📖

[논문 리뷰] CBAM [2018]

Abstract & Introduction

🌟해당 논문은 Feed-forward 방식인 simple하고, effective한, attention module을 다룹니다.

(구현 부분은 하단 링크를 참고하시기 바랍니다)

Channel, Spatial로 구성되어 있으며, 각각의 Attention Module은 attention map을 생성합니다. 생성한 attention map은 input feature map을 곱하여, 필요없는 정보는 억제하고, 중요한 정보만을 강조합니다.

CBAM(Convolutional Block Attention Module)은 여러 Dataset에서 검증되었으며, 적은 연산량을 가지고 있으며, 여러 CNN Model에 유용하게 활용하는 것이 가능합니다.

CNN의 performance를 향상시키기 위해, 최근 연구들은 **Depth, Width, cardinality**에 많이 investigated 해왔습니다. **(층, 필터, Group을 의미합니다.)**

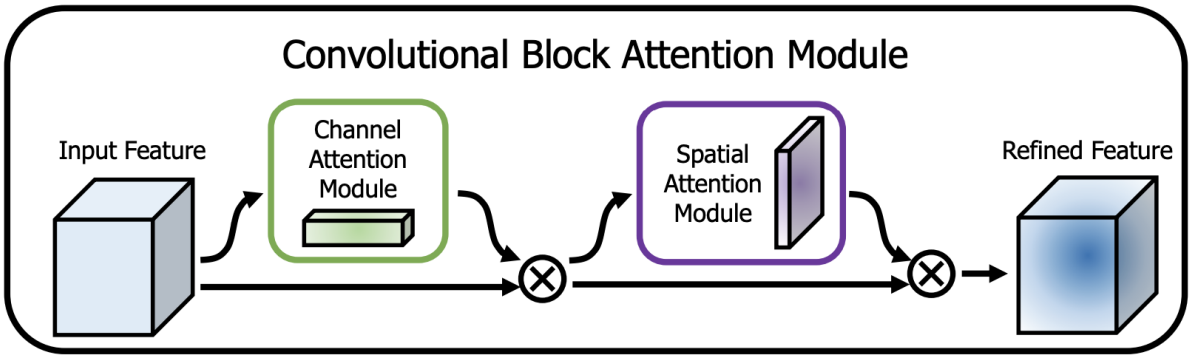


Fig. 1: **The overview of CBAM**. The module has two sequential sub-modules: *channel* and *spatial*. The intermediate feature map is adaptively refined through our module (CBAM) at every convolutional block of deep networks.

🌟 Attention의 목표는 **중요한 Feature map은 활용하고, supressing한 Feature map은 사용하지 않는 것에 초점을 둡니다.**

Convolutional Block Attention Module

Channel Attention Module은 **input feature에서 1D channel attention map을 생성**합니다.
생성한 1D channel attention map에 input feature을 곱하여, F`를 생성합니다.

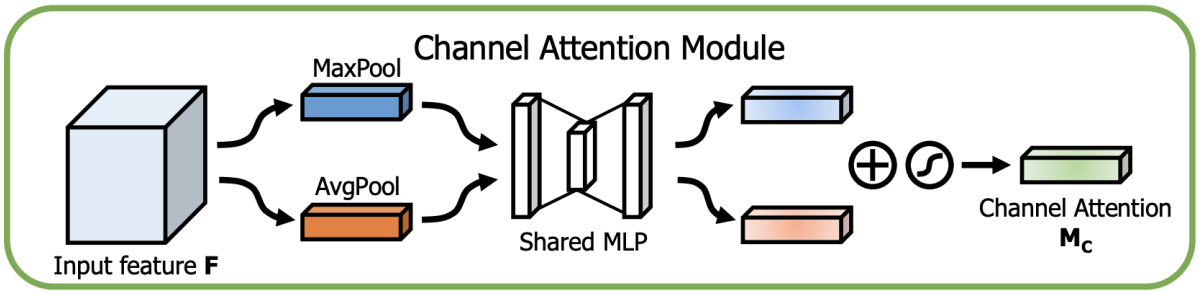
Spatial Attention Module은 F`를 입력 받아 2D spatial attention map을 생성합니다.
생성한 2D spatial attention map에 F를 곱하여, F``를 생성합니다.

Given an intermediate feature map $\mathbf{F} \in \mathbb{R}^{C \times H \times W}$ as input, CBAM sequentially infers a 1D channel attention map $\mathbf{M_c} \in \mathbb{R}^{C \times 1 \times 1}$ and a 2D spatial attention map $\mathbf{M_s} \in \mathbb{R}^{1 \times H \times W}$ as illustrated in Fig. 1. The overall attention process can be summarized as:

$$\begin{aligned} \mathbf{F}' &= \mathbf{M_c}(\mathbf{F}) \otimes \mathbf{F}, \\ \mathbf{F}'' &= \mathbf{M_s}(\mathbf{F}') \otimes \mathbf{F}', \end{aligned} \tag{1}$$

where $\otimes$ denotes element-wise multiplication. During multiplication, the attention values are broadcasted (copied) accordingly: channel attention values are broadcasted along the spatial dimension, and vice versa. $\mathbf{F}''$ is the final refined output. Fig. 2 depicts the computation process of each attention map. The following describes the details of each attention module.

🌟 **최종적으로 F는 Attention이 적용되어, 중요한 Feature Map은 강조되고, Noise는 억제된 Feature map이 됩니다.**

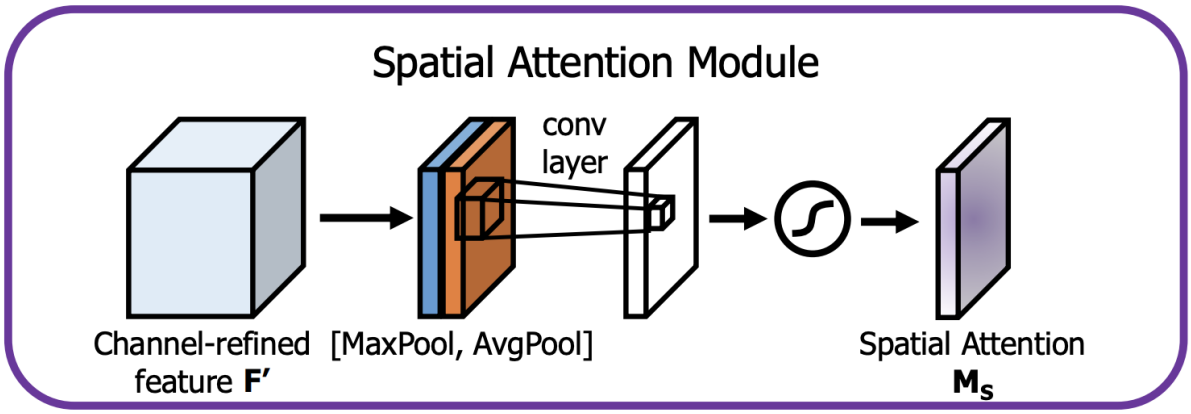


Channel Attention Module

Feature 내의 내부 channel 관계를 활용함으로서, channel attention map을 생성합니다.
효율적인 계산을 위하여, Spatial Dimenstion of input feature map을 squeez하도록 합니다.
Squeeze정보를 통합하기 위해, Average pooling 과 Max Pooling을 적용합니다. (동시 사용 성능향상)

$$\begin{aligned}\mathbf{M_c}(\mathbf{F}) &= \sigma(MLP(AvgPool(\mathbf{F})) + MLP(MaxPool(\mathbf{F}))) \\ &= \sigma(\mathbf{W_1}(\mathbf{W_0}(\mathbf{F^c_{avg}})) + \mathbf{W_1}(\mathbf{W_0}(\mathbf{F^c_{max}}))),\end{aligned} \tag{2}$$

where σ denotes the sigmoid function, $\mathbf{W_0} \in \mathbb{R}^{C/r \times C}$, and $\mathbf{W_1} \in \mathbb{R}^{C \times C/r}$. Note that the MLP weights, $\mathbf{W_0}$ and $\mathbf{W_1}$, are shared for both inputs and the ReLU activation function is followed by $\mathbf{W_0}$.



Spatial Attention Module

Feature 내의 내부 channel 관계를 활용함으로서, spatial attention map을 생성합니다.

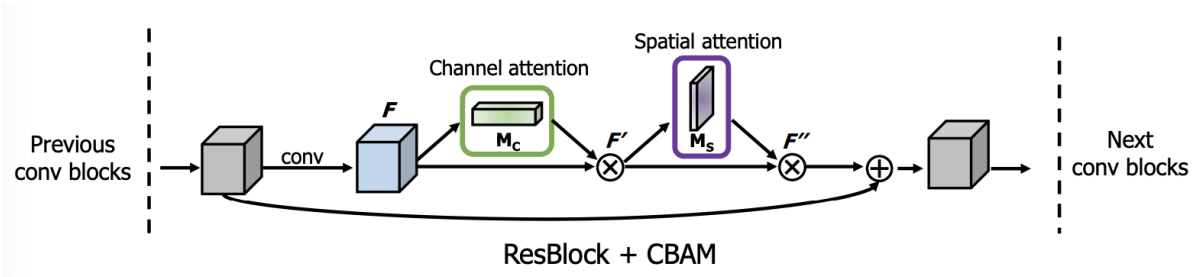
Different from the channel attention, Spatial Attention은 'Where'에 정보가 있는지 초점을 둡니다.

Average Pool 과 Max Pool을 적용한 후, DenseNet처럼 concatentate을 활용합니다.

$$\begin{aligned}\mathbf{M_s}(\mathbf{F}) &= \sigma(f^{7\times7}([AvgPool(\mathbf{F}); MaxPool(\mathbf{F})])) \\ &= \sigma(f^{7\times7}([\mathbf{F^s_{avg}}; \mathbf{F^s_{max}}])),\end{aligned} \tag{3}$$

where σ denotes the sigmoid function and $f^{7\times7}$ represents a convolution operation with the filter size of 7×7.

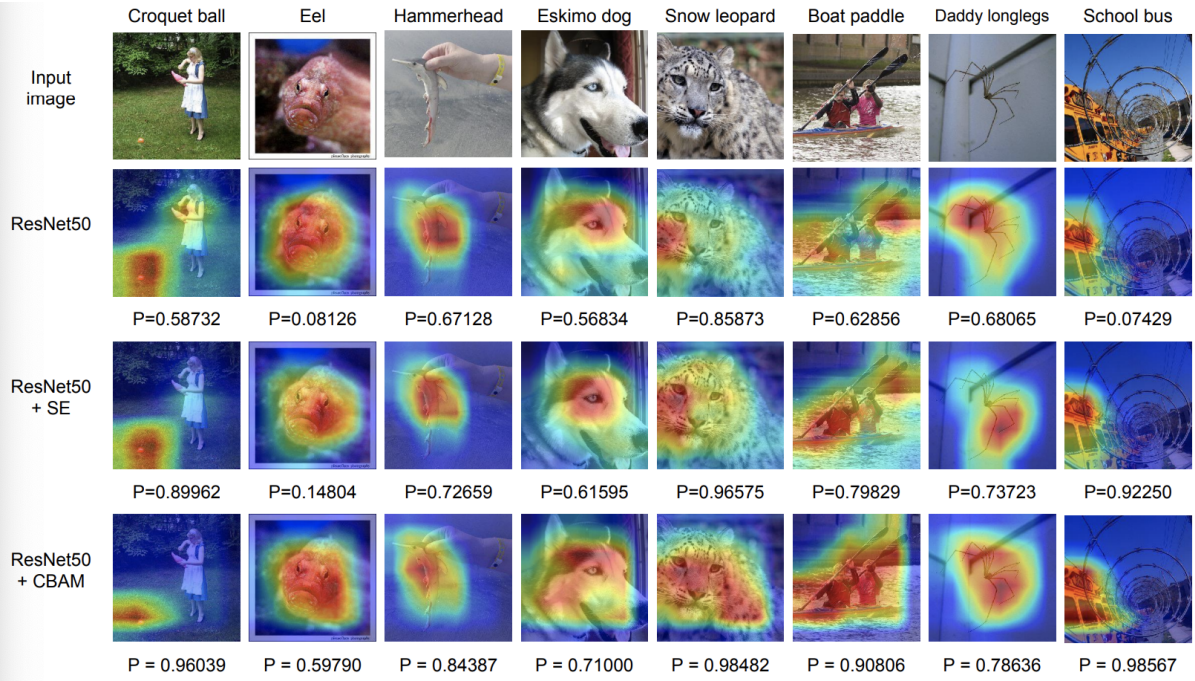
EX) ResBlock + CBAM



Experiments

- **CBAM(Convolutional Bottleneck attention module)은 CNN Network를 improve**하였습니다.
- **CBAM(Convolutional Bottleneck attention module)은 즉각적으로 효율적인 Feature Map들 사용합니다. (Using Spatial, Channels Attention Map)**

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**



CBAM: Convolutional Block Attention Module

We propose Convolutional Block Attention Module (CBAM), a simple yet effective attention module for feed-forward convolutional neural networks. Given an intermediate feature

https://arxiv.org/abs/1807.06521

https://bo-10000.tistory.com/126