Convolution of Convolution: Let Kernels Collaborate Spatially

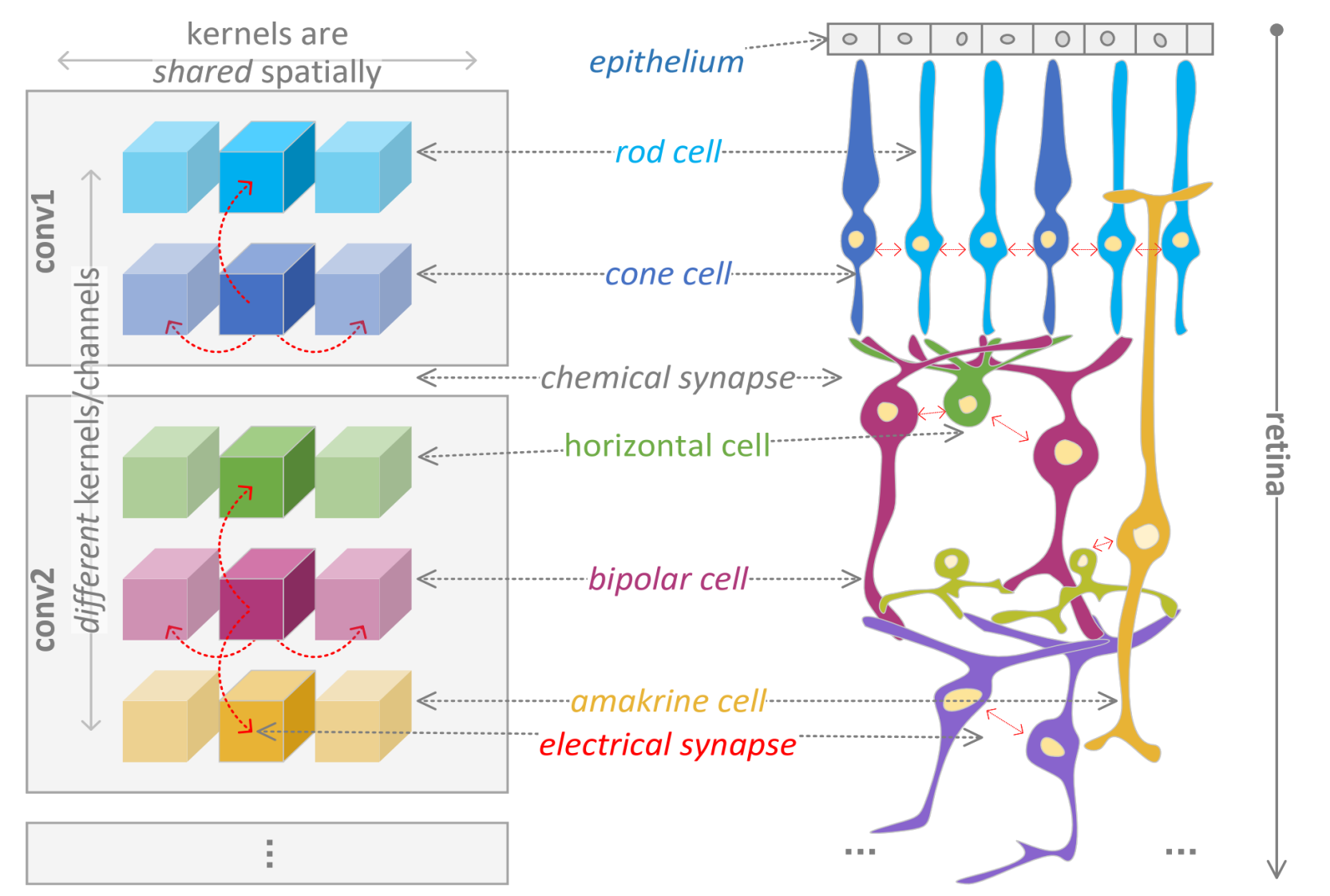
Zhao,Rongzhen   Li,Jian   Wu,Zhenzhi*   Lynxi Technologies, Beijing, China, 100097

MOTIVATION

The *retina* neurons tile spatially with *electrical coupling* (EC) as their local association (red arrows in fig right), while the conv kernels stack along the channel dimension singly.

EC among same type neurons (horizontal arrows in fig left) are implicated in sliding windows, but EC among different types (vertical arrows in fig left) hasn't been modeled yet.

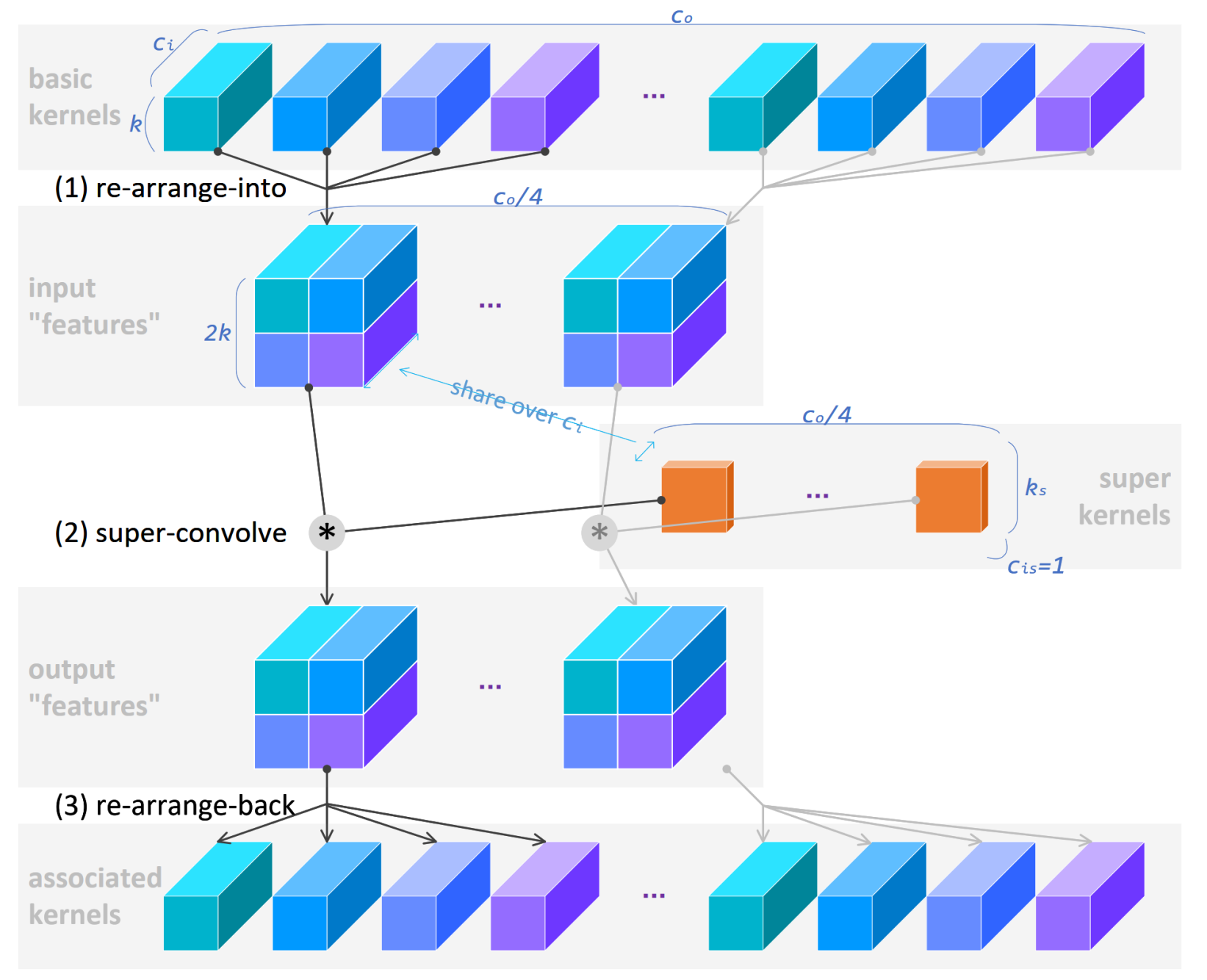
We handle this with method "convolution of convolution" (CoC), which exerts spatial associations among kernels within a layer to let them collaborate spatially.



PROPOSED METHOD

Let Kernels Collaborate in Spatial Dimensions

As drawn in fig given a basic convolution, of which the kernels' tensor is in shape (co, ci, k, k).



(0) *Divide* kernels *into* groups, e.g. every four as a group.

(1) *Re-arrange* each group along width and height *into* shape (co/4, ci, 2k, 2k), and treat them as "input features".

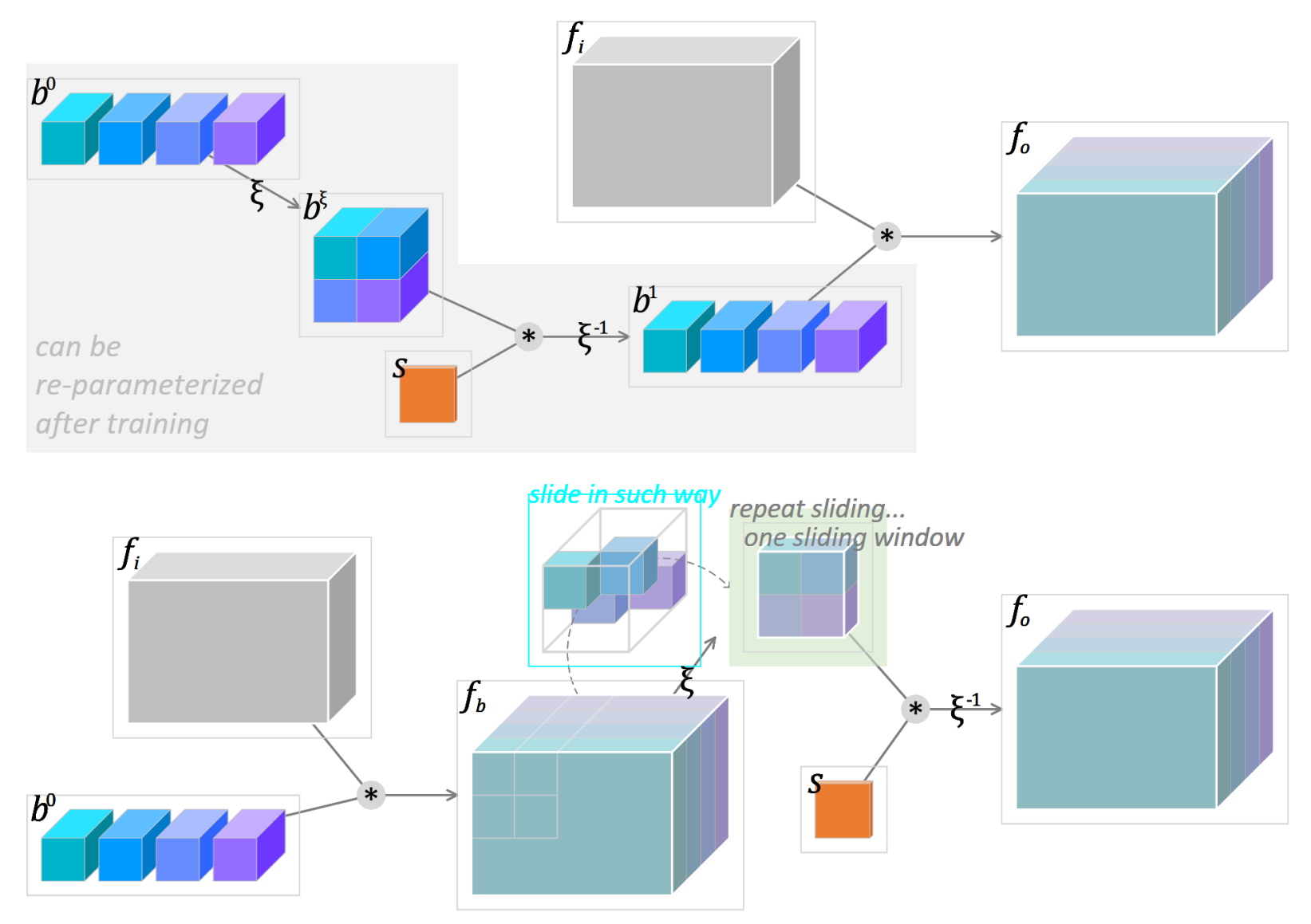
(2) For the "input features", each in shape (ci, 1, 2k, 2k) is *convolved* with a specific super kernel in shape (1, 1, ks, ks), which is shared over the "batch" dimension ci times.

(3) *Re-arrange* the "output features" *back* and spatially associated kernels are obtained.

(4) Conduct the *common convolution* with these kernels.

Before testing, step (1~3) are calculated in advance only once, i.e. *re-parameterizing* CoC back to common convolution.

FP: Provide Extra Transformations among Output Channels


can be re-parameterized after training
slide in such way
repeat sliding... one sliding window

In forward propagation, CoC is equal to a common convolution plus *extra transformations* on its outputs.

As shown in fig the upper, our CoC can be formulated as:

$$f_o = \mathrm{F}_{\mathrm{coc}}(f_i) = \xi^{-1}(s * \xi(b^0)) * f_i$$

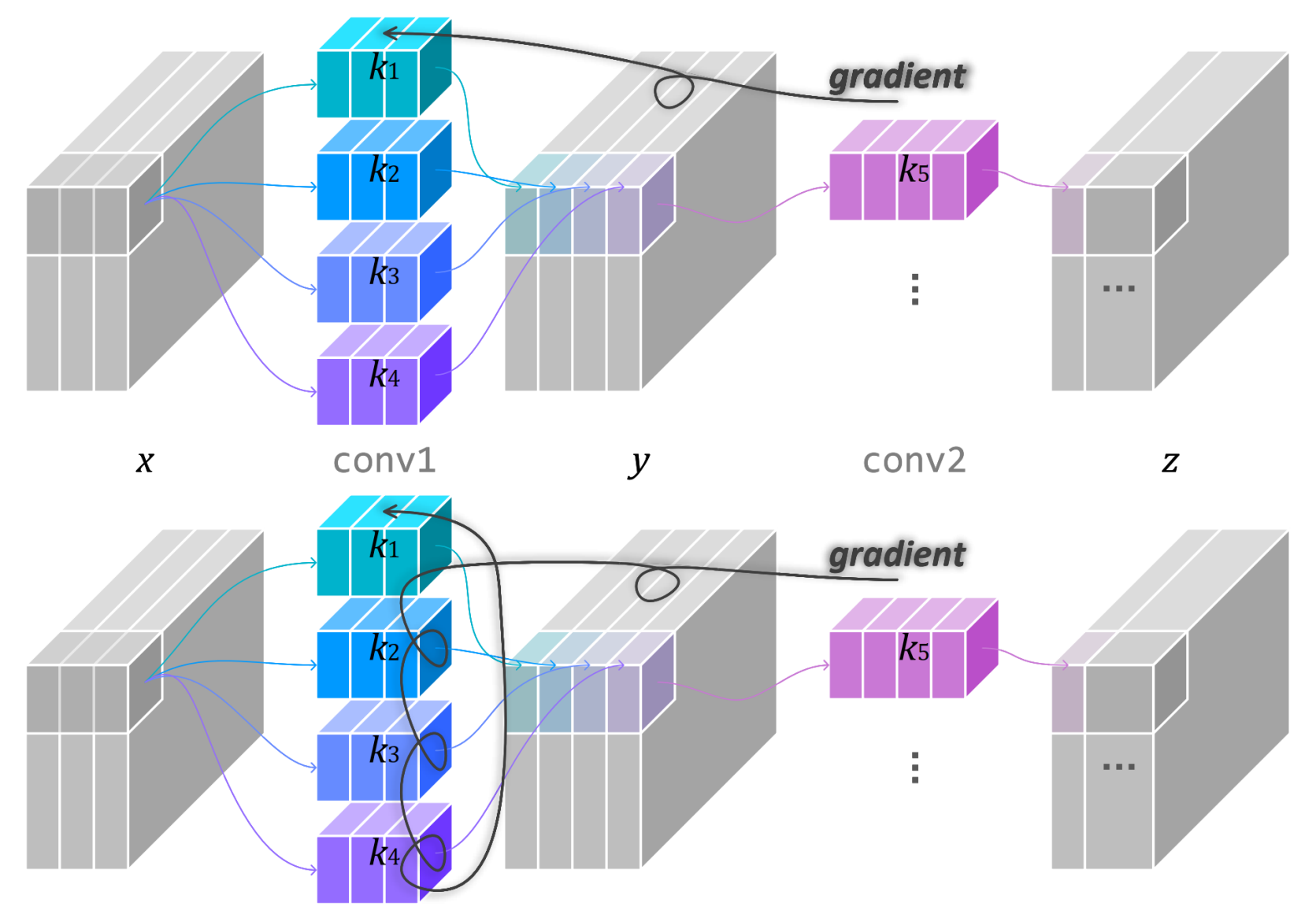
where s∗ξ(b0) is step (2) super convolution, and ξ−1(s∗ξ(b0)) is steps (1˜3) spatial association.

As in fig the lower, given super conv is linear, it is equal to

$$f_o = \mathrm{F}_{\mathrm{coc}}(f_i) = \xi^{-1}(s * \xi(b^0 * f_i))$$

where b0∗fi is the common conv, and the rest part is an extra transform. on the output channels.

BP: Learning Kernels in a Layer by Referring to One Another



In backward propagation, CoC makes that kernels in a layer are learnt by *referring to one another*.

For common conv in fig the upper, according to the chain rule, kernel k1's gradient in layer conv1 is

$$\begin{aligned} g_1 &= G \times \frac{\partial y}{\partial k_1} = G \times \frac{\partial(k_1 * x, k_2 * x, \dots k_4 * x)}{\partial k_1} \\ &= G \times x \end{aligned}$$

where G is the accumulated gradient. k1's gradient is independent of the other kernels in this layer.

For CoC in fig the lower, denote the aforementioned spatial association as α(·) = ξ−1(s∗ξ(·)). Then k1's gradient is

$$\begin{aligned} g_1 &= G \times \frac{\partial y}{\partial k_1} = G \times \frac{\partial(k'_1 * x, k'_2 * x, \dots k'_4 * x)}{\partial k_1} \\ &= G \times (\frac{\partial \alpha(k_1; k_2, \dots k_4) + \dots \alpha(k_4; k_1, \dots k_3)}{\partial k_1} * x) \end{aligned}$$

k1's gradient is explicitly dependent on other kernels.

Naive Implementation

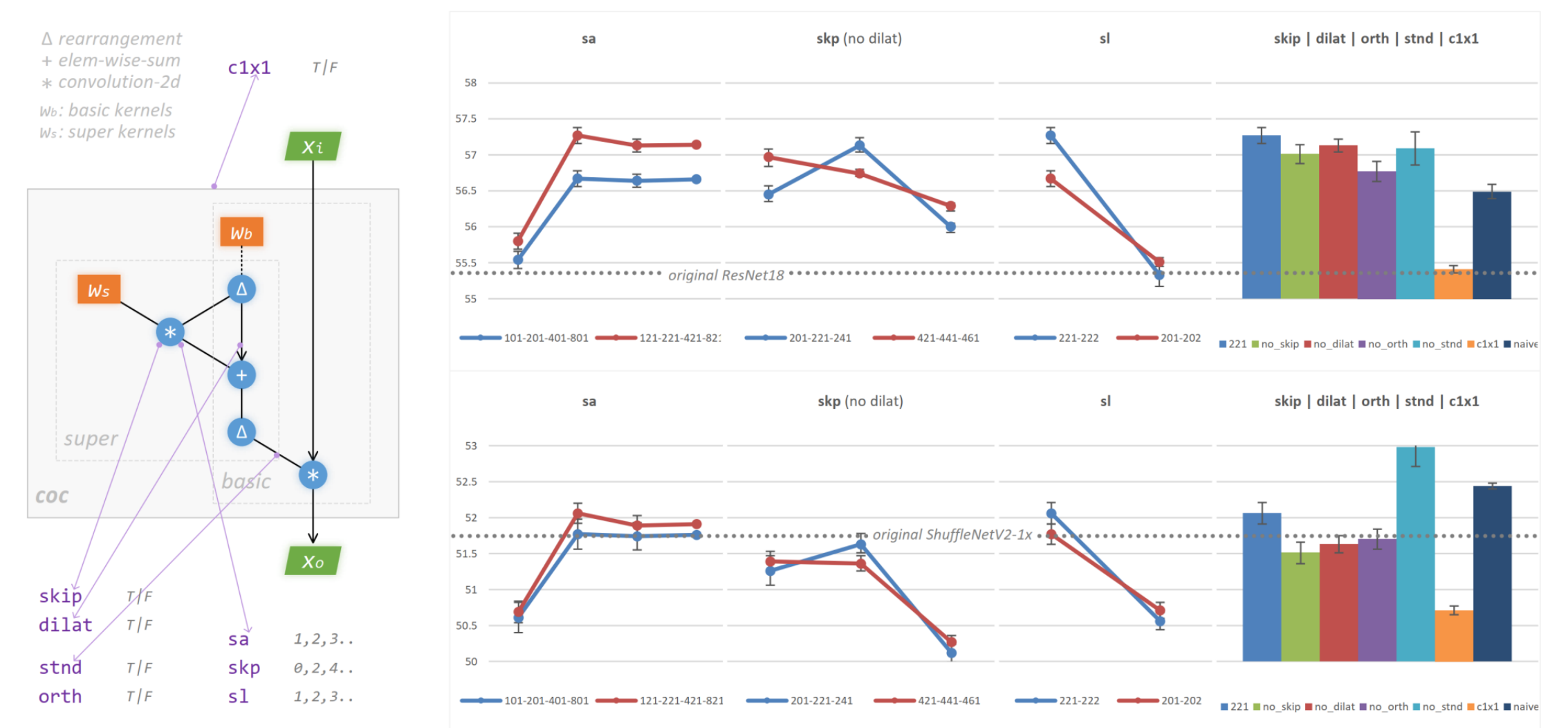
```
### __init__

co4 = out_channels // 2 ** 2
weight1 = Tensor(out_channels, in_channels, *kernel_size)
co, ci, kh, kw = weight1.shape

# 0 # create super conv
self.conv_super = Conv2d(co4, co4, super_kernel_size, super_stride,
    super_padding, super_dilation, groups=co4)  # XXX

# 1 # rearrange basic kernels for spatial association
weight2 = rearrange(weight1, '(co4 2 2) ci kh kw -> co4 ci (2 kh) (2 kw)')
weight3 = weight2.permute(1, 0, 2, 3)  # XXX
self.weight = Parameter(weight3)

### forward

# 2 # use super conv to associate the rearranged basic kernels
weight4 = self.conv_super(self.weight)  # XXX

# 3 # rearrange back to get the associated kernels
weight5 = weight4.permute(1, 0, 2, 3)  # XXX
weight6 = rearrange(weight5, 'co4 ci (2 kh) (2 kw) -> (co4 2 2) ci kh kw')

# 4 # use the associated kernels to do the common convolution
xo = F.conv2d(xi, weight6, self.bias, stride, padding, dilation)
```

EXPERIMENTS

Determine Hyper-Parameters

The best setting of CoC's hyper-params is:
sa=2, skp=2, sl=1, skip=true, dilat=true, c1x1=false.

Besides, for standard models, it is better to use CoC and *orthogonalization* together; for light-weight models, it is quite necessary to discard *standardization*.



Evaluate on Typical Tasks

CoC's advantage on classification like ImageNet is *not obvious*, just 0.2~0.6%. But on tasks demanding larger receptive fields like detection and segmentation, it's clearly a *superior* method.

Detection and instance segmentation results on COCO:

Semantic Segmentation results on Pascal VOC:

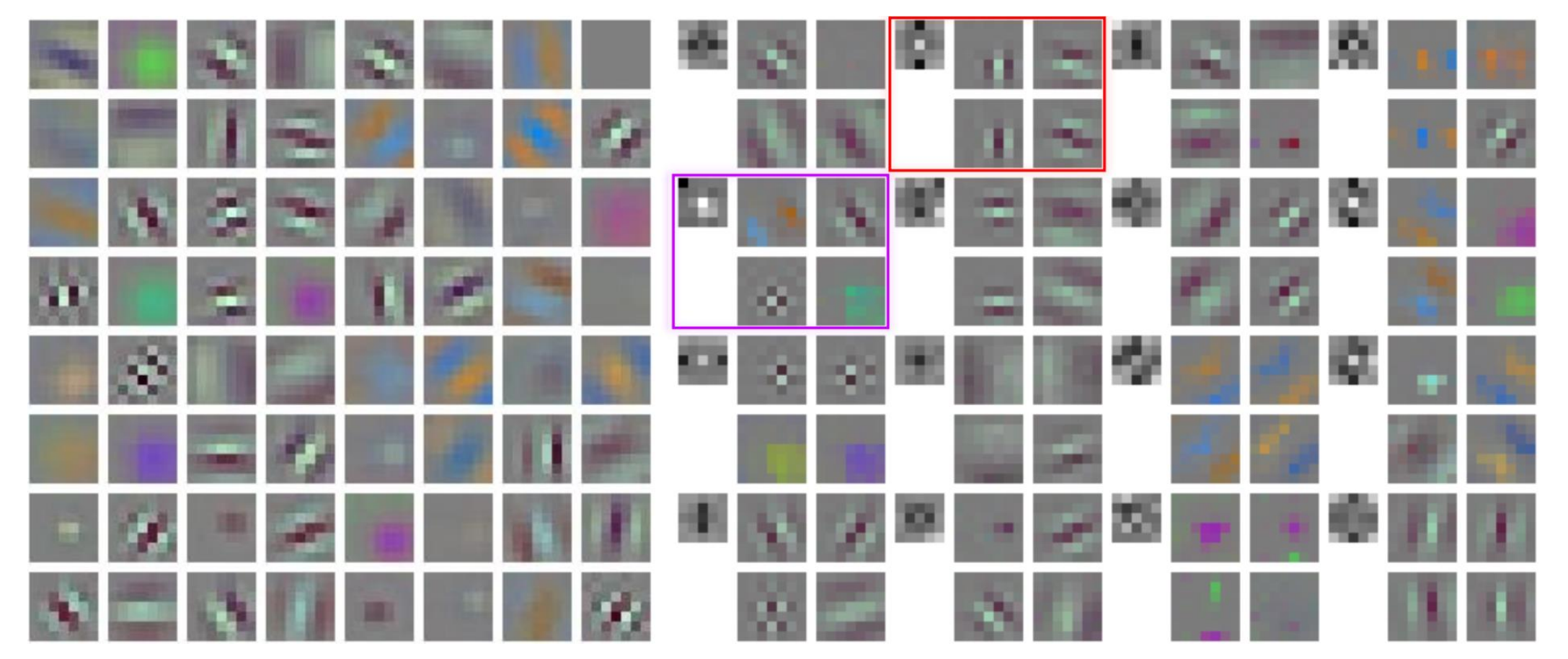
#	network	pretrain	mIoU	mAcc	aAcc
1	FCN-r50d8	✓	66.97	75.99	92.16
2			23.53	32.23	80.20
3	FCN-r50d8-stnd	✓	69.16	78.42	92.65
4	FCN-r50d8-orth	✓	69.01	78.36	92.62
5	FCN-r50d8-coc	✓	***69.52***	***79.14***	***92.83***
6			34.25	46.35	83.17
7	HRNet-w18	✓	72.13	82.33	93.59
8			35.23	49.74	82.28
9	HRNet-w18-stnd	✓	74.34	84.87	93.91
10	HRNet-w18-orth	✓	74.17	84.84	93.88
11	HRNet-w18-coc	✓	***74.81***	***85.65***	***93.97***
12			41.39	58.81	83.28

#	network	mAP	mAP_{50}	mAP_{75}	mAP_S	mAP_M	mAP_L	
1	RetinaNet-r50	36.3	55.1	38.8	20.1	40.1	47.8	bbox
2	RetinaNet-r50-stnd	37.9	56.9	40.6	21.3	**42.1**	50.0	
3	RetinaNet-r50-orth	37.6	56.4	40.7	21.0	41.9	49.9	
4	*RetinaNet-r50-coc*	***38.1***	***57.0***	***40.8***	***21.5***	*42.0*	***50.1***	
5	MaskRCNN-hr18	33.9	54.3	36.3	18.9	36.4	45.8	mask
6	MaskRCNN-hr18-stnd	34.9	56.0	37.4	19.9	37.9	47.2	
7	MaskRCNN-hr18-orth	35.1	55.8	37.9	20.0	37.6	47.5	
8	*MaskRCNN-hr18-coc*	***35.6***	***56.2***	***38.2***	***20.3***	***38.1***	***47.8***	

VISUALIZATION

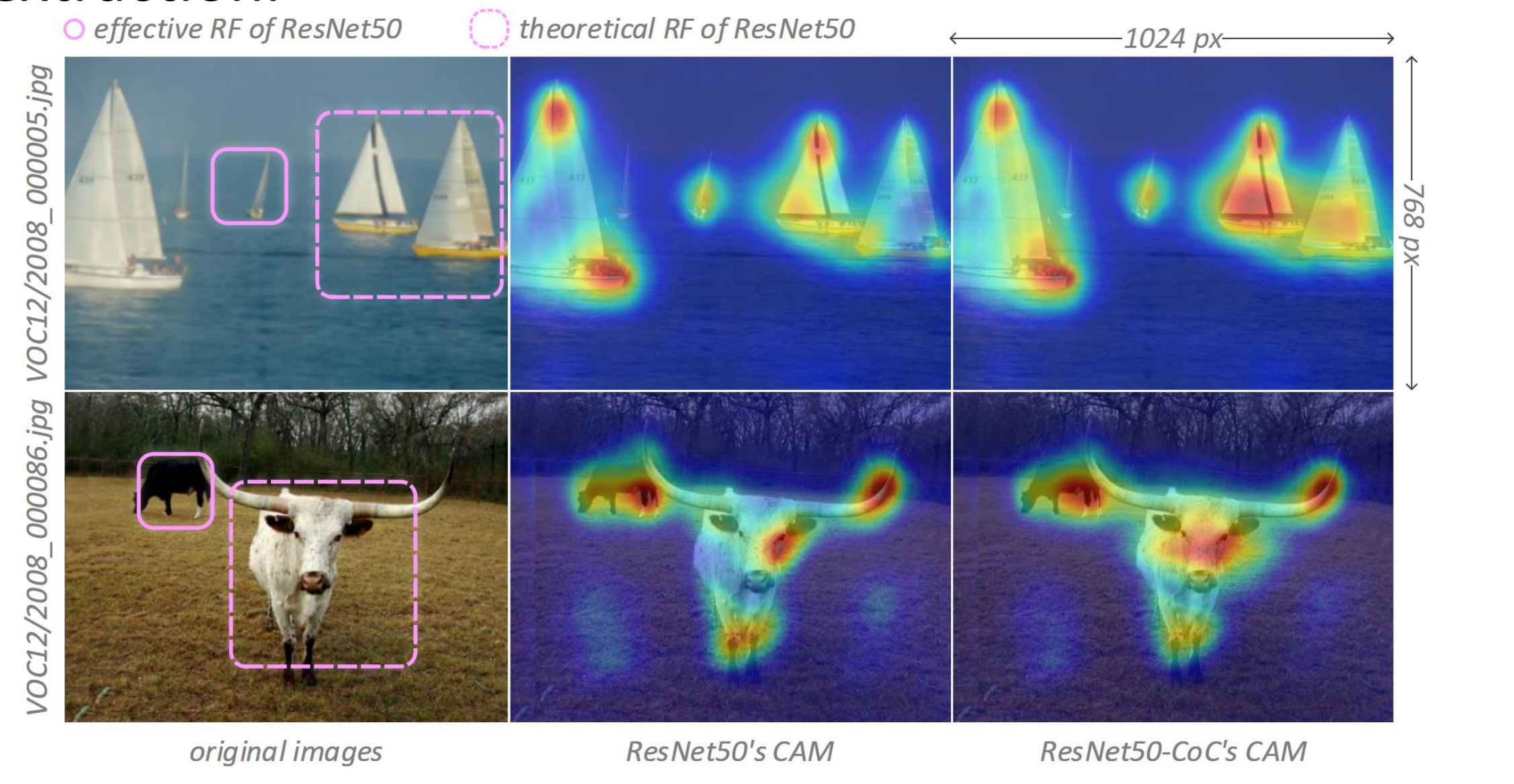
What Kernels Are Learnt under the Spatial Association?

In common convolution (fig left), patterns (grey vs. color, stripe vs. plane) scatter among channels irregularly. But patterns learnt in CoC (fig right) are always *similar* yet *complementary* within each spatial association. Besides, the spatial distribution of sub-patterns within a group clearly reflects their super kernel's pattern.



Why Relatively Better When Larger Receptive Fields Needed?

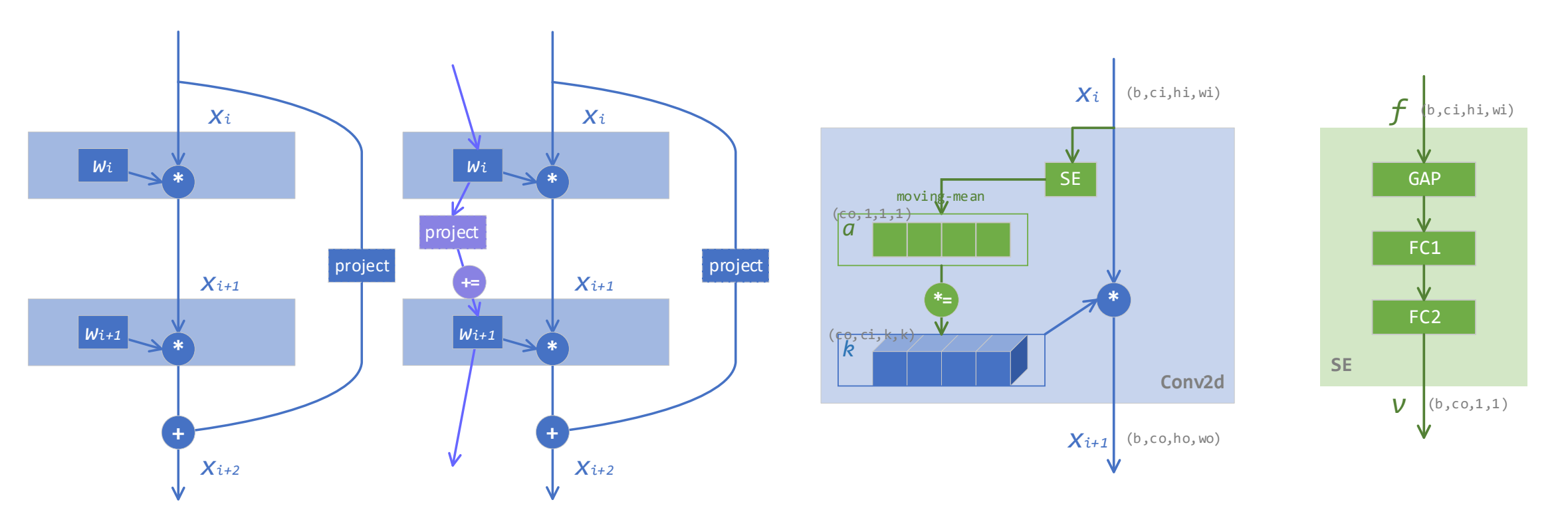
ResNet50 fully perceives small objects that fall into its ERF while partially perceives the large that exceed its ERF; our ResNet50-CoC works much better as shown in fig due to its *larger* ERF and better feature extraction.



FUTURE WORKS

We believe similar ways of thinking are also worth exploring: *channel attention for pruning*, *cross-layer connections among kernels* rather than activation features, and so on.

You are highly welcomed to make further explorations.


project
SE
Conv2d
GAP
FC1
FC2