Attention Augmented Convolutional Networks (ICCV 2019)

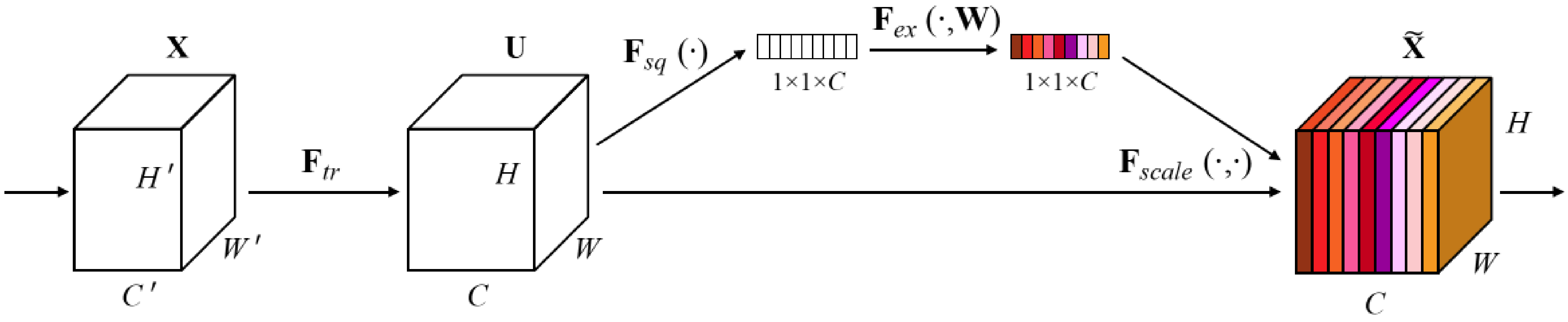
Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, Quoc V. Le

Google Brain

https://arxiv.org/abs/1904.09925

Squeeze-and-Excitation Network (CVPR2018)

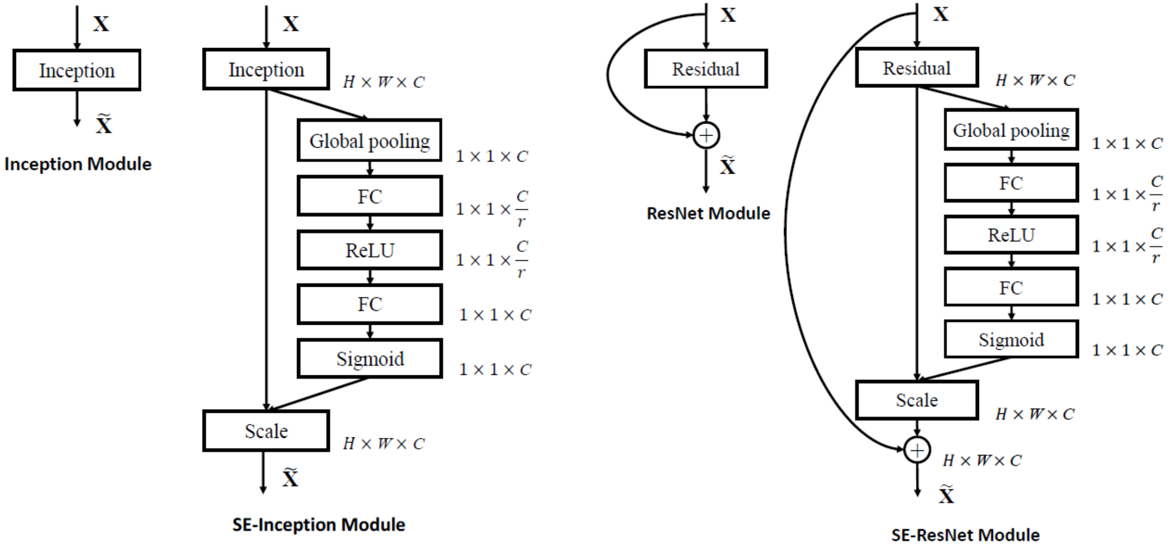
"Our goal is to improve the representational power of a network by explicitly modelling the interdependencies ***between the channels*** *of its convolutional features."*



https://arxiv.org/abs/1709.01507

Squeeze(압축) : Global Information Embedding

$$z_c = \mathbf{F}_{sq}(\mathbf{u}_c) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} u_c(i, j).$$



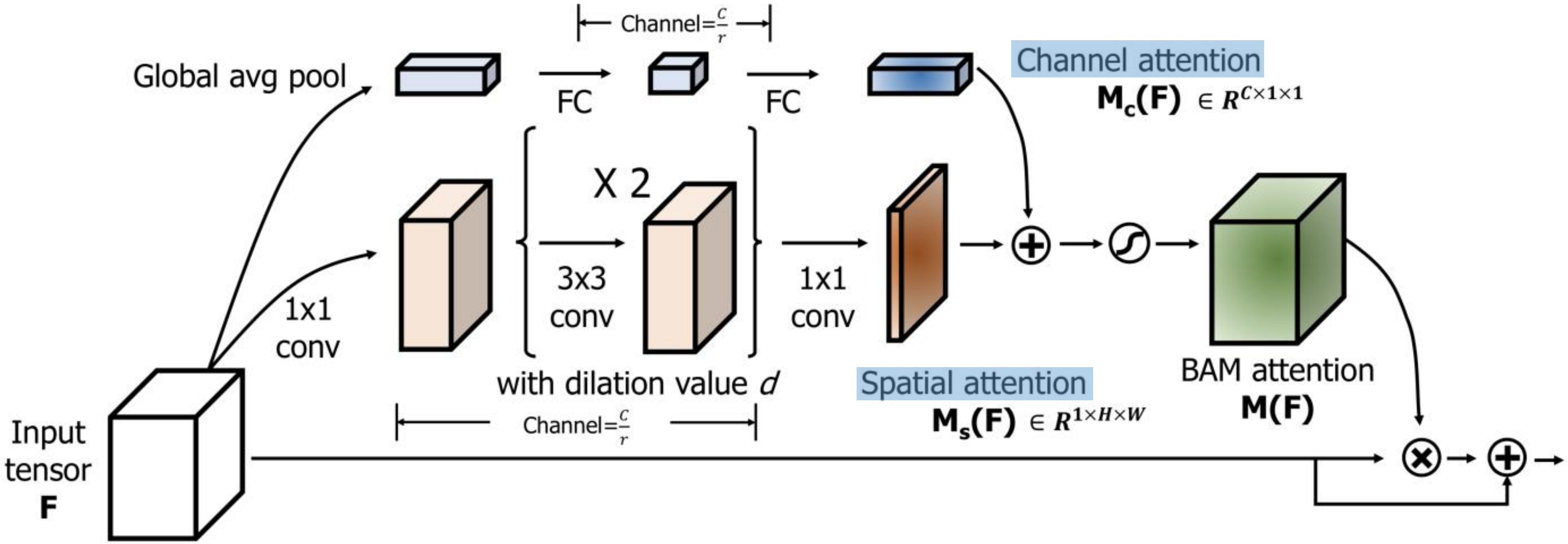
Excitation(재조정) : Adaptive Recalibration

$$\mathbf{s} = \mathbf{F}_{ex}(\mathbf{z}, \mathbf{W}) = \sigma(g(\mathbf{z}, \mathbf{W})) = \sigma(\mathbf{W}_2 \delta(\mathbf{W}_1 \mathbf{z})),$$

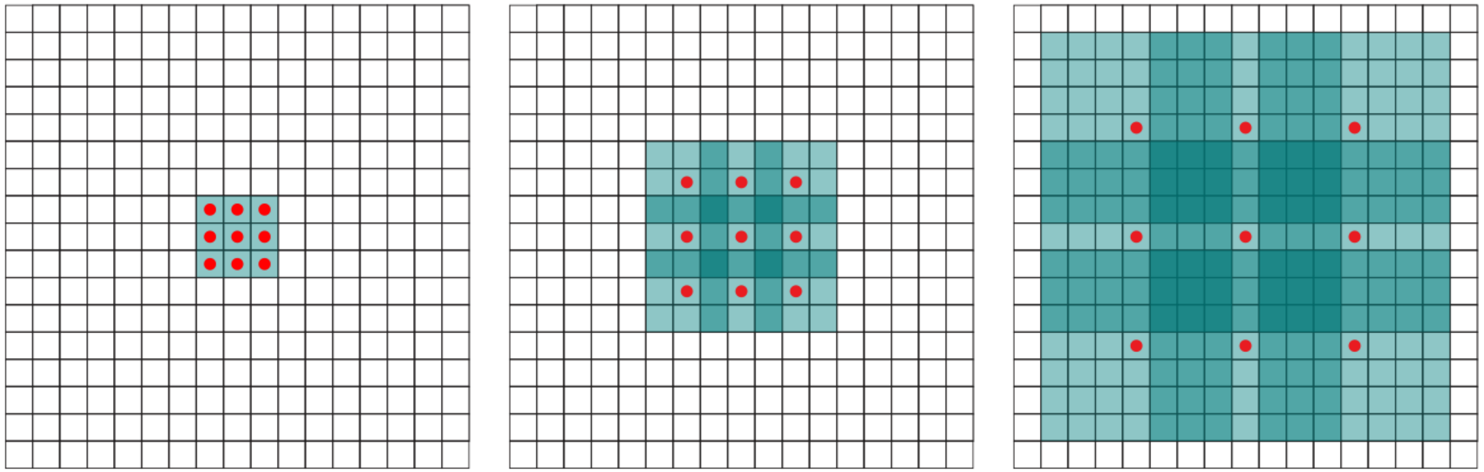
```python
from torch import nn


class SELayer(nn.Module):
    def __init__(self, channel, reduction=16):
        super(SELayer, self).__init__()
        self.avg_pool = nn.AdaptiveAvgPool2d(1)
        self.fc = nn.Sequential(
            nn.Linear(channel, channel // reduction, bias=False),
            nn.ReLU(inplace=True),
            nn.Linear(channel // reduction, channel, bias=False),
            nn.Sigmoid()
        )

    def forward(self, x):
        b, c, _, _ = x.size()
        y = self.avg_pool(x).view(b, c)
        y = self.fc(y).view(b, c, 1, 1)
        return x * y.expand_as(x)
```

	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

Bottleneck Attention Module (ECCV2018)



https://arxiv.org/abs/1807.06514



(a) (b) (c)

Figure 1: Systematic dilation supports exponential expansion of the receptive field without loss of resolution or coverage. (a) F_1 is produced from F_0 by a 1-dilated convolution; each element in F_1 has a receptive field of 3×3. (b) F_2 is produced from F_1 by a 2-dilated convolution; each element in F_2 has a receptive field of 7×7. (c) F_3 is produced from F_2 by a 4-dilated convolution; each element in F_3 has a receptive field of 15×15. The number of parameters associated with each layer is identical. The receptive field grows exponentially while the number of parameters grows linearly.

Architecture	Params	GFLOPs	Error
ResNet50[15]	23.71M	1.22	21.49
ResNet50[15] + BAM-C	28.98M	1.37	20.88
ResNet50[15] + BAM	**24.07M**	**1.25**	**20.00**
PreResNet110[16]	1.73M	0.245	22.22
PreResNet110[16] + BAM-C	**2.17M**	**0.275**	**21.29**
PreResNet110[16] + BAM	1.73M	0.246	21.96
WideResNet28 (w=8)[47]	23.40M	3.36	20.40
WideResNet28 (w=8)[47] + BAM-C	23.78M	3.39	20.06
WideResNet28 (w=8)[47] + BAM	**23.42M**	**3.37**	**19.06**
ResNext29 8x64d[43]	34.52M	4.99	18.18
ResNext29 8x64d[43] + BAM-C	35.60M	5.07	18.15
ResNext29 8x64d[43] + BAM	**34.61M**	**5.00**	**16.71**

CBAM: Convolutional Block Attention Module (2018)

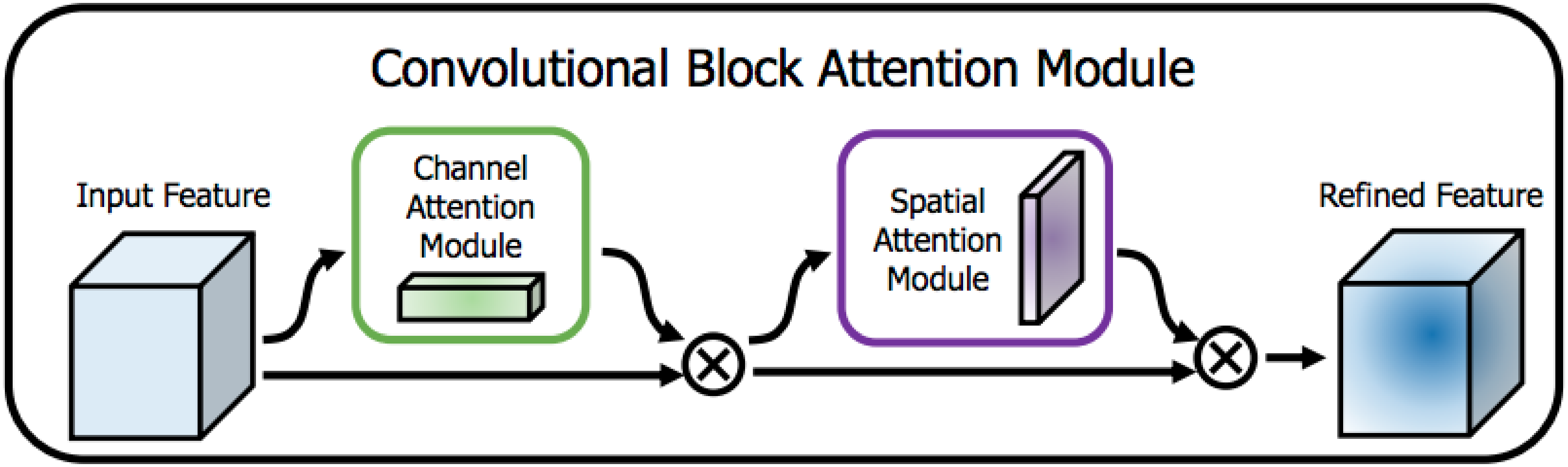
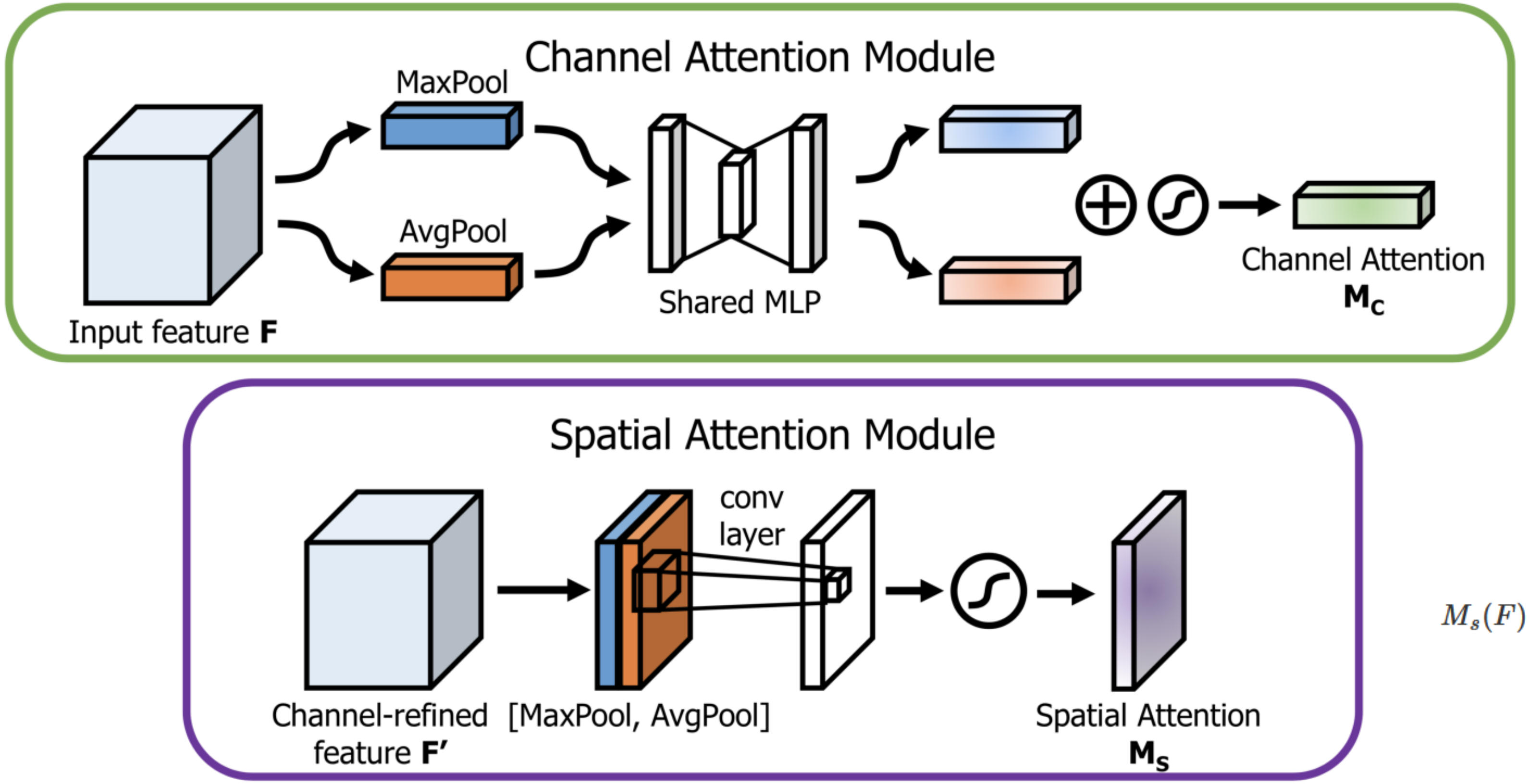


Fig. 1: **The overview of CBAM.** The module has two sequential sub-modules: *channel* and *spatial*. The intermediate feature map is adaptively refined through our module (CBAM) at every convolutional block of deep networks.

https://arxiv.org/abs/1807.06521

CBAM: Convolutional Block Attention Module (2018)



$$M_c(F) = \sigma(MLP(AvgPool(F)) + MLP(MaxPool(F))) \\ = \sigma(\mathbf{W_1}(\mathbf{W_0}(F^C_{avg})) + \mathbf{W_1}(\mathbf{W_0}(F^C_{max})))$$

$$\mathbf{W_0} \in R^{C/r \times C}, \mathbf{W_1} \in R^{C \times C/r}$$

$$M_s(F) = \sigma(f^{7\times7}([AvgPool(F); MaxPool(F)])) = \sigma(f^{7\times7}(F^S_{avg}; F^S_{max}))$$

https://arxiv.org/abs/1807.06521

Architecture	Param.	GFLOPs	Top-1 Error (%)	Top-5 Error (%)
ResNet18 [5]	11.69M	1.814	29.60	10.55
ResNet18 [5] + SE [28]	11.78M	1.814	29.41	10.22
ResNet18 [5] + CBAM	11.78M	1.815	**29.27**	**10.09**
ResNet34 [5]	21.80M	3.664	26.69	8.60
ResNet34 [5] + SE [28]	21.96M	3.664	26.13	8.35
ResNet34 [5] + CBAM	21.96M	3.665	**25.99**	**8.24**
ResNet50 [5]	25.56M	3.858	24.56	7.50
ResNet50 [5] + SE [28]	28.09M	3.860	23.14	6.70
ResNet50 [5] + CBAM	28.09M	3.864	**22.66**	**6.31**
ResNet101 [5]	44.55M	7.570	23.38	6.88
ResNet101 [5] + SE [28]	49.33M	7.575	22.35	6.19
ResNet101 [5] + CBAM	49.33M	7.581	**21.51**	**5.69**
WideResNet18 [6] (widen=1.5)	25.88M	3.866	26.85	8.88
WideResNet18 [6] (widen=1.5) + SE [28]	26.07M	3.867	26.21	8.47
WideResNet18 [6] (widen=1.5) + CBAM	26.08M	3.868	**26.10**	**8.43**
WideResNet18 [6] (widen=2.0)	45.62M	6.696	25.63	8.20
WideResNet18 [6] (widen=2.0) + SE [28]	45.97M	6.696	24.93	7.65
WideResNet18 [6] (widen=2.0) + CBAM	45.97M	6.697	**24.84**	**7.63**
ResNeXt50 [7] (32x4d)	25.03M	3.768	22.85	6.48
ResNeXt50 [7] (32x4d) + SE [28]	27.56M	3.771	**21.91**	6.04
ResNeXt50 [7] (32x4d) + CBAM	27.56M	3.774	21.92	**5.91**
ResNeXt101 [7] (32x4d)	44.18M	7.508	21.54	5.75
ResNeXt101 [7] (32x4d) + SE [28]	48.96M	7.512	21.17	5.66
ResNeXt101 [7] (32x4d) + CBAM	48.96M	7.519	**21.07**	**5.59**

* all results are reproduced in the PyTorch framework.

Table 4: **Classification results on ImageNet-1K.** Single-crop validation errors are reported.

Non-local Neural Networks (CVPR 2018)

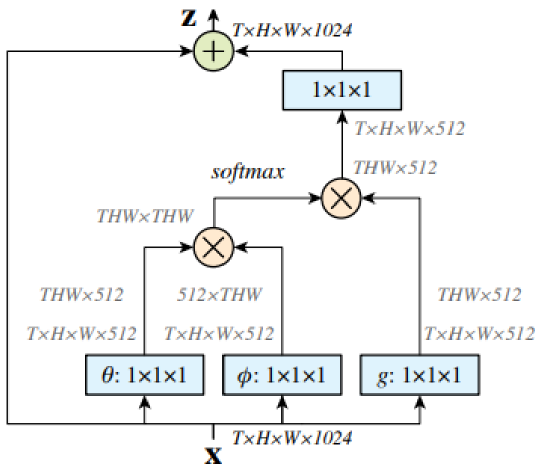


Figure 2. A spacetime **non-local block**. The feature maps are shown as the shape of their tensors, *e.g.*, $T\times H\times W\times 1024$ for 1024 channels (proper reshaping is performed when noted). "$\otimes$" denotes matrix multiplication, and "$\oplus$" denotes element-wise sum. The softmax operation is performed on each row. The blue boxes denote $1\times1\times1$ convolutions. Here we show the embedded Gaussian version, with a bottleneck of 512 channels. The vanilla Gaussian version can be done by removing θ and ϕ, and the dot-product version can be done by replacing softmax with scaling by $1/N$.

Gaussian. Following the non-local mean [4] and bilateral filters [47], a natural choice of f is the Gaussian function. In this paper we consider:

$$f(\mathbf{x}_i, \mathbf{x}_j) = e^{\mathbf{x}_i^T \mathbf{x}_j}. \tag{2}$$

Embedded Gaussian. A simple extension of the Gaussian function is to compute similarity in an embedding space. In this paper we consider:

$$f(\mathbf{x}_i, \mathbf{x}_j) = e^{\theta(\mathbf{x}_i)^T \phi(\mathbf{x}_j)}. \tag{3}$$

Here $\theta(\mathbf{x}_i) = W_\theta \mathbf{x}_i$ and $\phi(\mathbf{x}_j) = W_\phi \mathbf{x}_j$ are two embeddings. As above, we set $\mathcal{C}(\mathbf{x}) = \sum_{\forall j} f(\mathbf{x}_i, \mathbf{x}_j)$.

$$\mathbf{y} = softmax(\mathbf{x}^T W_\theta^T W_\phi \mathbf{x}) g(\mathbf{x})$$

https://arxiv.org/abs/1711.07971

Non-local Neural Networks (CVPR 2018)

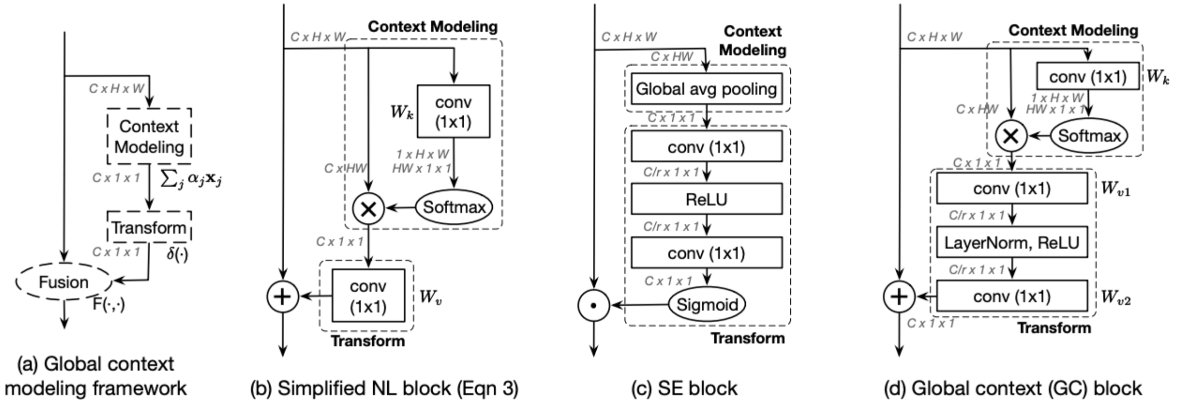
Figure 3. Examples of the behavior of a non-local block in res_3 computed by a 5-block non-local model trained on Kinetics. These examples are from held-out validation videos. The starting point of arrows represents one $\mathbf{x}_i$, and the ending points represent $\mathbf{x}_j$. The 20 highest weighted arrows for each $\mathbf{x}_i$ are visualized. The 4 frames are from a 32-frame input, shown with a stride of 8 frames. These visualizations show how the model finds related clues to support its prediction.

Non-local Neural Networks (CVPR 2018)

model	backbone	modality	top-1	top-5
I3D in [7]	Inception	RGB	71.1†	89.3†
2-Stream I3D in [7]	Inception	RGB + flow	74.2†	91.3†
RGB baseline in [3]	Inception-ResNet-v2	RGB	73.0	90.9
3-stream late fusion [3]	Inception-ResNet-v2	RGB + flow + audio	74.9	91.6
3-stream LSTM [3]	Inception-ResNet-v2	RGB + flow + audio	77.1	93.2
3-stream SATT [3]	Inception-ResNet-v2	RGB + flow + audio	77.7	93.2
NL I3D [ours]	ResNet-50	RGB	76.5	92.6
	ResNet-101	RGB	**77.7**	**93.3**

Table 3. Comparisons with state-of-the-art results in **Kinetics**. Numbers with † were reported on the test set; otherwise on the validation set. We include the Kinetics 2017 competition winner's results [3], but their best results exploited audio signals (marked in gray) so were not vision-only solutions. †: individual top-1 or top-5 numbers are not available from the test server at the time of submitting this manuscript.

GCNet: Non-local Networks Meet Squeeze-Excitation Networks and Beyond (ICCV 2019)



https://arxiv.org/abs/1904.11492

Attention Augmented Convolutional Networks

Irwan Bello Barret Zoph Ashish Vaswani Jonathon Shlens
Quoc V. Le
Google Brain
{ibello,barretzoph,avaswani,shlens,qvl}@google.com

We introduce a novel two-dimensional relative self-attention mechanism that proves competitive in replacing convolutions as a stand-alone computational primitive for image classification. We find in control experiments that the best results are obtained when combining both convolutions and self-attention. We therefore propose to augment convolutional operators with this self-attention mechanism by concatenating convolutional feature maps with a set of feature maps produced via self-attention

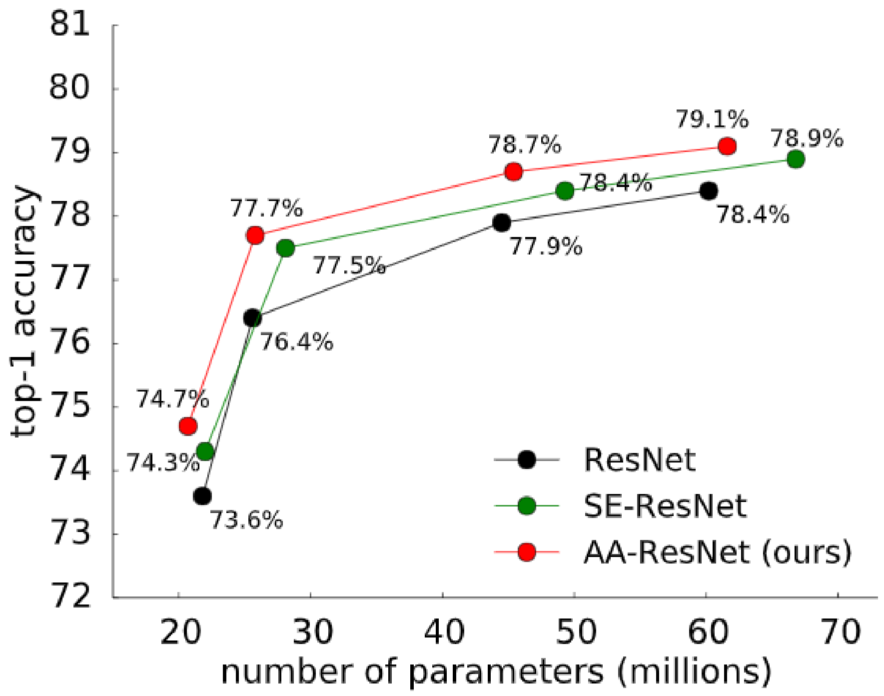


Figure 1. **Attention Augmentation systematically improves image classification across a large variety of networks of different scales.** ImageNet classification accuracy [9] versus the number of parameters for baseline models (ResNet) [14], models augmented with channel-wise attention (SE-ResNet) [17] and our proposed architecture (AA-ResNet).

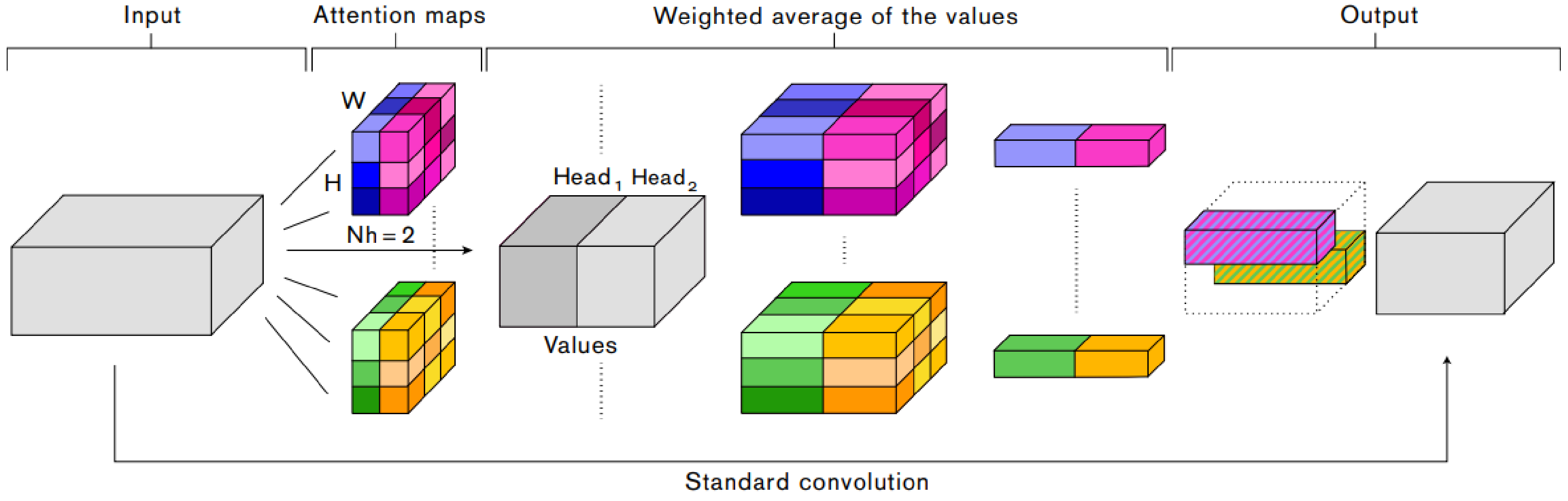


Figure 2. **Attention-augmented convolution**: For each spatial location (h, w), N_h attention maps over the image are computed from queries and keys. These attention maps are used to compute N_h weighted averages of the values V. The results are then concatenated, reshaped to match the original volume's spatial dimensions and mixed with a pointwise convolution. Multi-head attention is applied in parallel to a standard convolution operation and the outputs are concatenated.

$$O_h = \mathsf{Softmax}\left(\frac{(XW_q)(XW_k)^T}{\sqrt{d_k^h}}\right)(XW_v) \quad (1)$$

where $W_q, W_k \in \mathbb{R}^{F_{in} \times d_k^h}$ and $W_v \in \mathbb{R}^{F_{in} \times d_v^h}$ are learned linear transformations that map the input X to queries $Q = XW_q$, keys $K = XW_k$ and values $V = XW_v$. The outputs of all heads are then concatenated and projected again as follows:

$$\mathsf{MHA}(X) = \mathsf{Concat}\Big[O_1, \ldots, O_{Nh}\Big]W^O \quad (2)$$

where $W^O \in \mathbb{R}^{d_v \times d_v}$ is a learned linear transformation. $\mathsf{MHA}(X)$ is then reshaped into a tensor of shape (H, W, d_v) to match the original spatial dimensions. We note that multi-head attention incurs a complexity of $O((HW)^2 d_k)$ and a memory cost of $O((HW)^2 N_h)$ as it requires to store attention maps for each head.

Relative positional encodings

- two-dimensional relative self-attention
- relative height, relative width 정보 추가

for how much pixel $i = (i_x, i_y)$ attends to pixel $j = (j_x, j_y)$ is computed as:

$$l_{i,j} = \frac{q_i^T}{\sqrt{d_k^h}}(k_j + r^W_{j_x - i_x} + r^H_{j_y - i_y}) \tag{3}$$

where q_i is the query vector for pixel i (the i-th row of Q), k_j is the key vector for pixel j (the j-th row of K) and $r^W_{j_x - i_x}$ and $r^H_{j_y - i_y}$ are learned embeddings for relative width $j_x - i_x$ and relative height $j_y - i_y$, respectively. The output of head h now becomes:

$$O_h = \text{Softmax}\left(\frac{QK^T + S^{rel}_H + S^{rel}_W}{\sqrt{d_k^h}}\right)V \tag{4}$$

where $S^{rel}_H, S^{rel}_W \in \mathbb{R}^{HW \times HW}$ are matrices of relative position logits along height and width dimensions that satisfy $S^{rel}_H[i,j] = q_i^T r^H_{j_y - i_y}$ and $S^{rel}_W[i,j] = q_i^T r^W_{j_x - i_x}$.

Attention Augmented Convolution

$$\text{AAConv}(X) = \text{Concat}\big[\text{Conv}(X), \text{MHA}(X)\big].$$

```python
class AugmentedConv(nn.Module):
    def __init__(self, in_channels, out_channels, kernel_size,dk, dv, Nh, shape=0, relative=False, stride=1, padding=1):
        super(AugmentedConv, self).__init__()
        self.in_channels = in_channels
        self.out_channels = out_channels
        self.kernel_size = kernel_size
        self.dk = dk
        self.dv = dv
        self.Nh = Nh
        self.shape = shape
        self.relative = relative
        self.stride = stride
        self.padding = (self.kernel_size - 1) // 2

        assert self.Nh != 0, "integer division or modulo by zero, Nh >= 1"
        assert self.dk % self.Nh == 0, "dk should be divided by Nh. (example: out_channels: 20, dk: 40, Nh: 4)"
        assert self.dv % self.Nh == 0, "dv should be divided by Nh. (example: out_channels: 20, dv: 4, Nh: 4)"
        assert stride in [1, 2], str(stride) + " Up to 2 strides are allowed."

        self.conv_out = nn.Conv2d(self.in_channels, self.out_channels - self.dv, self.kernel_size, stride=stride, padding=self.padding)

        self.qkv_conv = nn.Conv2d(self.in_channels, 2 * self.dk + self.dv, kernel_size=self.kernel_size, stride=stride, padding=self.padding)

        self.attn_out = nn.Conv2d(self.dv, self.dv, kernel_size=1, stride=1)

        if self.relative:
            self.key_rel_w = nn.Parameter(torch.randn((2 * self.shape - 1, dk // Nh), requires_grad=True))
            self.key_rel_h = nn.Parameter(torch.randn((2 * self.shape - 1, dk // Nh), requires_grad=True))
```

Architecture	Params (M)	Δ_{Infer}	Δ_{Train}	top-1
ResNet-50	25.6	-	-	76.4
SE [17]	28.1	+12%	+92%	77.5 (77.0)
BAM [31]	25.9	+19%	+43%	77.3
CBAM [46]	28.1	+56%	+132%	77.4 (77.4)
GALA [28]	29.4	+86%	+133%	77.5 (77.3)
AA ($v = 0.25$)	**24.3**	**+29%**	**+25%**	**77.7**

Table 2. Image classification performance of different attention mechanisms on the ImageNet dataset. Δ refers to the increase in latency times compared to the ResNet50 on a single Tesla V100 GPU with Tensorflow using a batch size of 128. For fair comparison, we also include top-1 results (in parentheses) when scaling networks in width to match $\sim$ 25.6M parameters as the ResNet50 baseline.

Architecture	GFlops	Params	top-1	top-5
ResNet-34 [14]	7.4	21.8M	73.6	91.5
SE-ResNet-34 [17]	7.4	22.0M	74.3	91.8
AA-ResNet-34 (ours)	7.1	20.7M	74.7	92.0
ResNet-50 [14]	8.2	25.6M	76.4	93.1
SE-ResNet-50 [17]	8.2	28.1M	77.5	93.7
AA-ResNet-50 (ours)	8.3	25.8M	77.7	93.8
ResNet-101 [14]	15.6	44.5M	77.9	94.0
SE-ResNet-101 [17]	15.6	49.3M	78.4	94.2
AA-ResNet-101 (ours)	16.1	45.4M	78.7	94.4
ResNet-152 [14]	23.0	60.2M	78.4	94.2
SE-ResNet-152 [17]	23.1	66.8M	78.9	94.5
AA-ResNet-152 (ours)	23.8	61.6M	79.1	94.6

Table 3. Image classification on the ImageNet dataset [9] across a range of ResNet architectures: ResNet-34, ResNet-50, Resnet-101, and ResNet-152 [14, 47, 13].

Backbone architecture	GFlops	Params	$\mathrm{mAP}_{\mathrm{COCO}}$	mAP_{50}	mAP_{75}
ResNet-50 [26]	182	33.4M	36.8	54.5	39.5
SE-ResNet-50 [17]	183	35.9M	36.5	54.0	39.1
AA-ResNet-50 (ours)	182	33.1M	38.2	56.5	40.7
ResNet-101 [26]	243	52.4M	38.5	56.4	41.2
SE-ResNet-101 [17]	243	57.2M	37.4	55.0	39.9
AA-ResNet-101 (ours)	245	51.7M	39.2	57.8	41.9

Table 5. Object detection on the COCO dataset [27] using the RetinaNet architecture [26] with different backbone architectures. We report mean Average Precision at three different IoU values.