BAM: Bottleneck Attention Module

Jongchan Park*†[1]
jcpark@lunit.io
Sanghyun Woo*[2]
shwoo93@kaist.ac.kr
Joon-Young Lee[3]
jolee@adobe.com
In So Kweon[2]
iskweon77@kaist.ac.kr

[1] Lunit Inc.
Seoul, Korea
[2] Korea Advanced Institute of Science and Technology (KAIST)
Daejeon, Korea
[3] Adobe Research
San Jose, CA, USA

Abstract

Recent advances in deep neural networks have been developed via architecture search for stronger representational power. In this work, we focus on the effect of *attention* in general deep neural networks. We propose a simple and effective attention module, named *Bottleneck Attention Module* (BAM), that can be integrated with any feed-forward convolutional neural networks. Our module infers an attention map along two separate pathways, *channel* and *spatial*. We place our module at each *bottleneck* of models where the downsampling of feature maps occurs. Our module constructs a hierarchical attention at bottlenecks with a number of parameters and it is trainable in an end-to-end manner jointly with any feed-forward models. We validate our BAM through extensive experiments on CIFAR-100, ImageNet-1K, VOC 2007 and MS COCO benchmarks. Our experiments show consistent improvement in classification and detection performances with various models, demonstrating the wide applicability of BAM. The code and models will be publicly available[1].

1 Introduction

Deep learning has been a powerful tool for a series of pattern recognition applications including classification, detection, segmentation and control problems. Due to its data-driven nature and availability of large scale parallel computing, deep neural networks achieve state-of-the-art results in most areas. Researchers have done many efforts to boost the performance in various ways such as designing optimizers [28, 48], proposing adversarial training scheme [11], or task-specific meta architecture like 2-stage architectures [37] for detection.

A fundamental approach to boost performance is to design a good backbone architecture. Since the very first large-scale deep neural network AlexNet [30], various backbone architectures such as VGGNet [40], GoogLeNet [41], ResNet [16], DenseNet [20], have been proposed. All backbone architectures have their own design choices, and show significant performance boosts over the precedent architectures.

*Both authors have equally contributed.
†The work was done while the author was at KAIST.
[1] https://sites.google.com/view/bottleneck-attention-module

The most intuitive way to boost the network performance is to stack more layers. Deep neural networks then are able to approximate high-dimensional function using their *deep* layers. The philosophy of VGGNet [40] and ResNet [15] precisely follows this. Compared to AlexNet [30], VGGNet has twice more layers. Furthermore, ResNet has 22x more layers than VGGNet with improved gradient flow by adopting residual connections. GoogLeNet [41], which is also very deep, uses concatenation of features with various filter sizes at each convolutional block. The use of diverse features at the same layer shows increased performance, resulting in powerful representation. DenseNet [20] also uses the concatenation of diverse feature maps, but the features are from different layers. In other words, outputs of convolutional layers are iteratively concatenated upon the input feature maps. WideResNet [47] shows that using more channels, *wider* convolutions, can achieve higher performance than naively deepening the networks. Similarly, PyramidNet [13] shows that increasing channels in deeper layers can effectively boost the performance. Recent approaches with grouped convolutions, such as ResNeXt [43] or Xception [7], show state-of-the-art performances as backbone architectures. The success of ResNeXt and Xception comes from the convolutions with higher *cardinality* which can achieve high performance effectively. Besides, a practical line of research is to find mobile-oriented, computationally effective architectures. MobileNet [18], sharing a similar philosophy with ResNeXt and Xception, use *depthwise convolutions* with high cardinalities.

Apart from the previous approaches, we investigate the effect of *attention* in DNNs, and propose a simple, light-weight module for general DNNs. That is, the proposed module is designed for easy integration with existing CNN architectures. Attention mechanism in deep neural networks has been investigated in many previous works [2, 3, 12, 25, 35, 44]. While most of the previous works use attention with task-specific purposes, we explicitly investigate the use of attention as a way to improve network's representational power in an extremely efficient way. As a result, we propose "Bottleneck Attention Module" (BAM), a simple and efficient attention module that can be used in any CNNs. Given a 3D feature map, BAM produces a 3D attention map to emphasize important elements. In BAM, we decompose the process of inferring a 3D attention map in two streams (Fig. 2), so that the computational and parametric overhead are significantly reduced. As the channels of feature maps can be regarded as feature detectors, the two branches (spatial and channel) explicitly learn 'what' and 'where' to focus on.

We test the efficacy of BAM with various baseline architectures on various tasks. On the CIFAR-100 and ImageNet classification tasks, we observe performance improvements over baseline networks by placing BAM. Interestingly, we have observed that multiple BAMs located at different bottlenecks build a hierarchical attention as shown in Fig. 1. Finally, we validate the performance improvement of object detection on the VOC 2007 and MS COCO datasets, demonstrating a wide applicability of BAM. Since we have carefully designed our module to be light-weight, parameter and computational overheads are negligible.

Contribution. Our main contribution is three-fold.

1. We propose a simple and effective attention module, BAM, which can be integrated with any CNNs without bells and whistles.
2. We validate the design of BAM through extensive ablation studies.
3. We verify the effectiveness of BAM throughout extensive experiments with various baseline architectures on multiple benchmarks (CIFAR-100, ImageNet-1K, VOC 2007 and MS COCO).

2 Related Work

A number of studies [8, 24, 38] have shown that *attention* plays an important role in human perception. For example, the resolution at the foveal center of human eyes is higher than surrounding areas [17]. In order to efficiently and adaptively process visual information, human visual systems iteratively process spatial glimpses and focus on salient areas [31].

Cross-modal attention. Attention mechanism is a widely-used technique in multi-modal settings, especially where certain modalities should be processed conditioning on other modalities. Visual question answering (VQA) task is a well-known example for such tasks. Given an image and natural language question, the task is to predict an answer such as counting the number, inferring the position or the attributes of the targets. VQA task can be seen as a set of dynamically changing tasks where the provided image should be processed according to the given question. Attention mechanism softly chooses the task(question)-relevant aspects in the image features. As suggested in [45], attention maps for the image features are produced from the given question, and it act as *queries* to retrieve question-relevant features. The final answer is classified with the stacked images features. Another way of doing this is to use bi-directional inferring, producing attention maps for both text and images, as suggested in [36]. In such literatures, attention maps are used as an effective way to solve tasks in a conditional fashion, but they are trained in separate stages for task-specific purposes.

Self-attention. There have been various approaches to integrate *attention* in DNNs, jointly training the feature extraction and attention generation in an end-to-end manner. A few attempts [19, 42] have been made to consider *attention* as an effective solution for general classification task. Wang *et al*. have proposed *Residual Attention Networks* which use a hourglass module to generate 3D attention maps for intermediate features. Even the architecture is resistant to noisy labels due to generated attention maps, the computational/parameter overhead is large because of the heavy 3D map generation process. Hu *et al*. have proposed a compact 'Squeeze-and-Excitation' module to exploit the inter-channel relationships. Although it is not explicitly stated in the paper, it can be regarded as an *attention* mechanism applied upon channel axis. However, they miss the *spatial* axis, which is also an important factor in inferring accurate attention map.

Adaptive modules. Several previous works use adaptive modules that dynamically changes their output according to their inputs. Dynamic Filter Network [27] proposes to generate convolutional features based on the input features for flexibility. Spatial Transformer Network [26] adaptively generates hyper-parameters of affine transformations using input feature so that target area feature maps are well aligned finally. This can be seen as a *hard attention* upon the feature maps. Deformable Convolutional Network [9] uses *deformable convolution* where pooling offsets are dynamically generated from input features, so that only the relevant features are pooled for convolutions. Similar to the above approaches, BAM is also a self-contained adaptive module that dynamically suppress or emphasize feature maps through *attention* mechanism.

In this work, we exploit both channel and spatial axes of attention with a simple and lightweight design. Furthermore, we find an efficient location to put our module - *bottleneck* of the network.

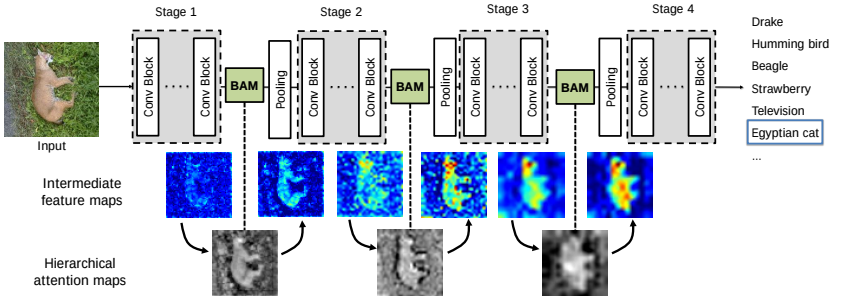

Figure 1: **BAM integrated with a general CNN architecture.** As illustrated, BAM is placed at every bottleneck of the network. Interestingly, we observe multiple BAMs construct a hierarchical attention which is similar to a human perception procedure. BAM denoises low-level features such as background texture features at the early stage. BAM then gradually focuses on the exact target which is a high-level semantic. ***More visualizations and analysis*** are included in the supplementary material due to space constraints.

3 Bottleneck Attention Module

The detailed structure of BAM is illustrated in Fig. 2. For the given input feature map $\mathbf{F} \in \mathbb{R}^{C\times H\times W}$, BAM infers a 3D attention map $\mathbf{M}(\mathbf{F}) \in \mathbb{R}^{C\times H\times W}$. The refined feature map $\mathbf{F}'$ is computed as:

$$\mathbf{F}' = \mathbf{F} + \mathbf{F} \otimes \mathbf{M}(\mathbf{F}), \tag{1}$$

where $\otimes$ denotes element-wise multiplication. We adopt a residual learning scheme along with the attention mechanism to facilitate the gradient flow. To design an efficient yet powerful module, we first compute the channel attention $\mathbf{M_c}(\mathbf{F}) \in \mathbb{R}^C$ and the spatial attention $\mathbf{M_s}(\mathbf{F}) \in \mathbb{R}^{H\times W}$ at two separate branches, then compute the attention map $\mathbf{M}(\mathbf{F})$ as:

$$\mathbf{M}(\mathbf{F}) = \sigma(\mathbf{M_c}(\mathbf{F}) + \mathbf{M_s}(\mathbf{F})), \tag{2}$$

where σ is a sigmoid function. Both branch outputs are resized to $\mathbb{R}^{C\times H\times W}$ before addition.

Channel attention branch. As each channel contains a specific feature response, we exploit the inter-channel relationship in the channel branch. To aggregate the feature map in each channel, we take global average pooling on the feature map $\mathbf{F}$ and produce a channel vector $\mathbf{F_c} \in \mathbb{R}^{C\times 1\times 1}$. This vector softly encodes global information in each channel. To estimate attention across channels from the channel vector $\mathbf{F_c}$, we use a multi-layer perceptron (MLP) with one hidden layer. To save a parameter overhead, the hidden activation size is set to $\mathbb{R}^{C/r\times 1\times 1}$, where r is the reduction ratio. After the MLP, we add a batch normalization (BN) layer [23] to adjust the scale with the spatial branch output. In short, the channel attention is computed as:

$$\begin{aligned}\mathbf{M_c}(\mathbf{F}) &= BN(MLP(AvgPool(\mathbf{F}))) \\ &= BN(\mathbf{W_1}(\mathbf{W_0}AvgPool(\mathbf{F}) + \mathbf{b_0}) + \mathbf{b_1}),\end{aligned} \tag{3}$$

where $\mathbf{W_0} \in \mathbb{R}^{C/r\times C}$, $\mathbf{b_0} \in \mathbb{R}^{C/r}$, $\mathbf{W_1} \in \mathbb{R}^{C\times C/r}$, $\mathbf{b_1} \in \mathbb{R}^C$.

Spatial attention branch. The spatial branch produces a spatial attention map $\mathbf{M_s}(\mathbf{F}) \in \mathbf{R}^{H\times W}$ to emphasize or suppress features in different spatial locations. It is widely known that [4, 14, 34, 46] utilizing contextual information is crucial to know which spatial locations

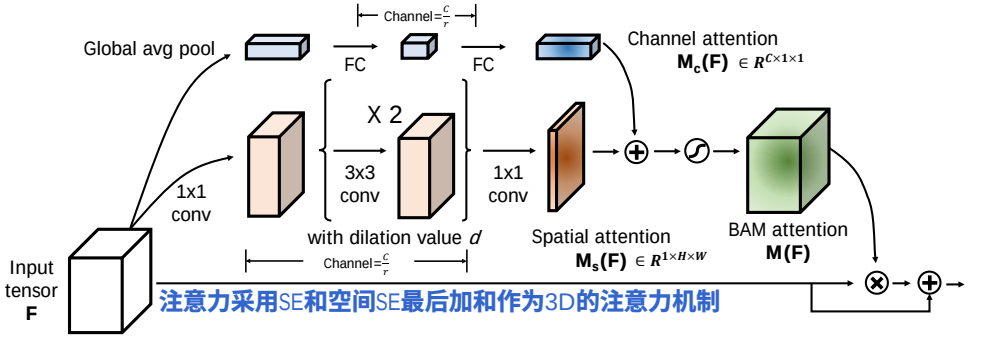

Figure 2: **Detailed module architecture.** Given the intermediate feature map **F**, the module computes corresponding attention map **M(F)** through the two separate attention branches – channel $\mathbf{M_c}$ and spatial $\mathbf{M_s}$. We have two hyper-parameters for the module: *dilation value (d)* and *reduction ratio (r)*. The dilation value determines the size of receptive fields which is helpful for the contextual information aggregation at the spatial branch. The reduction ratio controls the capacity and overhead in both attention branches. Through the experimental validation (see Sec. 4.1), we set {d = 4, r = 16}.

should be focused on. It is important to have a large receptive field to effectively leverage contextual information. We employ the dilated convolution [46] to enlarge the receptive fields with high efficiency. We observe that the dilated convolution facilitates constructing a more effective spatial map than the standard convolution (see Sec. 4.1). The "bottleneck structure" suggested by ResNet [15] is adopted in our spatial branch, which saves both the number of parameters and computational overhead. Specifically, the feature $\mathbf{F} \in \mathbb{R}^{C\times H\times W}$ is projected into a reduced dimension $\mathbb{R}^{C/r\times H\times W}$ using 1×1 convolution to integrate and compress the feature map across the channel dimension. We use the same reduction ratio r with the channel branch for simplicity. After the reduction, two 3×3 dilated convolutions are applied to utilize contextual information effectively. Finally, the features are again reduced to $\mathbb{R}^{1\times H\times W}$ spatial attention map using 1×1 convolution. For a scale adjustment, a batch normalization layer is applied at the end of the spatial branch. In short, the spatial attention is computed as:

$$\mathbf{M_s}(\mathbf{F}) = BN(f_3^{1\times1}(f_2^{3\times3}(f_1^{3\times3}(f_0^{1\times1}(\mathbf{F}))))), \tag{4}$$

where f denotes a convolution operation, BN denotes a batch normalization operation, and the superscripts denote the convolutional filter sizes. There are two 1×1 convolutions for channel reduction The intermediate 3×3 dilated convolutions are applied to aggregate contextual information with a larger receptive field.

Combine two attention branches. After acquiring the channel attention $\mathbf{M_c}(\mathbf{F})$ and the spatial attention $\mathbf{M_s}(\mathbf{F})$ from two attention branches, we combine them to produce our final 3D attention map $\mathbf{M}(\mathbf{F})$. Since the two attention maps have different shapes, we expand the attention maps to $\mathbb{R}^{C\times H\times W}$ before combining them. Among various combining methods, such as element-wise summation, multiplication, or max operation, we choose element-wise summation for efficient gradient flow [15]. We empirically verify that element-wise summation results in the best performance among three options (see Sec. 4). After the summation, we take a sigmoid function to obtain the final 3D attention map $\mathbf{M}(\mathbf{F})$ in the range from 0 to 1. This 3D attention map is element-wisely multiplied with the input feature map **F** then is added upon the original input feature map to acquire the refined feature map $\mathbf{F}'$ as Eq. (1).

4 Experiments

We evaluate BAM on the standard benchmarks: CIFAR-100, ImageNet-1K for image classification and VOC 2007, MS COCO for object detection. In order to perform better apple-to-apple comparisons, we first reproduce all the reported performance of networks in the PyTorch framework [1] and set as our baselines [15, 20, 43, 47]. Then we perform extensive experiments to thoroughly evaluate the effectiveness of our final module. Finally, we verify that BAM outperforms all the baselines without bells and whistles, demonstrating the general applicability of BAM across different architectures as well as different tasks. Table 5, Table 6, Table 7, Table 8, Table 9 can be found at supplemental material.

4.1 Ablation studies on CIFAR-100

The CIFAR-100 dataset [29] consists of 60,000 32×32 color images drawn from 100 classes. The training and test sets contain 50,000 and 10,000 images respectively. We adopt a standard data augmentation method of random cropping with 4-pixel padding and horizontal flipping for this dataset. For pre-processing, we normalize the data using RGB mean values and standard deviations.

Dilation value and Reduction ratio. In Table 1, we perform an experiment to determine two major hyper-parameters in our module, which are dilation value and reduction ratio, based on the ResNet50 architecture. The *dilation value* determines the sizes of receptive fields in the spatial attention branch. Table 1 shows the comparison result of four different dilation values. We can clearly see the performance improvement with larger dilation values, though it is saturated at the dilation value of 4. This phenomenon can be interpreted in terms of contextual reasoning, which is widely exploited in dense prediction tasks [4, 5, 34, 46, 49]. Since the sequence of dilated convolutions allows an exponential expansion of the receptive field, it enables our module to seamlessly aggregate contextual information. Note that the standard convolution (i.e. *dilation value* of 1) produces the lowest accuracy, demonstrating the efficacy of a context-prior for inferring the spatial attention map. The *reduction ratio* is directly related to the number of channels in both attention branches, which enable us to control the capacity and overhead of our module. In Table 1, we compare performance with four different reduction ratios. Interestingly, the reduction ratio of 16 achieves the best accuracy, even though the reduction ratios of 4 and 8 have higher capacity. We conjecture this result as over-fitting since the training losses converged in both cases. Based on the result in Table 1, we set the dilation value as 4 and the reduction ratio as 16 in the following experiments.

Separate or Combined branches. In Table 1, we conduct an ablation study to validate our design choice in the module. We first remove each branch to verify the effectiveness of utilizing both channel and spatial attention branches. As shown in Table 1, although each attention branch is effective to improve performance over the baseline, we observe significant performance boosting when we use both branches jointly. This shows that combining the channel and spatial branches together play a critical role in inferring the final attention map. In fact, this design follows the similar aspect of a human visual system, which has 'what' (channel) and 'where' (spatial) pathways and both pathways contribute to process visual information [6, 31].

Independent Variables	Value	Params	Error
Dilation value (d)	1	24.07M	20.47
	2	24.07M	20.28
	4	24.07M	**20**
	6	24.07M	20.08
Reduction ratio (r)	4	26.30M	20.46
	8	24.62M	20.56
	16	24.07M	**20**
	32	23.87M	21.24
Base (ResNet50[15])	-	23.68M	21.49

(a) Experiments on hyper-params.

	Base					**+BAM**
Channel		✓		✓	✓	✓
Spatial			✓	✓	✓	✓
MAX				✓		
PROD					✓	
SUM						✓
Error	21.49	21.29	21.24	20.28	20.21	**20**

(b) Experiments on each branch.

ConvBlock vs BAM	Params	Error
ResNet50[15]	23.68M	21.49
+ ResBlock	25.14M	21.02
+ BAM	**24.07M**	**20**
WideResNet28[47] (w=8)	23.4M	20.4
+ WideResBlock	24.88M	19.51
+ BAM	**23.56M**	**19.06**
ResNeXt[43] 8x64d	34.4M	18.18
+ ResNeXtBlock	37.3M	17.69
+ BAM	**34.61M**	**16.71**

(c) Experiments comparing conv blocks and BAM.

Table 1: Ablation studies on the structure and hyper parameters of BAM. **(a)** includes experiments for the optimal value for the two hyper parameters; **(b)** includes experiments to verify the effective of the spatial and channel branches; **(c)** includes experiments to compare the effectiveness of BAM over the original conv blocks.

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**

Table 2: **Bottleneck v.s. Inside each Convolution Block.** BAM-C denotes where the module is inserted to each convolution block.

Combining methods. We also explore three different combining strategies: element-wise maximum, element-wise product, and element-wise summation. Table 1 summarizes the comparison result for the three different implementations. We empirically confirm that element-wise summation achieves the best performance. In terms of the information flow, the element-wise summation is an effective way to integrate and secure the information from the previous layers. In the forward phase, it enables the network to use the information from two complementary branches, channel and spatial, without losing any of information. In the backward phase, the gradient is distributed equally to all of the inputs, leading to efficient training. Element-wise product, which can assign a large gradient to the small input, makes the network hard to converge, yielding the inferior performance. Element-wise maximum, which routes the gradient only to the higher input, provides a regularization effect to some extent, leading to unstable training since our module has few parameters. Note that all of three different implementations outperform the baselines, showing that utilizing each branch is crucial while the best-combining strategy further boosts performance.

Comparison with placing original convblocks. In this experiment, we empirically verify that the significant improvement does not come from the increased depth by naively adding the extra layers to the bottlenecks. We add auxiliary convolution blocks which have the same topology with their baseline convolution blocks, then compare it with BAM in Table 1. we can obviously notice that plugging BAM not only produces superior performance but also puts less overhead than naively placing the extra layers. It implies that the improvement of BAM is not merely due to the increased depth but because of the effective feature refinement.

Bottleneck: The efficient point to place BAM. We empirically verify that the bottlenecks of networks are the effective points to place our module BAM. Recent studies on attention mechanisms [19, 42] mainly focus on modifications within the '*convolution blocks*' rather

Architecture	Parameters	GFLOPs	Error
ResNet 50 [15]	23.71M	1.22	21.49
ResNet 50 [15] + BAM	24.07M$_{(+0.36)}$	1.25$_{(+0.03)}$	**20.00**
ResNet 101 [15]	42.70M	2.44	20.00
ResNet 101 [15] + BAM	43.06M$_{(+0.36)}$	2.46$_{(+0.02)}$	**19.61**
PreResNet 110[16]	1.726M	0.245	22.22
PreResNet 110[16] + BAM	1.733M$_{(+0.007)}$	0.246$_{(+0.01)}$	**21.96**
WideResNet 28 [47] (w=8)	23.40M	3.36	20.40
WideResNet 28 [47] (w=8) + BAM	23.42M$_{(+0.02)}$	3.37$_{(+0.01)}$	**19.06**
WideResNet 28 [47] (w=10)	36.54M	5.24	18.89
WideResNet 28 [47] (w=10) + BAM	36.57M$_{(+0.03)}$	5.25$_{(+0.01)}$	**18.56**
ResNeXt 29 [43] 8x64d	34.52M	4.99	18.18
ResNeXt 29 [43] 8x64d + BAM	34.61M$_{(+0.09)}$	5.00$_{(+0.01)}$	**16.71**
ResNeXt 29 [43] 16x64d	68.25M	9.88	17.25
ResNeXt 29 [43] 16x64d + BAM	68.34M$_{(+0.09)}$	9.90$_{(+0.02)}$	**16.39**
DenseNet 100-BC[20] (k=12)	0.8M	0.29	21.95
DenseNet 100-BC[20] (k=12) + BAM	0.84M$_{(+0.04)}$	0.30$_{(+0.01)}$	**20.65**

(a) CIFAR-100 experiment results

Architecture	Parameters	GFLOPs	Top-1(%)	Top-5(%)
ResNet18 [15]	11.69M	1.81	29.60	10.55
ResNet18 [15] + BAM	11.71M$_{(+0.02)}$	1.82$_{(+0.01)}$	**28.88**	**10.01**
ResNet50 [15]	25.56M	3.86	24.56	7.50
ResNet50 [15] + BAM	25.92M$_{(+0.36)}$	3.94$_{(+0.08)}$	**24.02**	**7.18**
ResNet101 [15]	44.55M	7.57	23.38	6.88
ResNet101 [15] + BAM	44.91M$_{(+0.36)}$	7.65$_{(+0.08)}$	**22.44**	**6.29**
WideResNet18 [47] (widen=1.5)	25.88M	3.87	26.85	8.88
WideResNet18 [47] (widen=1.5) + BAM	25.93M$_{(+0.05)}$	3.88$_{(+0.01)}$	**26.67**	**8.69**
WideResNet18 [47] (widen=2.0)	45.62M	6.70	25.63	8.20
WideResNet18 [47] (widen=2.0) + BAM	45.71M$_{(+0.09)}$	6.72$_{(+0.02)}$	**25.00**	**7.81**
ResNeXt50 [43] (32x4d)	25.03M	3.77	22.85	6.48
ResNeXt50 [43] (32x4d) + BAM	25.39M$_{(+0.36)}$	3.85$_{(+0.08)}$	**22.56**	**6.40**
MobileNet[18]	4.23M	0.569	31.39	11.51
MobileNet[18] + BAM	4.32M$_{(+0.09)}$	0.589$_{(+0.02)}$	**30.58**	**10.90**
MobileNet[18] $\alpha = 0.7$	2.30M	0.283	34.86	13.69
MobileNet[18] $\alpha = 0.7$ + BAM	2.34M$_{(+0.04)}$	0.292$_{(+0.009)}$	**33.09**	**12.69**
MobileNet[18] $\rho = 192/224$	4.23M	0.439	32.89	12.33
MobileNet[18] $\rho = 192/224$ + BAM	4.32M$_{(+0.09)}$	0.456$_{(+0.017)}$	**31.56**	**11.60**
SqueezeNet v1.1 [22]	1.24M	0.290	43.09	20.48
SqueezeNet v1.1 [22] + BAM	1.26M$_{(+0.02)}$	0.304$_{(+0.014)}$	**41.83**	**19.58**

(b) ImageNet classification results

* all results are reproduced in the PyTorch framework.

Table 3: Experiments on image classification tasks: CIFAR-100 and ImageNet 1K classification. The numbers inside the parentheses indicate the parameter/computational overhead. w denotes the widening factor in WideResNet [47] and k denotes the growth rate in DenseNet [20]. For the DenseNet [20], we put our module back and forth of the transition block.

than the '*bottlenecks*'. We compare those two different locations by using various models on CIFAR-100. In Table 2, we can clearly observe that placing the module at the bottleneck is effective in terms of overhead/accuracy trade-offs. It puts much less overheads with better accuracy in most cases except PreResNet 110 [16].

4.2 Classification Results on CIFAR-100

In Table 3, we compare the performance on CIFAR-100 after placing BAM at the bottlenecks of state-of-the-art models including [15, 16, 20, 43, 47]. Note that, while ResNet101 and ResNeXt29 16x64d networks achieve 20.00% and 17.25% error respectively, ResNet50 with BAM and ResNeXt29 8x64d with BAM achieve 20.00% and 16.71% error respectively using only half of the parameters. It suggests that our module BAM can efficiently raise the capacity of networks with a fewer number of network parameters. Thanks to our light-weight design, the overall parameter and computational overheads are trivial.

4.3 Classification Results on ImageNet-1K

The ILSVRC 2012 classification dataset [10] 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 [15, 16] for training and apply a single-crop evaluation with the size of 224×224 at test time. Following [15, 16, 21], we report classification errors on the validation set. ImageNet classification benchmark is one of the largest and most complex image classification benchmark, and we show the effectiveness of BAM in such a general and complex task. We use the baseline networks of ResNet [15], WideResNet [47], and ResNeXt [43] which are used for ImageNet classification task. More details are included in the supplementary material.

As shown in Table 3, the networks with BAM outperform all the baselines once again, demonstrating that BAM can generalize well on various models in the large-scale dataset. Note that the overhead of parameters and computation is negligible, which suggests that the proposed module BAM can significantly enhance the network capacity efficiently. Another

Architecture	mAP@.5	mAP@.75	mAP@[.5,.95]
ResNet101[15]	48.4	30.7	29.1
ResNet101 + BAM	**50.2**	**32.5**	**30.4**

(a) MS-COCO detection task

BackBone	Detector	mAP@.5	params(M)
VGG16[40]	SSD [33]	77.8	26.5
VGG16[40]	StairNet [39]	78.9	32.0
VGG16[40]	**StairNet [39] + BAM**	**79.3**	32.1
MobileNet[18]	SSD [33]	68.1	5.81
MobileNet[18]	StairNet [39]	70.1	5.98
MobileNet[18]	**StairNet [39] + BAM**	**70.6**	6.00

(b) VOC 2007 detection task

* all results are reproduced in the PyTorch framework.

Table 4: Experiments on detection tasks: MS-COCO and VOC 2007. Object detection mAP(%) is reported. We adopt ResNet101-Faster-RCNN as our baseline for MS-COCO, and adopt StairNet [39] for VOC 2007.

notable thing is that the improved performance comes from placing only three modules overall the network. Due to space constraints, ***further analysis and visualizations*** for success and failure cases of BAM are included in the supplementary material.

4.4 Effectiveness of BAM with Compact Networks

The main advantage of our module is that it significantly improves performance while putting trivial overheads on the model/computational complexities. To demonstrate the advantage in more practical settings, we incorporate our module with compact networks [18, 22], which have tight resource constraints. Compact networks are designed for mobile and embedded systems, so the design options have computational and parametric limitations.

As shown in Table 3, BAM boosts the accuracy of all the models with little overheads. Since we do not adopt any squeezing operation [18, 22] on our module, we believe there is more room to be improved in terms of efficiency.

4.5 MS COCO Object Detection

We conduct object detection on the Microsoft COCO dataset [32]. According to [4, 33], we trained our model using all the training images as well as a subset of validation images, holding out 5,000 examples for validation. We adopt *Faster-RCNN* [37] as our detection method and ImageNet pre-trained ResNet101 [15] as a baseline network. Here we are interested in improving performance by plugging BAM to the baseline. Because we use the same detection method of both models, the gains can only be attributed to our module BAM. As shown in the Table 4, we observe significant improvements from the baseline, demonstrating generalization performance of BAM on other recognition tasks.

4.6 VOC 2007 Object Detection

We further experiment BAM on the PASCAL VOC 2007 detection task. In this experiment, we apply BAM to the detectors. We adopt the StairNet [39] framework, which is one of the strongest multi-scale method based on the SSD [33]. We place BAM right before every classifier, refining the final features before the prediction, enforcing model to adaptively select only the meaningful features. The experimental results are summarized in Table 4. We can clearly see that BAM improves the accuracy of all strong baselines with two backbone networks. Note that accuracy improvement of BAM 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 [18] again shows that BAM can be an interesting method to low-end devices.

Architecture	Params	GFLOPs	Error	Architecture	Params	GFLOPs	Error
ResNet50	23.71M	1.22	21.49	WideResNet28 (w=8)	23.40M	3.36	20.40
ResNet50 + SE[19]	26.24M	1.23	20.72	WideResNet28 (w=8) + SE[19]	23.58M	3.36	19.85
ResNet50 + BAM	24.07M	1.25	**20.00**	WideResNet28 (w=8) + BAM	23.42M	3.37	**19.06**
PreResNet110	1.73M	0.245	22.22	ResNext29 16x64d	68.25M	9.88	17.25
PreResNet110 + SE[19]	1.93M	0.245	**21.85**	ResNext29 16x64d + SE[19]	68.81M	9.88	16.52
PreResNet110 + BAM	1.73M	0.246	21.96	ResNext29 16x64d + BAM	68.34M	9.9	**16.39**

* all results are reproduced in the PyTorch framework.

Table 5: **BAM v.s. SE [19].** CIFAR-100 experiment results. Top-1 errors are reported.

4.7 Comparison with Squeeze-and-Excitation[19]

We conduct additional experiments to compare our method with SE in CIFAR-100 classification task. Table 5 summarizes all the results showing that BAM outperforms SE in most cases with fewer parameters. Our module requires slightly more GFLOPS but has much less parameters than SE, as we place our module only at the bottlenecks not every conv blocks.

5 Conclusion

We have presented the bottleneck attention module (BAM), a new approach to enhancing the representation power of a network. Our module learns what and where to focus or suppress efficiently through two separate pathways and refines intermediate features effectively. Inspired by a human visual system, we suggest placing an attention module at the bottleneck of a network which is the most critical points of information flow. To verify its efficacy, we conducted extensive experiments with various state-of-the-art models and confirmed that BAM outperforms all the baselines on three different benchmark datasets: CIFAR-100, ImageNet-1K, VOC2007, and MS COCO. In addition, we visualize how the module acts on the intermediate feature maps to get a clearer understanding. Interestingly, we observed hierarchical reasoning process which is similar to human perception procedure. We believe our findings of adaptive feature refinement at the bottleneck is helpful to the other vision tasks as well.

References

[1] Pytorch. http://pytorch.org/. Accessed: 2018-04-20.

[2] Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. Multiple object recognition with visual attention. 2014.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. 2014.

[4] Sean Bell, C Lawrence Zitnick, Kavita Bala, and Ross Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2016.

[5] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016.

[6] Long Chen, Hanwang Zhang, Jun Xiao, Liqiang Nie, Jian Shao, and Tat-Seng Chua. Sca-cnn: Spatial and channel-wise attention in convolutional networks for image captioning. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2017.

[7] François Chollet. Xception: Deep learning with depthwise separable convolutions. *arXiv preprint arXiv:1610.02357*, 2016.

[8] Maurizio Corbetta and Gordon L Shulman. Control of goal-directed and stimulus-driven attention in the brain. In *Nature reviews neuroscience 3.3*, 2002.

[9] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. *CoRR, abs/1703.06211*, 1(2):3, 2017.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2009.

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[12] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. 2015.

[13] Dongyoon Han, Jiwhan Kim, and Junmo Kim. Deep pyramidal residual networks. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 6307–6315. IEEE, 2017.

[14] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2015.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2016.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Proc. of European Conf. on Computer Vision (ECCV)*, 2016.

[17] Joy Hirsch and Christine A Curcio. The spatial resolution capacity of human foveal retina. *Vision research*, 29(9):1095–1101, 1989.

[18] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[19] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. *arXiv preprint arXiv:1709.01507*, 2017.

[20] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993*, 2016.

[21] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *Proc. of European Conf. on Computer Vision (ECCV)*, 2016.

[22] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[23] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. 2015.

[24] Laurent Itti, Christof Koch, and Ernst Niebur. A model of saliency-based visual attention for rapid scene analysis. In *IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI)*, 1998.

[25] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Proc. of Neural Information Processing Systems (NIPS)*, 2015.

[26] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015.

[27] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. In *Advances in Neural Information Processing Systems*, pages 667–675, 2016.

[28] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[29] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images.

[30] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. of Neural Information Processing Systems (NIPS)*, 2012.

[31] Hugo Larochelle and Geoffrey E Hinton. Learning to combine foveal glimpses with a third-order boltzmann machine. In *Proc. of Neural Information Processing Systems (NIPS)*, 2010.

[32] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Proc. of European Conf. on Computer Vision (ECCV)*, 2014.

[33] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Proc. of European Conf. on Computer Vision (ECCV)*, 2016.

[34] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2015.

[35] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention." advances in neural information processing systems. In *Proc. of Neural Information Processing Systems (NIPS)*, 2014.

[36] Hyeonseob Nam, Jung-Woo Ha, and Jeonghee Kim. Dual attention networks for multimodal reasoning and matching. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2156–2164, 2017.

[37] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proc. of Neural Information Processing Systems (NIPS)*, 2015.

[38] Ronald A Rensink. The dynamic representation of scenes. In *Visual cognition 7.1-3*, 2000.

[39] Woo Sanghyun, Hwang Soonmin, and Kweon In So. Stairnet: Top-down semantic aggregation for accurate one shot detection. In *Proc. of Winter Conf. on Applications of Computer Vision (WACV)*, 2018.

[40] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[41] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2015.

[42] Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. Residual attention network for image classification. *arXiv preprint arXiv:1704.06904*, 2017.

[43] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *arXiv preprint arXiv:1611.05431*, 2016.

[44] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. 2015.

[45] Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. Stacked attention networks for image question answering. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2016.

[46] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. 2015.

[47] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[48] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[49] Yousong Zhu, Chaoyang Zhao, Jinqiao Wang, Xu Zhao, Yi Wu, and Hanqing Lu. Couplenet: Coupling global structure with local parts for object detection. In *Proc. of IntâĂŹl Conf. on Computer Vision (ICCV)*, 2017.