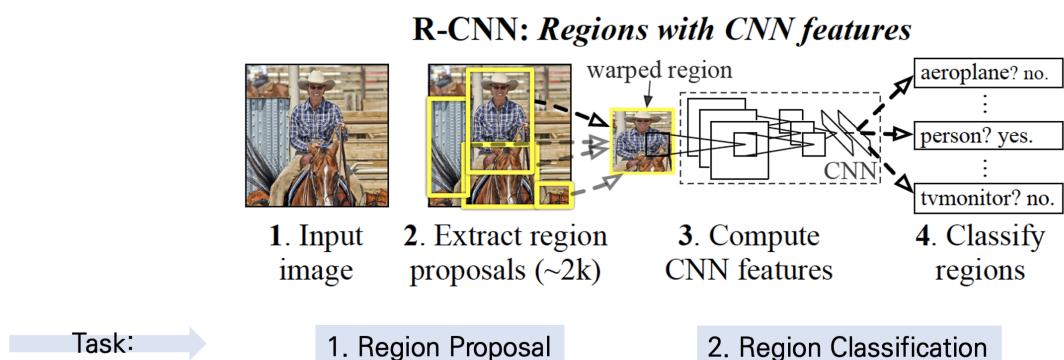
AP@IoU metric/AP metric

- 즉, 이 실험은 SE block의 일반성을 보여줌
 → 다양한 architecture/task/dataset에서 성능 향상을 이룰 수 있음

▼ R-CNN이란?

Object Detection 분야에 딥러닝을 최초로 적용시킨 모델이자 이전의 Object Detection 모델들과 비교해 성능을 상당히 향상시키고, 이후 여러 수정, 변형 모델들을 나오게 한, 의미있는 모델

R-CNN은 '**Regions with Convolutional Neural Networks features**'의 약자로, 즉 설정한 Region을 CNN의 feature(입력값)로 활용하여 Object Detection을 수행하는 신경망이라는 의미를 담고 있음



R-CNN의 기본적인 구조는 **2-stage Detector**

전체 task를 두 단계로 나눌 수 있는데, 우선 물체의 위치를 찾는 Region Proposal, 그리고 물체를 분류하는 Region Classification

6-4. Analysis and Interpretation

Reduction ratio

reduction ratio r 은 SE block의 **capacity 및 computational cost 변화에 관련된 hyperparameter**

이 hyperparameter에 따른 [performance / computational cost] 간의 trade-off를 알아보기 위해, 다양한 r 값에 대한 SE-ResNet-50을 실험

Ratio r	top-1 err.	top-5 err.	Params
2	22.29	6.00	45.7M
4	22.25	6.09	35.7M
8	22.26	5.99	30.7M
16	22.28	6.03	28.1M
32	22.72	6.20	26.9M
original	23.30	6.55	25.6M

r 이 작을수록 모델의 parameter 수는 크게 늘어나지만, complexity 증가에 따른 성능 향상이 단조롭진 않았으며, $r = 16$ 인 경우가 performance/complexity의 균형이 잘 맞았다.

Role of Excitation

SE block에서 **excitation operator의 기능을 명확한 그림으로 보여줌**

ImageNet dataset으로부터 의미와 모양이 다양한 4개의 class를 sampling



(a) goldfish

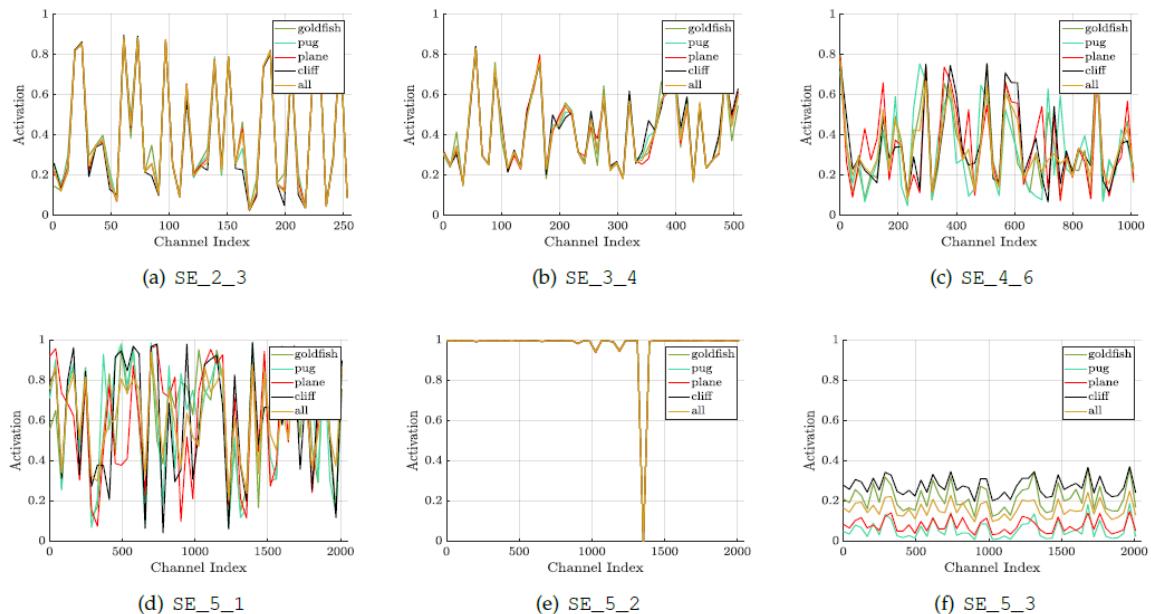
(b) pug

(c) plane

(d) cliff

금붕어, 얼굴이 납작하고 주름이 많은 개, 평지, 절벽

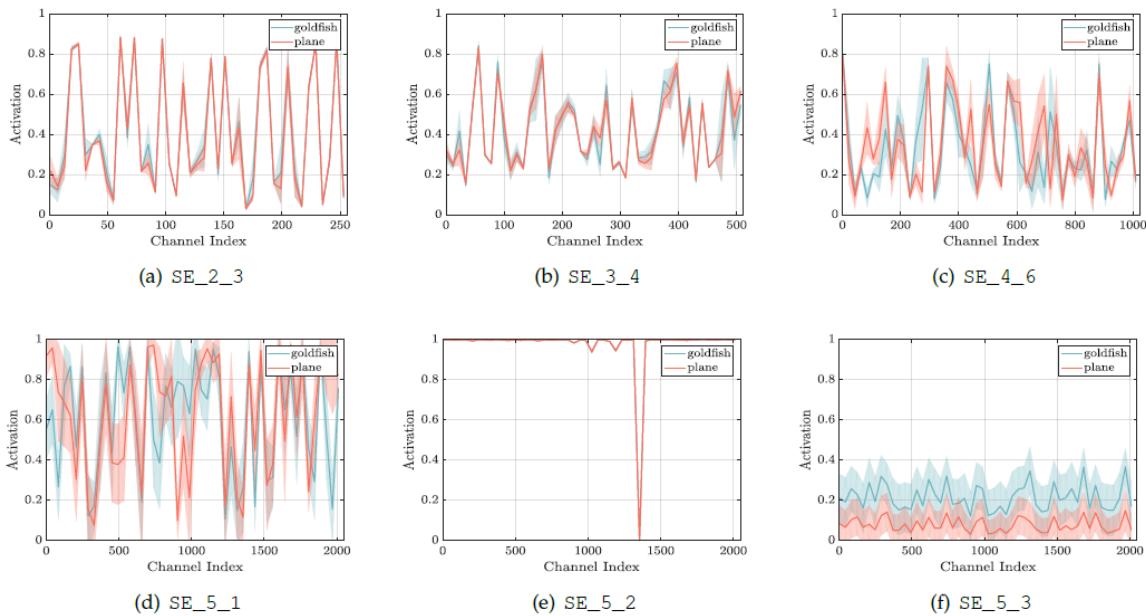
validation set에서 각 class에 대한 50개의 sample을 추출하여 각 stage의 마지막 SE block (downsampling 직전)에서 50개의 uniformly sampled channel에 대한 average activation을 계산



계산된 average activation distribution

- 네트워크 초반부의 layer에서는, 다른 class에 걸친 distribution이 매우 유사하다. (**SE_2_3**)
 - 하위 layer일 경우 더 일반적인 feature를 학습하기 때문
- 보다 중간부에서는, 각 channel의 가치가 class에 따라 훨씬 크게 달라진다. (**SE_4_6**) 및 (**SE_5_1**)
 - 서로 다른 클래스 간에는 feature들에 대한 선호도가 다르기 때문
 - 상위 layer일 경우 더 고차원의 feature 학습

- 네트워크의 마지막 stage에서는 약간 다른 현상을 관찰
 - (SE_5_2)에서는 대부분의 activation이 1에 가깝게 포화됨
 - (SE_5_3)에서는 다른 class 간에도 약간의 scale 변화 외에는 거의 유사한 패턴을 보였다.
 - 이는 (SE_5_2)와 (SE_5_3)이 네트워크 recalibration에 덜 중요하다는 것을 나타냄



goldfish와 plane에 대해 동일한 distribution

SE block은 instance-specific response를 생성하지만 아키텍처의 여러 layer에서 모델의 class-specific needs를 지원

7. Conclusion

- 본 논문에서는 dynamic한 **channel-wise feature recalibration**을 수행함으로써, 네트워크의 **representational power**를 향상시키는 architectural unit인 **SE block**을 제안
- 다양한 실험에서 SENet의 효과가 입증됐으며, **여러 종류의 dataset/task**에서 **SOTA 성능을 달성**
- SE block에 의해 생성 된 **feature importance**(특징 중요도) 값은 model compression을 위한 pruning과 같은 다른 task에서도 사용될 수 있음

