Attentional Feature Fusion: supplemental document

1. IMPLEMENTATION DETAILS

All network architectures in this work are implemented based on MXNet [1] and GluonCV [2]. Since most of the experimental architectures cannot take advantage of pre-trained weights, each implementation is trained from scratch for fairness. We have introduced most of the experimental settings in Table 2 of the manuscript. Here, in the supplemental document, we introduce the left settings that not mentioned before.

For the experiments on the CIFAR-100 dataset, the weight decay is 1e-4, and we decay the learning rate by a factor of 0.1 at epoch 300 and 350.

For the experiments on the ImageNet, we use the label smoothing trick and a cosine annealing schedule for the learning rate without weight decay.

For the semantic segmentation experiment, the StopSign dataset is a subset of the COCO dataset [3], which has a large scale variation issue, as shown in Fig. S1. We use the cross entropy as loss function and the mean intersection over union (mIoU) as evaluation metric.

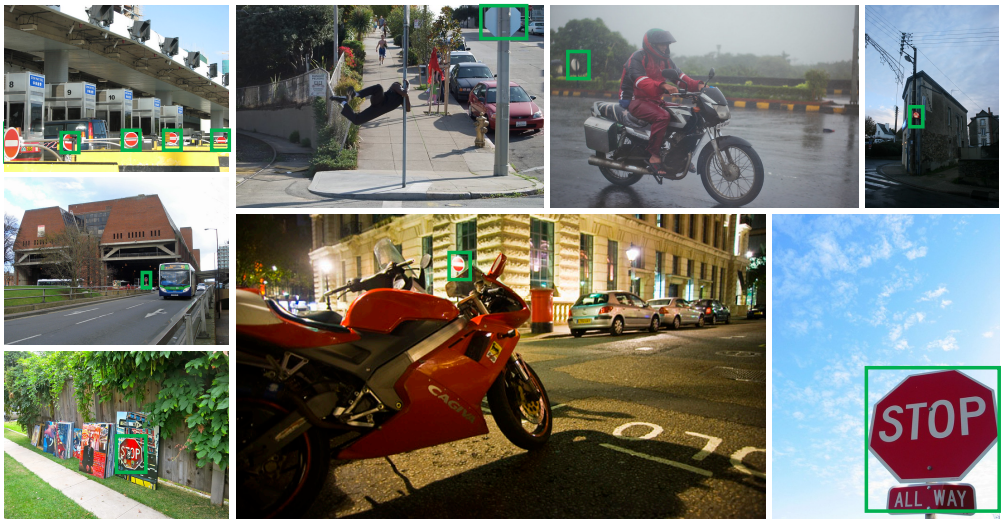


Fig. S1. Illustration for the StopSign dataset

It should be noted that the proposed networks in Table 5 and Table 6 are trained with mixup [4]. The rest experiments, including all the ablation studies and the exprimental results in Figure 7 (in the manuscript) are trained without mixup.

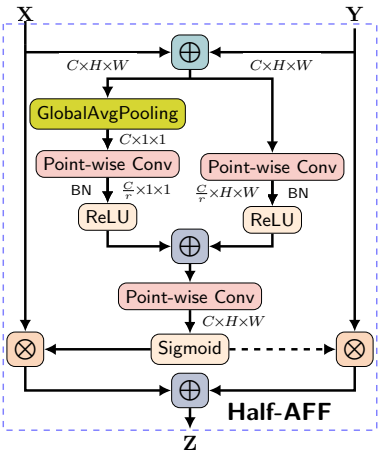
2. FUSION STRATEGY FOR THE LOCAL AND GLOBAL CONTEXTS INSIDE ATTENTION MODULE

We also investigate the fusion strategy for the local and global contexts inside the attention module. We explored four strategies as shown in Fig. S2, in which:

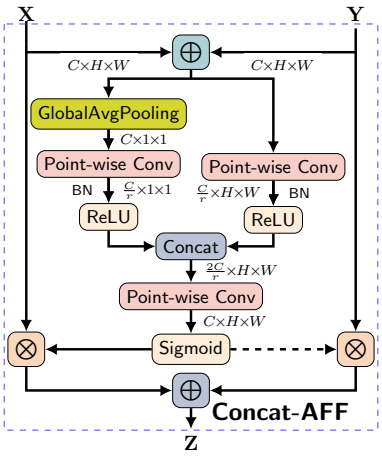
1. Half-AFF, AFF, and Iterative AFF apply addition to fuse the local and global contexts, which allocate the same weights (a constant 0.5) for local and global contexts.

2. Concat-AFF concatenates the local and global contexts followed by a point-wise convolution, in which the fusing weights are learned during training and fixed after training.

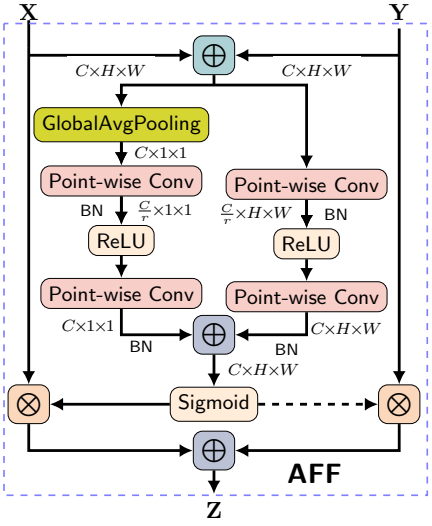
3. Recursive AFF allocates dynamic fusion weights for the local and global contexts during inference based on the proposed MS-CAM.



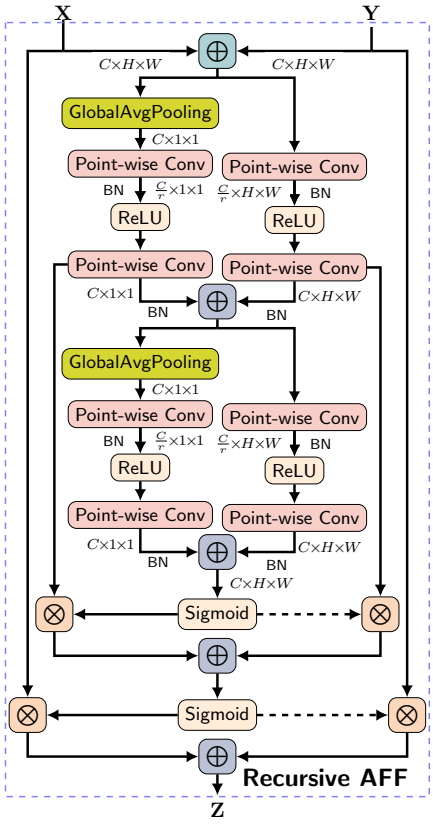
(a) Half-AFF



(b) Concat-AFF



(c) AFF



(d) Recursive AFF

Fig. S2. Architectures for the ablation study on the fusion manner of the local and global channel contexts.

Table S1 provides the experimental results of these modules on CIFAR-100, from which it can be seen that the iterative AFF (iAFF) module presented in the manuscript achieves the best performance. On the contrary, the Recursive AFF which can dynamically allocate fusion weights for local and global contexts are almost the worst among these modules. We believe the reason is that Recursive AFF has two successive nested Sigmoid functions (see Fig. S2(d)), which increases the difficulty in optimization due to Sigmoid's saturation function form, whereas the iterative AFF presented in the manuscript does not suffer from this problem.

AFF and Concat-AFF have a very similar performance. Therefore, for simplicity, we choose the squeeze-and-excitation form (current MS-CAM module) instead of the Inception-style form (Concat-AFF) for the proposed attentional feature fusion. In future work, we will investigate their performance difference on larger datasets like ImageNet. However, this point is not the main issue that we would like to discuss in the manuscript, so we didn't include this part in the manuscript.

Table S1. Experimental results for the ablation study on the fusion manner of the local and global channel contexts on CIFAR-100

Module	Fusion weights of local and global channel contexts	$b = 1$	$b = 2$	$b = 3$
Half-AFF	Constant, 0.5 for each	0.759	0.798	0.813
Concat-AFF	Learned, fixed after training	0.765	0.792	0.817
AFF	Constant, 0.5 for each	0.764	0.799	0.816
Recursive AFF	Dynamic, depending on the local and global channel contexts	0.764	0.797	0.812
Iterative AFF	Constant, 0.5 for each	**0.772**	**0.807**	**0.822**

3. ANALYSIS ON THE FLOPS

The point-wise convolution inside our multi-scale channel attention module can bring additional FLOPs, but at a marginal level, not a significant magnitude. The FLOPs of our AAF-ResNet-50 is 4.3 GFlops, and the Flops of ResNet-50 in our implementation is 4.1 GFlops. Actually, depending on how many tricks are used in ResNet, the Flops of ResNet-50 can vary from 3.9 GFlops to 4.3 GFlops [2]. Therefore, taking ResNet-50 vs our AFF-ResNet-50 for example, integrating the AFF module only brings additional 4.88% Flops from 4.1 GFlops to 4.3 GFlops. Considering the performance boost by the AFF module, we think additional 4.88% Flops is a good trade-off.

Given an output channel number C and the size $H \times W$ of a output feature map, if the input channel number and output channel number are the same, the Flops of a 3×3 convolution layer is $18C^2HW$ (multiplication and addition), and a ResBlock consists of two or three convolution layers. Meanwhile, the Flops of two point-wise convolutions of a bottleneck structure is $\frac{2}{r}C^2HW$, where $r = 4$ or $r = 16$ depending on the dataset and network. Therefore, comparing the Flops of convolutions in the host network, the Flops brought by the AFF module is marginal.

In Table S2, we list the Flops of convolutions in BasicResBlock / BottleneckResBlock, Flops of point-wise convolution in our AFF module, and the relative increasing percentage. It can be seen that the maximum additional flops brought by the AFF module in percentage is around 7.7% if we use AFF module in each ResBlock from beginning to end. However, it is not necessary to replace every ResBlock with AFF-ResBlock. In our AFF-ResNet, we do this replacement from the middle of the network (last two stages), while leaving the first two stages the original BottleneckResBlock. It further reduces the Flops of AFF-ResNet-50.

To conclude, the AFF module will bring additional Flops but at a marginal level, around 3% to 5%. We think it is a good trade-off since the AFF module boosts the representation power of the convolution networks.

Table S2. Additional Flops brought by the proposed AFF module in an AFF-ResBlock

ResBlock Type	Layer doubling channel number ?	Flops of Conv in ResBlock	Flops of Point-wise Convin AFF module	Percentage
BasicResBlock (CIFAR, $r = 4$)	Yes	$27C^2HW$	C^2HW	3.70%
	No	$36C^2HW$	C^2HW	2.78%
BottleneckResBlock (ImageNet, $r = 16$)	Yes	$51C^2HW$	$4C^2HW$	7.84%
	No	$52C^2HW$	$4C^2HW$	7.69%

REFERENCES

1. Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems. In *In Neural Information Processing Systems, Workshop on Machine Learning Systems*, volume abs/1512.01274, 2015.
2. Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 558–567, 2019.
3. Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pages 740–755, Cham, 2014.
4. Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.