Noname manuscript No.
(will be inserted by the editor)

A Simple and Light-weight Attention Module for Convolutional Neural Networks

Jongchan Park* · **Sanghyun Woo*** · **Joon-Young Lee** · **In So Kweon**

Received: date / Accepted: date

Abstract Many aspects of deep neural networks, such as depth, width, or cardinality, have been studied to strengthen the representational power. In this work, we study the effect of *attention* in convolutional neural networks and present our idea in a simple self-contained module, called *Bottleneck Attention Module* (BAM). Given an intermediate feature map, BAM efficiently produces the attention map along two factorized axes, *channel* and *spatial*, with negligible overheads. BAM is placed at *bottlenecks* of various models where the downsampling of feature maps occurs, and is jointly trained in an end-to-end manner. Ablation studies and extensive experiments are conducted in CIFAR-100/ImageNet classification, VOC2007/MS-COCO detection, super resolution and scene parsing with various architectures including mobile-oriented networks. BAM shows consistent improvements over all experiments, demonstrating the wide applicability of BAM. The code and models are publicly available.

Keywords attention mechanism · deep learning

Jongchan Park
Lunit, 175 Yeoksam-Ro, Gangnam-Gu, Seoul, Korea
Tel.: +82 2-21382702
E-mail: jcpark@lunit.io

Sanghyun Woo and In So Kweon
#211, N1, KAIST, Yuseong-Gu, Daejeon, Korea
Tel.: +82-42-350-5465
E-mail: {shwoo93,iskweon77}@kaist.ac.kr

Joon-Young Lee
Adobe Research, 345 Park Ave, San Jose, CA 95110
Tel.: +1 408-536-5928
E-mail: jolee@adobe.com

* Both authors have equally contributed.

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 (Zeiler, 2012; Kingma and Ba, 2014), proposing adversarial training scheme (Goodfellow et al., 2014), or task-specific meta architecture such as 2-stage architectures (Ren et al., 2015) for object detection.

However, fundamental approach to boost performance is to design a good backbone architecture. Since the very first large-scale deep neural network AlexNet (Krizhevsky et al., 2012), various backbone architectures such as VGGNet (Simonyan and Zisserman, 2015), GoogLeNet (Szegedy et al., 2015), ResNet (He et al., 2016b), DenseNet (Huang et al., 2017), have been proposed. All those have their own design choices, and shown significant performance boosts over the precedent architectures.

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 (Simonyan and Zisserman, 2015) and ResNet (He et al., 2016a) precisely follows this. Compared to AlexNet, VGGNet has twice more layers. Furthermore, ResNet has 22x more layers than VGGNet with improved gradient flow by adopting residual connections. GoogLeNet (Szegedy et al., 2015), which is also very deep, uses concatenation of features with various filter sizes at each convolutional block. The use of diverse features

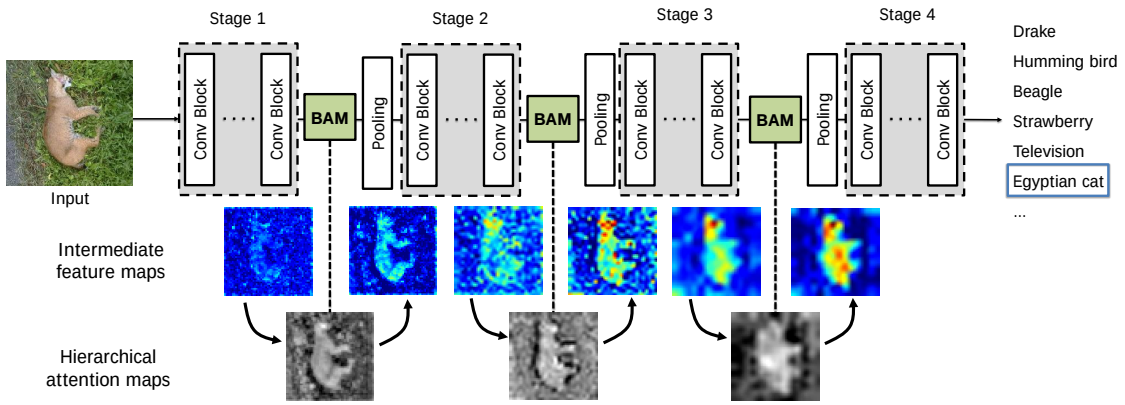


Fig. 1: **BAM integrated with a general CNN architecture.** As illustrated, BAM is placed at every *bottleneck* of the network. Interestingly, we observe sequential BAMs construct hierarchical attention maps which is similar to the human perception procedure. BAM denoises low-level features such as background 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 Fig. 5, Fig. 6.

at the same layer shows increased performance, resulting in powerful representation. DenseNet (Huang et al., 2017) 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 (Zagoruyko and Komodakis, 2016) shows that using more channels, *wider* convolutions, can achieve higher performance than naively deepening the networks. Similarly, PyramidNet (Han et al., 2017) shows that increasing channels in deeper layers can effectively boost the performance. Recent approaches with grouped convolutions, such as ResNeXt (Xie et al., 2017) or Xception (Chollet, 2017), 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 (Howard et al., 2017), 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 (Mnih et al., 2014; Ba et al., 2015; Bahdanau et al., 2014; Xu et al., 2015; Gregor et al., 2015; Jaderberg et al., 2015a). 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. We validate the performance improvement of object detection on the VOC 2007 and MS COCO datasets. We further apply bam to the pixel-level prediction tasks; super resolution and scene parsing and show consistent performance improvement over the baselines, demonstrating a wide applicability of BAM. Since we have carefully designed our module to be light-weight, parameter and computational overheads are negligible.

In short, we investigate the effect of attention with the proposed module BAM. BAM is a simple self-contained module to be inserted at any feed-forward convolutional neural networks without bells and whis-

tles. We extensively validate several design choices via ablation studies, and demonstrate the effectiveness of BAM in various vision tasks including classification, detection, segmentation, and super resolution. Moreover, we analyze and explain the difference between the baseline and the BAM-integrated network in terms of class-selectivity index (Morcos et al., 2018). Finally, we analyze the effect of attention with visualizations.

2 Related Works

A number of studies (Itti et al., 1998; Rensink, 2000; Corbetta and Shulman, 2002) 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 (Hirsch and Curcio, 1989). In order to efficiently and adaptively process visual information, human visual systems iteratively process spatial glimpses and focus on salient areas (Larochelle and Hinton, 2010).

Cross-modal attention. Attention mechanism is a widely-used technique in multi-modal settings, especially when certain modalities should be conditioned on the 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 (Yang et al., 2016), 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 (Nam et al., 2017). 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 (Wang et al., 2017; Hu et al., 2018b,a) 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 hour-glass 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. Recently, the Gather-Excite framework by Hu et al. (2018a) further improved this approach by replacing the global average pooling with depth-wise convolution, enhancing a gathering operation in the attention module. However, the method still misses the *spatial* axis, which is also an important factor in inferring accurate attention map.

SCA-CNN (Chen et al., 2017b) and HANet (Li et al., 2018) have shown that using both the spatial and the channel attention is effective for image captioning and person re-identification tasks respectively. Here, we carefully design a module that outputs both the spatial and the channel attention maps for image classification tasks. Our method greatly reduces the heavy computation of 3D attention map inference (Wang et al., 2017) and improves the baseline significantly. We also investigate the effective point to place the module that is before the pooling occurs (see Fig. 1). Recently proposed CBAM method (Woo et al., 2018b) is an extended version of BAM. It improves on BAM with their module design and placement (i.e., the modules are placed at every convolution block). However, it introduces much more parameter overhead than BAM.

Adaptive modules. Several previous works use adaptive modules that dynamically changes their output according to their inputs. Dynamic Filter Network (Jia et al., 2016) proposes to generate convolutional features based on the input features for flexibility. Spatial Transformer Network (Jaderberg et al., 2015b) 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 (Dai et al., 2017) 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 light-weight design. Furthermore, we find an efficient location to put our module - *bottleneck* of the network.

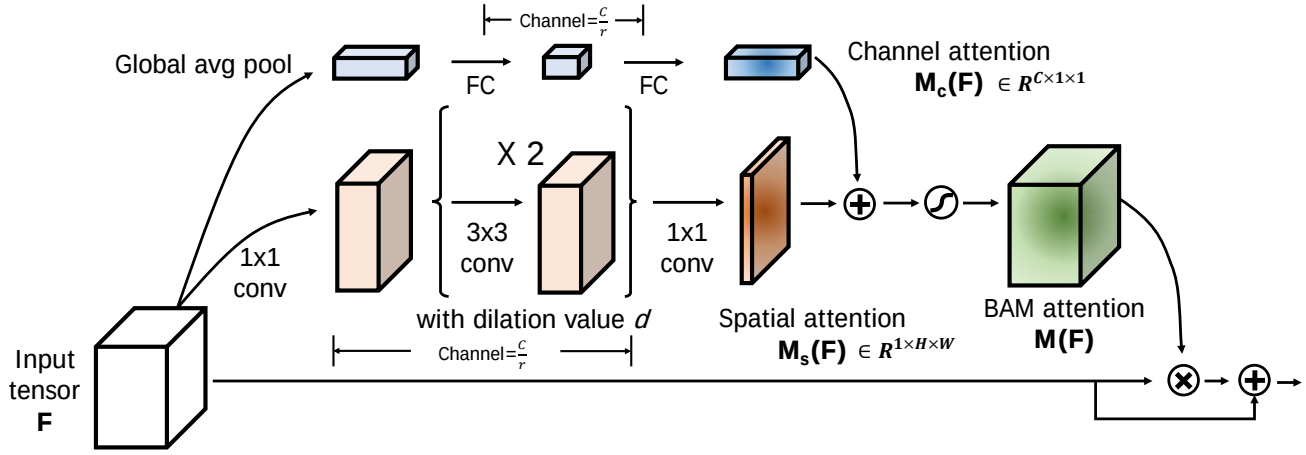


Fig. 2: **Detailed module architecture.** Given the intermediate feature map $\mathbf{F}$, the module computes corresponding attention map $\mathbf{M(F)}$ through the two separate attention branches – channel $\mathbf{M_c}$ and spatial $\mathbf{M_s}$. Two intermediate tensors from channel and spatial branches are properly broadcasted to match the final tensor shape.t 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. 5.1), we set $\{\mathrm{d} = 4, \mathrm{r} = 16\}$.

3 Bottleneck Attention Module

We design a module that learns spatial (where) and channel-wise (what) attention separately. The intuition behind the factorization is that those two attentions have different properties. Thus, separation can make them focus on their own objectives more clearly.

It is well known that each channel of the feature maps corresponds to a certain visual pattern (Simon and Rodner, 2015; Zhang et al., 2016). Therefore, estimating and applying the channel-wise attention can be viewed as a process of picking up the necessary semantic attributes for the target task. The spatial attention, on the other hand, attempts to select the important spatial locations rather than considering each image region equally. Thus, it can be seen as a clutter removal that is quite different from the channel-attention. Therefore, it is obvious that using these two complementary attentions in combination is crucial for many classification tasks, and we empirically confirm that it provides the best result in Table 1b.

We implement the attention map generation of each branch to be highly efficient. For the channel attention, we squeeze the spatial axis using global average pooling. We then regress the channel attention using two fully connected layers. For the spatial attention, we gradually reduce the channel dimension to be 1 at the final. Here, we adopt the atrous convolution to enlarge the receptive field and effectively decide spatially important part.

The overall 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 $\mathbf{F_c} \in \mathbb{R}^C$. 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 $\mathbb{R}^{C/r}$, where r is the reduction ratio. After the MLP, we add a batch normalization (BN) layer (Ioffe

and Szegedy, 2015) 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 (Yu and Koltun, 2016; Long et al., 2015; Bell et al., 2016; Hariharan et al., 2015) utilizing contextual information is crucial to know which spatial locations should be focused on. It is important to have a large receptive field to effectively leverage contextual information. We employ the dilated convolution (Yu and Koltun, 2016) 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. 5.1). The "bottleneck structure" suggested by ResNet (He et al., 2016a) 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 (He et al., 2016a). We empirically verify that element-wise summation results in the best performance among three options (see Sec. 5). 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 $\mathbf{F}$ then is added upon the original input feature map to acquire the refined feature map $\mathbf{F'}$ as Eq. (1).

Module placement. As BAM is a self-contained module, it can be placed at any point of the network. Through ablation experiments in Table 2, we empirically found that the best location for BAM is the bottlenecks (i.e. right before spatial pooling).

4 Benefits of Using Self-Attention

The two main advantages of using self-attention mechanism in the CNN are: 1) efficient global context modeling, and 2) effective back-propagation (i.e., model training).

The global context allows the model to better recognize patterns that would be locally ambiguous and to attend on important parts. Therefore, capturing and utilizing the global context is crucial for various vision tasks. In this respect, CNN models typically stack many convolution layers or use pooling operations to ensure the features to have a large receptive field. Although doing so provides the model to equip with the global view at the end, there are several drawbacks. First, naively stacking the convolution layers significantly increases the space (i.e., parameters) and time (i.e., computational overheads) complexities. Second, the features at lower layers still have limited receptive fields. On the other hand, our proposed method BAM alleviates the above issues nicely. Specifically, a small meta-network (or module) is designed to refine the input feature map based on its global feature statistics. The module is placed at the bottlenecks of the model, making lower layer features to benefit from the contextual information. The overall procedure operates in a highly efficient manner thanks to the light-weight module design. We empirically verify that using BAM is more effective than simply deepening the models (i.e., using more convolutions) as shown in Table 1c.

Moreover, our method eases model optimization. In particular, the predicted attention map modulates the training signal (i.e., gradients) to focus on more important regions (Wang et al., 2017). We formulate the

Independent Variables	Value	Params	Error
Dilation value (d)	1	34.61	17.24
	2	34.61	16.92
	4	34.61	**16.71**
	6	34.61	16.97
Reduction ratio (r)	4	35.14	16.88
	8	34.74	17.14
	16	34.61	**16.71**
	32	34.56	16.92
Base (ResNeXt29 8x64d)	-	34.52	18.18

a Experiments on hyper-params

	Base							**BAM**
Channel		✓		✓	✓	✓	✓	✓
Spatial			✓	✓	✓	✓	✓	✓
$\sigma(max(C,S))$				✓				
$\sigma(C*S)$					✓			
$\sigma(C)+\sigma(S)$						✓		
$\sigma(C+S)$							✓	✓
No identity							✓	
Error	18.18	16.82	17.00	17.44	17.55	17.02	16.89	**16.71**

b Experiments on each branch

ConvBlock vs BAM	Params	Error
ResNet50	23.68M	21.49
+ ResBlock	25.14M	21.02
+ **BAM**	**24.07M**	**20**
WideResNet28 (w=8)	23.4M	20.4
+ WideResBlock	24.88M	19.51
+ **BAM**	**23.56M**	**19.06**
ResNeXt 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 in CIFAR100 benchmark. **(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. All the experiments are reproduced in PyTorch.

attentioning process as follows:

$$\mathbf{F}' = (1 + \mathbf{M}(\mathbf{F}))\mathbf{F}(x, \phi), \tag{5}$$

where ϕ is the parameters of the feature extractor. Then, the gradient can be computed as:

$$\frac{\partial \mathbf{M}(F)\mathbf{F}(x,\phi)))}{\partial \phi} = \mathbf{M}(\mathbf{F})\frac{\partial \mathbf{F}(x,\phi))}{\partial \phi} \tag{6}$$

The equation indicates that the higher the attention value (important regions), the greater the gradient value flows in there.

5 Experiments

In this section, we empirically verify the design choices of BAM, and show the efficacy of BAM across architectures and tasks. We conduct extensive experiments on the standard benchmarks: CIFAR-100 (Sec. 5.1 and 5.2), ImageNet-1K (Sec. 5.3 and 5.4) for image classification; VOC 2007 (Sec. 5.6), MS COCO (Sec. 5.5) for object detection; Set5 and Set14 (Sec. 5.7) for super resolution; ADE20K (Sec. 5.8) for scene parsing.

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 (He et al., 2016a; Zagoruyko and Komodakis, 2016; Xie et al., 2017; Huang et al., 2017). When training the baseline models (or BAM-integrated models), we follow their training schemes (i.e., hyper-parameter settings), if not otherwise specified. Throughout all experiments, 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.

[1] `https://pytorch.org/`

5.1 Ablation studies on CIFAR-100

The CIFAR-100 dataset (Krizhevsky and Hinton, 2009) 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 1a, 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 1a 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 (Yu and Koltun, 2016; Long et al., 2015; Bell et al., 2016; Chen et al., 2016; Zhu et al., 2017). 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 1a, 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 1a, we set the dilation value as 4 and the reduction ratio as 16 in the following experiments.

Separate or Combined branches. In Table 1b, 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 1b, 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 (Larochelle and Hinton, 2010; Chen et al., 2017a).

Combining methods. We also explore *four* different combining strategies: *maximum-and-sigmoid*, *product-and-sigmoid*, *sum-and-sigmoid*, and *sigmoid-and-sum*. Table 1b summarizes the result of them. We empirically confirm that *sum-and-sigmoid* achieves the best performance. In terms of the information flow, the *sum-and-sigmoid* 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. Product-and-sigmoid, which can assign a large gradient to the small input, makes the network hard to converge, yielding the inferior performance. Maximum-and-sigmoid, 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. *Sigmoid-and-sum* still improves over the baseline, but is worse than other combining options. The main difference with the *sum-and-sigmoid* lies on where we place the sigmoid operation. Applying the sigmoid to each branch before element-wise summation may affect the original feature representation (i.e., restricting the feature value range between 0 and 1) and may affect the gradient updates. Though, note that all of four different implementations outperform the baselines. This implies that utilizing both stream is important while the best-combining strategy further boosts the final performance.

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

Identity connection. In the early stage of the training, the BAM might produce inaccurate attention map which may negatively affect both the forward and backward (i.e., back propagation) information flow. Therefore, by introducing the residual connection, we are able to alleviate the possibly detrimental initial behavior of the model, easing the overall model training. Note that the attention value now ranges from 1 to 2 instead of 0 to 1. However, the relative importance is still maintained. We empirically verify that residual connection indeed is effective in Table 1b.

Comparison with placing original convblocks. It is widely know that larger networks with more parameters have better performances. Although BAM introduces negligible overheads, it does bring some extra layers to the networks. 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 1c. 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. Bottleneck is where the feature downsampling occurs. For example, pooling operations or convolutions with stride larger than 1. Specifically, we place BAM right before the downsampling. Recent studies on attention mechanisms (Hu et al., 2018b; Wang et al., 2017) mainly focus

on modifications within the '*convolution blocks*' rather than the '*bottlenecks*'. We compare those two different locations by using various models on CIFAR-100. In the BAM-C ('convolution blocks') case, we place BAM in every convolutional block, so there is much more overhead. 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 (He et al., 2016b).

5.2 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 (He et al., 2016a,b; Zagoruyko and Komodakis, 2016; Xie et al., 2017; Huang et al., 2017). 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.

5.3 Classification Results on ImageNet-1K

The ILSVRC 2012 classification dataset (Deng et al., 2009) 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 (He et al., 2016a,b) for training and apply a single-crop evaluation with the size of 224×224 at test time. Following (He et al., 2016a,b; Huang et al., 2016), 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 (He et al., 2016a), WideResNet (Zagoruyko and Komodakis, 2016), and ResNeXt (Xie et al., 2017) which are used for ImageNet classification task. More details are included in the supplementary material.

As shown in Table 4, 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 notable thing is that the improved performance comes from placing only three modules overall the network.

5.4 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 (Howard et al., 2017; Iandola et al., 2016), 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 4, BAM boosts the accuracy of all the models with little overheads. Since we do not adopt any squeezing operation (Howard et al., 2017; Iandola et al., 2016) on our module, we believe there is more room to be improved in terms of efficiency.

5.5 MS COCO Object Detection

We conduct object detection on the Microsoft COCO dataset (Lin et al., 2014). According to (Bell et al., 2016; Liu et al., 2016), 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* (Ren et al., 2015) as our detection method and ImageNet pre-trained ResNet101 (He et al., 2016a) 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 6, we observe significant improvements from the baseline, demonstrating generalization performance of BAM on other recognition tasks.

In Table 5, we compute mAP over different IoU thresholds and coco object size criteria (Lin et al., 2014). We confirm that the performance enhancement is not at a certain threshold but in overall. Note that the relative improvement, which we define as accuracy improvement over the baseline performance, are amplified at higher IoU thresholds, demonstrating that attention module is effective for accurate bounding box prediction. The BAM also improves the baseline model over all the different object sizes rather than improving it at a specific object size.

architecture	original	re-implement			with BAM		
	error	error	params	GFLOPs	error	params	GFLOPS
ResNet 50	-	21.49	23.71M	1.22	20.00	24.07M(+0.36)	1.25(+0.03)
ResNet 101	-	20.00	42.70M	2.44	**19.61**	43.06M(+0.36)	2.46(+0.02)
PreResNet 110	-	22.22	1.726M	0.245	**21.96**	1.733M(+0.007)	0.246(+0.01)
WideResNet 28 (w=8)	-	20.40	23.40M	3.36	**19.06**	23.42M(+0.02)	3.37(+0.01)
WideResNet 28 (w=10)	18.85	18.89	36.54M	5.24	**18.56**	36.57M(+0.03)	5.25(+0.01)
WideResNet 40 (w=10)	18.30	18.29	55.90M	8.07	**18.17**	55.94M(+0.04)	8.08(+0.01)
ResNeXt 29 8x64d	17.77	18.18	34.52M	4.99	**16.71**	34.61M(+0.09)	5.00(+0.01)
ResNeXt 29 16x64d	17.31	17.25	68.25M	9.88	**16.39**	68.34M(+0.09)	9.90(+0.02)
DenseNet 100-BC (k=12)	22.27	21.95	0.8M	0.29	**20.65**	0.84M(+0.04)	0.30(+0.01)

Table 3: Experiments on image classification tasks: CIFAR-100 classification. The numbers inside the parentheses indicate the parameter/computational overhead. w denotes the widening factor in WideResNet (Zagoruyko and Komodakis, 2016) and k denotes the growth rate in DenseNet (Huang et al., 2017). For the DenseNet (Huang et al., 2017), we put our module back and forth of the transition block.

architecture	original		re-implement				with BAM			
	top-1 err.	top-5 err.	top-1	top-5	params	GFLOPs	top-1	top-5	params	GFLOPS
ResNet18	-	-	29.60	10.55	11.69M	1.81	**28.88**	**10.01**	11.71M(+0.02)	1.82(+0.01)
ResNet50	24.7	7.8	24.56	7.50	25.56M	3.86	**24.02**	**7.18**	25.92M(+0.36)	3.94(+0.08)
ResNet101	23.6	7.1	23.38	6.88	44.55M	7.57	**22.44**	**6.29**	44.91M(+0.36)	7.65(+0.08)
WideResNet18 (widen=1.5)	27.06	9.0	26.85	8.88	25.88M	3.87	**26.67**	**8.69**	25.93M(+0.05)	3.88(+0.01)
WideResNet18 (widen=2.0)	25.58	8.06	25.63	8.20	45.62M	6.70	**25.00**	**7.81**	45.71M(+0.09)	6.72(+0.02)
ResNeXt50 (32x4d)	22.2	-	22.35	6.01	25.03M	3.77	21.92	5.89	25.39M(+0.36)	3.85(+0.08)
MobileNetV2 (Sandler et al., 2018)	28.0	-	28.27	9.63	3.505M	0.300	27.50	9.35	3.516M(+0.011)	0.307(+0.007)
SqueezeNet v1.1	-	19.7	40.86	18.88	1.24M	0.290	39.82	17.68	1.26M(+0.02)	0.304(+0.014)

Table 4: Experiments on image classification tasks: ImageNet 1K classification. The numbers inside the parentheses indicate the parameter/computational overhead. w denotes the widening factor in WideResNet (Zagoruyko and Komodakis, 2016). ResNet (He et al., 2016a) results are obtained from the Github page https://github.com/Kaiminghe/deep-residual-networks. SqueezeNet v1.1 (Iandola et al., 2016) result is obtained from the Github page https://github.com/DeepScale/SqueezeNet/tree/master/SqueezeNet_v1.1

Architecture	mAP@0.5	@0.55	@0.60	@0.65	@0.70	@0.75	@0.80	@0.85	@0.90	@0.95	small	medium	large
ResNet101	48.4	45.9	43.4	40	35.7	30.7	24.1	15.6	6.5	0.59	11.5	33.2	44.3
ResNet101 + BAM	**50.2**	**47.5**	**44.7**	**41.7**	**37.4**	**32.5**	**25.5**	**17.0**	**7.3**	**0.76**	**12.6**	**34.6**	**46.4**
Relative improvement	+3.7%	+3.5%	+3.0%	+4.3%	+4.8%	+5.7%	+5.8%	+9.0%	+12.3%	+28.8%	9.6%	4.2%	+4.7%

Table 5: **Detailed MS COCO detection results.** The *small*, *medium*, and *large* indicates mAP over different IoU thresholds from 0.5 to 0.95 which are computed based on a coco object size criteria (Lin et al., 2014).

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

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

Table 6: **MS COCO detection results.** Object detection mAP(%) is reported. We adopt Faster-RCNN(Ren et al., 2015) as our detection method and ImageNet pre-trained ResNet101 (He et al., 2016a) as baseline network.

BackBone	Detector	mAP@.5	params(M)
VGG16	SSD	77.8	26.5
VGG16	StairNet*	78.8	32.0
VGG16	StairNet	78.9	32.0
VGG16	**StairNet + BAM**	**79.3**	32.1
MobileNet	SSD	68.1	5.81
MobileNet	StairNet	70.1	5.98
MobileNet	**StairNet + BAM**	**70.6**	6.00

***** indicates the original paper performance.

Table 7: **VOC2007 detection test set results.** Object detection mAP(%) is reported. We adopt StairNet (Woo et al., 2018a) as our baseline.

5.6 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 (Woo et al., 2018a) framework, which is one of the strongest multi-

Architecture	SET5	SET14
SRResNet	31.5548/0.8901	28.2529/0.7807
SRResNet + BAM	**31.6688/0.8915**	**28.2827/0.7814**

Table 8: **Super Resolution experiments.** We compare the effect of BAM integrated ResNet with baseline SRResNet model.

Encoder	Decoder	mIoU	Pixel Accuracy
ResNet50	UperNet	0.3157	75.33%
ResNet50+BAM	UperNet	**0.3497**	**76.60%**

Table 9: **ADE20K scene parsing experiments.** We compare the effect of BAM integrated ResNet (He et al., 2016b) encoder with UperNet (Xiao et al., 2018) decoder. Single scale evaluation results are reported.

scale method based on the SSD (Liu et al., 2016). 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 7. 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 (Howard et al., 2017) again shows that BAM can be an interesting method to low-end devices.

5.7 Super Resolution

For the classification and detection tasks, CNNs are used to recognize single or multiple target in the given image respectively. We further explore the applicability of BAM in more challenging pixel-level prediction tasks. We first apply BAM in super resolution task. We set SRResNet (Ledig et al., 2017) as our baseline model and place one BAM module at every 4th ResBlock to construct SRResNet + BAM model. We perform experiments on two widely used benchmark datasets: Set5 and Set14. All experiments are performed with a scale factor of 4 between low- and high-resolution images. This corresponds to a total 16 reduction in image pixels. For fair comparison, all reported PSNR [dB] and SSIM scores were calculated on the y-channel of center-cropped images, removing a 4-pixel wide strip from each border. We use 2017 COCO train dataset (118k images) for training both baseline and BAM-integrated model. We follow the training details as given in the original paper (Ledig et al., 2017). We confirm that PSNR and SSIM scores of reproduced baseline match closely to the reported values.

As we can see in Table 8, BAM improves over the baseline performance in the super resolution task. Please note that BAM is proposed and ablated on semantic tasks, and is not optimized for pixel-level prediction task, but it still shows improvement over the baseline. We believe moderate changes to the design of

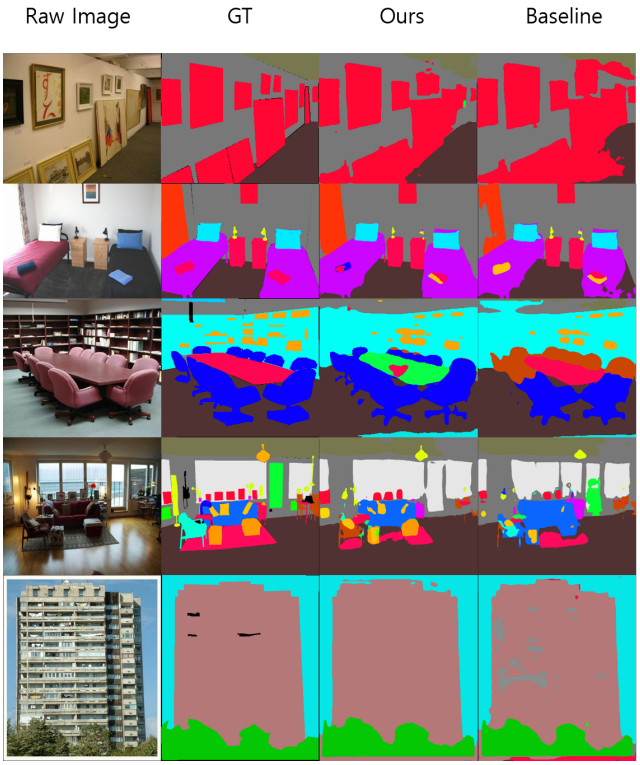


Fig. 3: **Qualitative evaluation on ADE20K validation set.** Several validation examples are shown above. Baseline is ResNet50(encoder) + UperNet, and ours is ResNet50 & BAM + UperNet. We can see that BAM induces the network to capture a finer object extent.

attention module can further improve the performance. Here, we focus on showing the attention process can be an effective solution for the pixel-level inference task.

5.8 ADE20K Scene Parsing

We now investigate the effectiveness of BAM in ADE20K scene parsing task (Zhou et al., 2019). We adopt a recent state-of-the-art architecture UperNet (Xiao et al., 2018) and place BAM to the encoder part. We use the official PyTorch code provided by the authors (Zhou, Bolei and Zhao, Hang and Puig, Xavier and Fidler, Sanja and Barriuso, Adela and Tor-

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

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

Table 10: **BAM v.s. SE (Hu et al., 2018b).** CIFAR-100 experiment results. Top-1 errors are reported.

ralba, Antonio, 2018). We use the encoder architecture of ResNet50, and the decoder architecture of UperNet. Following the default hyper-parameters (segmentation downsampling 4, padding 32).

The experiment results are summarized in Table 9. The results again shows that attention process is effective for pixel-level inference task. We also provide qualitative results in Fig. 3. We can see that BAM helps the model to capture a finer object extent such as boundary shape, edges, and small targets. We see that attention process enables contextual reasoning and provides strong global cue to resolve local ambiguities.

5.9 Comparison with Squeeze-and-Excitation

We conduct additional experiments to compare our method with SE in CIFAR-100 classification task. Table 10 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.

6 Analysis on the Effect of BAM

We have shown that BAM can improve the performance of a deep network for various vision tasks. Now, we provide in-depth analysis of how a BAM-integrated model (*i.e.,* ResNet50 + BAM) may differ from a vanilla baseline model (*i.e.,* ResNet50) in several aspects. We first explore the features of these models using a class selectivity index proposed by Morcos et al. (2018). Next, we provide visualization results of the attention process with regard to the case of when the BAM-integrated model succeeds in classification but the baseline fails. Finally, we investigate the channel attentions and the spatial attentions of the BAM-integrated model.

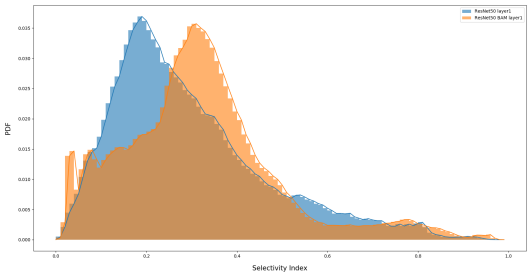
6.1 Class-selectivity index.

Class-selectivity is a neuro-science inspired metric proposed by Morcos et al. (2018). For each feature map, the metric computes the normalized difference between the highest class-conditional mean activity and the mean of all other classes over a given data distribution. The resulting value varies between zero and one, where zero indicates that the filter produced same value for every class (*i.e.,* feature re-use) and one indicates that a filter only activates for a single class. We compute the class-selectivity index for the features generated from two models (*i.e.,* ResNet50 with and without BAM). The distribution of class-selectivity is illustrated in Fig. 4.

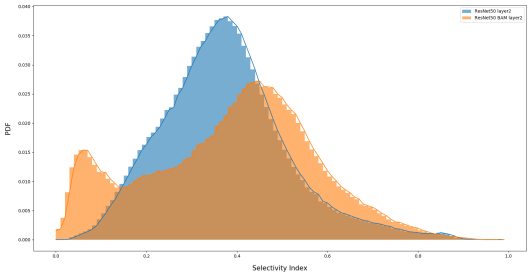
We observe a common underlying trend in both models: the class-selectivity increases gradually as the stages progress. It is well known that the filters of deep networks tend to extract class-agnostic features at the early stage (*i.e.,* low-level features) while class-specific features are extracted at the last stage. In contrast to the baseline model, at the stage 2 and 3, the distributions of class-selectivity for the BAM-integrated model appears to be separated. We conjecture that the attention module helps feature re-use within the network and prevents allocating highly specialized units. As a result, the sub-features of intermediate stages from the BAM-integrated model shows less class selectivity than the ResNet50 baseline (see Fig. 4).

6.2 Qualitative results.

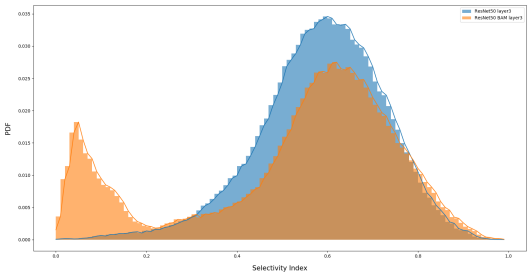
In Fig. 5, we visualize our attention maps and compare with the baseline feature maps for thorough analysis of accuracy improvement. We compare two models trained on ImageNet-1K: ResNet50 and ResNet50 + BAM. We select three examples that the baseline model fails to correctly classify while the model with BAM succeeds. We gather all the 3D attention maps at the bottlenecks and examine their distributions with respect to the channel and spatial axes respectively. For visualizing the 2D spatial attention maps, we averaged attention maps over the channel axis and resized them. All the 2D maps are normalized according to the global statistics at each stage computed from the whole ImageNet-1K training set. For visualizing the channel attention profiles, we averaged our attention map over the spatial axis and uniformly sampled 200 channels similar to Hu et al. (2018b).



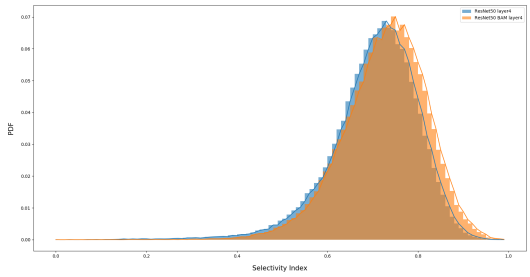
a Class-selectivity index at ResNet50 stage 1



b Class-selectivity index at ResNet50 stage 2



c Class-selectivity index at ResNet50 stage 3



d Class-selectivity index at ResNet50 stage 4

Fig. 4: Class-selectivity index plot of ResNet50 and ResNet50+BAM in ImageNet.

As shown in Fig. 5, we can observe that the module BAM drives the network to focus on the target gradually while the baseline model shows more scattered feature activations. Note that accurate targeting is important for the fine-grained classification, as the incorrect answers of the baseline are reasonable errors. At the first stage, we observe high variance along the channel axis and enhanced 2D feature maps after BAM. Since the theoretical receptive field size at the first bottleneck is 35, compared to the input image size of 224, the features contain only local information of the input. Therefore, the filters of attention map at this stage act as a local feature denoiser. We can infer that both channel and spatial attention contributes together to selectively refine local features, learning *what* ('channel') and *where* ('spatial') to *focus* or *suppress*. The second stage shows an intermediate characteristic of the first and final stages. At the final stage, the module generates binary-like 2D attention maps focusing on the target object. In terms of channel, the attention profile shows few spikes with low variance. We conjecture that this is because there is enough information about 'what' to focus at this stage. Even it is noisy, note that the features before applying the module show high activations around the target, indicating that the network already has a strong clue in what to focus on. By comparing the features of the baseline and before/after BAM, we verify that BAM accurately focuses on the target object while the baseline features are still scattered. The visualization of the overall attention process demonstrates the efficacy of BAM, which refines the features using two complementary attentions jointly to focus on more meaningful information. Moreover, the stage-by-stage gradual focusing resembles a hierarchical human perception process (Hubel and Wiesel, 1959; Riesenhuber and Poggio, 1999; Marr and Vision, 1982), suggesting that BAM drives the network to mimic the human visual system effectively.

6.3 Visualization Results

We show more visualization results of the attention process in Fig. 6 from ImageNet validation set. The listed samples are correctly classified by the BAM-integrated model ResNet50 + BAM, but incorrectly classified by the baseline model of ResNet50. The examples are listed with intermediate features and attention maps (averaged over channel axis for visualization). Starting from the early stage 1, we can clearly observe that the attention module acts as a feature denoiser, successfully suppressing much of the noise and highlighting on visually meaningful contents. Figures are best viewed in color.

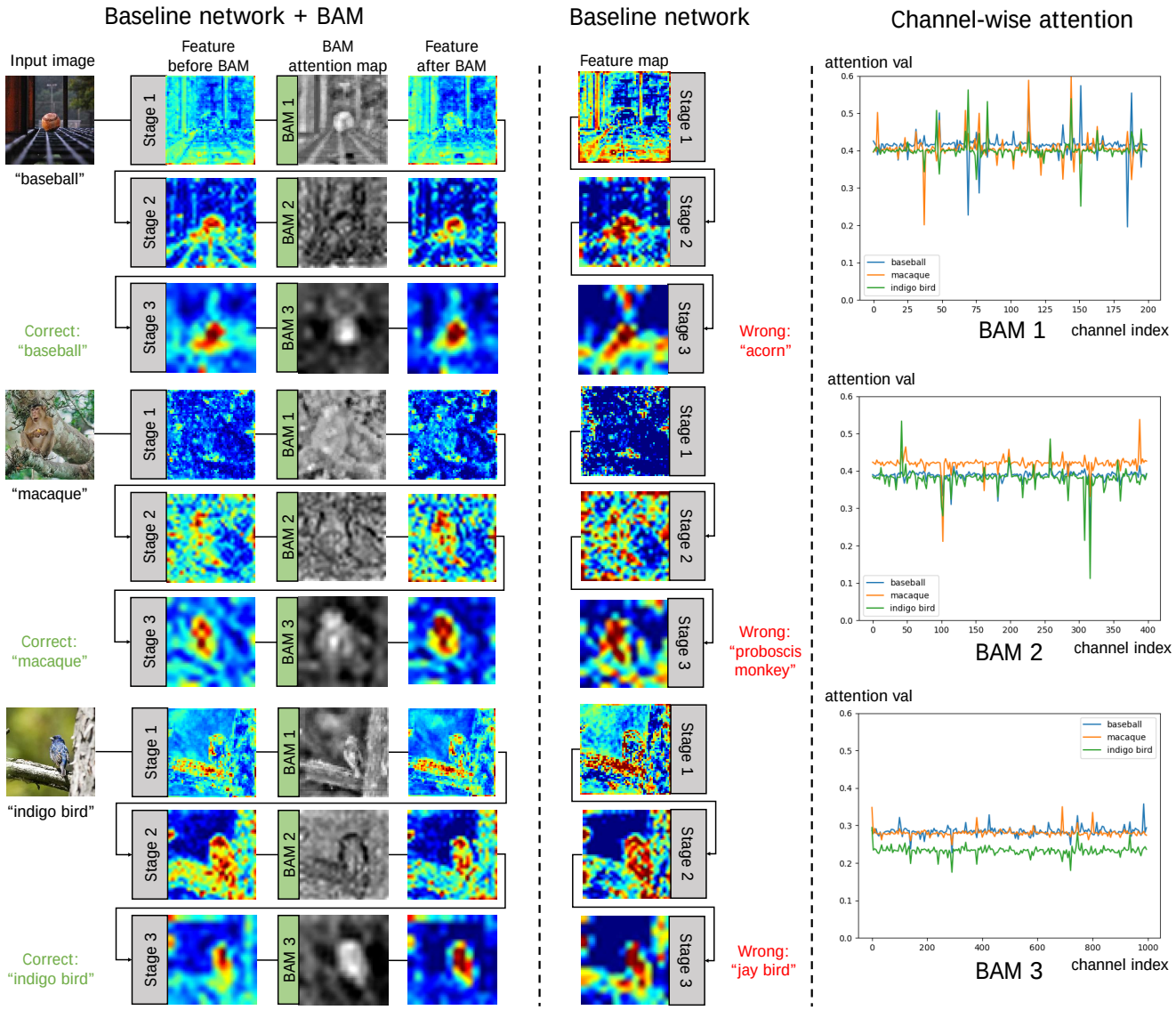


Fig. 5: **Visualizing the attention process of BAM.** In order to provide an intuitive understanding of BAM's role, we visualize image classification process using the images that baseline (ResNet50) fails to classify correctly while the model with BAM succeeds. Using the models trained on ImageNet-1K, we gather all the 3D attention maps from each bottleneck and examine their distribution spatially and channel-wise. We can clearly observe that the module BAM successfully drives the network to focus on the target while the baseline model fails.

7 Conclusion

In this work, we propose a simple and light-weight attention module, named *Bottleneck Attention Module*, to improve the performance of CNNs. BAM is a self-contained module composed of off-the-shelf CNN layers, so it can be easily implemented and added upon any CNN architectures. 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, and empirically verified it. To show its efficacy, we conducted extensive experiments with various state-of-the-art models and confirmed that BAM outperforms all the baselines on four different types of vision tasks: classification, detection, super-resolution, and scene parsing. Moreover, we analyze and visualize how the module acts on the intermediate feature maps to get a clearer understanding. We believe our findings of adaptive feature refinement at the bottleneck is helpful to the other vision tasks as well.

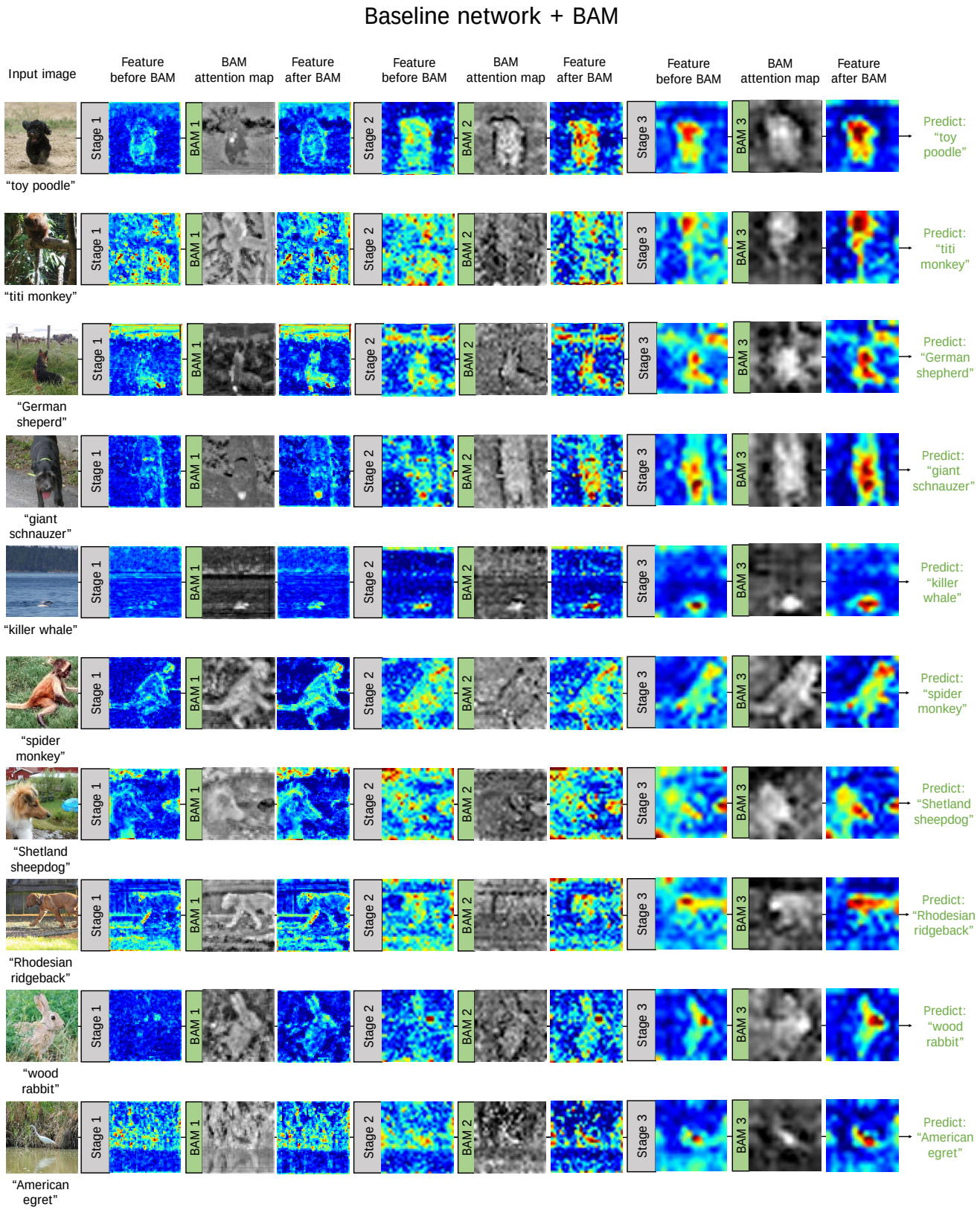


Fig. 6: **Successful cases with BAM.** The shown examples are the intermediate activations and BAM attention maps when the baseline+BAM succeeds and the baseline fails. Figure best viewed in color.

References

Ba J, Mnih V, Kavukcuoglu K (2015) Multiple object recognition with visual attention. In: Proc. of Int'l Conf. on Learning Representations (ICLR)

Bahdanau D, Cho K, Bengio Y (2014) Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:14090473

Bell S, Lawrence Zitnick C, Bala K, Girshick R (2016) Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In: Proc. of Computer Vision and Pattern Recognition (CVPR)

Chen L, Zhang H, Xiao J, Nie L, Shao J, Chua TS (2017a) Sca-cnn: Spatial and channel-wise attention in convolutional networks for image captioning. In: Proc. of Computer Vision and Pattern Recognition (CVPR)

Chen L, Zhang H, Xiao J, Nie L, Shao J, Liu W, Chua TS (2017b) Sca-cnn: Spatial and channel-wise attention in convolutional networks for image captioning. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 5659–5667

Chen LC, Papandreou G, Kokkinos I, Murphy K, Yuille AL (2016) Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. arXiv preprint arXiv:160600915

Chollet F (2017) Xception: Deep learning with depthwise separable convolutions. In: Proc. of Computer Vision and Pattern Recognition (CVPR)

Corbetta M, Shulman GL (2002) Control of goal-directed and stimulus-driven attention in the brain. In: Nature reviews neuroscience 3.3

Dai J, Qi H, Xiong Y, Li Y, Zhang G, Hu H, Wei Y (2017) Deformable convolutional networks. CoRR, abs/170306211 1(2):3

Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L (2009) Imagenet: A large-scale hierarchical image database. In: Proc. of Computer Vision and Pattern Recognition (CVPR)

Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial nets. In: Advances in neural information processing systems, pp 2672–2680

Gregor K, Danihelka I, Graves A, Rezende DJ, Wierstra D (2015) Draw: A recurrent neural network for image generation. In: Proc. of International Conference on Machine Learning (ICML)

Han D, Kim J, Kim J (2017) Deep pyramidal residual networks. In: Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on, IEEE, pp 6307–6315

Hariharan B, Arbeláez P, Girshick R, Malik J (2015) Hypercolumns for object segmentation and fine-grained localization. In: Proc. of Computer Vision and Pattern Recognition (CVPR)

He K, Zhang X, Ren S, Sun J (2016a) Deep residual learning for image recognition. In: Proc. of Computer Vision and Pattern Recognition (CVPR)

He K, Zhang X, Ren S, Sun J (2016b) Identity mappings in deep residual networks. In: Proc. of European Conf. on Computer Vision (ECCV)

Hirsch J, Curcio CA (1989) The spatial resolution capacity of human foveal retina. Vision research 29(9):1095–1101

Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H (2017) Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:170404861

Hu J, Shen L, Albanie S, Sun G, Vedaldi A (2018a) Gather-excite: Exploiting feature context in convolutional neural networks. In: Advances in Neural Information Processing Systems, pp 9422–9432

Hu J, Shen L, Sun G (2018b) Squeeze-and-excitation networks. In: Proc. of Computer Vision and Pattern Recognition (CVPR)

Huang G, Sun Y, Liu Z, Sedra D, Weinberger KQ (2016) Deep networks with stochastic depth. In: Proc. of European Conf. on Computer Vision (ECCV)

Huang G, Liu Z, Weinberger KQ, van der Maaten L (2017) Densely connected convolutional networks. In: Proc. of Computer Vision and Pattern Recognition (CVPR)

Hubel DH, Wiesel TN (1959) Receptive fields of single neurones in the cat's striate cortex. The Journal of physiology 148(3):574–591

Iandola FN, Han S, Moskewicz MW, Ashraf K, Dally WJ, Keutzer K (2016) Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size. arXiv preprint arXiv:160207360

Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proc. of International Conference on Machine Learning (ICML)

Itti L, Koch C, Niebur E (1998) A model of saliency-based visual attention for rapid scene analysis. In: IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI)

Jaderberg M, Simonyan K, Zisserman A, et al. (2015a) Spatial transformer networks. In: Proc. of Neural Information Processing Systems (NIPS)

Jaderberg M, Simonyan K, Zisserman A, et al. (2015b) Spatial transformer networks. In: Advances in neural information processing systems, pp 2017–2025

Jia X, De Brabandere B, Tuytelaars T, Gool LV (2016) Dynamic filter networks. In: Advances in Neural Information Processing Systems, pp 667–675

Kingma DP, Ba J (2014) Adam: A method for stochastic optimization. arXiv preprint arXiv:14126980

Krizhevsky A, Hinton G (2009) Learning multiple layers of features from tiny images. Technical report, University of Toronto

Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Proc. of Neural Information Processing Systems (NIPS)

Larochelle H, Hinton GE (2010) Learning to combine foveal glimpses with a third-order boltzmann machine. In: Proc. of Neural Information Processing Systems (NIPS)

Ledig C, Theis L, Huszár F, Caballero J, Cunningham A, Acosta A, Aitken AP, Tejani A, Totz J, Wang Z, et al. (2017) Photo-realistic single image super-resolution using a generative adversarial network. In: Proc. of Computer Vision and Pattern Recognition (CVPR)

Li W, Zhu X, Gong S (2018) Harmonious attention network for person re-identification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 2285–2294

Lin TY, Maire M, Belongie S, Hays J, Perona P, Ramanan D, Dollár P, Zitnick CL (2014) Microsoft coco: Common objects in context. In: Proc. of European Conf. on Computer Vision (ECCV)

Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, Berg AC (2016) Ssd: Single shot multibox detector. In: Proc. of European Conf. on Computer Vision (ECCV)

Long J, Shelhamer E, Darrell T (2015) Fully convolutional networks for semantic segmentation. In: Proc. of Computer Vision and Pattern Recognition (CVPR)

Marr D, Vision A (1982) A computational investigation into the human representation and processing of visual information. WH San Francisco: Freeman and Company 1(2)

Mnih V, Heess N, Graves A, et al. (2014) Recurrent models of visual attention." advances in neural information processing systems. In: Proc. of Neural Information Processing Systems (NIPS)

Morcos AS, Barrett DG, Rabinowitz NC, Botvinick M (2018) On the importance of single directions for generalization. In: Proc. of Int'l Conf. on Learning Representations (ICLR)

Nam H, Ha JW, Kim J (2017) Dual attention networks for multimodal reasoning and matching. In: Proc. of Computer Vision and Pattern Recognition (CVPR), pp 2156–2164

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

Rensink RA (2000) The dynamic representation of scenes. In: Visual cognition 7.1-3

Riesenhuber M, Poggio T (1999) Hierarchical models of object recognition in cortex. Nature neuroscience 2(11):1019–1025

Sandler M, Howard A, Zhu M, Zhmoginov A, Chen LC (2018) Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 4510–4520

Simon M, Rodner E (2015) Neural activation constellations: Unsupervised part model discovery with convolutional networks. In: Proceedings of the IEEE International Conference on Computer Vision, pp 1143–1151

Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: Proc. of Int'l Conf. on Learning Representations (ICLR)

Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: Proc. of Computer Vision and Pattern Recognition (CVPR)

Wang F, Jiang M, Qian C, Yang S, Li C, Zhang H, Wang X, Tang X (2017) Residual attention network for image classification. In: Proc. of Computer Vision and Pattern Recognition (CVPR)

Woo S, Hwang S, Kweon IS (2018a) Stairnet: Top-down semantic aggregation for accurate one shot detection. In: Proc. of Winter Conf. on Applications of Computer Vision (WACV)

Woo S, Park J, Lee JY, Kweon IS (2018b) Cbam: Convolutional block attention module. In: Proc. of European Conf. on Computer Vision (ECCV)

Xiao T, Liu Y, Zhou B, Jiang Y, Sun J (2018) Unified perceptual parsing for scene understanding. In: Proc. of European Conf. on Computer Vision (ECCV)

Xie S, Girshick R, Dollár P, Tu Z, He K (2017) Aggregated residual transformations for deep neural networks. In: Proc. of Computer Vision and Pattern Recognition (CVPR)

Xu K, Ba J, Kiros R, Cho K, Courville A, Salakhudinov R, Zemel R, Bengio Y (2015) Show, attend and tell: Neural image caption generation with visual attention. In: Proc. of International Conference on Machine Learning (ICML)

Yang Z, He X, Gao J, Deng L, Smola A (2016) Stacked attention networks for image question answering. In:

Proc. of Computer Vision and Pattern Recognition (CVPR)

Yu F, Koltun V (2016) Multi-scale context aggregation by dilated convolutions. In: Proc. of Int'l Conf. on Learning Representations (ICLR)

Zagoruyko S, Komodakis N (2016) Wide residual networks. In: Proc. of British Machine Vision Conference (BMVC)

Zeiler MD (2012) Adadelta: an adaptive learning rate method. arXiv preprint arXiv:12125701

Zhang X, Xiong H, Zhou W, Lin W, Tian Q (2016) Picking deep filter responses for fine-grained image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 1134–1142

Zhou B, Zhao H, Puig X, Fidler S, Barriuso A, Torralba A (2019) Semantic understanding of scenes through the ade20k dataset. Int'l Journal of Computer Vision (IJCV)

Zhou, Bolei and Zhao, Hang and Puig, Xavier and Fidler, Sanja and Barriuso, Adela and Torralba, Antonio (2018) Semantic Segmentation on MIT ADE20K dataset in PyTorch. `https://github.com/CSAILVision/semantic-segmentation-pytorch/`

Zhu Y, Zhao C, Wang J, Zhao X, Wu Y, Lu H (2017) Couplenet: Coupling global structure with local parts for object detection. In: Proc. of Intl Conf. on Computer Vision (ICCV)