SimAM: A Simple, Parameter-Free Attention Module for Convolutional Neural Networks

Lingxiao Yang [1 2 3] **Ru-Yuan Zhang** [4 5] **Lida Li** [6] **Xiaohua Xie** [1 2 3]

Abstract

In this paper, we propose a conceptually simple but very effective attention module for Convolutional Neural Networks (ConvNets). In contrast to existing channel-wise and spatial-wise attention modules, our module instead infers 3-D attention weights for the feature map in a layer without adding parameters to the original networks. Specifically, we base on some well-known neuroscience theories and propose to optimize an energy function to find the importance of each neuron. We further derive a fast closed-form solution for the energy function, and show that the solution can be implemented in less than ten lines of code. Another advantage of the module is that most of the operators are selected based on the solution to the defined energy function, avoiding too many efforts for structure tuning. Quantitative evaluations on various visual tasks demonstrate that the proposed module is flexible and effective to improve the representation ability of many ConvNets. Our code is available at Pytorch-SimAM.

1. Introduction

Convolutional Neural Networks (ConvNets) trained on large-scale datasets (*e.g.,* ImageNet (Russakovsky et al., 2015)) have greatly boosted the performance on many vision tasks, such as image classification (Krizhevsky et al., 2012; Simonyan & Zisserman, 2014; He et al., 2016b; Huang et al., 2017; Szegedy et al., 2015; Sandler et al., 2018), object detection (Ren et al., 2015; Liu et al., 2016; He et al., 2017), and video understanding (Feichtenhofer et al., 2016; Wang et al., 2018a). Multiple studies have demonstrated that a better ConvNet structure can significantly improve performance on various problems. Therefore, constructing a strong ConvNet is an essential task in vision research.

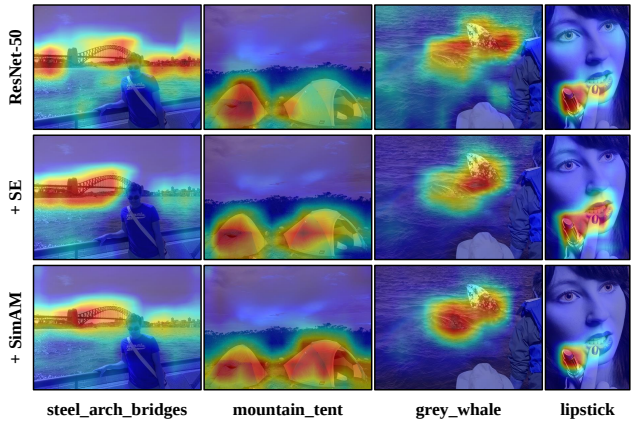


Figure 1. **Visualization of feature activations obtained by different networks.** All compared networks are trained on ImageNet (Russakovsky et al., 2015) under a consistent setting. The features are extracted on the validation set and shown by Grad-CAM (Selvaraju et al., 2017). Our SimAM helps the network focus on some primary regions which are close to the image labels shown below.

A modern ConvNet typically has multiple stages, and each stage consists of a few blocks. Such block is constructed by several operators like convolution, pooling, activation or some customized meta-structure (referred as **module** in this paper). Recently, instead of designing the whole architecture as (Krizhevsky et al., 2012), many works focus on building advanced blocks to improve the representational power of ConvNets. Stacked convolutions (Simonyan & Zisserman, 2014), residual units (He et al., 2016b;a; Zagoruyko & Komodakis, 2016; Sandler et al., 2018), and dense connections (Huang et al., 2017; 2018) are the most representative ones that have been widely applied in existing architectures. However, designing those blocks requires rich expert knowledge and enormous time. To circumvent this, many researchers seek for some search strategies to automatically build archi-

[1]School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China [2]Guangdong Province Key Laboratory of Information Security Technology, Sun Yat-sen University, Guangzhou, China [3]Key Laboratory of Machine Intelligence and Advanced Computing, Ministry of Education, Sun Yat-sen University, Guangzhou, China [4]Institute of Psychology and Behavioral Science, Shanghai Jiao Tong University, Shanghai, China [5]Shanghai Key Laboratory of Psychotic Disorders, Shanghai Mental Health Center, Shanghai Jiao Tong University, Shanghai, China [6]The Hong Kong Polytechnic University, Hong Kong, China. Correspondence to: Xiaohua Xie <xiexiaoh6@mail.sysu.edu.cn>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

tectures (Zoph & Le, 2016; Liu et al., 2018b; Dong & Yang, 2019; Tan & Le, 2019; Guo et al., 2020; Liu et al., 2019; Feichtenhofer, 2020; Tan et al., 2020).

Besides designing sophisticated blocks, another line of research focuses on building plug-and-play modules (Hu et al., 2018b; Woo et al., 2018; Cao et al., 2020; Lee et al., 2019; Wang et al., 2020; Yang et al., 2020) that can refine convolutional outputs within a block and enable the whole network to learn more informative features. For example, Squeeze-and-Excitation (SE) module (Hu et al., 2018b) allows a network to capture task-relevant features (see "mountain_tent" in Figure 1) and suppress many background activations (see "steel_arch_bridges" in Figure 1). This module is independent of network architecture, and therefore can be plugged into a broad range of networks such as VGG (Simonyan & Zisserman, 2014), ResNets (He et al., 2016b), and ResNeXts (Xie et al., 2017). More recently, the SE module is included as a component in AutoML to search for better network structures (Howard et al., 2019; Tan & Le, 2019).

However, existing attention modules have two problems. First, they can only refine features along either channel or spatial dimensions, limiting their flexibility of learning attention weights that vary across both channel and space. Second, their structures are built by a series of complex factors, *e.g.,* the choice for pooling. We address these issues by proposing a module based on well-established neuroscience theories. Specifically, to make the network learn more discriminative neurons, we propose to directly infer 3-D weights (*i.e.*, considering both spatial and channel dimensions) from current neurons and then in turn refine those neurons. To efficiently infer such 3-D weights, we define an energy function guided by the knowledge from neuroscience and derive a closed-form solution. As shown in Figure 1, our module helps the network capture many valuable cues which are consistent with image labels (see examples of "mountain_tent" and "grey_whale"). Moreover, most of the operators used in our module are obtained from the solution to the energy function without other bells and whistles.

It is worth emphasizing that we mainly focus on a small plug-and-play module rather than a new architecture beyond existing ConvNets. One previous study (Wang et al., 2017) also attempts to infer 3-D weights. Their promising results are based on a hand-crafted encoder-decoder structure. Compared to that study, our work provides an alternative and efficient way to generate 3-D weights. Our module is more flexible and modularized, and still remains lightweight. To sum up, our main contributions are:

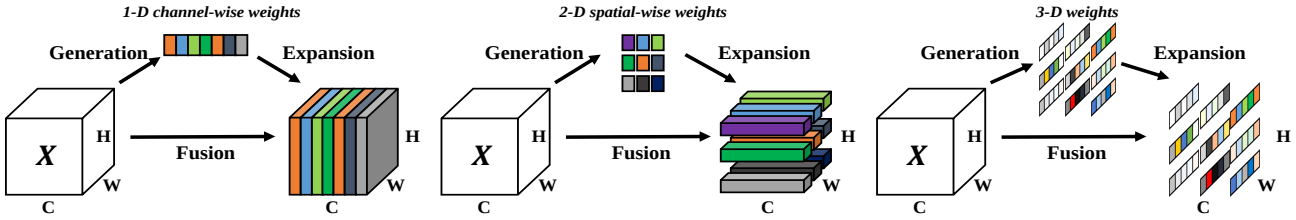
- Inspired by the attention mechanisms in human brain, we propose an attention module with full 3-D weights and design an energy function to calculate the weights.
- We derive a closed-form solution of the energy function that speedup the weight calculation and allows for a lightweight form of the whole module.
- We integrate the proposed module into some well-known networks and evaluate them on various tasks. Our module performs favourably against other popular modules in terms of accuracy, model size, and speed.

2. Related Work

In this section, we briefly discuss representative works on network architectures and plug-and-play attention modules.

Network Architectures. In 2012, a modern deep ConvNet, AlexNet (Krizhevsky et al., 2012), was released for large-scale image classification. It is a simple feedforward structure similar to the setup in LeNet (LeCun et al., 1998). After that, multiple approaches have been proposed to strengthen the power of ConvNets. Some works focus on finding the optimal filter shapes (Zeiler & Fergus, 2014; Chatfield et al., 2014), and some other methods attempt to design much deeper networks. For example, VGG (Simonyan & Zisserman, 2014) and Inception Net (Szegedy et al., 2015) use stacked convolutions to reduce the risk of gradient vanishing/exploding (Bengio et al., 1994; Glorot & Bengio, 2010). Following this up, ResNet (He et al., 2016b) and Highway network (Srivastava et al., 2015) add shortcut connections from input to output within each block. The shortcut connections enable ConvNet to scale up to hundreds of layers. Their results reveal that increasing network depth can substantially boost representational power of a ConvNet. Besides network depth, some works propose to increase the number of filters (Zagoruyko & Komodakis, 2016) for wider block, to add more connections within each block (Huang et al., 2017), or to explore group/depth-wise convolutions (Xie et al., 2017; Chollet, 2017). More recently, a bunch of works use AutoML (Zoph & Le, 2016; Liu et al., 2018b;a; Tan et al., 2019; Howard et al., 2019; Wu et al., 2019) to save the manual efforts in network design. Different from such mentioned works, we aim at designing a lightweight plug-and-play module. This module can be adopted for many ConvNets to further boost their performance in various tasks without big changes in architecture.

Attention and Recalibration Modules. Previous works also design some computational modules that refine feature maps. They are usually referred as attention module or recalibration module. For simplicity, we call them as attention module in this paper. In fact, human attention acts as one of the most important selection mechanisms that prioritize task-relevant information and attenuate irrelevant signals (Reynolds & Chelazzi, 2004; Chun et al., 2011). The attentional mechanisms in human visual processing inspire researchers to design similar attention modules in ConvNets. One representative work, Squeeze-and-Excitation (SE) (Hu et al., 2018b), learns the importance of different channels by



(a) Channel-wise attention (b) Spatial-wise attention (c) Full 3-D weights for attention

Figure 2. **Comparisons of different attention steps**. Most of existing attention modules generate 1-D or 2-D weights from features $\mathbf{X}$, and then expand the generated weights for channel (a) and spatial (b) attention. Our module instead directly estimates 3-D weights (c). In each subfigure, same color denotes that a single scalar is employed for each channel, for spatial location or for each point on that features.

firstly capturing some context cues from a global view, and then using two fully-connected layers to model interactions between channels. The outputs are in turn used to refine those features at the channel level. This module is further extended by other methods, e.g., capturing global contexts with convolutional aggregators (Hu et al., 2018a), learning interactions with a channel-based convolution (Wang et al., 2020), adding spatial-wise attention (Woo et al., 2018), incorporating long-range dependencies (Cao et al., 2020; Wang et al., 2018b), unifying attention and normalization process (Li et al., 2019a), or utilizing style cues of that features (Lee et al., 2019). However, all these methods treat either all neurons in one channel or all neurons at one spatial position equally such that they cannot effectively compute true 3-D weights. Moreover, their algorithms to calculate attention weights are mostly hand-crafted, requiring extensive computational power. In contrast, we design our module based on some well-known neuroscience theories, which is more interpretable. It is noticed that some modules are also inspired by neuroscience theories, such as convolution driven by adaptive contexts (Lin et al., 2020), and receptive field adjusting by selective kernels (Li et al., 2019b). Different to them, our module is based on spatial suppression observed from mammal brains and formulates the weight generation as an energy function. A closed-form solution is derived for this energy function. Thanks to the fast closed-form solution, our module introduces no additional parameters, a compelling property that differs from those previous works.

3. Method

In this section, we first summarize some representative attention modules, such as SE (Hu et al., 2018b), CBAM (Woo et al., 2018), GC (Cao et al., 2020). We then introduce our new module that shares the similar philosophy as previous methods, but has a very different formulation.

3.1. Overview of existing attention modules

Existing attention modules are often integrated into each block to refine outputs from previous layers. Such refinement step is usually operated along either the channel dimension (Figure 5a) or the spatial dimension (Figure 5b). As a result, those methods generate 1-D or 2-D weights and treat neurons in each channel or spatial location equally, which may limit their capability of learning more discriminative cues. For example, Figure 1 shows SE loses some main components of "grey_whale". Therefore, we argue that full 3-D weights is better than conventional 1-D and 2-D attentions, and propose to refine that features with full 3-D weights, as shown in Figure 2c.

Table 1. **Comparisons of different attention modules in their structure design and parameters**. Operators are: spatial (GAP) or channel average pooling (CAP), spatial (GMP) or channel max pooling (CMP), standard deviation calculated along spatial dimension (GSP), standard convolution (C2D) or channel-based convolution (C1D), standard (FC) or channel-wise (CFC) fully-connected layers, layer normalization (LN), batch normalization (BN), Softmax and ReLU. k and r are convolutional filter numbers and reduction ratio respectively. C is the current feature channels.

Attention Modules	Operators	Parameters	Design
SE (Hu et al., 2018b)	GAP, FC, ReLU	$2C^2/r$	handcrafted
CBAM (Woo et al., 2018)	GAP, GMP, FC, ReLU CAP, CMP, BN, C2D	$2C^2/r + 2k^2$	handcrafted
GC (Cao et al., 2020)	C1D, Softmax, LN, FC, ReLU	$2C^2/r + C$	handcrafted
ECA (Wang et al., 2020)	GAP, C1D	k	handcrafted
SRM (Lee et al., 2019)	GAP, GSP, CFC, BN	$6C$	handcrafted
SimAM (this paper)	GAP, $/, \odot, +$	0	Eqn (5)

Another important factor for attention modules is the weight generation method. Most existing works calculate attention weights based on some ungrounded heuristics. For example, SE uses global average pooling (GAP) to capture the global context. The context aggregation is further improved by adding a global max pooling (GMP) and a softmax-based pooling in CBAM and GC respectively. In table 1, we list main operators used in previous works. It can be seen that existing modules built on many common operators such as FC, Conv2D, BN *etc.*, as well as some highly customized operators like channel-wise fully-connected layers (CFC). In

sum, the choices of the structure design prequire significant engineering. We argue that implementations of attention mechanisms should be guided by some unified principles in neural computation. Therefore, we propose a new method based on some well-established neuroscience theories.

3.2. Our attention module

As our illustrated before, existing attention modules in computer vision focus on either the channel domain or the spatial domain. These two attention mechanisms correspond exactly to the feature-based attention and the spatial-based attention in the human brain (Carrasco, 2011). However, in humans, these two mechanisms coexist and jointly contribute to information selection during visual processing. Therefore, we propose an attention module to operate similarly such that each neuron is assigned with a unique weight.

However, it is challenging to directly estimate full 3-D weights. (Wang et al., 2017) proposes to employ the encoder-decoder framework to learn 3-D weights. But this approach adds different subnetworks from lower to higher layers of a ResNet, which cannot be easily extended to other modularized pnetworks. Another example, CBAM, separately estimates 1-D and 2-D weights, and then combines them. This approach does not directly generate true 3-D weights (Woo et al., 2018). The two-step manner in CBAM takes too much calculation time. Hence, we argue that the calculation of 3-D weights should be straightforward, and at the same time allow the module to keep a lightweight property.

According to above arguments, here we propose a module that can efficiently produce true 3-D weights. To successfully implement attention, we need to estimate the importance of individual neurons. How to calculate importance of individual neurons based on the feature map in a layer? In visual neuroscience, the most informative neurons are usually the ones that show distinctive firing patterns from surrounding neurons. Moreover, an active neuron may also suppress the activities of surrounding neurons, a phenomenon termed as spatial suppression (Webb et al., 2005). In other words, the neurons displaying clear spatial suppression effects should be given higher priority (*i.e.,* importance) in visual processing. The simplest implementation to find these neurons is to measure the linear separability between one target neuron and other neurons. Based on these neuroscience findings, we define the following energy function for each neuron:

$$e_t(w_t, b_t, \mathbf{y}, x_i) = (y_t - \hat{t})^2 + \frac{1}{M-1}\sum_{i=1}^{M-1}(y_o - \hat{x}_i)^2. \quad (1)$$

Here, $\hat{t} = w_t t + b_t$ and $\hat{x}_i = w_t x_i + b_t$ are linear transforms of t and x_i, where t and x_i are the target neuron and other neurons in a single channel of the input feature $\mathbf{X} \in \mathbb{R}^{C\times H\times W}$. i is index over spatial dimension and $M = H \times W$ is the number of neurons on that channel. w_t and b_t are weight and bias the transform. All values in Eqn (1) are scalars. The Eqn (1) attains the minimal value when the $\hat{t}$ equals to y_t, and all other $\hat{x}_i$ are y_o, where y_t and y_o are two different values. By minimizing this equation, Eqn (1) is equivalent to find the linear separability between the target neuron t and all other neurons in the same channel. For simplicity, we adopt binary labels (*i.e.,* 1 and -1) for y_t and y_o and also add a regularizer into Eqn (1). The final energy function is given by:

$$\begin{aligned} e_t(w_t, b_t, \mathbf{y}, x_i) = \frac{1}{M-1}\sum_{i=1}^{M-1}(-1-(w_t x_i + b_t))^2 \\ + (1-(w_t t + b_t))^2 + \lambda w_t^2. \end{aligned} \quad (2)$$

In theory, we have M energy functions for each channel. It is computationally burdensome to solve all these equations via some iterative solvers like SGD. Luckily, Eqn (2) has a fast closed-form solution with respect to w_t and b_t, which can be easily obtained by:

$$w_t = -\frac{2(t-\mu_t)}{(t-\mu_t)^2 + 2\sigma_t^2 + 2\lambda}, \quad (3)$$

$$b_t = -\frac{1}{2}(t+\mu_t)w_t. \quad (4)$$

$\mu_t = \frac{1}{M-1}\sum_{i=1}^{M-1} x_i$ and $\sigma_t^2 = \frac{1}{M-1}\sum_{i}^{M-1}(x_i - \mu_t)^2$ are mean and variance calculated over all neurons except t in that channel. Since existing solutions shown in Eqn (3) and Eqn (4) are obtained on a single channel, it is reasonable to assume that all pixels in a single channel follows the same distribution. Given this assumption, mean and variance can be calculated over all neurons and reused for all neurons on that channel (Hariharan et al., 2012). It can significantly reduce the computation costs to avoid iteratively calculating μ and σ for each position. As a result, the minimal energy can be computed with the following:

$$e_t^* = \frac{4(\hat{\sigma}^2 + \lambda)}{(t-\hat{\mu})^2 + 2\hat{\sigma}^2 + 2\lambda}, \quad (5)$$

where $\hat{\mu} = \frac{1}{M}\sum_{i=1}^{M} x_i$ and $\hat{\sigma}^2 = \frac{1}{M}\sum_{i=1}^{M}(x_i - \hat{\mu})^2$. Eqn (5) indicates that the lower energy e_t^*, the neuron t is more distinctive from surround neurons, and more important for visual processing. Therefore, the importance of each neuron can be obtained by $1/e_t^*$. Akin to our method, (Aubry et al., 2014) investigate a similar function for semantic part matching. But their method needs to compute large covariance matrix which is not suitable for deep neural networks. Different to (Aubry et al., 2014), we operate on the single neuron and integrate this linear separability into an end-to-end framework. In addition, we provide comprehensive understandings from neuroscience.

```
# X: input feature [N, C, H, W]
# lambda: coefficient λ in Eqn (5)

def forward (X, lambda):
    # spatial size
    n = X.shape[2] * X.shape[3] - 1
    # square of (t - u)
    d = (X - X.mean(dim=[2,3])).pow(2)
    # d.sum() / n is channel variance
    v = d.sum(dim=[2,3]) / n
    # E_inv groups all importance of X
    E_inv = d / (4 * (v + lambda)) + 0.5
    # return attended features
    return X * sigmoid(E_inv)
```

Figure 3. A pytorch-like implementation of our SimAM.

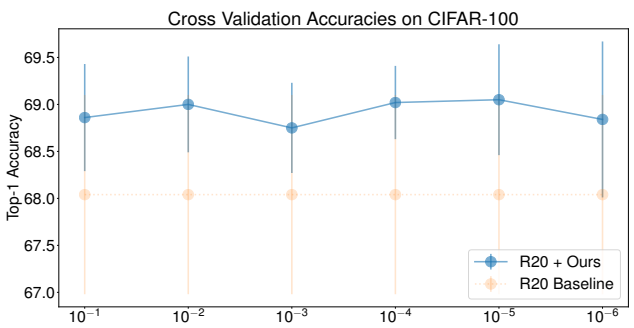


Figure 4. **Visualization of cross validation for λ search.** This process is achieved based on ResNet-20 network on a split from train set. Mean and standard deviation over 5 folds are reported.

By far, we derive an energy function and find the importance of each neuron. According to (Hillyard et al., 1998), attention modulation in mammalian brain typically manifests as a gain (i.e., scaling) effect on neuronal responses. We thus use a scaling operator rather than an addition for feature refinement. The whole refinement phase of our module is:

$$\widetilde{\mathbf{X}} = sigmoid(\frac{1}{\mathbf{E}}) \odot \mathbf{X}, \tag{6}$$

where $\mathbf{E}$ groups all e_t^* across channel and spatial dimensions. $sigmoid$ is added to restrict too large value in $\mathbf{E}$. It will not influence the relative importance of each neuron because $sigmoid$ is a monofonic function.

In fact, except the calculations of channel mean $\hat{\mu}$ and variance $\hat{\sigma}$, all computing in our module are element-wise operations. Therefore, we can take the advantage of current machine learning libraries like pytorch to implement our module (Eqn (6)) in just a few lines, as shown in Figure 3. We add this implementation after second convolutional layer within each block. In summary, the proposed module is derived from basic theories of neuroscience which is quite different to previous manners. Moreover, our module is simple to implement and use along with existing networks.

4. Experiments

In this section, we conduct a series of experiments across a wide range of tasks to verify the effectiveness of our SimAM. For fair comparisons, we re-implement all compared methods using pytorch with consistent settings.

4.1. CIFAR Classification

To begin, we test our methods on image classification tasks based on CIFAR (Krizhevsky et al., 2009). There are two variants: one has 10 categories and the other one contains 100 classes. Both variants have 50k training and 10k validation images. Our main focus is to verify our simple yet effective attention module. Therefore, we incorporate our attention module into some well-established architectures, including ResNet (He et al., 2016b), Pre-activation ResNet (He et al., 2016a), WideResNet (Zagoruyko & Komodakis, 2016), and MobileNetV2 (Sandler et al., 2018) [1].

Implementation details. We follow the standard training pipeline (Lee et al., 2015; He et al., 2016b) for all models. Specifically, each image is zero-padded with 4 pixels on each side, and a 32×32 image fed for training is randomly cropped from that padded image or its horizontal flip. During evaluation, all models accept the original images for testing. Optimization is done by a SGD solver with a momentum of 0.9, a batch size of 128, and a weight decay of 0.0005. All networks are trained on a single GPU except that the WideResNet is optimized on two GPUs. The learning rate is started with 0.1 and divided by 10 at 32,000 and 48,000 iterations (the division is stopped at 64,000 iterations). For our SimAM, the hyper-parameter λ in Eqn (5) is set to 0.0001, searched using ResNet-20 on a 45k/5k train/split set. Detailed analysis of λ will be discussed in the later section. For other modules including SE (Hu et al., 2018b), CBAM (Woo et al., 2018), ECA (Wang et al., 2020), and GC (Cao et al., 2020), we use the public codes provided by the authors with default settings. Because of randomness, we report average accuracy and standard derivation over 5 times for each method. All results are shown in Table 2.

Analysis of Attention Modules. The proposed SimAM consistently improves the top-1 accuracies over all employed baseline networks on both CIFAR-10 and CIFAR-100 datasets. In addition, our SimAM also obtains very competitive results against other compared attention modules. Specifically, SimAM achieves the best accuracies compared to other modules based onp small networks (ResNet-20 and PreResNet-20). For 56-layer networks, our method

[1] We use the model from https://github.com/kuangliu/pytorch-cifar/blob/master/models/mobilenetv2.py for CIFAR datasets and employ the default structure for ImageNet.

Table 2. Top-1 accuracies (%) for different networks with 5 attention modules, SE (Hu et al., 2018b), CBAM (Woo et al., 2018), ECA (Wang et al., 2020), GC (Cao et al., 2020) and our SimAM, on CIFAR-10 (C10) and CIFAR-100 (C100) datasets. All results are reported as *mean*±*std* by calculating the top1 accuracy of each model over 5 trials . Our SimAM achieves very competitive results against other modules without introducing any parameter into the baseline models.

Attention	ResNet-20		ResNet-56		ResNet-110		MobileNetV2	
Module	C10	C100	C10	C100	C10	C100	C10	C100
Baseline	92.33±0.19	68.88±0.15	93.58±0.20	72.24±0.37	94.51±0.31	75.54±0.24	91.86±0.12	71.32±0.09
+ SE	92.42±0.14	69.45±0.11	93.69±0.17	**72.84±0.51**	94.68±0.22	**76.56±0.30**	91.79±0.20	71.54±0.31
+ CBAM	92.60±0.31	69.47±0.35	**93.82±0.10**	72.47±0.52	94.83±0.18	76.45±0.54	91.88±0.16	71.79±0.22
+ ECA	92.35±0.35	68.89±0.57	93.68±0.08	72.45±0.38	94.72±0.15	76.33±0.65	92.34±0.23	71.24±0.45
+ GC	92.47±0.19	69.16±0.48	93.58±0.08	72.50±0.50	**94.78±0.25**	76.21±0.17	91.73±0.14	71.78±0.28
+ **SimAM**	**92.73±0.18**	**69.57±0.40**	93.76±0.13	72.82±0.25	94.72±0.18	76.42±0.27	**92.36±0.20**	**72.08±0.28**
Attention	PreResNet-20		PreResNet-56		PreResNet-110		WideResNet-20x10	
Module	C10	C100	C10	C100	C10	C100	C10	C100
Baseline	92.14±0.25	68.70±0.30	93.71±0.24	71.83±0.23	94.22±0.18	75.95±0.22	95.78±0.10	81.31±0.39
+ SE	92.24±0.06	68.70±0.21	93.57±0.15	**72.57±0.32**	94.40±0.18	**76.68±0.30**	**96.24±0.04**	81.30±0.08
+ CBAM	92.19±0.11	68.76±0.56	93.67±0.08	72.16±0.12	94.37±0.33	76.01±0.57	95.98±0.17	80.54±0.23
+ ECA	92.16±0.25	68.31±0.46	93.78±0.17	72.43±0.45	94.70±0.31	76.11±0.54	96.12±0.15	80.35±0.22
+ GC	92.19±0.20	68.96±0.48	93.77±0.12	72.44±0.19	94.85±0.19	75.88±0.28	96.12±0.18	79.98±0.17
+ **SimAM**	**92.47±0.12**	**69.13±0.50**	**93.80±0.30**	72.36±0.19	**94.90±0.19**	76.24±0.31	96.09±0.21	**81.51±0.25**

also achieves the best result based on PreResNet-56 in CIFAR-10, and performs favorably against other modules on another settings. In larger networks, SimAM still attains very promising results. For example, in MobileNetV2, ECA and CBAM achieve the best accuracies on CIFAR-10 (92.34±0.23) and CIFAR-100 (71.79±0.22), respectively. Our SimAM outperforms SE and CBAM with top-1 results of 92.36±0.20 and 72.02±0.28 on those datasets respectively. We consider MobileNetV2 as a large network because it contains nearly 2.4 M parameters, which is even bigger than ResNet-110 with ∼1.17 M parameters. Table 2 also demonstrates that our module can improve a very big network - WideResNet-20x10, which has about 36 M free parameters on the CIFAR datasets. All these results demonstrate that the effectiveness of our parameter-free SimAM is not confined to some specific networks.

Analysis of λ. Figure 4 shows our searched results of λ with ResNet-20 on the training set from the CIFAR-100 dataset. It is intractable to verify all real values of λ. Therefore, we set the pool of λ from 10^{-1} to 10^{-6}, as shown in the "x-axis" of this figure. For each λ value, we repeat 5 times and report the mean and standard deviations. Note that, all these tests are done on the split set from the training data. Based on Figure 4, we make two conclusions: (1) our module can significantly boost performance using a wide range of λ (from 10^{-1} to 10^{-6}), and (2) $\lambda = 10^{-4}$ provides a good balance between the top-1 accuracy and the standard deviation. Following these observations, we set $\lambda = 10^{-4}$ in all networks on the CIFAR datasets.

4.2. ImageNet Classification

In this section, we evaluate our SimAM on ImageNet (Russakovsky et al., 2015) that consists of 1000 classes. All models are trained on ∼1.2M training images and tested on 50K validation images with the standard setup. In detail, during training, input images are randomly cropped to 224×224, from the original image or its horizontal flipped one. All ResNets are optimized by SGD with a batch size of 256 on 4 GPUs (Quadro RTX 8000). Momentum and weight decay are set to 0.9 and 0.0001 respectively. In addition, the batch mean and variance for BN are computed within each GPU. We train the networks with a initial learning rate 0.1 and then decrease the learning rate by $10\times$ at 30, 60, 90 epochs. The training is stopped after 100 epochs. For MobileNetV2, we use the same training pipeline as in (Zhang, 2019). It is trained with a cosine learning rate schedule, starting with a learning rate 0.5. Weight decay and the number of epochs are set to 4e-5 and 150 respectively.

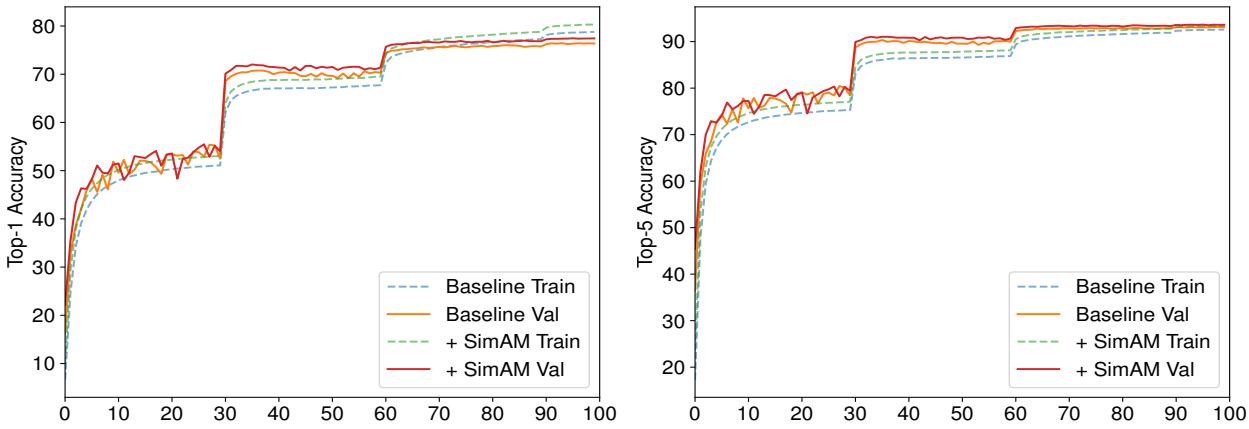
For the proposed SimAM, we still employ cross-validation to find a good λ. Instead of searching all networks on the original training images from this dataset, we base on

Table 3. Top-1 and Top-5 accuracies (%) for different networks with various attention modules, including SE (Hu et al., 2018b), CBAM (Woo et al., 2018), ECA (Wang et al., 2020), SRM (Lee et al., 2019) and the proposed SimAM, on ImageNet-1k (Russakovsky et al., 2015) dataset. All networks are trained under the same settings in our own computers. Our SimAM achieves very competitive results against other attention modules, while being efficiency in terms of speed and parameters. Speed is tested on a GTX 1080 TI gpu with a single image. We forward 500 images and report the average FPS.

Model	Top-1 Acc.	Top-5 Acc.	# Parameters	+ Parameters-to-Baseline	# FLOPs	Inference Speed
ResNet-18	70.33 %	89.58 %	**11.69 M**	**0**	**1.82 G**	**215** FPS
+ SE (Hu et al., 2018b)	71.19 %	**90.21** %	11.78 M	0.087 M	1.82 G	144 FPS
+ CBAM (Woo et al., 2018)	71.24 %	90.04 %	11.78 M	0.090 M	1.82 G	78 FPS
+ ECA (Wang et al., 2020)	70.71 %	89.85 %	11.69 M	36	1.82 G	148 FPS
+ SRM (Lee et al., 2019)	71.09 %	89.98 %	11.69 M	0.004 M	1.82 G	115 FPS
+ **SimAM**	**71.31** %	89.88 %	**11.69 M**	**0**	1.82 G	147 FPS
ResNet-34	73.75 %	91.60 %	**21.80 M**	**0**	**3.67 G**	119 FPS
+ SE (Hu et al., 2018b)	74.32 %	91.99 %	21.95 M	0.157 M	3.67 G	81 FPS
+ CBAM (Woo et al., 2018)	74.41 %	91.85 %	21.96 M	0.163 M	3.67 G	38 FPS
+ ECA (Wang et al., 2020)	74.03 %	91.73 %	21.80 M	74	3.67 G	82 FPS
+ SRM (Lee et al., 2019)	**74.49** %	92.01 %	21.81 M	0.008 M	3.67 G	59 FPS
+ **SimAM**	74.46 %	**92.02** %	**21.80 M**	**0**	3.67 G	78 FPS
ResNet-50	76.34 %	93.12 %	**25.56 M**	**0**	**4.11 G**	**89 FPS**
+ SE (Hu et al., 2018b)	77.51 %	93.74 %	28.07 M	2.515 M	4.12 G	64 FPS
+ CBAM (Woo et al., 2018)	**77.63** %	**93.88** %	28.09 M	2.533 M	4.12 G	33 FPS
+ ECA (Wang et al., 2020)	77.17 %	93.52 %	25.56 M	88	4.12 G	64 FPS
+ SRM (Lee et al., 2019)	77.51 %	93.06 %	25.59 M	0.030 M	4.11 G	56 FPS
+ **SimAM**	77.45 %	93.66 %	**25.56 M**	**0**	4.11 G	64 FPS
ResNet-101	77.82 %	93.85 %	**44.55 M**	**0**	**7.83 G**	**47 FPS**
+ SE (Hu et al., 2018b)	78.39 %	94.13 %	49.29 M	4.743 M	7.85 G	33 FPS
+ CBAM (Woo et al., 2018)	78.57 %	**94.18** %	49.33 M	4.781 M	7.85 G	14 FPS
+ ECA (Wang et al., 2020)	78.46 %	94.12 %	44.55 M	171	7.84 G	33 FPS
+ SRM (Lee et al., 2019)	78.58 %	94.15 %	44.68 M	0.065 M	7.83 G	25 FPS
+ **SimAM**	**78.65** %	94.11 %	**44.55 M**	**0**	7.83 G	32 FPS
ResNeXt-50 (32x4d)	77.47 %	93.52 %	**25.03 M**	**0**	**4.26 G**	**70 FPS**
+ SE (Hu et al., 2018b)	77.96 %	93.93 %	27.54 M	2.51 M	4.27 G	53 FPS
+ CBAM (Woo et al., 2018)	**78.06** %	**94.07** %	27.56 M	2.53 M	4.27 G	32 FPS
+ ECA (Wang et al., 2020)	77.74 %	93.87 %	25.03 M	86	4.27 G	54 FPS
+ SRM (Lee et al., 2019)	78.04 %	93.91 %	25.06 M	0.030 M	4.26 G	46 FPS
+ **SimAM**	78.00 %	93.93 %	**25.03 M**	**0**	**4.26 G**	53 FPS
MobileNetV2	71.90 %	90.51 %	**3.50 M**	**0**	**0.31 G**	**99 FPS**
+ SE (Hu et al., 2018b)	72.46 %	**90.85** %	3.53 M	0.028 M	0.31 G	65 FPS
+ CBAM (Woo et al., 2018)	**72.49** %	90.78 %	3.54 M	0.032 M	0.32 G	35 FPS
+ ECA (Wang et al., 2020)	72.01 %	90.46 %	3.50 M	59	0.31 G	66 FPS
+ SRM (Lee et al., 2019)	72.32 %	90.70 %	3.51 M	0.003 M	0.31 G	53 FPS
+ **SimAM**	72.36 %	90.74 %	**3.50 M**	**0**	**0.31 G**	66 FPS

ResNet-18 model and search the λ on the image with half resolutions. To further reduce the search time, we train ResNet-18 model with 50 epochs. The learning rate starts with 0.1, but is decayed by 10 factor at 15, 30, 45 epochs. The parameter pool is the same to what we used on CIFAR and we run 3 times for each parameter. The whole search can be done in less than 1 week and $\lambda = 0.1$ is selected as it provides a good tradeoff between accuracy and robustness.

On ImageNet, we compare our module with two representative attention modules, SE (Hu et al., 2018b) and CBAM (Woo et al., 2018), and two recently introduced methods, ECA (Wang et al., 2020) and SRM (Lee et al., 2019). We do not include GC (Cao et al., 2020) because their performance is obtained by a very different training pipeline to the standard setup. All results are shown in Table 3. Generally speaking, all attention modules can improve the



(a) Top-1 accuracy. (b) Top-5 accuracy.

Figure 5. **Training curve comparisons for ResNet-50 with and without our module on ImageNet-1K.** Top-1 (%) and Top-5 (%) accuracies of both networks are shown in the left and right, respectively. As can be seen, integrating our SimAM into ResNet-50 works better than the baseline model on both training and validation.

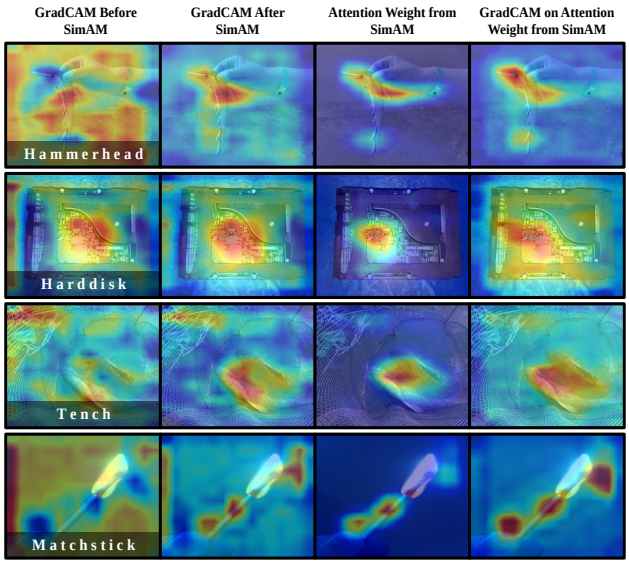


Figure 6. **Visualization of feature activations using the trained ResNet-50 with our SimAM.** For each image, maps (from left to right) are GradCAM on features before and after SimAM, attention weights, and GradCAM on attention weights. The attention maps are obtained by averaging the 3-D attention weights along the channel dimension (see more channel-wise attention weights in **supplementary materials**).

baseline models with a clear margin. In addition, the proposed SimAM achieves leading performance in ResNet-18 and also obtains slightly better results in ResNet-101. For ResNet-34, ResNet-50, ResNeXt-50 and MobileNetV2, the proposed SimAM still performs favorably against other attention modules. Moreover, our module does not add any parameters into existing networks, which is a great advantage over other modules. Table 3 also compares the inference speed of the attention modules. SE and ECA often show better inference speed than CBAM and SRM. Our SimAM obtains a similar inference FPS as compared to SE and ECA. We believe our module can be further speedup as other modules because many operators employed in other modules are highly tuned on GPUs. In Figure 5, we also present the training and validation curves of ResNet-50 with our module. Our module improves the baseline model in both training and validation accuracies.

To see the effect of our SimAM on the model, we visualize different features from ResNet-50 trained with our SimAM module by Grad-CAM (Selvaraju et al., 2017). As shown in Figure 6, our SimAM refines the features that focus on main objects (1st *v.s.* 2nd in each group). The heatmaps generated by GradCAM are consistent with SimAM masks (2nd v.s. 3rd v.s. 4th in each group). Both the quantitative and qualitative results presented above pdemonstrate that our SimAM can effectively enhance representational power in various networks, without adding any extra parameter. **In our supplementary materials, we show more results with some other re-calibration modules.**

4.3. Object Detection and Instance Segmentation

In this subsection, we base on the popular Faster R-CNN (Ren et al., 2015) and Mask R-CNN (He et al., 2017) with feature pyramid networks (FPNs) (Lin et al., 2017) to evaluate SimAM on object detection tasks. Besides, we also report instance segmentation results using Mask R-CNN. We mainly compare our module with SE and baseline models. All networks are pre-trained on ImageNet-1K (Russakovsky et al., 2015) and transferred to the COCO (Lin et al., 2014) dataset by fune-tuning. We train each detector on the "coco_2017_train" set and evaluate on "coco_2017_val" set. For fair comparisons, we adopt mmdetection (Chen et al., 2019) to train and evaluate all models

Table 4. Comparisons ofp ResNet-50/101 with SE and the proposed SimAM on COCO dataset. All backbone networks are pre-trained on ImageNet-1K dataset. APs refer to box IoU and mask IoU in detection and segmentation tasks respectively. We also include the number additional parameters in this table.

Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	+ Params
Faster RCNN for Object Detection							
ResNet-50	37.8	58.5	40.8	21.9	41.7	48.2	**0**
+ SE	**39.4**	60.6	**43.0**	**23.6**	**43.5**	50.4	2.5 M
+ SimAM	39.2	**60.7**	42.8	22.8	43.0	**50.6**	**0**
ResNet-101	39.6	60.3	43.0	22.5	43.7	51.4	**0**
+ SE	41.1	62.0	**45.2**	**24.1**	45.4	**53.0**	4.7 M
+ SimAM	**41.2**	**62.4**	45.0	24.0	**45.6**	52.8	**0**
Mask RCNN for Object Detection							
ResNet-50	38.1	58.9	41.3	22.2	41.5	49.4	**0**
+ SE	**39.9**	**61.1**	**43.4**	**24.6**	43.6	51.3	2.5 M
+ SimAM	39.8	61.0	43.4	23.1	**43.7**	**51.4**	**0**
ResNet-101	40.3	60.8	44.0	22.8	44.2	52.9	**0**
+ SE	41.8	62.6	45.5	24.3	**46.3**	**54.1**	4.7 M
+ SimAM	**41.8**	**62.8**	**46.0**	**24.8**	46.2	53.9	**0**
Mask RCNN for Instance Segmentation							
ResNet-50	34.6	55.6	36.7	18.8	37.7	46.8	**0**
+ SE	36.0	57.8	38.1	**20.7**	39.4	48.5	2.5 M
+ SimRM	**36.0**	**57.9**	**38.2**	19.1	**39.7**	**48.6**	**0**
ResNet-101	36.3	57.6	38.8	18.8	39.9	49.6	**0**
+ SE	37.2	59.4	39.7	20.2	41.2	50.4	4.7 M
+ SimRM	**37.6**	**59.5**	**40.1**	**20.5**	**41.5**	**50.8**	**0**

on 4 GPUs with a batch size of 8 (2/gpu). The start learning rate is set to 0.01 according to the linear scaling rule (Goyal et al., 2017). Other hyper-parameters are the same to the default settings of each detector.

As shown in Table 4, integrating either SE or our SimAM into ResNets can largely boost the baseline performance in object detection and instance segmentation. For object detection task, the compared two attention modules obtain very similar performance by using both Faster RCNN and Mask RCNN detectors. For instance segmentation task, the proposed SimAM module achieves slightly better results than SE module by using both ResNets. It is worth noting that our SimAM does not introduce any additional parameters as compared to SE module. For example, SE-ResNet-50 and SE-ResNet-101 add 2.5 M and 4.7 M more parameters to ResNet-50/101 respectively. These results demonstrate that SimAM is a lightweight module for various vision tasks.

5. Conclusion

In this paper, we propose a new attention module - SimAM, inspired by neuroscience theories in the mammalian brain. In particular, we base on the well-established spatial suppression theory and design an energy function to implement this theory. We also derive a simple solution to the function, where this function is further employed as attentional importance for each neuron within a feature map. Our attention module is implemented guided by this energy function, avoiding too much heuristics. Extensive experiments are conducted to verify the effectiveness and efficiency of the proposed SimAM. Our results show that the proposed SimAM performs comparably pagainst other attention modules in various networks for different vision tasks.

Acknowledgements

We thank all reviewers for their kindly and constructive suggestions. This work is supported by the Key-Area Research and Development Program of Guangzhou (202007030004) China, and also supported by Natural Science Foundation of China (62072482).

References

Aubry, M., Russell, B. C., and Sivic, J. Painting-to-3D Model Alignment via Discriminative Visual Elements. *ACM Transactions on Graphics (ToG)*, 33(2):1–14, 2014.

Bengio, Y., Simard, P., and Frasconi, P. Learning Long-term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

Cao, Y., Xu, J., Lin, S., Wei, F., and Hu, H. Global Context Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

Carrasco, M. Visual Attention: The Past 25 Years. *Vision Research*, 51(13):1484–1525, 2011.

Chatfield, K., Simonyan, K., Vedaldi, A., and Zisserman, A. Return of the Devil in the Details: Delving Deep into Convolutional Nets. *arXiv:1405.3531*, 2014.

Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., Zhang, Z., Cheng, D., Zhu, C., Cheng, T., Zhao, Q., Li, B., Lu, X., Zhu, R., Wu, Y., Dai, J., Wang, J., Shi, J., Ouyang, W., Loy, C. C., and Lin, D. MMDetection: Open MMLab Detection Toolbox and Benchmark. *arXiv:1906.07155*, 2019.

Chollet, F. Xception: Deep Learning with Depthwise Separable Convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1251–1258, 2017.

Chun, M. M., Golomb, J. D., and Turk-Browne, N. B. A Taxonomy of External and Internal Attention. *Annual Review of Psychology*, 62:73–101, 2011.

Dong, X. and Yang, Y. Searching for a Robust Neural Architecture in Four GPU Hours. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1761–1770, 2019.

Feichtenhofer, C. X3D: Expanding Architectures for Efficient Video Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 203–213, 2020.

Feichtenhofer, C., Pinz, A., and Zisserman, A. Convolutional Two-stream Network Fusion for Video Action Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1933–1941, 2016.

Glorot, X. and Bengio, Y. Understanding the Difficulty of Training Deep Feedforward Neural Networks. In *Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. Accurate, Large Minibatch SGD: Training Imagenet in 1 Hour. *arXiv:1706.02677*, 2017.

Guo, Z., Zhang, X., Mu, H., Heng, W., Liu, Z., Wei, Y., and Sun, J. Single Path One-Shot Neural Architecture Search with Uniform Sampling. In *European Conference on Computer Vision*, pp. 544–560. Springer, 2020.

Hariharan, B., Malik, J., and Ramanan, D. Discriminative Decorrelation for Clustering and Classification. In *European Conference on Computer Vision*, pp. 459–472. Springer, 2012.

He, K., Zhang, X., Ren, S., and Sun, J. Identity Mappings in Deep Residual Networks. In *European Conference on Computer Vision*, pp. 630–645. Springer, 2016a.

He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016b.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. Mask R-CNN. In *IEEE International Conference on Computer Vision*, pp. 2961–2969, 2017.

Hillyard, S. A., Vogel, E. K., and Luck, S. J. Sensory Gain Control (Amplification) as a Mechanism of Selective Attention: Electrophysiological and Neuroimaging evidence. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 353(1373): 1257–1270, 1998.

Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al. Searching for MobileNetV3. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1314–1324, 2019.

Hu, J., Shen, L., Albanie, S., Sun, G., and Vedaldi, A. Gather-Excite: Exploiting Feature Context in Convolutional Neural Networks. *arXiv:1810.12348*, 2018a.

Hu, J., Shen, L., and Sun, G. Squeeze-and-Excitation Networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7132–7141, 2018b.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely Connected Convolutional Networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4700–4708, 2017.

Huang, G., Liu, S., Van der Maaten, L., and Weinberger, K. Q. CondenseNet: An Efficient Densenet using Learned Group Convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2752–2761, 2018.

Krizhevsky, A., Hinton, G., et al. Learning Multiple Layers of Features from Tiny Images. 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, 25: 1097–1105, 2012.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Lee, C.-Y., Xie, S., Gallagher, P., Zhang, Z., and Tu, Z. Deeply-Supervised Nets. In *Artificial Intelligence and Statistics*, pp. 562–570. PMLR, 2015.

Lee, H., Kim, H.-E., and Nam, H. SRM: A Style-based Recalibration Module for Convolutional Neural Networks. In *IEEE International Conference on Computer Vision*, pp. 1854–1862, 2019.

Li, X., Sun, W., and Wu, T. Attentive Normalization. *arXiv:1908.01259*, 2019a.

Li, X., Wang, W., Hu, X., and Yang, J. Selective Kernel Networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 510–519, 2019b.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft COCO: Common Objects in Context. In *European Conference on Computer Vision*, pp. 740–755. Springer, 2014.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. Feature Pyramid Networks for Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2117–2125, 2017.

Lin, X., Ma, L., Liu, W., and Chang, S.-F. Context-Gated Convolution. In *European Conference on Computer Vision*, pp. 701–718. Springer, 2020.

Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.-J., Fei-Fei, L., Yuille, A., Huang, J., and Murphy, K. Progressive Neural Architecture Search. In *European Conference on Computer Vision*, pp. 19–34, 2018a.

Liu, C., Chen, L.-C., Schroff, F., Adam, H., Hua, W., Yuille, A. L., and Fei-Fei, L. Auto-DeepLab: Hierarchical Neural Architecture Search for Semantic Image Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 82–92, 2019.

Liu, H., Simonyan, K., and Yang, Y. DARTs: Differentiable Architecture Search. *arXiv:1806.09055*, 2018b.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. SSD: Single Shot Multibox Detector. In *European Conference on Computer Vision*, pp. 21–37. Springer, 2016.

Ren, S., He, K., Girshick, R., and Sun, J. Faster R-CNN: Towards Real-time Object Detection with Region Proposal Networks. *arXiv:1506.01497*, 2015.

Reynolds, J. H. and Chelazzi, L. Attentional Modulation of Visual Processing. *Annu. Rev. Neurosci.*, 27:611–647, 2004.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3): 211–252, 2015.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 618–626, 2017.

Simonyan, K. and Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv:1409.1556*, 2014.

Srivastava, R. K., Greff, K., and Schmidhuber, J. Highway Networks. *arXiv:1505.00387*, 2015.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going Deeper with Convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, 2015.

Tan, M. and Le, Q. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *International Conference on Machine Learning*, pp. 6105–6114. PMLR, 2019.

Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., and Le, Q. V. MnasNet: Platform-aware Neural Architecture Search for Mobile. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2820–2828, 2019.

Tan, M., Pang, R., and Le, Q. V. EfficientDet: Scalable and Efficient Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10781–10790, 2020.

Wang, F., Jiang, M., Qian, C., Yang, S., Li, C., Zhang, H., Wang, X., and Tang, X. Residual Attention Network for Image Classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3156–3164, 2017.

Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., and Van Gool, L. Temporal Segment Networks for Action Recognition in Videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(11):2740–2755, 2018a.

Wang, Q., Wu, B., Zhu, P., Li, P., Zuo, W., and Hu, Q. ECA-Net: Efficient Channel Attention for Deep Convolutional Neural Networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 11534–11542, 2020.

Wang, X., Girshick, R., Gupta, A., and He, K. Non-Local Neural Networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7794–7803, 2018b.

Webb, B. S., Dhruv, N. T., Solomon, S. G., Tailby, C., and Lennie, P. Early and Late Mechanisms of Surround Suppression in Striate Cortex of Macaque. *Journal of Neuroscience*, 25(50):11666–11675, 2005.

Woo, S., Park, J., Lee, J.-Y., and So Kweon, I. CBAM: Convolutional Block Attention Module. In *European Conference on Computer Vision*, pp. 3–19, 2018.

Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y., and Keutzer, K. FBNet: Hardware-aware Efficient ConvNet Design via Differentiable Neural Architecture Search. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10734–10742, 2019.

Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. Aggregated Residual Transformations for Deep Neural Networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1492–1500, 2017.

Yang, Z., Zhu, L., Wu, Y., and Yang, Y. Gated Channel Transformation for Visual Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 11794–11803, 2020.

Zagoruyko, S. and Komodakis, N. Wide Residual Networks. *arXiv:1605.07146*, 2016.

Zeiler, M. D. and Fergus, R. Visualizing and Understanding Convolutional Networks. In *European Conference on Computer Vision*, pp. 818–833. Springer, 2014.

Zhang, R. Making Convolutional Networks Shift-Invariant Again. In *International Conference on Machine Learning*, pp. 7324–7334. PMLR, 2019.

Zoph, B. and Le, Q. V. Neural Architecture Search with Reinforcement Learning. *arXiv:1611.01578*, 2016.

SimAM: A Simple, Parameter-Free Attention Module for Convolutional Neural Networks (Supplementary Materials)

In this supplementary material, we first compare more recently similar re-calibration modules by using ResNet-50 (He et al., 2016) as backbone networks. These modules include selective kernel (SK) (Li et al., 2019b), attentive normalization (AttNorm) (Li et al., 2019a), dropout (Srivastava et al., 2014) as well as Swish function (Ramachandran et al., 2017). We also insert SimAM after all convolutional layers (SimAM-all). Moreover, we compare SE (Hu et al., 2018) with the proposed SimAM on another two variants of ResNet, ResNext-50 (32x4d) and Res2Net (Gao et al., 2019). We follow the standard training pipeline (100 epochs) to train all models from **scratch**. Results are shown in Table 1. As one can see, our SimAM-all achieves the best results and our SimAM also performs comparably against other modules based on ResNet-50 network. Moreover, the SimAM can also enhance the representation power of ResNeXt-50 and Res2Net-50.

Table 1. Top-1 accuracies (%) on ImageNet dataset with different options .

Model	Baseline	+SimAM	+SimAM-all	+SK	+AttNorm	+Dropout0.1	+Swish
ResNet-50	76.34%	77.46%	77.62%	77.50%	77.12%	75.51%	76.71%
Model	Baseline	+SimAM	+SE	Model	Baseline	+SimAM	+SE
ResNeXt-50	77.47%	78.00%	77.96%	**Res2Net-50**	77.94%	78.54%	78.41%

Second, we visualize the estimated 3-D weights by the proposed SimAM based on ResNet-50 network to demonstrate our mechanism. For each stage (model.layer1 to model.layer4) in ResNet-50, we show the weights generated in the last block on randomly selected 64 images from the ImageNet validation set. To show different channels of our 3-D weights, for each image, we randomly select 6 channels and show their selected index on each weight image. As one can see, our SimAM can attend some important points, curve (first two pictures), as well as some discriminative components (last two pictures).

References

Gao, S., Cheng, M.-M., Zhao, K., Zhang, X.-Y., Yang, M.-H., and Torr, P. H. Res2net: A new multi-scale backbone architecture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Hu, J., Shen, L., and Sun, G. Squeeze-and-excitation networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7132–7141, 2018.

Li, X., Sun, W., and Wu, T. Attentive normalization. *arXiv:1908.01259*, 2019a.

Li, X., Wang, W., Hu, X., and Yang, J. Selective kernel networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 510–519, 2019b.

Ramachandran, P., Zoph, B., and Le, Q. V. Searching for activation functions. *arXiv:1710.05941*, 2017.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

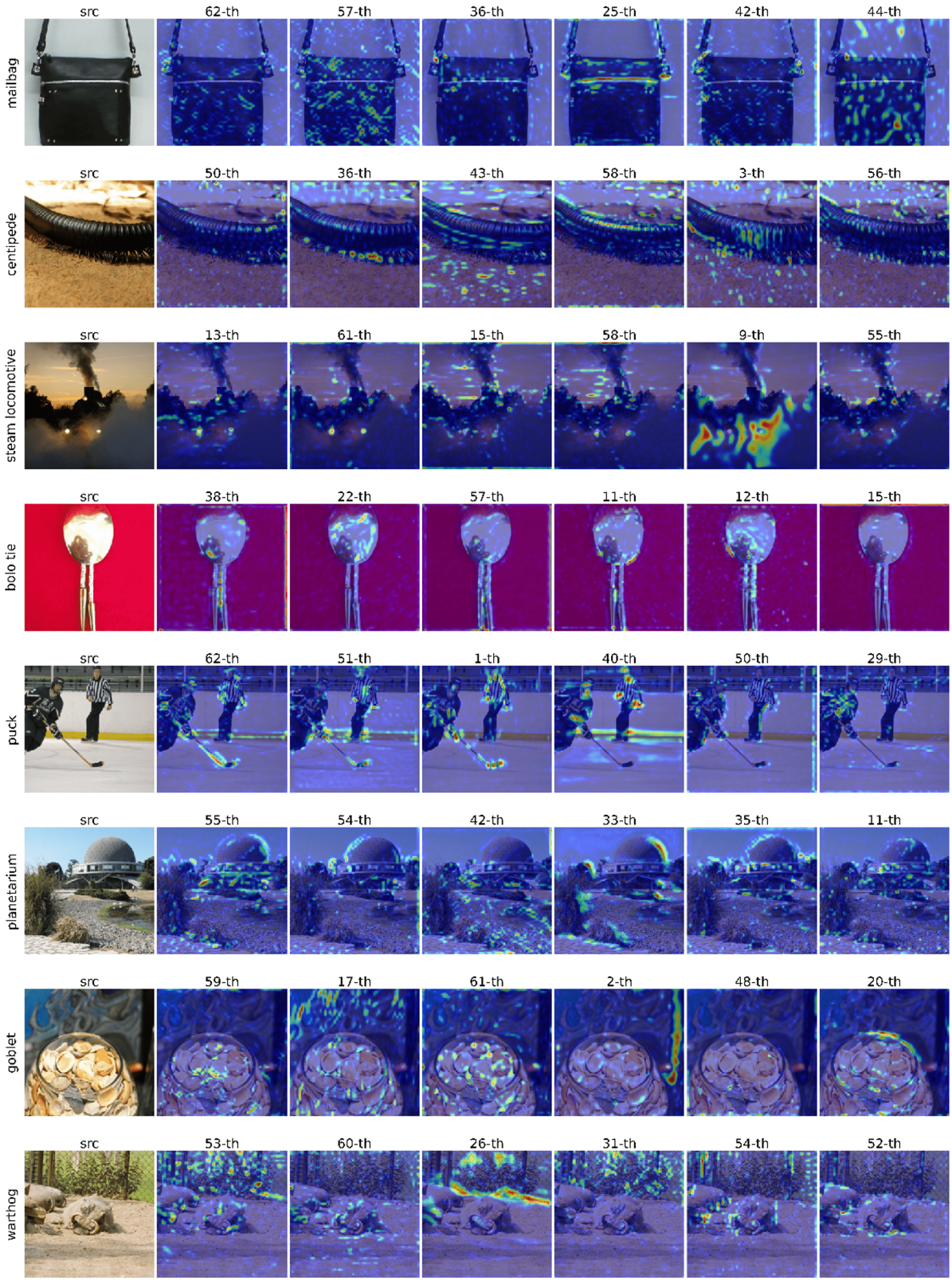


Figure 1. Visualization of attention weights generated by our method (from the last block in **"model.layer1"**). Each row represents one image with its correspondence attention weights. We randomly select different (shown as "X"-th) channels from our 3-D weights to show.

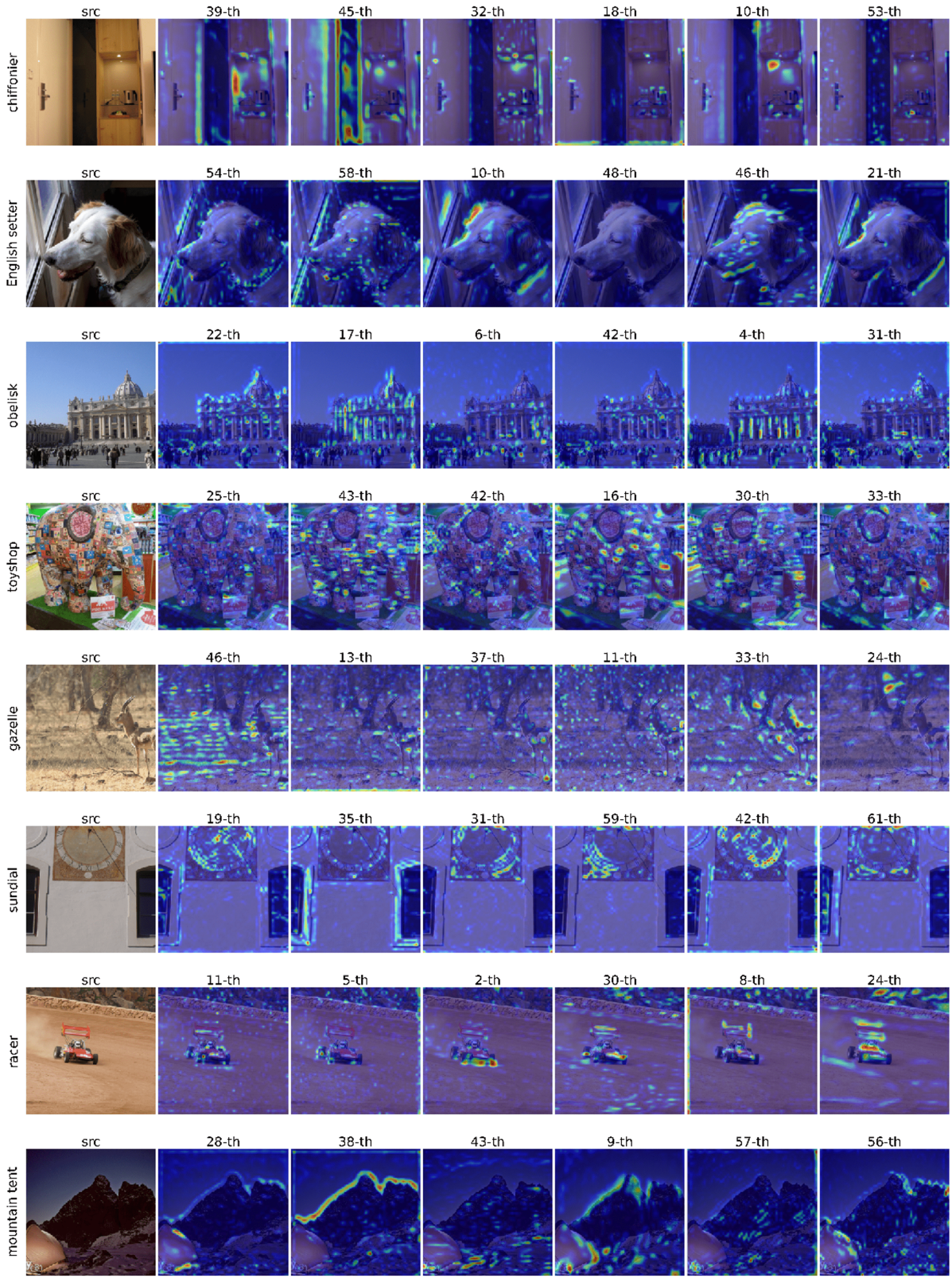


Figure 2. Visualization of attention weights generated by our method (from the last block in **"model.layer1"**). Each row represents one image with its correspondence attention weights. We randomly select different (shown as "X"-th) channels from our 3-D weights to show.

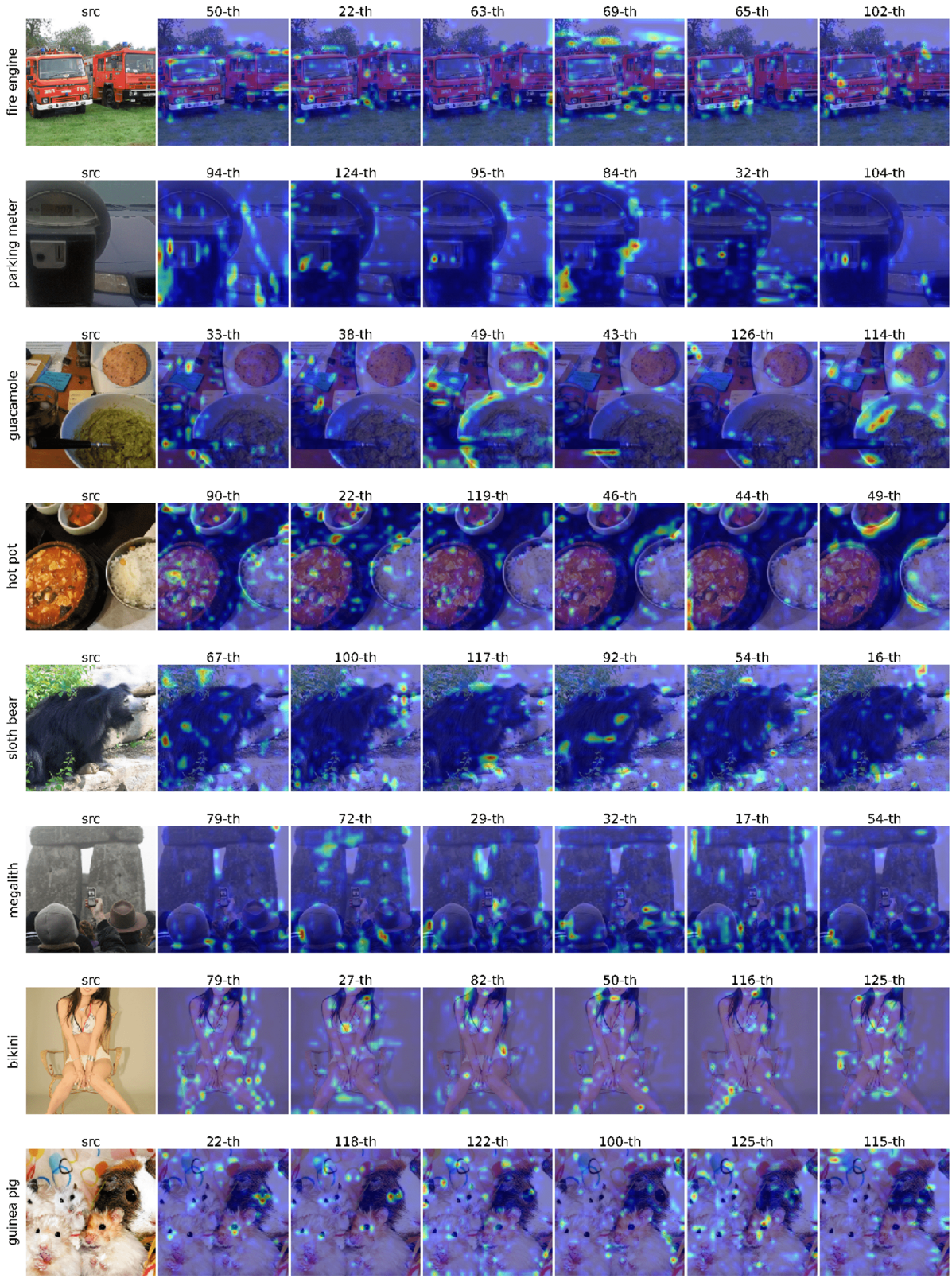


Figure 3. Visualization of attention weights generated by our method (from the last block in **"model.layer2"**). Each row represents one image with its correspondence attention weights. We randomly select different (shown as "X"-th) channels from our 3-D weights to show.

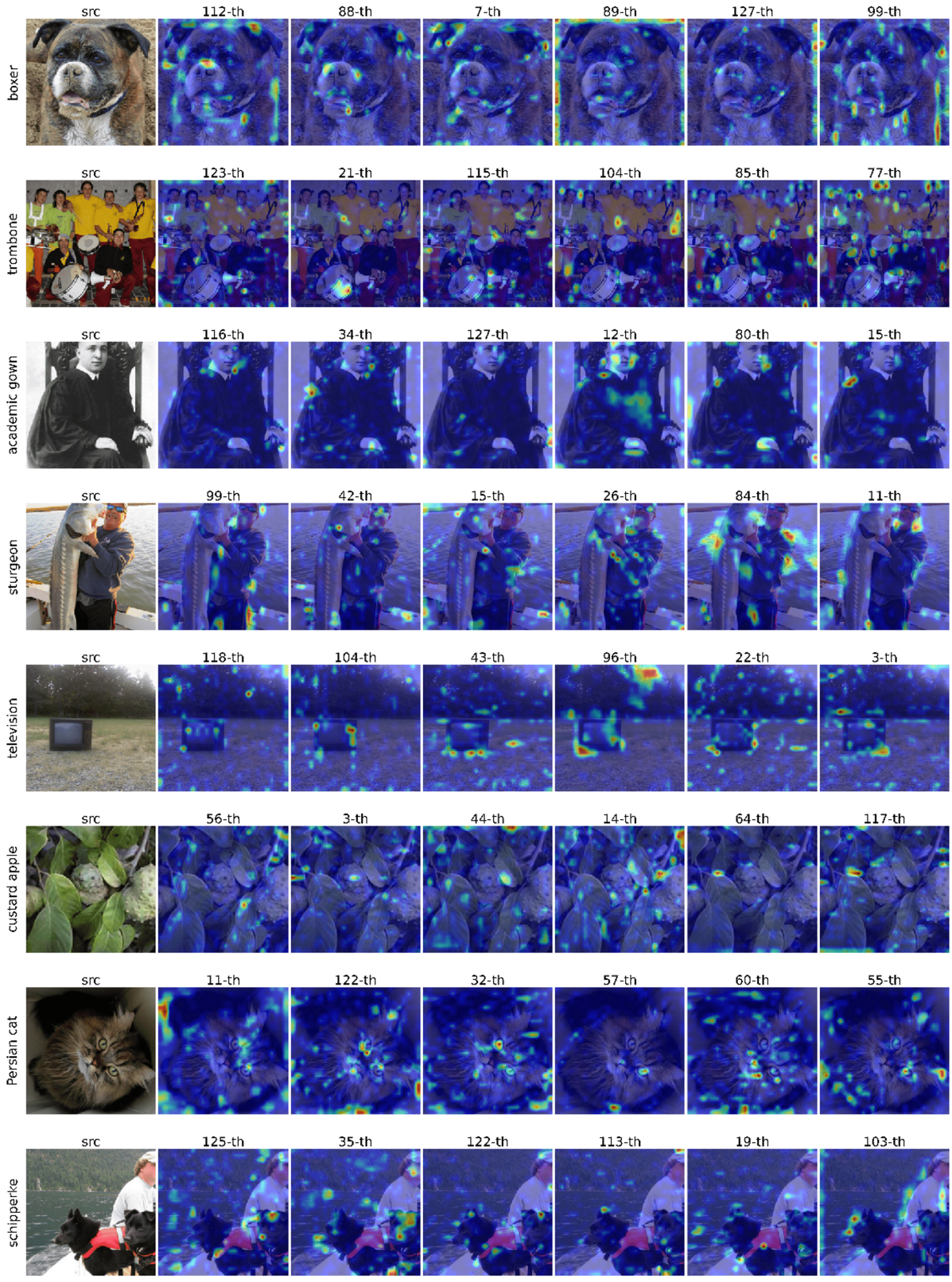


Figure 4. Visualization of attention weights generated by our method (from the last block in **"model.layer2"**). Each row represents one image with its correspondence attention weights. We randomly select different (shown as "X"-th) channels from our 3-D weights to show.

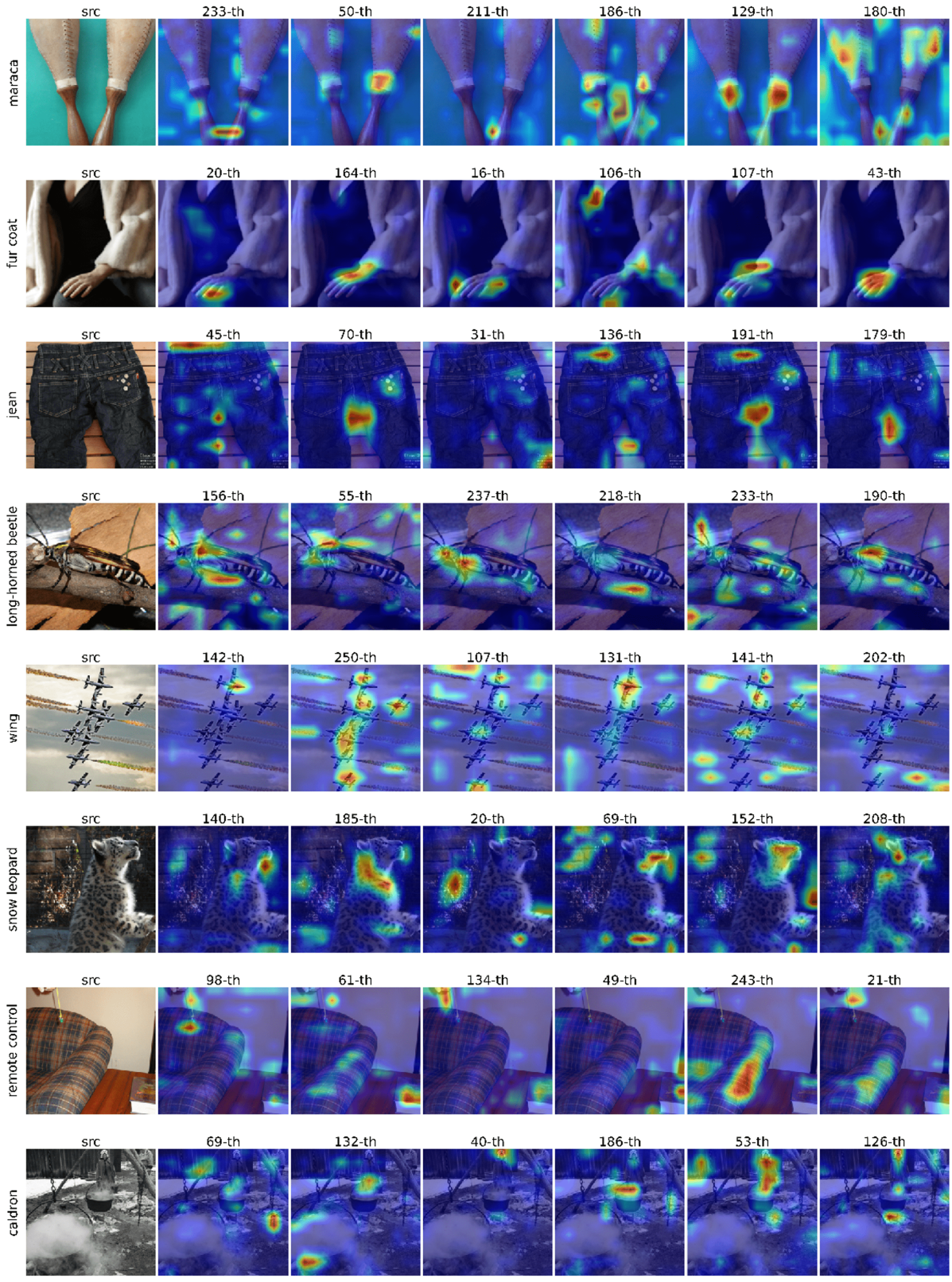


Figure 5. Visualization of attention weights generated by our method (from the last block in **"model.layer3"**). Each row represents one image with its correspondence attention weights. We randomly select different (shown as "X"-th) channels from our 3-D weights to show.

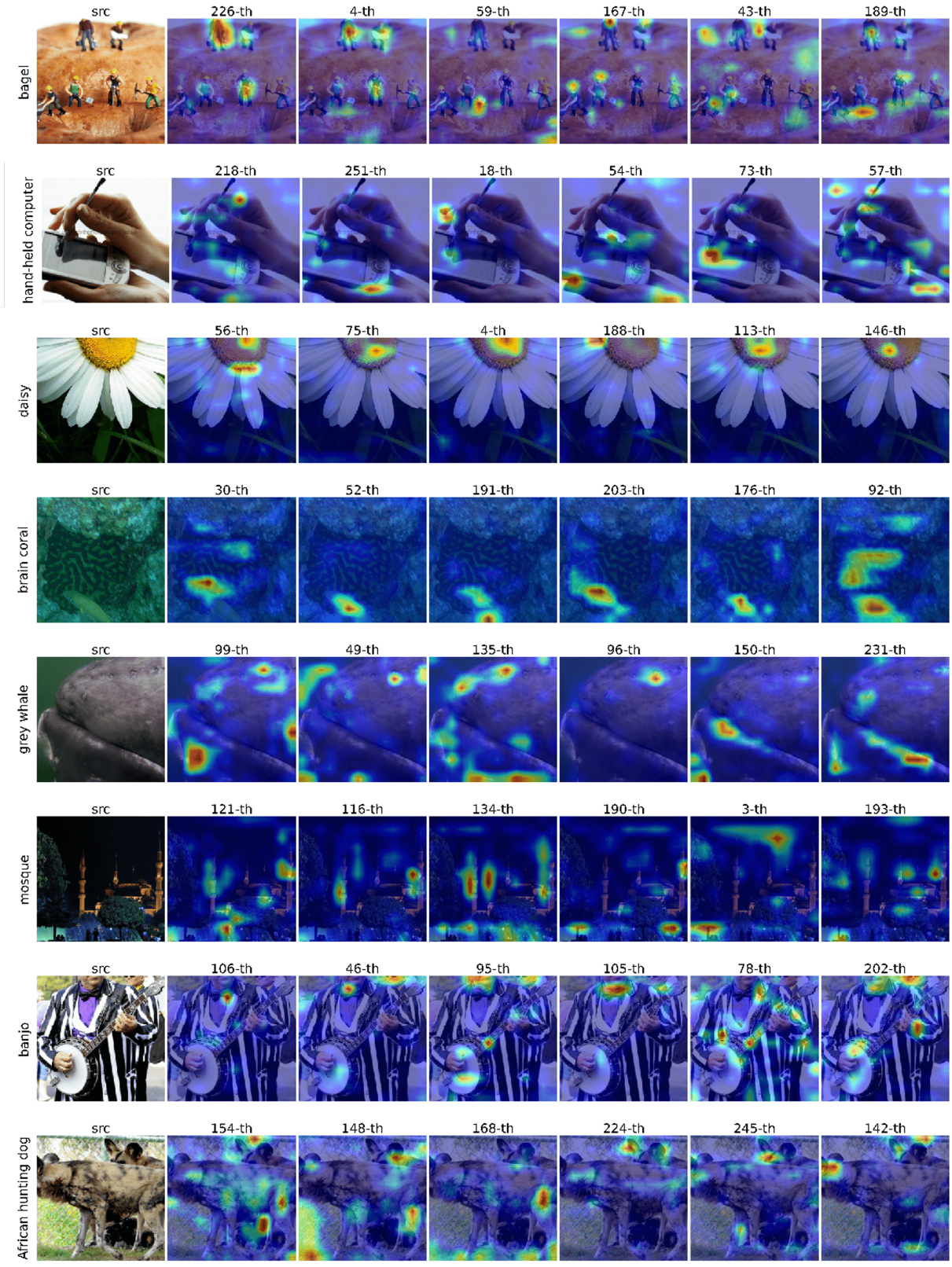


Figure 6. Visualization of attention weights generated by our method (from the last block in **"model.layer3"**). Each row represents one image with its correspondence attention weights. We randomly select different (shown as "X"-th) channels from our 3-D weights to show.

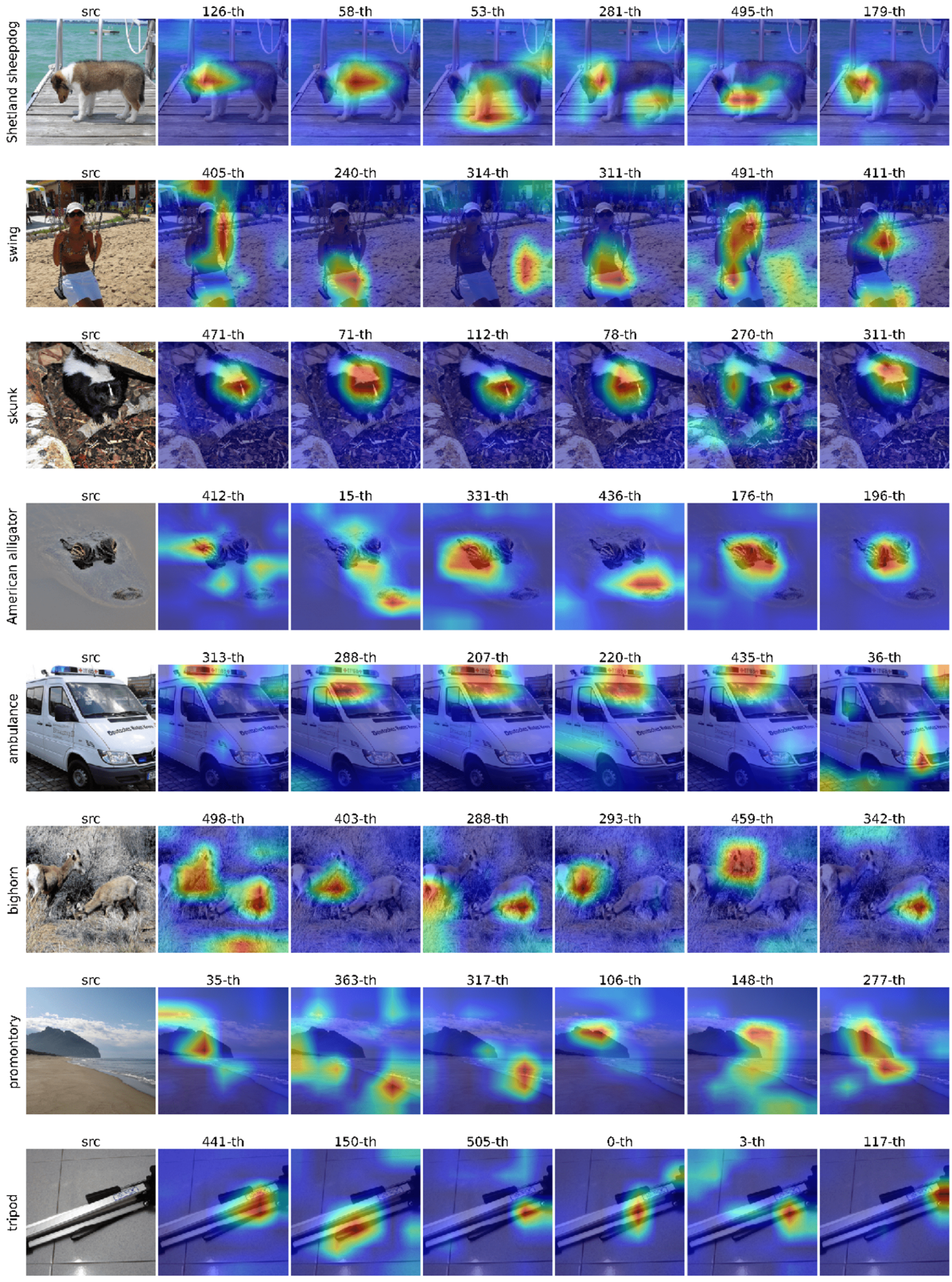


Figure 7. Visualization of attention weights generated by our method (from the last block in **"model.layer4"**). Each row represents one image with its correspondence attention weights. We randomly select different (shown as "X"-th) channels from our 3-D weights to show.

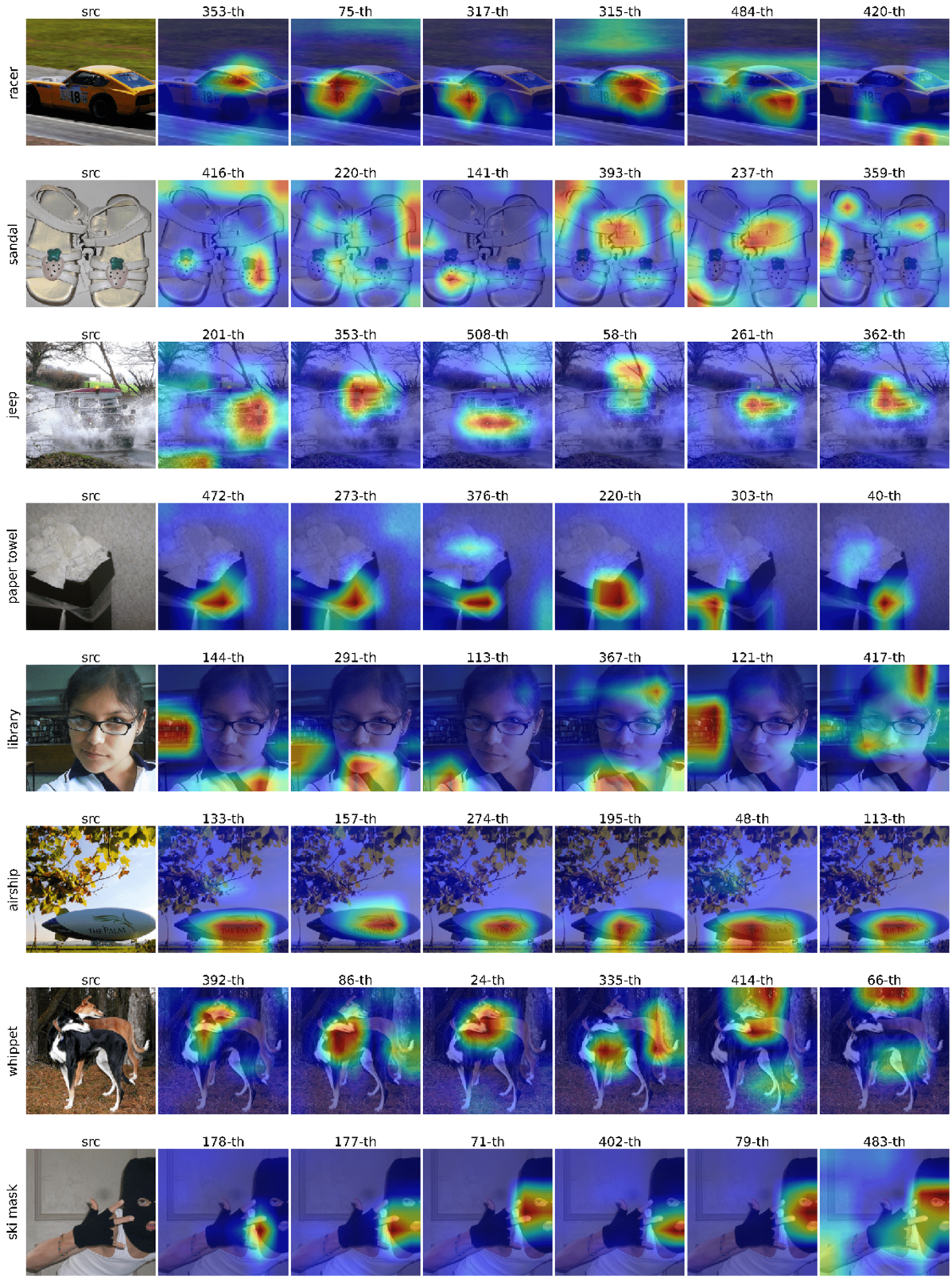


Figure 8. Visualization of attention weights generated by our method (from the last block in **"model.layer4"**). Each row represents one image with its correspondence attention weights. We randomly select different (shown as "X"-th) channels from our 3-D weights to show.