BAM: Bottleneck Attention Module

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

[1] Lunit Inc.
Seoul, Korea

[2] School of Electrical Engineering
Korea Advanced Institute of Science and Technology
Daejeon, Korea

[3] Adobe Research
San Hose, CA, USA

1 Implementation Details

In order to perform fair comparisons, we have created our benchmark platform in the Pytorch fromework [10] based on the open-sourced projects [1, 2, 5, 7, 8, 10, 12, 13]. Our unified framework has allowed us to simply plug our module (BAM) while keeping all other settings same. All the networks are trained using Stochastic Gradient Descent. On CIFAR, we train for 300 epochs. The initial learning rate is set to 0.1, and is divided by 10 at 50% and 75% of the total number of training epochs. On ImageNet, we train models for 90 epochs. The learning rate is initially set to 0.1, and is decreased by 10 times at epoch 30 and 60. On the MS COCO detection dataset, we take our ImageNet pretrained models and train for 490K iterations. The initial learning rate is set to 0.001 and lowered by 10 times at 350K iteration. We use a weight decay of 10^{-4} and a Nesterov momentum [11] of 0.9 without dampening. Throughout the experiments, we used a fixed random seed.

2 The effectiveness of BAM

In Fig. 1, 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 [3].

*Both authors have equally contributed.

†The work was done while the author was at KAIST.

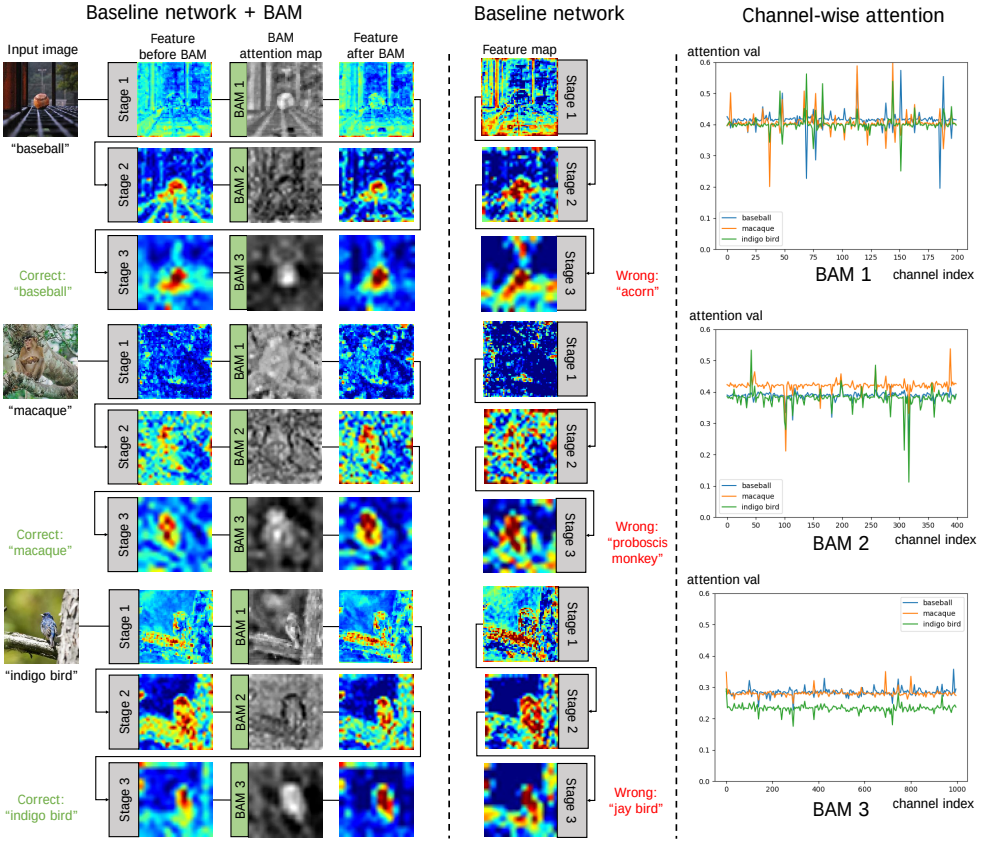

Figure 1: **Visualizing the attention process of BAM.** In order to provide an intuitive understanding of BAM's role, we visualize image classification process using the images that baseline (ResNet50) fails to classify correctly while the model with BAM succeeds. Using the models trained on ImageNet-1K, 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.

As shown in Fig. 1, 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 [4, 6, 9], suggesting that BAM drives the network to mimic the human visual system effectively.

3 Additional Visualization Results

We show more visualization results of attention process. All the results in this section are produced from ResNet50 baseline (with BAM) tested with the ImageNet validation set. In Sec. 3.1, correctly classified examples with BAM are listed with intermediate activations and attention maps. We have selected examples where the baseline with BAM succeeds and the baseline fails. On the contrary, in Sec. 3.2, examples are selected where the baseline with BAM fails and the baseline succeeds. Figures are best viewed in color.

3.1 Successful Cases with BAM

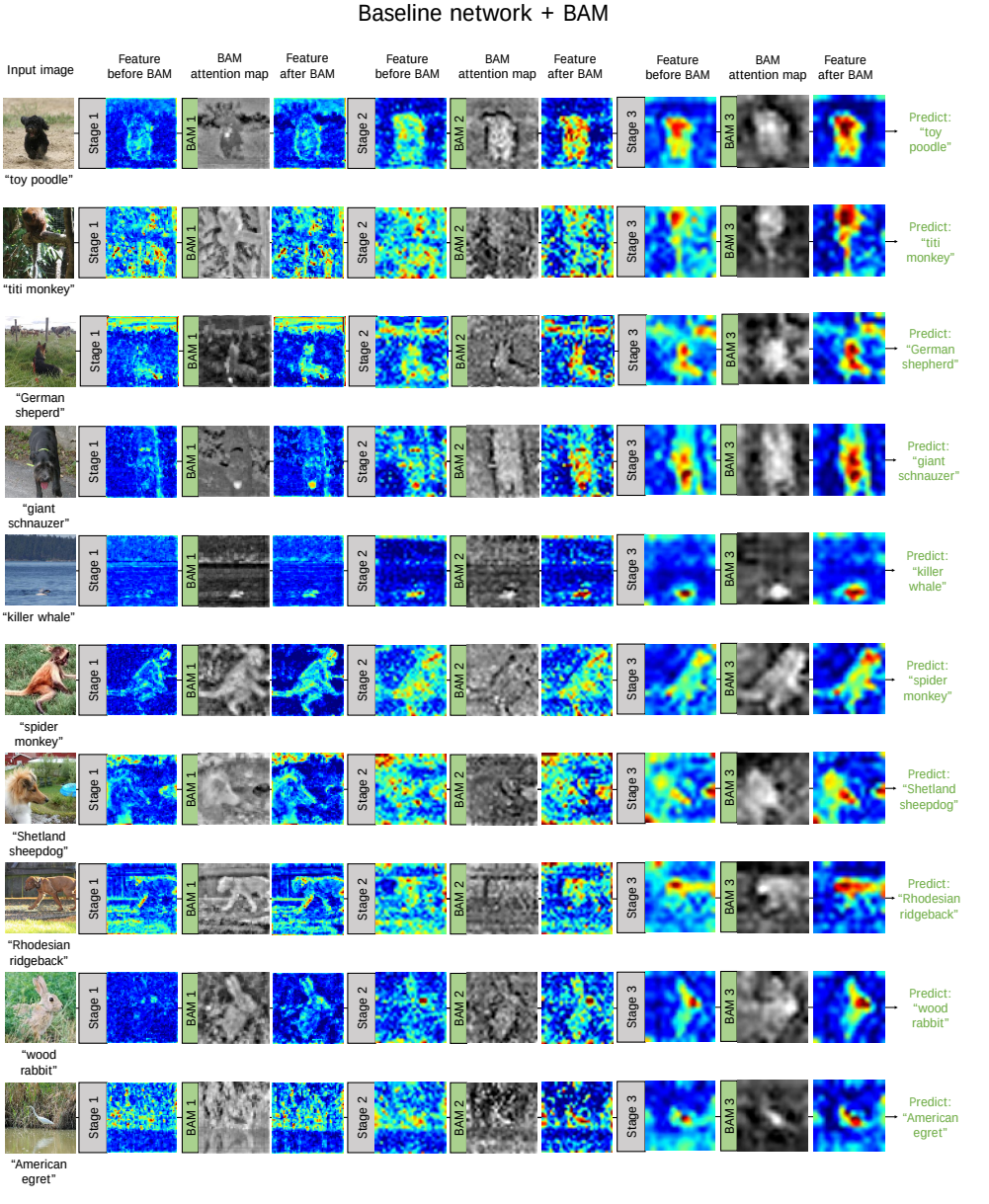

Figure 2: **Successful cases with BAM.** The shown examples are the intermediate activations and BAM attention maps when the baseline+BAM succeeds and the baseline fails. Figure best viewed in color.

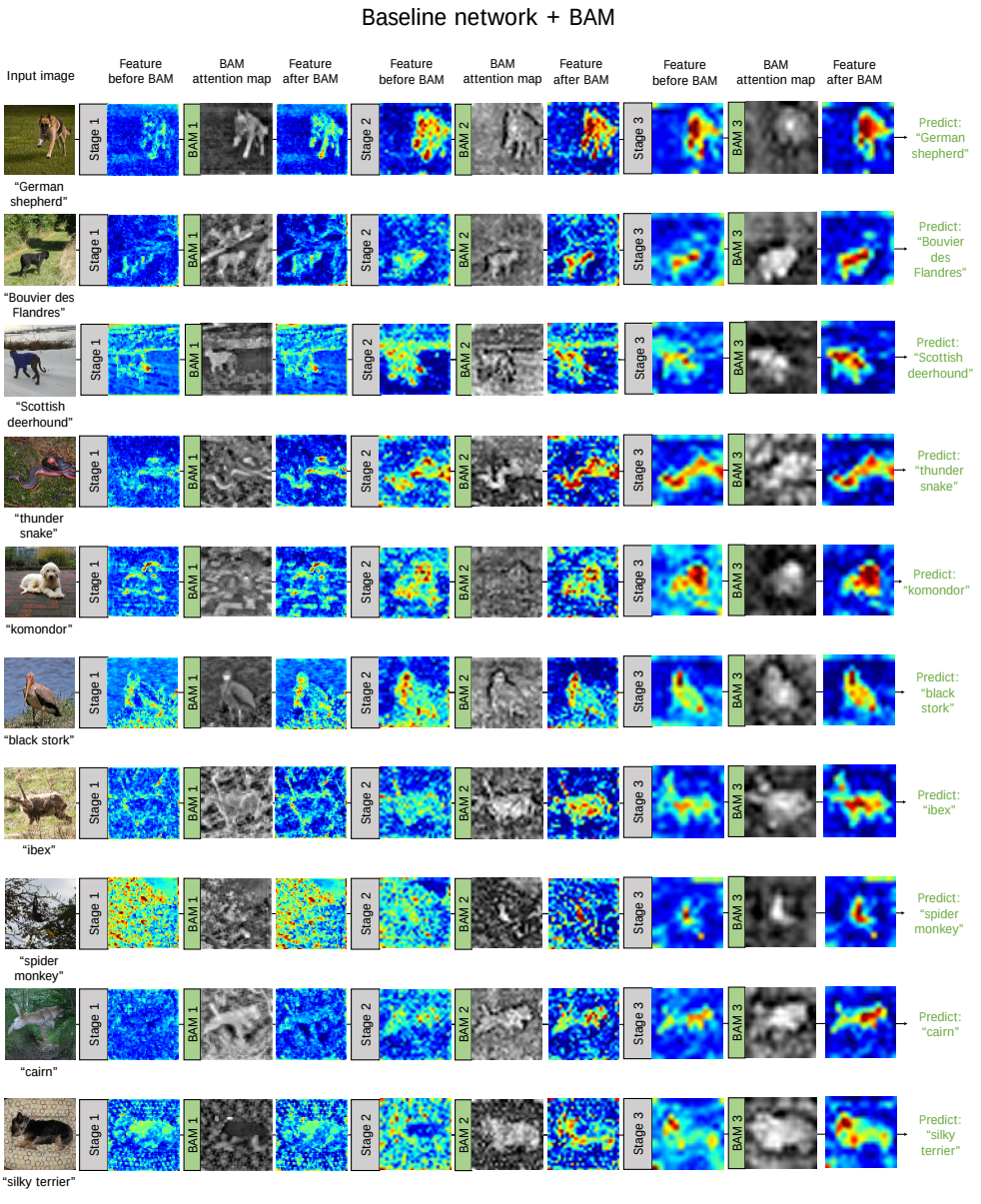

Figure 3: **Successful cases with BAM.** The shown examples are the intermediate activations and BAM attention maps when the baseline+BAM succeeds and the baseline fails. Figure best viewed in color.

3.2 Failure Cases with BAM

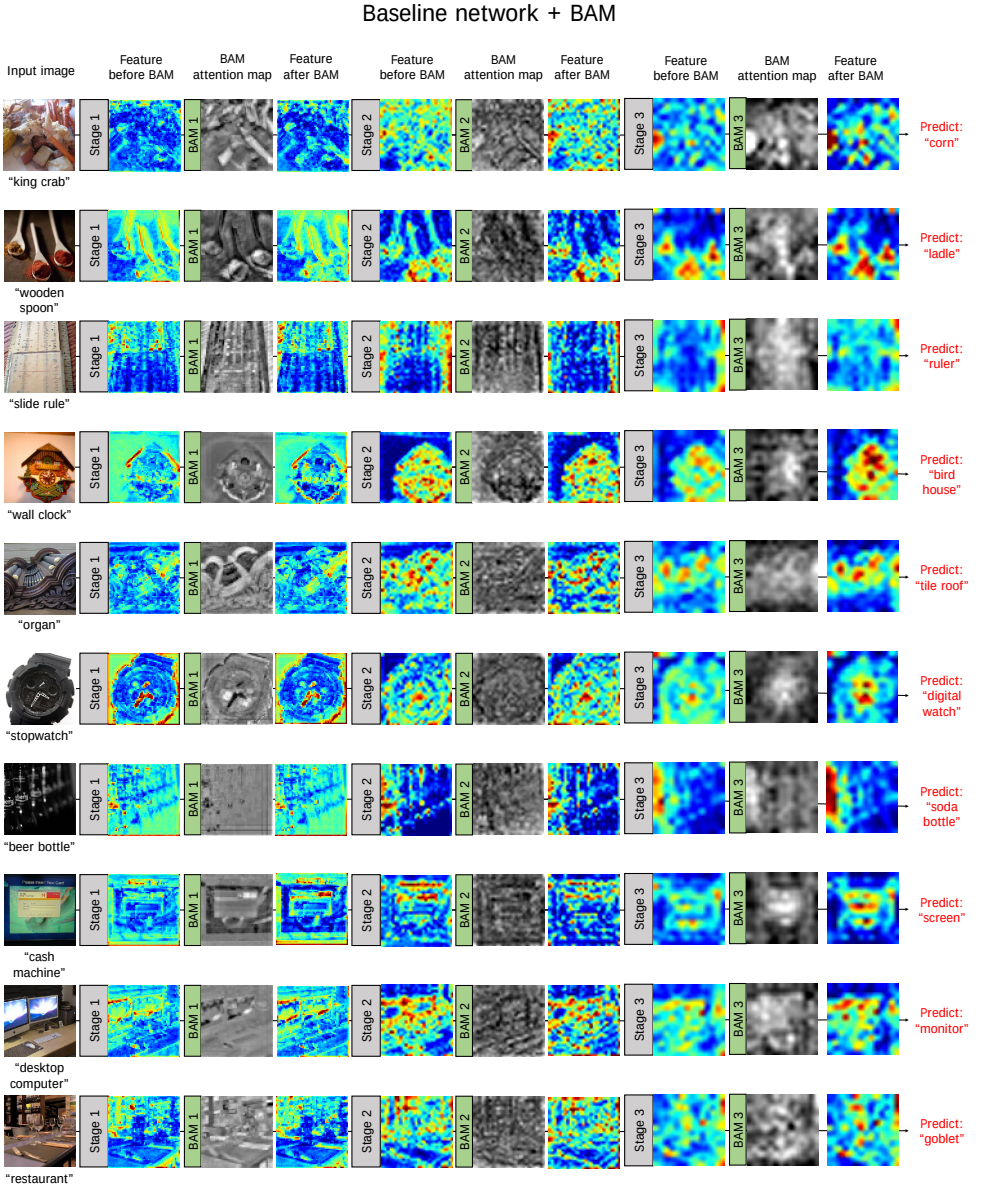

Figure 4: **Failure cases with BAM.** The shown examples are the intermediate activations and BAM attention maps when baseline+BAM fails and baseline succeeds. Figure best viewed in color.

References

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

[2] Xinlei Chen and Abhinav Gupta. An implementation of faster rcnn with study for region sampling. *arXiv preprint arXiv:1702.02138*, 2017.

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

[4] David H Hubel and Torsten N Wiesel. Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, 148(3):574–591, 1959.

[5] Jason Kuen. Wideresnet pytorch implementation. https://github.com/xternalz/WideResNet-pytorch.git, 2017.

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

[7] marvis. Mobilenet pytorch implementation. https://github.com/marvis/pytorch-mobilenet, 2017.

[8] PyTorch. torchvision. https://github.com/pytorch/vision, 2017.

[9] Maximilian Riesenhuber and Tomaso Poggio. Hierarchical models of object recognition in cortex. *Nature neuroscience*, 2(11):1019–1025, 1999.

[10] Pau Rodriguez. Resnext pytorch implementation. https://github.com/prlz77/ResNeXt.pytorch, 2017.

[11] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

[12] Andreas Veit. Densenet pytorch implementation. https://github.com/andreasveit/densenet-pytorch.git, 2017.

[13] Wei Yang. Preresnet pytorch implementation. https://github.com/bearpaw/pytorch-classification, 2017.