Image Classification with Squeeze-and-Excitation Networks and Efficient Networks

CV/DL Reading Group

November 5th, 2019

Contents

Squeeze-and-Excitation Networks

Overview

General Scope of SENets is to:

- Replace standard convolutional blocks with a new block called the SE block.
- Model non-linear dependencies over channels to assist the classification task.
- Improve a network's performance by adding only a small computational burden on it.

SE block

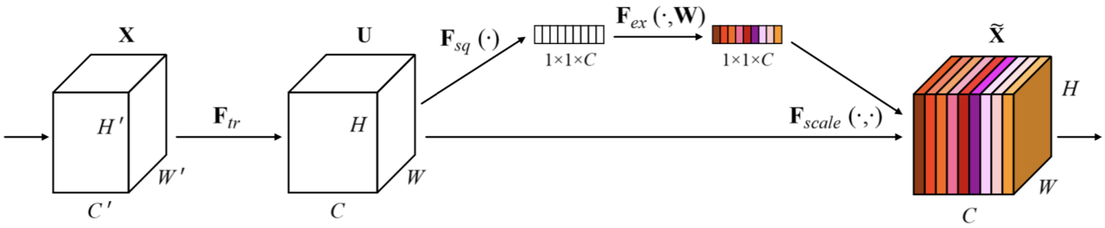


Figure 1: Form of a Squeeze-and-Excitation block

Steps

1. Convolve[1] over the input with $\mathbf{F_{tr}}$ to form $U \in R^{CxHxW}$
2. 'Squeeze' channels to scalar values.
3. 'Excitation' to emphasize several channels in terms of importance to the task.
4. Scale U into $\widetilde{X}$ and feed $\widetilde{X}$ to following layers.

[1]Taken care by the original network

Operations

Squeeze Operation:

1. Simply perform global average pooling to get the mean pixel value per channel.

Excitation Operation:

1. Use ratio r to perform dimensionality decrease and increase of around the non-linearity with FC layers.
2. Use sigmoid (non-linear and non-sparse) to learn the scale values.

SE blocks are plug-n-play

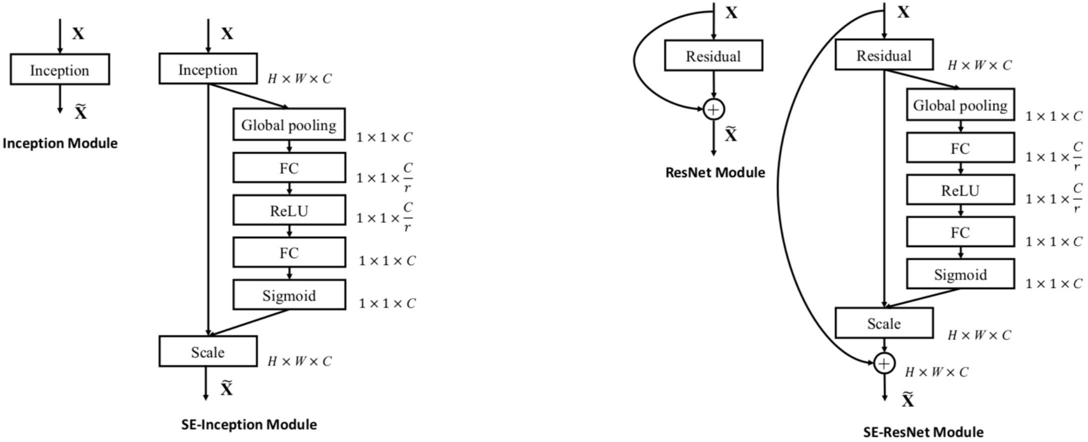


Figure 2: SE block as applied on an Inception-Net block (left) and a Res-Net block (right)

Results: ImageNet

	original		re-implementation			SENet		
	top-1 err.	top-5 err.	top-1err.	top-5 err.	GFLOPs	top-1 err.	top-5 err.	GFLOPs
ResNet-50 [10]	24.7	7.8	24.80	7.48	3.86	$23.29_{(1.51)}$	$6.62_{(0.86)}$	3.87
ResNet-101 [10]	23.6	7.1	23.17	6.52	7.58	$22.38_{(0.79)}$	$6.07_{(0.45)}$	7.60
ResNet-152 [10]	23.0	6.7	22.42	6.34	11.30	$21.57_{(0.85)}$	$5.73_{(0.61)}$	11.32
ResNeXt-50 [47]	22.2	-	22.11	5.90	4.24	$21.10_{(1.01)}$	$5.49_{(0.41)}$	4.25
ResNeXt-101 [47]	21.2	5.6	21.18	5.57	7.99	$20.70_{(0.48)}$	$5.01_{(0.56)}$	8.00
VGG-16 [39]	-	-	27.02	8.81	15.47	$25.22_{(1.80)}$	$7.70_{(1.11)}$	15.48
BN-Inception [16]	25.2	7.82	25.38	7.89	2.03	$24.23_{(1.15)}$	$7.14_{(0.75)}$	2.04
Inception-ResNet-v2 [42]	$19.9^{\dagger}$	$4.9^{\dagger}$	20.37	5.21	11.75	$19.80_{(0.57)}$	$4.79_{(0.42)}$	11.76

Results: COCO

	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

Relative improvement of 6.3% and 5% on ResNet-50 and ResNet-101.

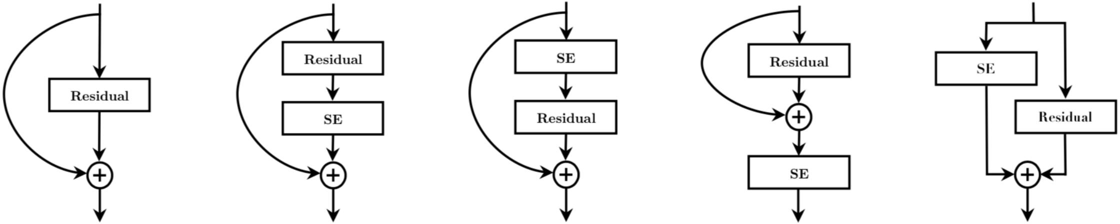
Results

And many more!

Ablation Study and Discussion

1. Reduction ratio r
2. Squeeze Operator (Max vs Global pooling)
3. Excitation Operator (Sigmoid vs Tanh vs ReLU)
4. SE blocks at different stages of the original network
5. SE operation at different places of the SE block

Ablation study on position of SE block



(a) Residual block (b) Standard SE block (c) SE-PRE block (d) SE-POST block (e) SE-Identity block

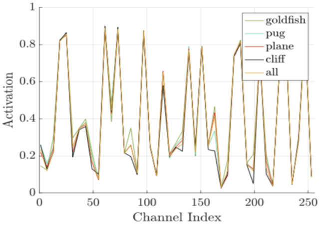
Figure 3: Different try-outs on place of SE block

Assessing the Squeeze operation

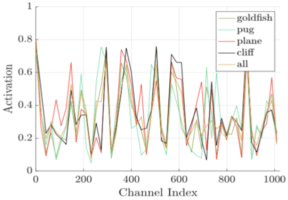
Is it possible that Squeeze helps to improve network performance simply due to the addition of extra parameters at each layer? 'No Squeeze' test (1x1 convs, no global pooling) to assess whether global information is the key method (*it is*!) at this step.

Assessing the Excitation operation

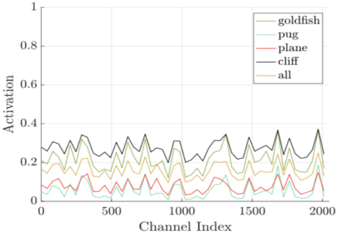
Sample 4 diverse classes of ImageNet (namely *goldfish*, *pub*, *plane* and *cliff*) and keep track of the activations at SE blocks of different depths.



(a) SE_2_3



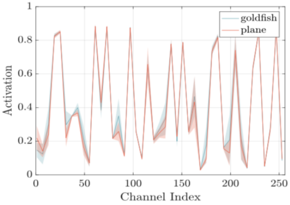
(c) SE_4_6



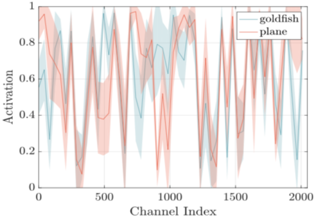
(f) SE_5_3

Figure 4: Activation of SE blocks at layers of different depth

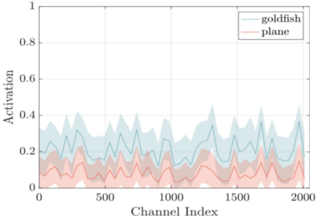
Assessing the Excitation operation



(a) SE_2_3



(d) SE_5_1



(f) SE_5_3

Figure 5: Mean/Std of instance activations of SE blocks at layers of different depth

Implementation

Official:

https://github.com/hujie-frank/SENet

TensorFlow:

https://github.com/taki0112/SENet-Tensorflow

PyTorch:

https://github.com/moskomule/senet.pytorch

Efficient Networks

Overview

Scope of Efficient Networks is to scale networks towards depth, width and resolution *concurrently* to improve performance.

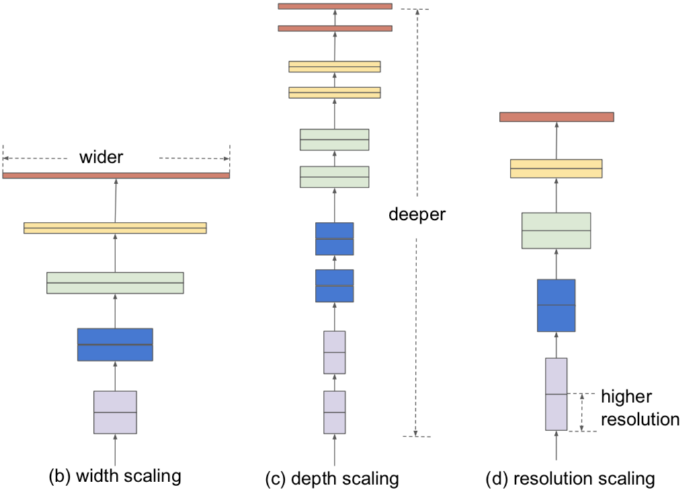


Figure 6: Different ways of model scaling

Network Scaling

Neither of these can grow arbitrarily large: Extremely deep networks do not improve performance, filters correlate and increase ratio of image resolution has its own threshold.

Question

Authors **empirically** observe that scaling in each direction in **not** independent from the reamining two directions. Thus, how can depth scaling d, width scaling w and resolution ratio r be tuned to maximize performance under specific memory and FLOPS constraints?

Efficient Net design

1. Grid search to define values baseline values (EB0 network) α for depth, β for width and γ for resolution.
2. Compound coefficient ϕ that scales all of α, β and γ concurrently to form larger versions (EB1 up to EB7) of the baseline network.

α, β and γ should fall into the constraint $\alpha * \beta^2 * \gamma^2 \approx 2$ since $\mathit{FLOPS} \approx d * w^2 * r^2$ and the author's target is to increase FLOPs by 2^ϕ when increasing ϕ.

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

Figure 7: Efficient Net B0

Training

Efficient Nets use:

- *Swish* activations
- *AutoAugment* policy
- Stochastic depth

Results: HeatMap activation

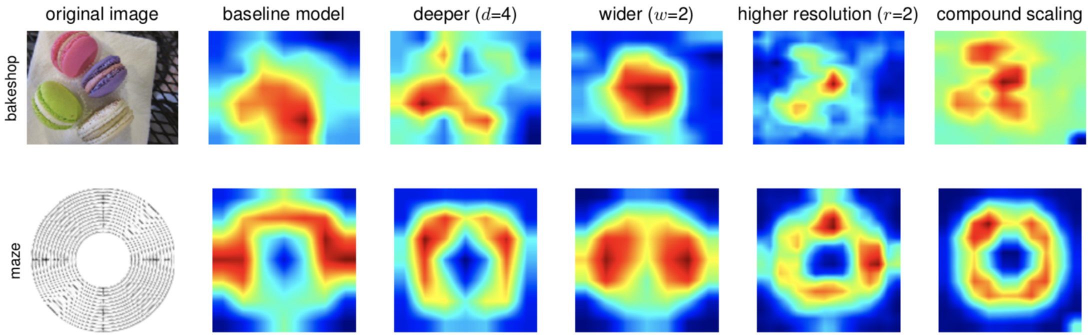


Figure 8: Heatmap activations of different model scalings

Results: ImageNet

Model	Top-1 Acc.	Top-5 Acc.	#Params	Ratio-to-EfficientNet	#FLOPS	Ratio-to-EfficientNet
EfficientNet-B0	**76.3%**	**93.2%**	**5.3M**	**1x**	**0.39B**	**1x**
ResNet-50 (He et al., 2016)	76.0%	93.0%	26M	4.9x	4.1B	11x
DenseNet-169 (Huang et al., 2017)	76.2%	93.2%	14M	2.6x	3.5B	8.9x
EfficientNet-B1	**78.8%**	**94.4%**	**7.8M**	**1x**	**0.70B**	**1x**
ResNet-152 (He et al., 2016)	77.8%	93.8%	60M	7.6x	11B	16x
DenseNet-264 (Huang et al., 2017)	77.9%	93.9%	34M	4.3x	6.0B	8.6x
Inception-v3 (Szegedy et al., 2016)	78.8%	94.4%	24M	3.0x	5.7B	8.1x
Xception (Chollet, 2017)	79.0%	94.5%	23M	3.0x	8.4B	12x
EfficientNet-B2	**79.8%**	**94.9%**	**9.2M**	**1x**	**1.0B**	**1x**
Inception-v4 (Szegedy et al., 2017)	80.0%	95.0%	48M	5.2x	13B	13x
Inception-resnet-v2 (Szegedy et al., 2017)	80.1%	95.1%	56M	6.1x	13B	13x
EfficientNet-B3	**81.1%**	**95.5%**	**12M**	**1x**	**1.8B**	**1x**
ResNeXt-101 (Xie et al., 2017)	80.9%	95.6%	84M	7.0x	32B	18x
PolyNet (Zhang et al., 2017)	81.3%	95.8%	92M	7.7x	35B	19x
EfficientNet-B4	**82.6%**	**96.3%**	**19M**	**1x**	**4.2B**	**1x**
SENet (Hu et al., 2018)	82.7%	96.2%	146M	7.7x	42B	10x
NASNet-A (Zoph et al., 2018)	82.7%	96.2%	89M	4.7x	24B	5.7x
AmoebaNet-A (Real et al., 2019)	82.8%	96.1%	87M	4.6x	23B	5.5x
PNASNet (Liu et al., 2018)	82.9%	96.2%	86M	4.5x	23B	6.0x
EfficientNet-B5	**83.3%**	**96.7%**	**30M**	**1x**	**9.9B**	**1x**
AmoebaNet-C (Cubuk et al., 2019)	83.5%	96.5%	155M	5.2x	41B	4.1x
EfficientNet-B6	**84.0%**	**96.9%**	**43M**	**1x**	**19B**	**1x**
EfficientNet-B7	**84.4%**	**97.1%**	**66M**	**1x**	**37B**	**1x**

Results: Transfer Learning of ImageNet PreTrained Networks

	Comparison to best public-available results						Comparison to best reported results					
	Model	Acc.	#Param	Our Model	Acc.	#Param(ratio)	Model	Acc.	#Param	Our Model	Acc.	#Param(ratio)
CIFAR-10	NASNet-A	98.0%	85M	EfficientNet-B0	98.1%	4M (21x)	†Gpipe	**99.0%**	556M	EfficientNet-B7	98.9%	64M (8.7x)
CIFAR-100	NASNet-A	87.5%	85M	EfficientNet-B0	88.1%	4M (21x)	Gpipe	91.3%	556M	EfficientNet-B7	**91.7%**	64M (8.7x)
Birdsnap	Inception-v4	81.8%	41M	EfficientNet-B5	82.0%	28M (1.5x)	GPipe	83.6%	556M	EfficientNet-B7	**84.3%**	64M (8.7x)
Stanford Cars	Inception-v4	93.4%	41M	EfficientNet-B3	93.6%	10M (4.1x)	‡DAT	**94.8%**	-	EfficientNet-B7	94.7%	-
Flowers	Inception-v4	98.5%	41M	EfficientNet-B5	98.5%	28M (1.5x)	DAT	97.7%	-	EfficientNet-B7	**98.8%**	-
FGVC Aircraft	Inception-v4	90.9%	41M	EfficientNet-B3	90.7%	10M (4.1x)	DAT	92.9%	-	EfficientNet-B7	**92.9%**	-
Oxford-IIIT Pets	ResNet-152	94.5%	58M	EfficientNet-B4	94.8%	17M (5.6x)	GPipe	**95.9%**	556M	EfficientNet-B6	95.4%	41M (14x)
Food-101	Inception-v4	90.8%	41M	EfficientNet-B4	91.5%	17M (2.4x)	GPipe	93.0%	556M	EfficientNet-B7	**93.0%**	64M (8.7x)
Geo-Mean						**(4.7x)**						**(9.6x)**

Figure 10: Heatmap activations of different model scalings

Implementation

Official:

https://github.com/tensorflow/tpu/tree/master/models/official/efficientnet

PyTorch:

https://github.com/lukemelas/EfficientNet-PyTorch

Thank you!