# CBAM: Convolutional Block Attention Module

Sanghyun Woo*[1], Jongchan Park*†[2], Joon-Young Lee[3], and In So Kweon[1]

[1] Korea Advanced Institute of Science and Technology, Daejeon, Korea
{shwoo93, iskweon77}@kaist.ac.kr
[2] Lunit Inc., Seoul, Korea
jcpark@lunit.io
[3] Adobe Research, San Jose, CA, USA
jolee@adobe.com

**Abstract.** We propose Convolutional Block Attention Module (CBAM), a simple yet effective attention module for feed-forward convolutional neural networks. Given an intermediate feature map, our module sequentially infers attention maps along two separate dimensions, channel and spatial, then the attention maps are multiplied to the input feature map for adaptive feature refinement. Because CBAM is a lightweight and general module, it can be integrated into any CNN architectures seamlessly with negligible overheads and is end-to-end trainable along with base CNNs. We validate our CBAM through extensive experiments on ImageNet-1K, MS COCO detection, and VOC 2007 detection datasets. Our experiments show consistent improvements in classification and detection performances with various models, demonstrating the wide applicability of CBAM. The code and models will be publicly available.

**Keywords:** Object recognition, attention mechanism, gated convolution

## 1 Introduction

Convolutional neural networks (CNNs) have significantly pushed the performance of vision tasks [1–3] based on their rich representation power. To enhance performance of CNNs, recent researches have mainly investigated three important factors of networks: *depth*, *width*, and *cardinality*.

From the LeNet architecture [4] to Residual-style Networks [5–8] so far, the network has become deeper for rich representation. VGGNet [9] shows that stacking blocks with the same shape gives fair results. Following the same spirit, ResNet [5] stacks the same topology of residual blocks along with skip connection to build an extremely deep architecture. GoogLeNet [10] shows that width is another important factor to improve the performance of a model. Zagoruyko and Komodakis [6] propose to increase the width of a network based on the ResNet architecture. They have shown that a 28-layer ResNet with increased

*Both authors have equally contributed.
†The work was done while the author was at KAIST.

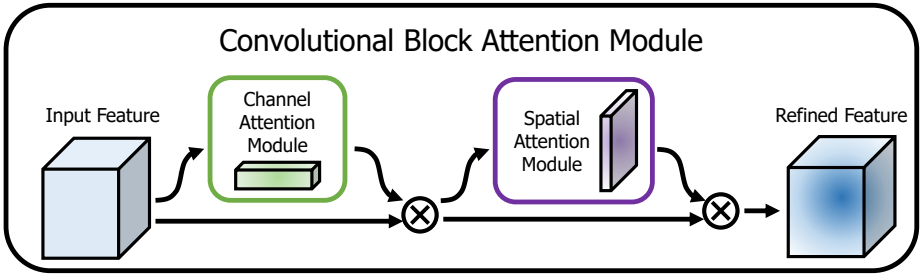



Fig. 1: **The overview of CBAM**. The module has two sequential sub-modules: *channel* and *spatial*. The intermediate feature map is adaptively refined through our module (CBAM) at every convolutional block of deep networks.

width can outperform an extremely deep ResNet with 1001 layers on the CIFAR benchmarks. Xception [11] and ResNeXt [7] come up with to increase the cardinality of a network. They empirically show that cardinality not only saves the total number of parameters but also results in stronger representation power than the other two factors: depth and width.

Apart from these factors, we investigate a different aspect of the architecture design, *attention*. The significance of attention has been studied extensively in the previous literature [12–17]. Attention not only tells where to focus, it also improves the representation of interests. Our goal is to increase representation power by using attention mechanism: focusing on important features and suppressing unnecessary ones. In this paper, we propose a new network module, named "Convolutional Block Attention Module". Since convolution operations extract informative features by blending cross-channel and spatial information together, we adopt our module to emphasize meaningful features along those two principal dimensions: channel and spatial axes. To achieve this, we sequentially apply channel and spatial attention modules (as shown in Fig. 1), so that each of the branches can learn 'what' and 'where' to attend in the channel and spatial axes respectively. As a result, our module efficiently helps the information flow within the network by learning which information to emphasize or suppress.

In the ImageNet-1K dataset, we obtain accuracy improvement from various baseline networks by plugging our tiny module, revealing the efficacy of CBAM. We visualize trained models using the grad-CAM [18] and observe that CBAM-enhanced networks focus on target objects more properly than their baseline networks. Taking this into account, we conjecture that the performance boost comes from accurate attention and noise reduction of irrelevant clutters. Finally, we validate performance improvement of object detection on the MS COCO and the VOC 2007 datasets, demonstrating a wide applicability of CBAM. Since we have carefully designed our module to be light-weight, the overhead of parameters and computation is negligible in most cases.

**Contribution.** Our main contribution is three-fold.

1. We propose a simple yet effective attention module (CBAM) that can be widely applied to boost representation power of CNNs.

2. We validate the effectiveness of our attention module through extensive ablation studies.
3. We verify that performance of various networks is greatly improved on the multiple benchmarks (ImageNet-1K, MS COCO, and VOC 2007) by plugging our light-weight module.

## 2 Related Work

**Network engineering.** "Network engineering" has been one of the most important vision research, because well-designed networks ensure remarkable performance improvement in various applications. A wide range of architectures has been proposed since the successful implementation of a large-scale CNN [19]. An intuitive and simple way of extension is to increase the depth of neural networks [9]. Szegedy *et al.* [10] introduce a deep Inception network using a multi-branch architecture where each branch is customized carefully. While a naive increase in depth comes to saturation due to the difficulty of gradient propagation, ResNet [5] proposes a simple identity skip-connection to ease the optimization issues of deep networks. Based on the ResNet architecture, various models such as WideResNet [6], Inception-ResNet [8], and ResNeXt [7] have been developed. WideResNet [6] proposes a residual network with a larger number of convolutional filters and reduced depth. PyramidNet [20] is a strict generalization of WideResNet where the width of the network gradually increases. ResNeXt [7] suggests to use grouped convolutions and shows that increasing the cardinality leads to better classification accuracy. More recently, Huang *et al.* [21] propose a new architecture, DenseNet. It iteratively concatenates the input features with the output features, enabling each convolution block to receive raw information from all the previous blocks. While most of recent network engineering methods mainly target on three factors *depth* [19, 9, 10, 5], *width* [10, 22, 6, 8], and *cardinality* [7, 11], we focus on the other aspect, '*attention*', one of the curious facets of a human visual system.

**Attention mechanism.** It is well known that attention plays an important role in human perception [23–25]. One important property of a human visual system is that one does not attempt to process a whole scene at once. Instead, humans exploit a sequence of partial glimpses and selectively focus on salient parts in order to capture visual structure better [26].

Recently, there have been several attempts [27, 28] to incorporate attention processing to improve the performance of CNNs in large-scale classification tasks. Wang *et al.* [27] propose *Residual Attention Network* which uses an encoder-decoder style attention module. By refining the feature maps, the network not only performs well but is also robust to noisy inputs. Instead of directly computing the 3d attention map, we decompose the process that learns channel attention and spatial attention separately. The separate attention generation process for 3D feature map has much less computational and parameter over-

head, and therefore can be used as a plug-and-play module for pre-existing base CNN architectures.

More close to our work, Hu *et al.* [28] introduce a compact module to exploit the inter-channel relationship. In their *Squeeze-and-Excitation* module, they use global average-pooled features to compute channel-wise attention. However, we show that those are suboptimal features in order to infer fine channel attention, and we suggest to use max-pooled features as well. They also miss the spatial attention, which plays an important role in deciding 'where' to focus as shown in [29]. In our CBAM, we exploit both spatial and channel-wise attention based on an efficient architecture and empirically verify that exploiting both is superior to using only the channel-wise attention as [28]. Moreover, we empirically show that our module is effective in detection tasks (MS-COCO and VOC). Especially, we achieve state-of-the-art performance just by placing our module on top of the existing one-shot detector [30] in the VOC2007 test set.

## 3 Convolutional Block Attention Module

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.

**Channel attention module.** We produce a channel attention map by exploiting the inter-channel relationship of features. As each channel of a feature map is considered as a feature detector [31], channel attention focuses on 'what' is meaningful given an input image. To compute the channel attention efficiently, we squeeze the spatial dimension of the input feature map. For aggregating spatial information, average-pooling has been commonly adopted so far. Zhou *et al.* [32] suggest to use it to learn the extent of the target object effectively and Hu *et al.* [28] adopt it in their attention module to compute spatial statistics. Beyond the previous works, we argue that max-pooling gathers another important clue about distinctive object features to infer finer channel-wise attention. Thus, we use both average-pooled and max-pooled features simultaneously. We empirically confirmed that exploiting both features greatly improves representation power of networks rather than using each independently (see Sec. 4.1), showing the effectiveness of our design choice. We describe the detailed operation below.

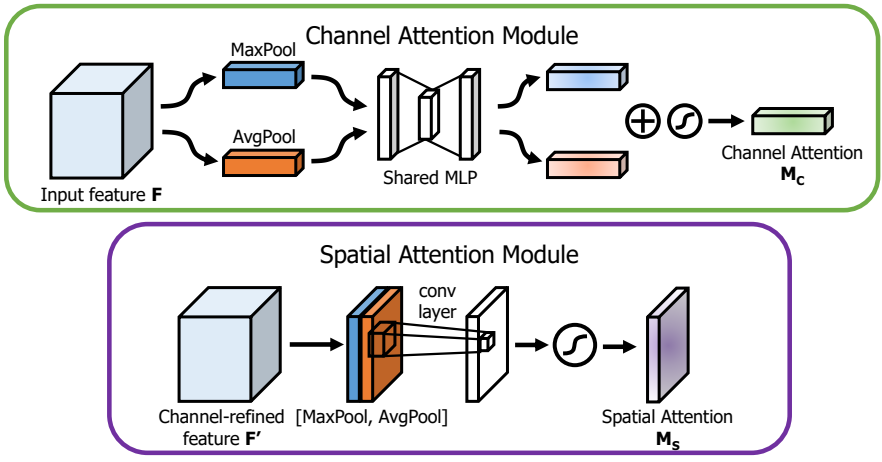



Fig. 2: **Diagram of each attention sub-module.** As illustrated, the channel sub-module utilizes both max-pooling outputs and average-pooling outputs with a shared network; the spatial sub-module utilizes similar two outputs that are pooled along the channel axis and forward them to a convolution layer.

We first aggregate spatial information of a feature map by using both average-pooling and max-pooling operations, generating two different spatial context descriptors: $\mathbf{F^c_{avg}}$ and $\mathbf{F^c_{max}}$, which denote average-pooled features and max-pooled features respectively. Both descriptors are then forwarded to a shared network to produce our channel attention map $\mathbf{M_c} \in \mathbb{R}^{C\times1\times1}$. The shared network is composed of multi-layer perceptron (MLP) with one hidden layer. To reduce parameter overhead, the hidden activation size is set to $\mathbb{R}^{C/r\times1\times1}$, where $r$ is the reduction ratio. After the shared network is applied to each descriptor, we merge the output feature vectors using element-wise summation. In short, the channel attention is computed as:

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

where $\sigma$ 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.** We generate a spatial attention map by utilizing the inter-spatial relationship of features. Different from the channel attention, the spatial attention focuses on 'where' is an informative part, which is complementary to the channel attention. To compute the spatial attention, we first apply average-pooling and max-pooling operations along the channel axis and concatenate them to generate an efficient feature descriptor. Applying pooling operations along the channel axis is shown to be effective in highlighting informative regions [33]. On the concatenated feature descriptor, we apply a convolution

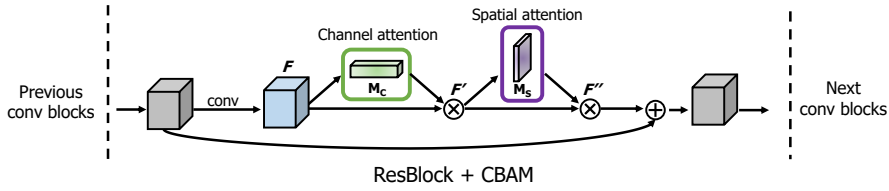



Fig. 3: **CBAM integrated with a ResBlock in ResNet[5].** This figure shows the exact position of our module when integrated within a ResBlock. We apply CBAM on the convolution outputs in each block.

layer to generate a spatial attention map $\mathbf{M_s}(\mathbf{F}) \in \mathbf{R}^{H \times W}$ which encodes where to emphasize or suppress. We describe the detailed operation below.

We aggregate channel information of a feature map by using two pooling operations, generating two 2D maps: $\mathbf{F^s_{avg}} \in \mathbb{R}^{1 \times H \times W}$ and $\mathbf{F^s_{max}} \in \mathbb{R}^{1 \times H \times W}$. Each denotes average-pooled features and max-pooled features across the channel. Those are then concatenated and convolved by a standard convolution layer, producing our 2D spatial attention map. In short, the spatial attention is computed as:

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

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

**Arrangement of attention modules.** Given an input image, two attention modules, channel and spatial, compute complementary attention, focusing on 'what' and 'where' respectively. Considering this, two modules can be placed in a parallel or sequential manner. We found that the sequential arrangement gives a better result than a parallel arrangement. For the arrangement of the sequential process, our experimental result shows that the channel-first order is slightly better than the spatial-first. We will discuss experimental results on network engineering in Sec. 4.1.

## 4 Experiments

We evaluate CBAM on the standard benchmarks: ImageNet-1K for image classification; MS COCO and VOC 2007 for object detection. In order to perform better apple-to-apple comparisons, we reproduced all the evaluated networks [5–7, 34, 28] in the PyTorch framework [35] and report our reproduced results in the whole experiments.

To thoroughly evaluate the effectiveness of our final module, we first perform extensive ablation experiments. Then, we verify that CBAM outperforms all the

baselines without bells and whistles, demonstrating the general applicability of CBAM across different architectures as well as different tasks. One can seamlessly integrate CBAM in any CNN architectures and jointly train the combined CBAM-enhanced networks. Fig. 3 shows a diagram of CBAM integrated with a ResBlock in ResNet [5] as an example.

### 4.1 Ablation studies

In this subsection, we empirically show the effectiveness of our design choice. For this ablation study, we use the ImageNet-1K dataset and adopt ResNet-50 [5] as the base architecture. The ImageNet-1K classification dataset [1] consists of 1.2 million images for training and 50,000 for validation with 1,000 object classes. We adopt the same data augmentation scheme with [5, 36] for training and apply a single-crop evaluation with the size of 224×224 at test time. The learning rate starts from 0.1 and drops every 30 epochs. We train the networks for 90 epochs. Following [5, 36, 37], we report classification errors on the validation set.

Our module design process is split into three parts. We first search for the effective approach to computing the channel attention, then the spatial attention. Finally, we consider how to combine both channel and spatial attention modules. We explain the details of each experiment below.

**Channel attention.** We experimentally verify that using both average-pooled and max-pooled features enables finer attention inference. We compare 3 variants of channel attention: average pooling, max pooling, and joint use of both poolings. Note that the channel attention module with an average pooling is the same as the SE [28] module. Also, when using both poolings, we use a shared MLP for attention inference to save parameters, as both of aggregated channel features lie in the same semantic embedding space. We only use channel attention modules in this experiment and we fix the reduction ratio to 16.

Experimental results with various pooling methods are shown in Table 1. We observe that max-pooled features are as meaningful as average-pooled features, comparing the accuracy improvement from the baseline. In the work of SE [28], however, they only exploit the average-pooled features, missing the importance

| Description | Parameters | GFLOPs | Top-1 Error(%) | Top-5 Error(%) |
|---|---|---|---|---|
| ResNet50 (baseline) | 25.56M | 3.86 | 24.56 | 7.50 |
| ResNet50 + AvgPool (SE [28]) | 25.92M | 3.94 | 23.14 | 6.70 |
| ResNet50 + MaxPool | 25.92M | 3.94 | 23.20 | 6.83 |
| ResNet50 + AvgPool & MaxPool | 25.92M | 4.02 | **22.80** | **6.52** |

Table 1: **Comparison of different channel attention methods**. We observe that using our proposed method outperforms recently suggested Squeeze and Excitation method [28].

| Description | Param. | GFLOPs | Top-1 Error(%) | Top-5 Error(%) |
|---|---|---|---|---|
| ResNet50 + channel (SE [28]) | 28.09M | 3.860 | 23.14 | 6.70 |
| ResNet50 + channel | 28.09M | 3.860 | 22.80 | 6.52 |
| ResNet50 + channel + spatial (1x1 conv, k=3) | 28.10M | 3.862 | 22.96 | 6.64 |
| ResNet50 + channel + spatial (1x1 conv, k=7) | 28.10M | 3.869 | 22.90 | 6.47 |
| ResNet50 + channel + spatial (avg&max, k=3) | 28.09M | 3.863 | 22.68 | 6.41 |
| ResNet50 + channel + spatial (avg&max, k=7) | 28.09M | 3.864 | **22.66** | **6.31** |

Table 2: **Comparison of different spatial attention methods**. Using the proposed channel-pooling (*i.e.* average- and max-pooling along the channel axis) along with the large kernel size of 7 for the following convolution operation performs best.

| Description | Top-1 Error(%) | Top-5 Error(%) |
|---|---|---|
| ResNet50 + channel (SE [28]) | 23.14 | 6.70 |
| ResNet50 + channel + spatial | **22.66** | **6.31** |
| ResNet50 + spatial + channel | 22.78 | 6.42 |
| ResNet50 + channel & spatial in parallel | 22.95 | 6.59 |

Table 3: **Combining methods of channel and spatial attention**. Using both attention is critical while the best-combining strategy (*i.e.* sequential, channel-first) further improves the accuracy.

of max-pooled features. We argue that max-pooled features which encode the degree of the most salient part can compensate the average-pooled features which encode global statistics softly. Thus, we suggest to use both features simultaneously and apply a shared network to those features. The outputs of a shared network are then merged by element-wise summation. We empirically show that our channel attention method is an effective way to push performance further from SE [28] without additional learnable parameters. As a brief conclusion, we use both average- and max-pooled features in our channel attention module with the reduction ratio of 16 in the following experiments.

**Spatial attention.** Given the channel-wise refined features, we explore an effective method to compute the spatial attention. The design philosophy is symmetric with the channel attention branch. To generate a 2D spatial attention map, we first compute a 2D descriptor that encodes channel information at each pixel over all spatial locations. We then apply one convolution layer to the 2D descriptor, obtaining the raw attention map. The final attention map is normalized by the sigmoid function.

We compare two methods of generating the 2D descriptor: *channel pooling* using average- and max-pooling across the channel axis and *standard* $1 \times 1$ *convolution* reducing the channel dimension into 1. In addition, we investigate the effect of a kernel size at the following convolution layer: kernel sizes of 3 and 7. In the experiment, we place the spatial attention module after the previously

designed channel attention module, as the final goal is to use both modules together.

Table 2 shows the experimental results. We can observe that the channel pooling produces better accuracy, indicating that explicitly modeled pooling leads to finer attention inference rather than learnable weighted channel pooling (implemented as $1 \times 1$ convolution). In the comparison of different convolution kernel sizes, we find that adopting a larger kernel size generates better accuracy in both cases. It implies that a broad view (*i.e.* large receptive field) is needed for deciding spatially important regions. Considering this, we adopt the channel-pooling method and the convolution layer with a large kernel size to compute spatial attention. In a brief conclusion, we use the average- and max-pooled features across the channel axis with a convolution kernel size of 7 as our spatial attention module.

**Arrangement of the channel and spatial attention.** In this experiment, we compare three different ways of arranging the channel and spatial attention submodules: sequential channel-spatial, sequential spatial-channel, and parallel use of both attention modules. As each module has different functions, the order may affect the overall performance. For example, from a spatial viewpoint, the channel attention is globally applied, while the spatial attention works locally. Also, it is natural to think that we may combine two attention outputs to build a 3D attention map. In the case, both attentions can be applied in parallel, then the outputs of the two attention modules are added and normalized with the sigmoid function.

Table 3 summarizes the experimental results on different attention arranging methods. From the results, we can find that generating an attention map sequentially infers a finer attention map than doing in parallel. In addition, the channel-first order performs slightly better than the spatial-first order. Note that all the arranging methods outperform using only the channel attention independently, showing that utilizing both attentions is crucial while the best-arranging strategy further pushes performance.

**Final module design.** Throughout the ablation studies, we have designed the channel attention module, the spatial attention module, and the arrangement of the two modules. Our final module is as shown in Fig. 1 and Fig. 2: we choose average- and max-pooling for both channel and spatial attention module; we use convolution with a kernel size of 7 in the spatial attention module; we arrange the channel and spatial submodules sequentially. Our final module(*i.e.* ResNet50 + CBAM) achieves top-1 error of 22.66%, which is much lower than SE [28](*i.e.* ResNet50 + SE), as shown in Table 4.

### 4.2 Image Classification on ImageNet-1K

We perform ImageNet-1K classification experiments to rigorously evaluate our module. We follow the same protocol as specified in Sec. 4.1 and evaluate our

| 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.

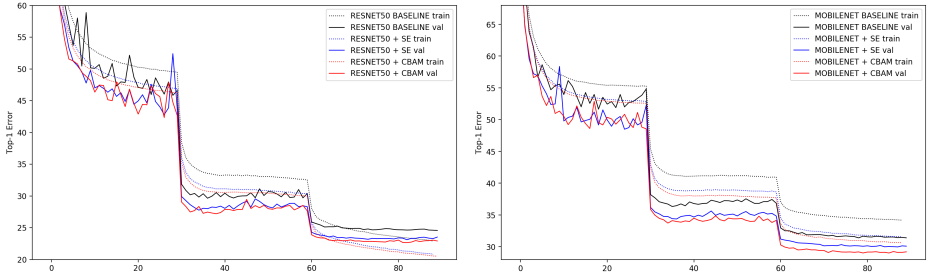
Table 4: **Classification results on ImageNet-1K.** Single-crop validation errors are reported.

module in various network architectures including ResNet [5], WideResNet [6], and ResNext [7].

Table 4 summarizes the experimental results. The networks with CBAM outperform all the baselines significantly, demonstrating that the CBAM can generalize well on various models in the large-scale dataset. Moreover, the models with CBAM improve the accuracy upon the one of the strongest method – SE [28] which is the winning approach of the ILSVRC 2017 classification task. It implies that our proposed approach is powerful, showing the efficacy of *new pooling method* that generates richer descriptor and *spatial attention* that complements the channel attention effectively.

Fig. 4 depicts the error curves of various networks during ImageNet-1K training. We can clearly see that our method exhibits lowest training and validation error in both error plots. It shows that CBAM has greater ability to improve generalization power of baseline models compared to SE [28].

We also find that the overall overhead of CBAM is quite small in terms of both parameters and computation. This motivates us to apply our proposed module CBAM to the light-weight network, MobileNet [34]. Table 5 summarizes the experimental results that we conducted based on the MobileNet architecture. We have placed CBAM to two models, basic and capacity-reduced model(*i.e.* adjusting width multiplier($\alpha$) to 0.7). We observe similar phenomenon as shown




(a) ResNet50 [5] (b) MobileNet [34]

Fig. 4: **Error curves during ImageNet-1K training**. Best viewed in color.

| Architecture | Parameters | GFLOPs | Top-1 Error (%) | Top-5 Error (%) |
|---|---|---|---|---|
| MobileNet [34] $\alpha = 0.7$ | 2.30M | 0.283 | 34.86 | 13.69 |
| MobileNet [34] $\alpha = 0.7$ + SE [28] | 2.71M | 0.283 | 32.50 | 12.49 |
| MobileNet [34] $\alpha = 0.7$ + CBAM | 2.71M | 0.289 | **31.51** | **11.48** |
| MobileNet [34] | 4.23M | 0.569 | 31.39 | 11.51 |
| MobileNet [34] + SE [28] | 5.07M | 0.570 | 29.97 | 10.63 |
| MobileNet [34] + CBAM | 5.07M | 0.576 | **29.01** | **9.99** |

***** all results are reproduced in the PyTorch framework.

Table 5: **Classification results on ImageNet-1K using the light-weight network, MobileNet [34].** Single-crop validation errors are reported.

in Table 4. CBAM not only boosts the accuracy of baselines significantly but also favorably improves the performance of SE [28]. This shows the great potential of CBAM for applications on low-end devices.

### 4.3 Network Visualization with Grad-CAM [18]

For the qualitative analysis, we apply the Grad-CAM [18] to different networks using images from the ImageNet validation set. Grad-CAM is a recently proposed visualization method which uses gradients in order to calculate the importance of the spatial locations in convolutional layers. As the gradients are calculated with respect to a unique class, Grad-CAM result shows attended regions clearly. By observing the regions that network has considered as important for predicting a class, we attempt to look at how this network is making good use of features. We compare the visualization results of CBAM-integrated network (ResNet50 + CBAM) with baseline (ResNet50) and SE-integrated network (ResNet50 + SE). Fig. 5 illustrate the visualization results. The softmax scores for a target class are also shown in the figure.

In Fig. 5, we can clearly see that the Grad-CAM masks of the CBAM-integrated network cover the target object regions better than other methods. That is, the CBAM-integrated network learns well to exploit information in target object regions and aggregate features from them. Note that target class

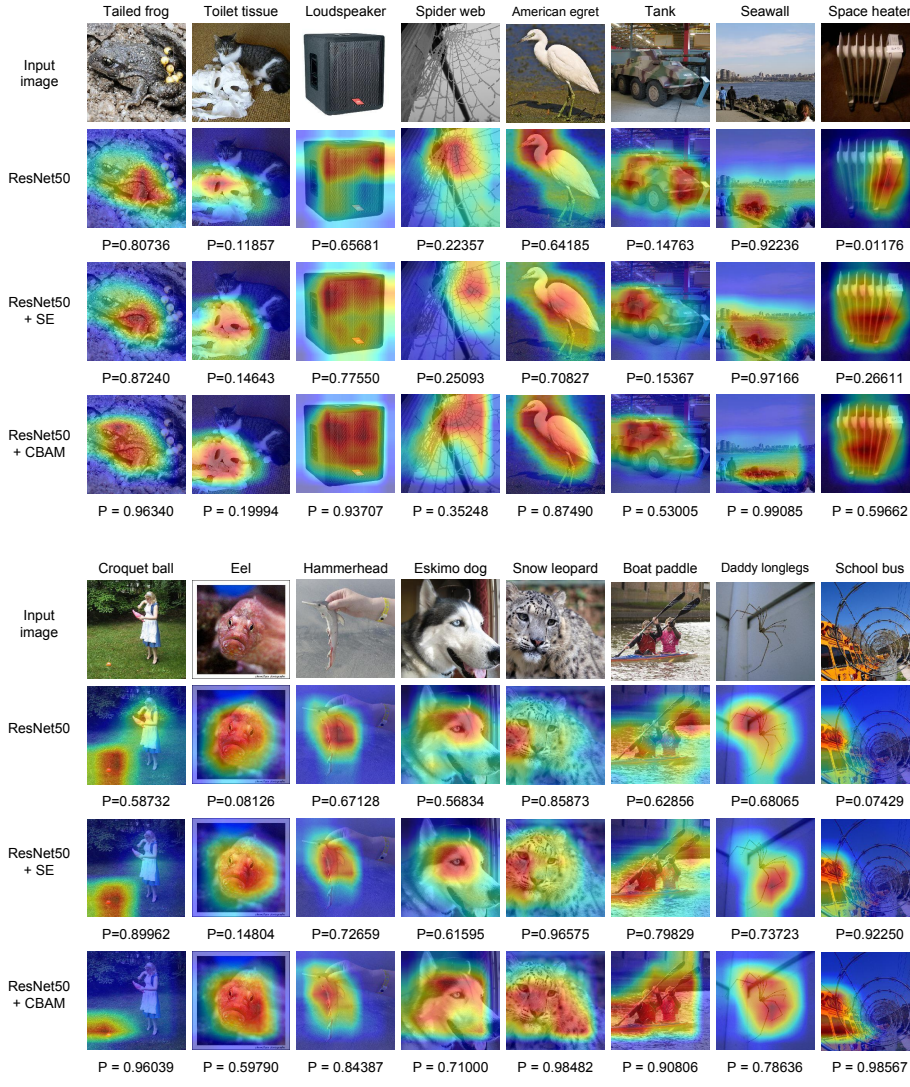



Fig. 5: **Grad-CAM [18] visualization results.** We compare the visualization results of CBAM-integrated network (ResNet50 + CBAM) with baseline (ResNet50) and SE-integrated network (ResNet50 + SE). The grad-CAM visualization is calculated for the last convolutional outputs. The ground-truth label is shown on the top of each input image and $P$ denotes the softmax score of each network for the ground-truth class.

| Backbone | Detector | mAP@.5 | mAP@.75 | mAP@[.5, .95] |
|---|---|---|---|---|
| ResNet50 [5] | Faster-RCNN [41] | 46.2 | 28.1 | 27.0 |
| ResNet50 [5] + CBAM | Faster-RCNN [41] | **48.2** | **29.2** | **28.1** |
| ResNet101 [5] | Faster-RCNN [41] | 48.4 | 30.7 | 29.1 |
| ResNet101 [5] + CBAM | Faster-RCNN [41] | **50.5** | **32.6** | **30.8** |

* all results are reproduced in the PyTorch framework.

Table 6: **Object detection mAP(%) on the MS COCO validation set**. We adopt the Faster R-CNN [41] detection framework and apply our module to the base networks. CBAM boosts mAP@[.5, .95] by 0.9 for both baseline networks.

| Backbone | Detector | mAP@.5 | Parameters (M) |
|---|---|---|---|
| VGG16 [9] | SSD [39] | 77.8 | 26.5 |
| VGG16 [9] | StairNet [30] | 78.9 | 32.0 |
| VGG16 [9] | StairNet [30] + SE [28] | 79.1 | 32.1 |
| VGG16 [9] | StairNet [30] + CBAM | **79.3** | 32.1 |
| MobileNet [34] | SSD [39] | 68.1 | 5.81 |
| MobileNet [34] | StairNet [30] | 70.1 | 5.98 |
| MobileNet [34] | StairNet [30] + SE [28] | 70.0 | 5.99 |
| MobileNet [34] | StairNet [30] + CBAM | **70.5** | 6.00 |

* all results are reproduced in the PyTorch framework.

Table 7: **Object detection mAP(%) on the VOC 2007 test set**. We adopt the StairNet [30] detection framework and apply SE and CBAM to the detectors. CBAM favorably improves all the strong baselines with negligible additional parameters.

scores also increase accordingly. From the observations, we conjecture that the feature refinement process of CBAM eventually leads the networks to utilize given features well.

### 4.4 MS COCO Object Detection

We conduct object detection on the Microsoft COCO dataset [3]. This dataset involves 80k training images ("2014 train") and 40k validation images ("2014 val"). The average mAP over different IoU thresholds from 0.5 to 0.95 is used for evaluation. According to [38, 39], we trained our model using all the training images as well as a subset of validation images, holding out 5,000 examples for validation. Our training code is based on [40] and we train the network for 490K iterations for fast performance validation. We adopt *Faster-RCNN* [41] as our detection method and ImageNet pre-trained ResNet50 and ResNet101 [5] as our baseline networks. Here we are interested in performance improvement by plugging CBAM to the baseline networks. Since we use the same detection method in all the models, the gains can only be attributed to the enhanced representation power, given by our module CBAM. As shown in the Table 6, we observe significant improvements from the baseline, demonstrating generalization performance of CBAM on other recognition tasks.

### 4.5 VOC 2007 Object Detection

We further perform experiments on the PASCAL VOC 2007 test set. In this experiment, we apply CBAM to the detectors, while the previous experiments (Table 6) apply our module to the base networks. We adopt the StairNet [30] framework, which is one of the strongest multi-scale method based on the SSD [39]. For the experiment, we reproduce SSD and StairNet in our PyTorch platform in order to estimate performance improvement of CBAM accurately and achieve 77.8% and 78.9% mAP@.5 respectively, which are higher than the original accuracy reported in the original papers. We then place SE [28] and CBAM right before every classifier, refining the final features which are composed of up-sampled global features and corresponding local features before the prediction, enforcing model to adaptively select only the meaningful features. We train all the models on the union set of VOC 2007 trainval and VOC 2012 trainval ("07+12"), and evaluate on the VOC 2007 test set. The total number of training epochs is 250. We use a weight decay of 0.0005 and a momentum of 0.9. In all the experiments, the size of the input image is fixed to 300 for the simplicity.

The experimental results are summarized in Table 7. We can clearly see that CBAM improves the accuracy of all strong baselines with two backbone networks. Note that accuracy improvement of CBAM comes with a negligible parameter overhead, indicating that enhancement is not due to a naive capacity-increment but because of our effective feature refinement. In addition, the result using the light-weight backbone network [34] again shows that CBAM can be an interesting method to low-end devices.

## 5 Conclusion

We have presented the convolutional bottleneck attention module (CBAM), a new approach to improve representation power of CNN networks. We apply attention-based feature refinement with two distinctive modules, channel and spatial, and achieve considerable performance improvement while keeping the overhead small. For the channel attention, we suggest to use the max-pooled features along with the average-pooled features, leading to produce finer attention than SE [28]. We further push the performance by exploiting the spatial attention. Our final module (CBAM) learns what and where to emphasize or suppress and refines intermediate features effectively. To verify its efficacy, we conducted extensive experiments with various state-of-the-art models and confirmed that CBAM outperforms all the baselines on three different benchmark datasets: ImageNet-1K, MS COCO, and VOC 2007. In addition, we visualize how the module exactly infers given an input image. Interestingly, we observed that our module induces the network to focus on target object properly. We hope CBAM become an important component of various network architectures.

## References


1. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: Proc. of Computer Vision and Pattern Recognition (CVPR). (2009)
2. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images
3. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: Proc. of European Conf. on Computer Vision (ECCV). (2014)
4. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11) (1998) 2278–2324
5. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proc. of Computer Vision and Pattern Recognition (CVPR). (2016)
6. Zagoruyko, S., Komodakis, N.: Wide residual networks. arXiv preprint arXiv:1605.07146 (2016)
7. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. arXiv preprint arXiv:1611.05431 (2016)
8. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.A.: Inception-v4, inception-resnet and the impact of residual connections on learning. In: Proc. of Association for the Advancement of Artificial Intelligence (AAAI). (2017)
9. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
10. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proc. of Computer Vision and Pattern Recognition (CVPR). (2015)
11. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. arXiv preprint arXiv:1610.02357 (2016)
12. Mnih, V., Heess, N., Graves, A., et al.: Recurrent models of visual attention." advances in neural information processing systems. In: Proc. of Neural Information Processing Systems (NIPS). (2014)
13. Ba, J., Mnih, V., Kavukcuoglu, K.: Multiple object recognition with visual attention. (2014)
14. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. (2014)
15. Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., Bengio, Y.: Show, attend and tell: Neural image caption generation with visual attention. (2015)
16. Gregor, K., Danihelka, I., Graves, A., Rezende, D.J., Wierstra, D.: Draw: A recurrent neural network for image generation. (2015)
17. Jaderberg, M., Simonyan, K., Zisserman, A., et al.: Spatial transformer networks. In: Proc. of Neural Information Processing Systems (NIPS). (2015)
18. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: Visual explanations from deep networks via gradient-based localization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2017) 618–626
19. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Proc. of Neural Information Processing Systems (NIPS). (2012)
20. Han, D., Kim, J., Kim, J.: Deep pyramidal residual networks. In: Proc. of Computer Vision and Pattern Recognition (CVPR). (2017)

21. Huang, G., Liu, Z., Weinberger, K.Q., van der Maaten, L.: Densely connected convolutional networks. arXiv preprint arXiv:1608.06993 (2016)
22. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proc. of Computer Vision and Pattern Recognition (CVPR). (2016)
23. Itti, L., Koch, C., Niebur, E.: A model of saliency-based visual attention for rapid scene analysis. In: IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI). (1998)
24. Rensink, R.A.: The dynamic representation of scenes. In: Visual cognition 7.1-3. (2000)
25. Corbetta, M., Shulman, G.L.: Control of goal-directed and stimulus-driven attention in the brain. In: Nature reviews neuroscience 3.3. (2002)
26. Larochelle, H., Hinton, G.E.: Learning to combine foveal glimpses with a third-order boltzmann machine. In: Proc. of Neural Information Processing Systems (NIPS). (2010)
27. Wang, F., Jiang, M., Qian, C., Yang, S., Li, C., Zhang, H., Wang, X., Tang, X.: Residual attention network for image classification. arXiv preprint arXiv:1704.06904 (2017)
28. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. arXiv preprint arXiv:1709.01507 (2017)
29. Chen, L., Zhang, H., Xiao, J., Nie, L., Shao, J., Chua, T.S.: Sca-cnn: Spatial and channel-wise attention in convolutional networks for image captioning. In: Proc. of Computer Vision and Pattern Recognition (CVPR). (2017)
30. Sanghyun, W., Soonmin, H., So, K.I.: Stairnet: Top-down semantic aggregation for accurate one shot detection. In: Proc. of Winter Conference on Applications of Computer Vision (WACV). (2018)
31. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: Proc. of European Conf. on Computer Vision (ECCV). (2014)
32. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A.: Learning deep features for discriminative localization. In: Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on, IEEE (2016) 2921–2929
33. Zagoruyko, S., Komodakis, N.: Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In: ICLR. (2017)
34. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
35. : Pytorch. http://pytorch.org/ Accessed: 2017-11-08.
36. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: Proc. of European Conf. on Computer Vision (ECCV). (2016)
37. Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K.Q.: Deep networks with stochastic depth. In: Proc. of European Conf. on Computer Vision (ECCV). (2016)
38. Bell, S., Lawrence Zitnick, C., Bala, K., Girshick, R.: Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In: Proc. of Computer Vision and Pattern Recognition (CVPR). (2016)
39. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: Proc. of European Conf. on Computer Vision (ECCV). (2016)
40. Chen, X., Gupta, A.: An implementation of faster rcnn with study for region sampling. arXiv preprint arXiv:1702.02138 (2017)

41. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Proc. of Neural Information Processing Systems (NIPS). (2015)