Outline

01 Motivation

02 what and how

03 Experiments

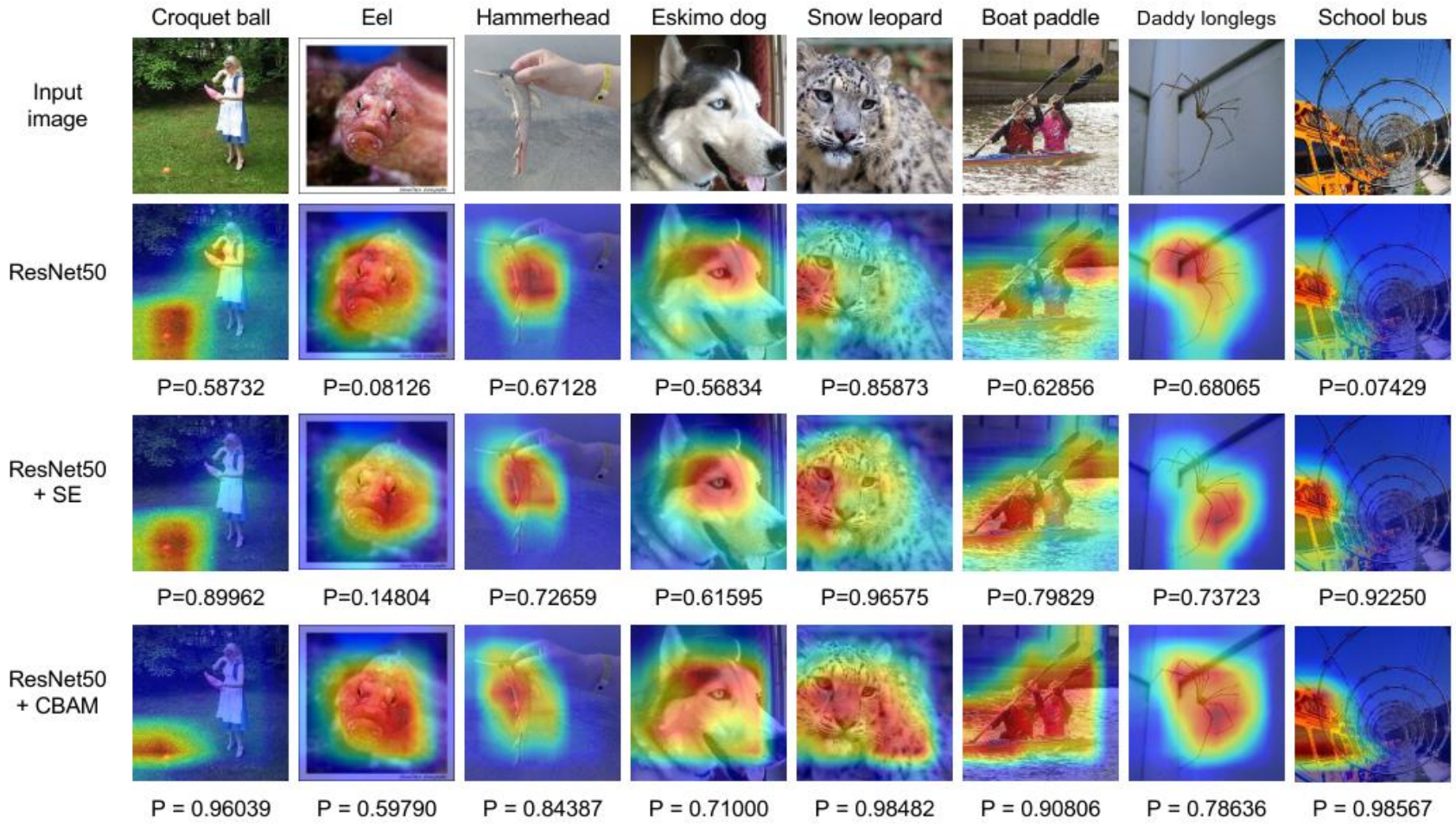
04 similar work

motivation

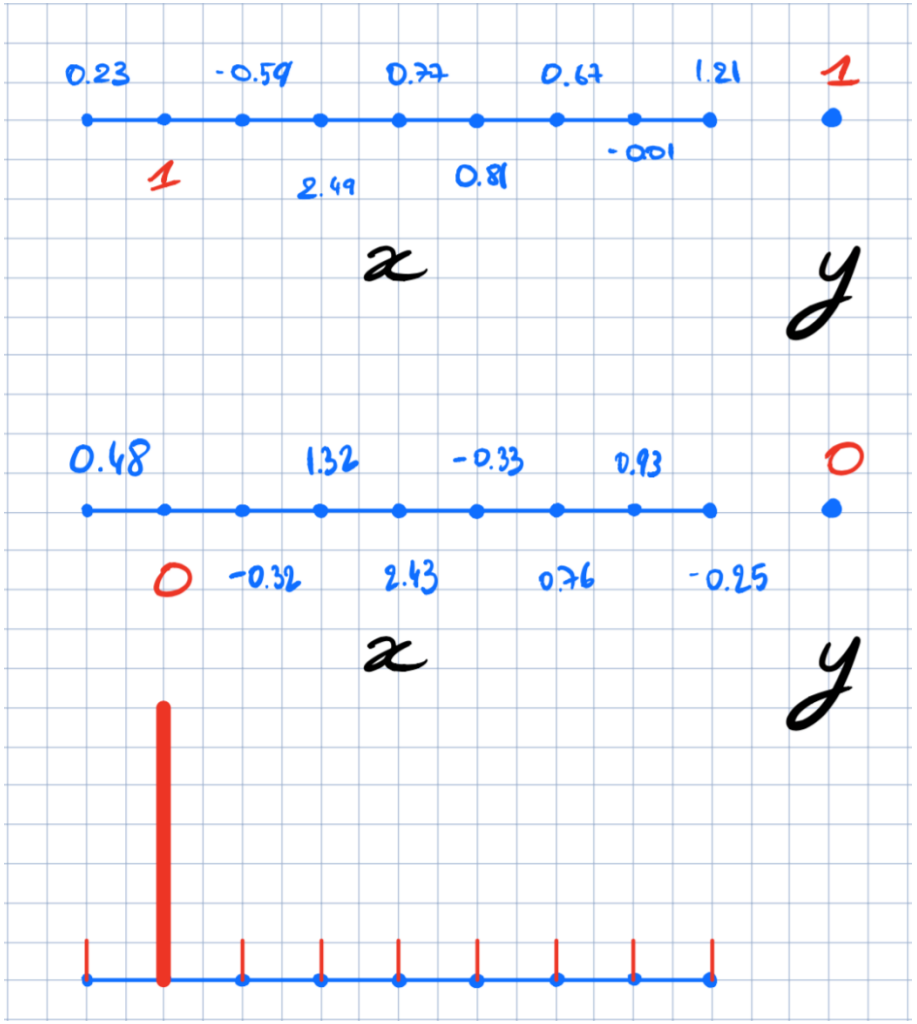
- Network engineering>
  - Depth: VGG, ResNet
  - Width: GoogLeNet
  - Cardinality: Xception ResNeXt

This paper investigate a different factor: **attention**
to increase representation power by using attention mechanism:
focusing on important features and suppressing unnecessary ones.

Example of attention



Example of attention



- https://github.com/philipperemy/keras-attention-mechanism

Decomposition, lightweight

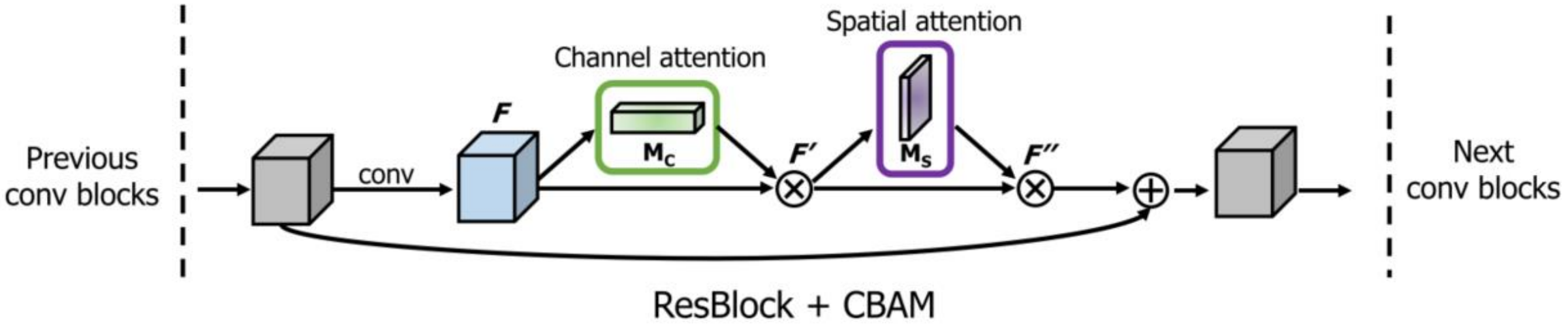
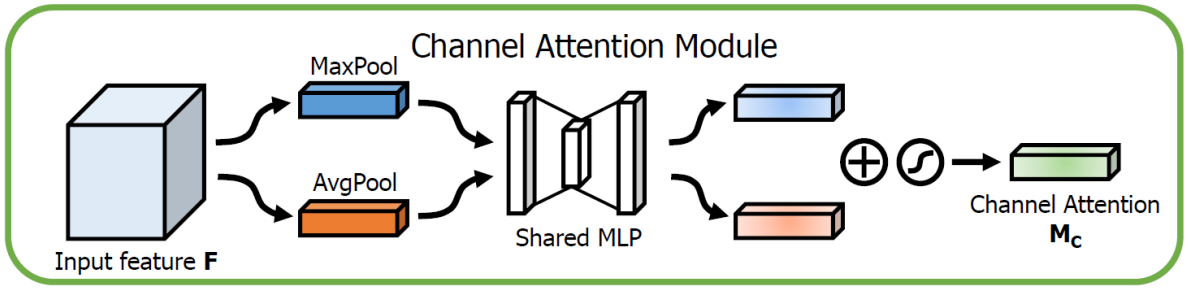


Fig. 3: **CBAM integrated with a ResBlock in ResNet[5].** This figure shows the exact position of our module when integrated within a ResBlock. We apply CBAM on the convolution outputs in each block.

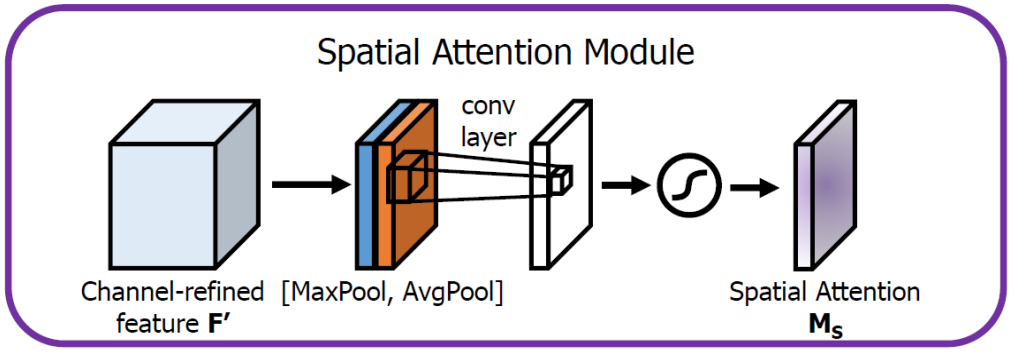
Channel attention



Similar to Squeeze-and-Excitation Neworks (SENet)

$$\mathbf{M_c}(\mathbf{F}) = \sigma(MLP(AvgPool(\mathbf{F})) + MLP(MaxPool(\mathbf{F})))$$
$$= \sigma(\mathbf{W_1}(\mathbf{W_0}(\mathbf{F^c_{avg}})) + \mathbf{W_1}(\mathbf{W_0}(\mathbf{F^c_{max}}))),$$

Spatial attention



$$\mathbf{M_s}(\mathbf{F}) = \sigma(f^{7\times7}([AvgPool(\mathbf{F}); MaxPool(\mathbf{F})])) = \sigma(f^{7\times7}([\mathbf{F^s_{avg}}; \mathbf{F^s_{max}}])),$$

Decomposition, lightweight

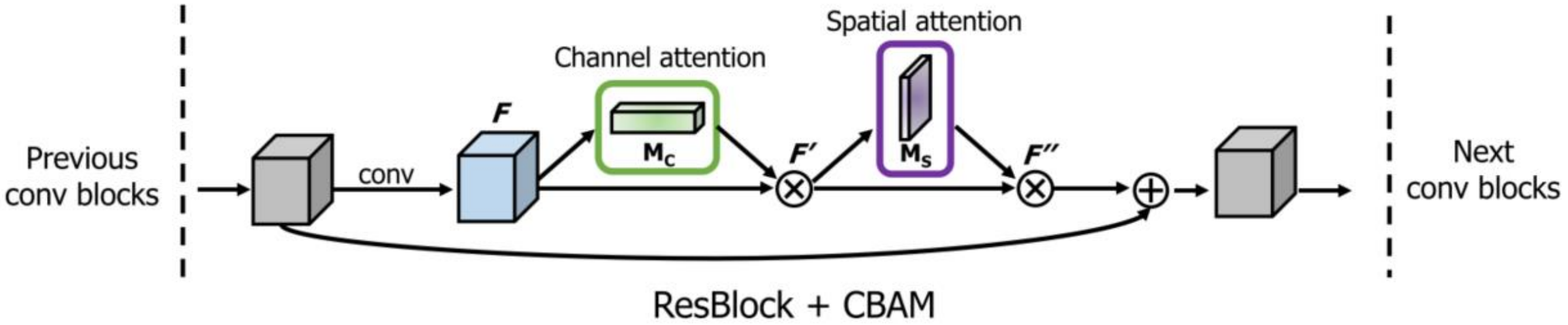


Fig. 3: **CBAM integrated with a ResBlock in ResNet[5].** This figure shows the exact position of our module when integrated within a ResBlock. We apply CBAM on the convolution outputs in each block.

Architecture	Param.	GFLOPs	Top-1 Error (%)	Top-5 Error (%)
ResNet18 [5]	11.69M	1.814	29.60	10.55
ResNet18 [5] + SE [28]	11.78M	1.814	29.41	10.22
ResNet18 [5] + CBAM	11.78M	1.815	**29.27**	**10.09**
ResNet34 [5]	21.80M	3.664	26.69	8.60
ResNet34 [5] + SE [28]	21.96M	3.664	26.13	8.35
ResNet34 [5] + CBAM	21.96M	3.665	**25.99**	**8.24**
ResNet50 [5]	25.56M	3.858	24.56	7.50
ResNet50 [5] + SE [28]	28.09M	3.860	23.14	6.70
ResNet50 [5] + CBAM	28.09M	3.864	**22.66**	**6.31**
ResNet101 [5]	44.55M	7.570	23.38	6.88
ResNet101 [5] + SE [28]	49.33M	7.575	22.35	6.19
ResNet101 [5] + CBAM	49.33M	7.581	**21.51**	**5.69**
WideResNet18 [6] (widen=1.5)	25.88M	3.866	26.85	8.88
WideResNet18 [6] (widen=1.5) + SE [28]	26.07M	3.867	26.21	8.47
WideResNet18 [6] (widen=1.5) + CBAM	26.08M	3.868	**26.10**	**8.43**
WideResNet18 [6] (widen=2.0)	45.62M	6.696	25.63	8.20
WideResNet18 [6] (widen=2.0) + SE [28]	45.97M	6.696	24.93	7.65
WideResNet18 [6] (widen=2.0) + CBAM	45.97M	6.697	**24.84**	**7.63**
ResNeXt50 [7] (32x4d)	25.03M	3.768	22.85	6.48
ResNeXt50 [7] (32x4d) + SE [28]	27.56M	3.771	**21.91**	6.04
ResNeXt50 [7] (32x4d) + CBAM	27.56M	3.774	21.92	**5.91**
ResNeXt101 [7] (32x4d)	44.18M	7.508	21.54	5.75
ResNeXt101 [7] (32x4d) + SE [28]	48.96M	7.512	21.17	5.66
ResNeXt101 [7] (32x4d) + CBAM	48.96M	7.519	**21.07**	**5.59**

* all results are reproduced in the PyTorch framework.

Table 4: **Classification results on ImageNet-1K.** Single-crop validation errors are reported.

Personal opinion and Open question

- Not using the cbam in every block, in order to avoid overfitting

Applications in other fields

- 《Neural machine translation by jointly learning to align and translate》
- 《Attend and Tell: Neural Image Caption Generation with Visual Attention》
- 《Attention-Based Models for Speech Recognition》
- 《AttentionNet: Aggregating Weak Directions for Accurate Object Detection》
- 《Multi-context attention for human pose estimation》
- 《Hierarchical Attentive Recurrent Tracking》

Image Captioning

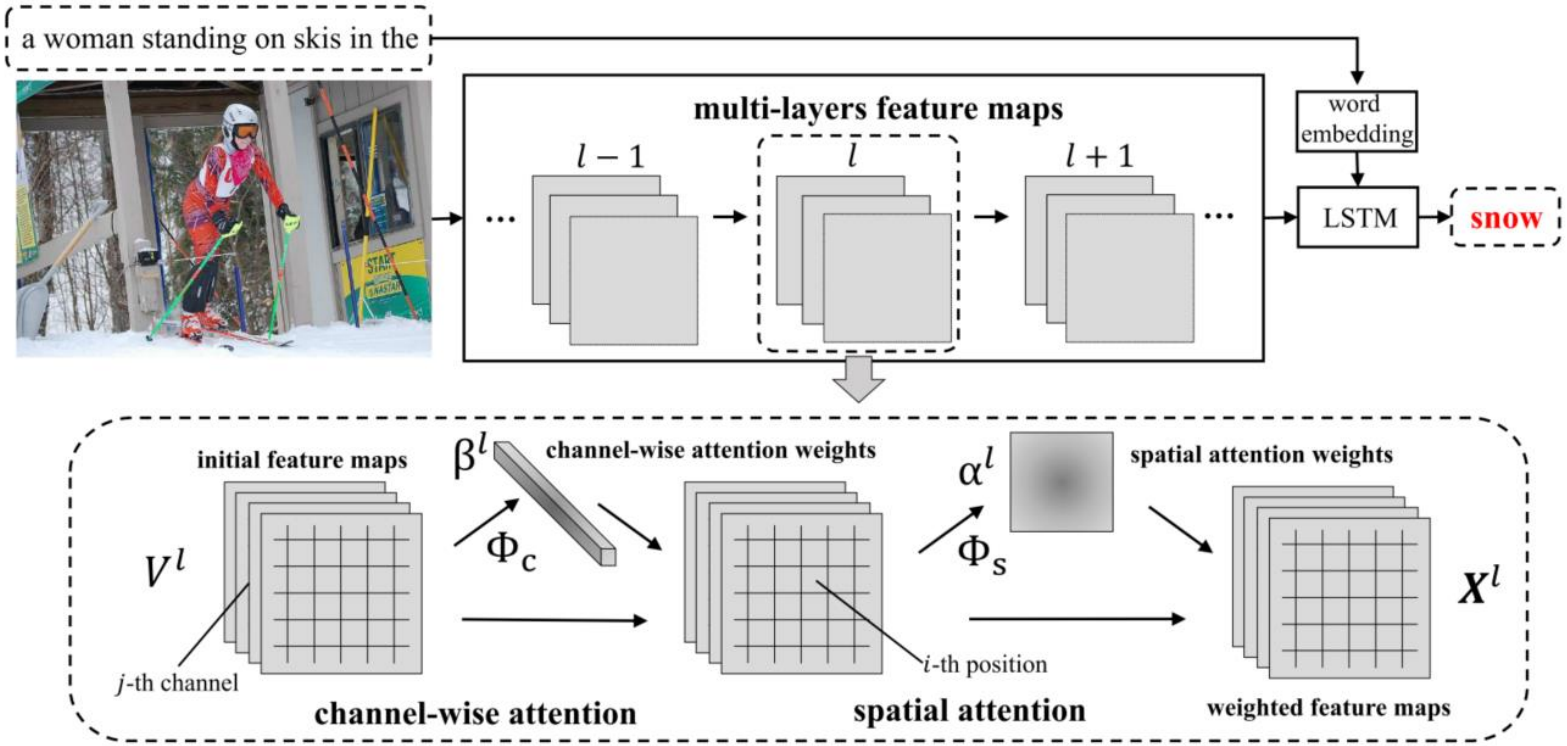
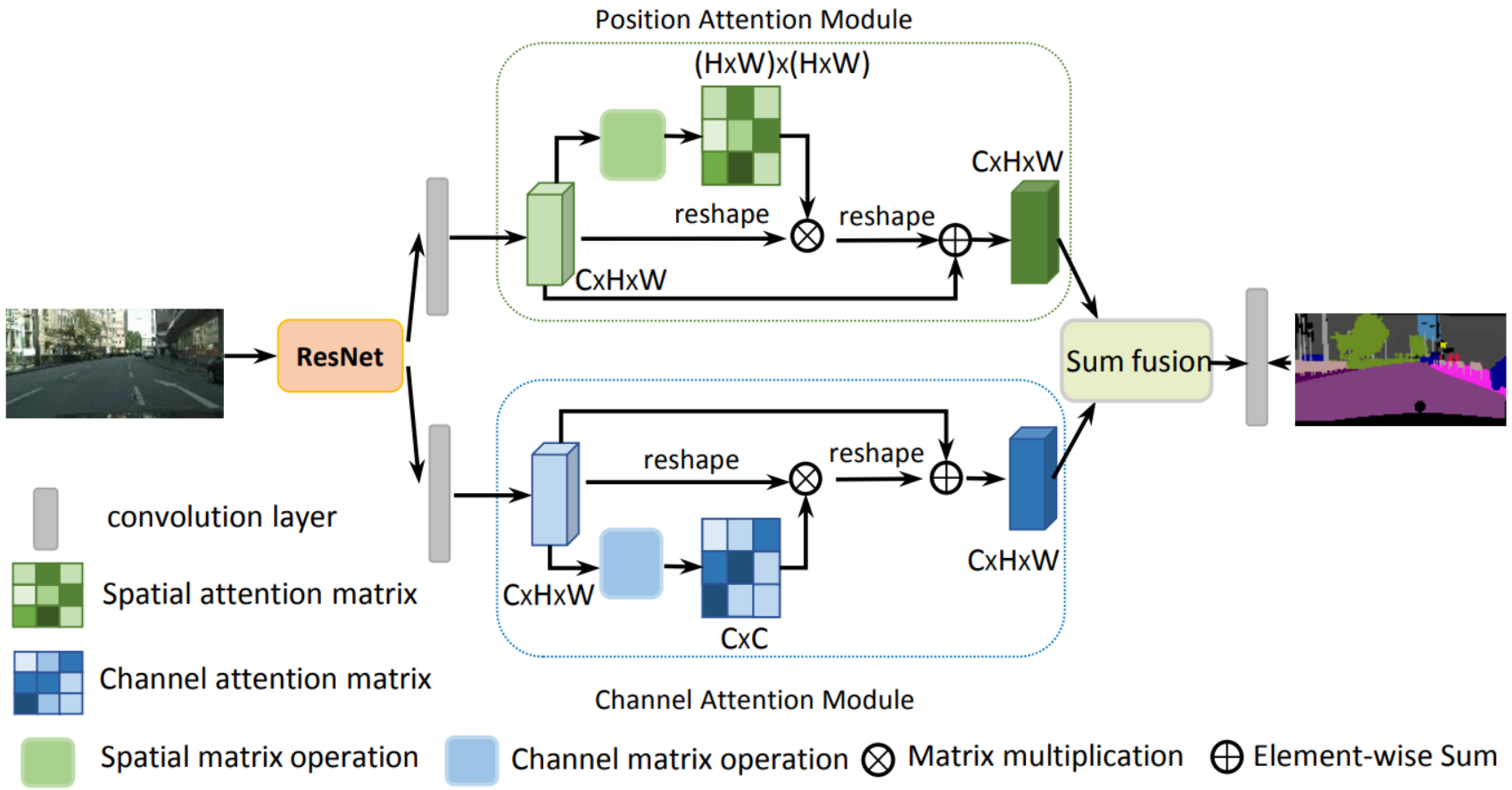


Figure 2. The overview of our proposed SCA-CNN. For the l-th layer, initial feature map $\mathbf{V}^l$ is the output of $(l-1)$-th conv-layer. We first use the channel-wise attention function Φ_c to obtain the channel-wise attention weights β^l, which are multiplied in channel-wise of the feature map. Then, we use the spatial attention function Φ_s to obtain the spatial attention weights α^l, which are multiplied in each spatial regions, resulting in an attentive feature map $\mathbf{X}^l$. Different orders of two attention mechanism are discussed in Section 3.3.

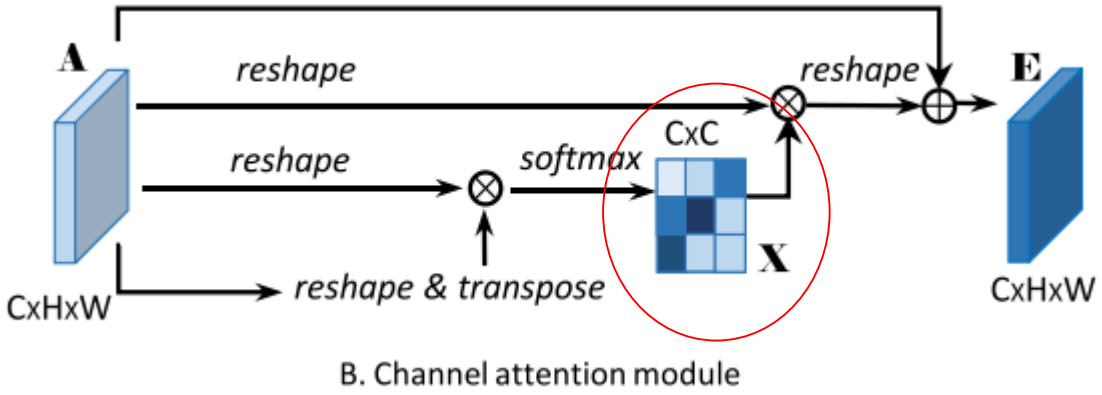
- Long Chen; Hanwang Zhang; Jun Xiao; Liqiang Nie; Jian Shao; Wei Liu; Tat-Seng Chua: SCA-CNN: Spatial and Channel-Wise Attention in Convolutional Networks for Image Captioning.

Dual attention

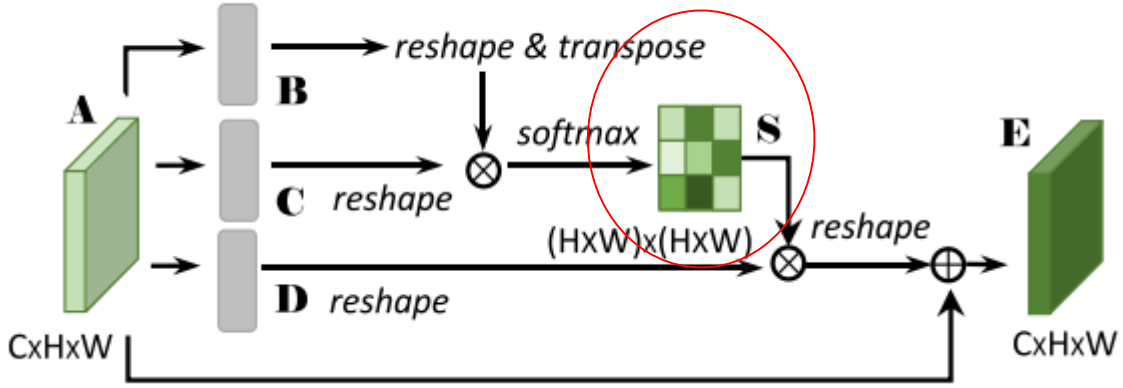


- https://github.com/junfu1115/DANet

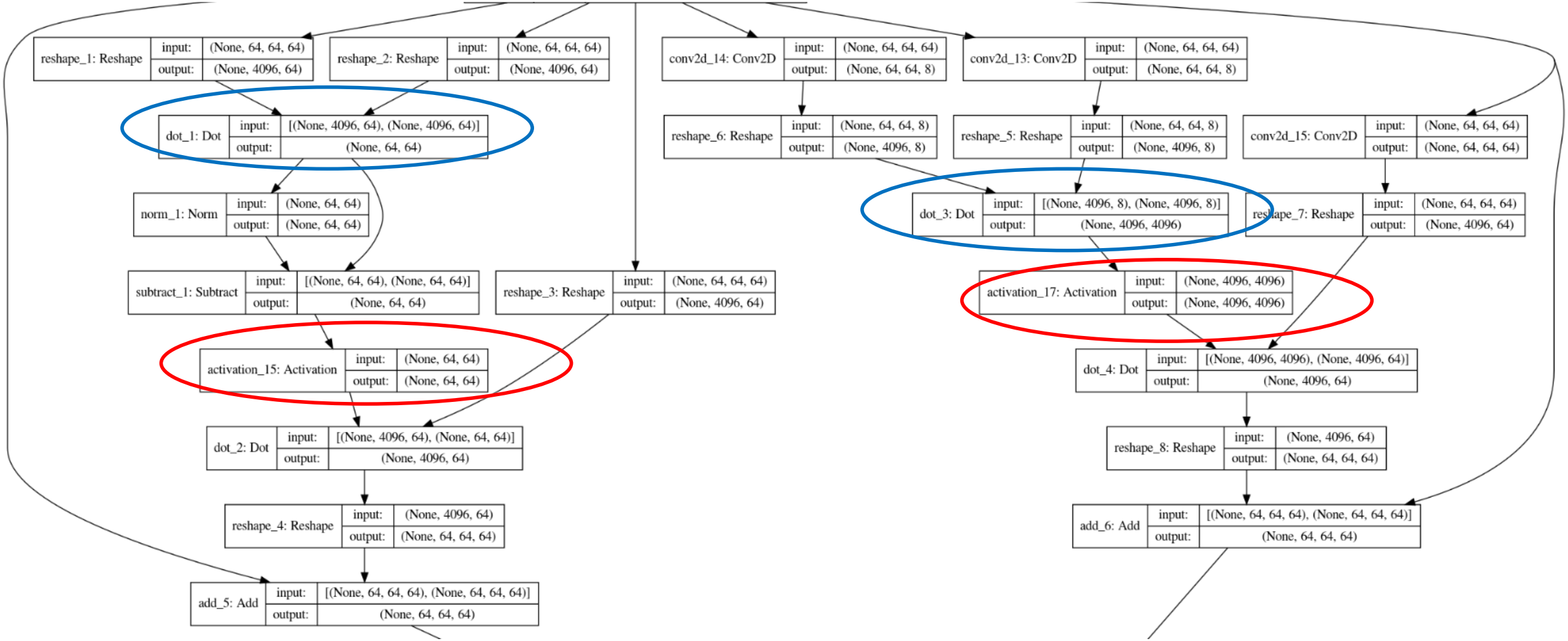
channel attention & spatial attention = dual attention



B. Channel attention module



- https://zhuanlan.zhihu.com/p/48056789


reshape_1: Reshape
input: (None, 64, 64, 64)
output: (None, 4096, 64)
reshape_2: Reshape
input: (None, 64, 64, 64)
output: (None, 4096, 64)
conv2d_14: Conv2D
input: (None, 64, 64, 64)
output: (None, 64, 64, 8)
conv2d_13: Conv2D
input: (None, 64, 64, 64)
output: (None, 64, 64, 8)
dot_1: Dot
input: [(None, 4096, 64), (None, 4096, 64)]
output: (None, 64, 64)
reshape_6: Reshape
input: (None, 64, 64, 8)
output: (None, 4096, 8)
reshape_5: Reshape
input: (None, 64, 64, 8)
output: (None, 4096, 8)
conv2d_15: Conv2D
input: (None, 64, 64, 64)
output: (None, 64, 64, 64)
norm_1: Norm
input: (None, 64, 64)
output: (None, 64, 64)
dot_3: Dot
input: [(None, 4096, 8), (None, 4096, 8)]
output: (None, 4096, 4096)
reshape_7: Reshape
input: (None, 64, 64, 64)
output: (None, 4096, 64)
subtract_1: Subtract
input: [(None, 64, 64), (None, 64, 64)]
output: (None, 64, 64)
reshape_3: Reshape
input: (None, 64, 64, 64)
output: (None, 4096, 64)
activation_17: Activation
input: (None, 4096, 4096)
output: (None, 4096, 4096)
activation_15: Activation
input: (None, 64, 64)
output: (None, 64, 64)
dot_4: Dot
input: [(None, 4096, 4096), (None, 4096, 64)]
output: (None, 4096, 64)
dot_2: Dot
input: [(None, 4096, 64), (None, 64, 64)]
output: (None, 4096, 64)
reshape_8: Reshape
input: (None, 4096, 64)
output: (None, 64, 64, 64)
reshape_4: Reshape
input: (None, 4096, 64)
output: (None, 64, 64, 64)
add_6: Add
input: [(None, 64, 64, 64), (None, 64, 64, 64)]
output: (None, 64, 64, 64)
add_5: Add
input: [(None, 64, 64, 64), (None, 64, 64, 64)]
output: (None, 64, 64, 64)