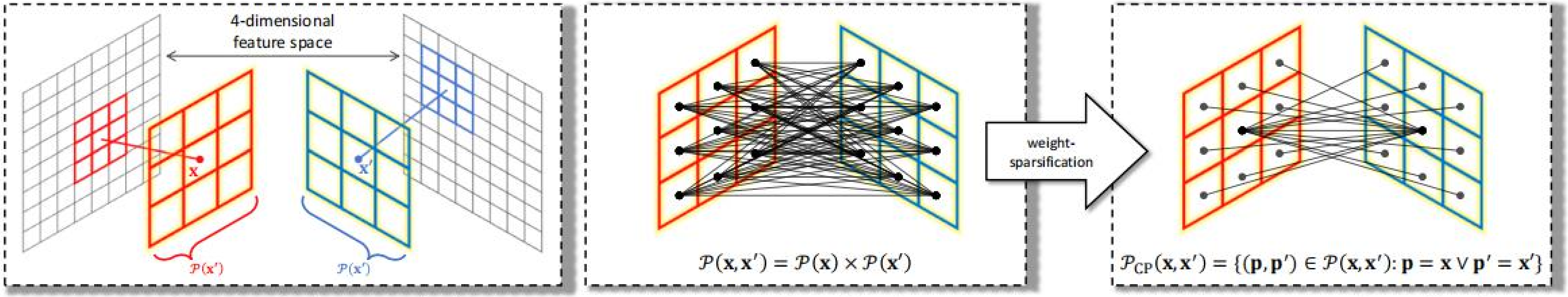
01 Hypercorrelation Squeeze for Few-Shot Segmentation

- ICCV 2021

Hypercorrelation Squeeze for Few-Shot Segmentation

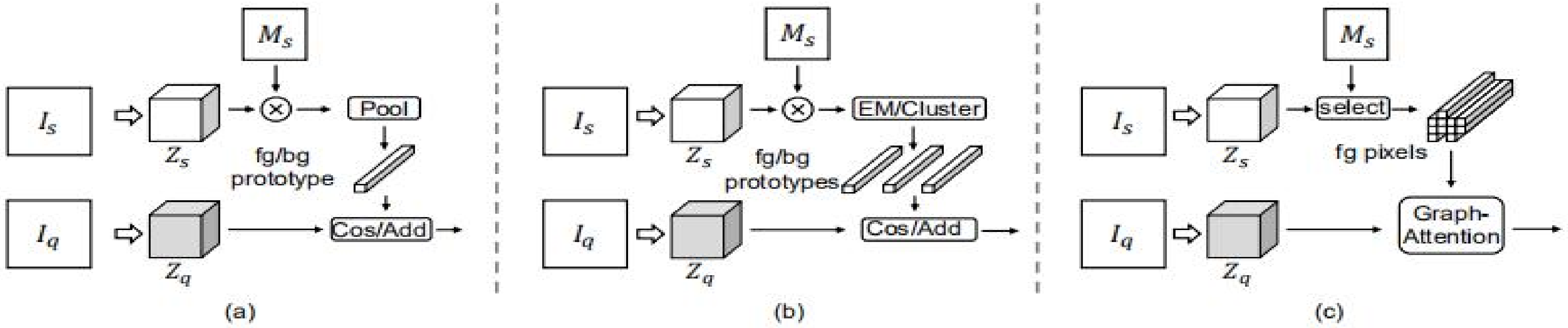
Juhong Min Dahyun Kang Minsu Cho
Pohang University of Science and Technology (POSTECH), South Korea
http://cvlab.postech.ac.kr/research/HSNet/



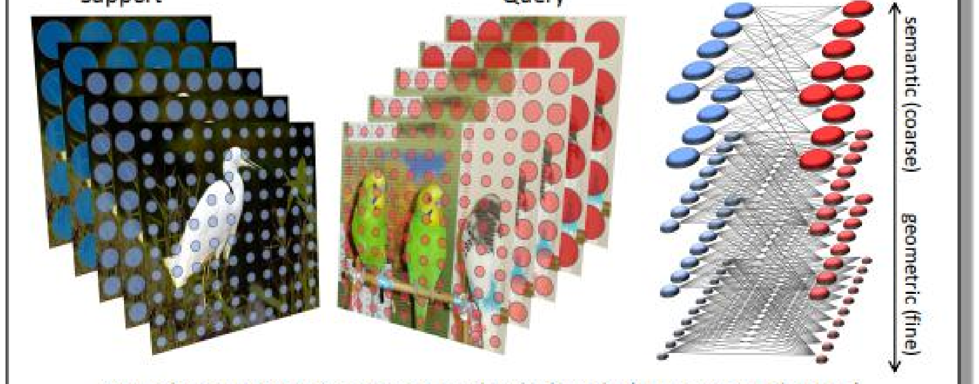
4D convolution convolved at position $(\mathbf{x},\mathbf{x}')$ 4D convolutional kernel Center-pivot 4D convolutional kernel

01 Hypercorrelation Squeeze for Few-Shot Segmentation

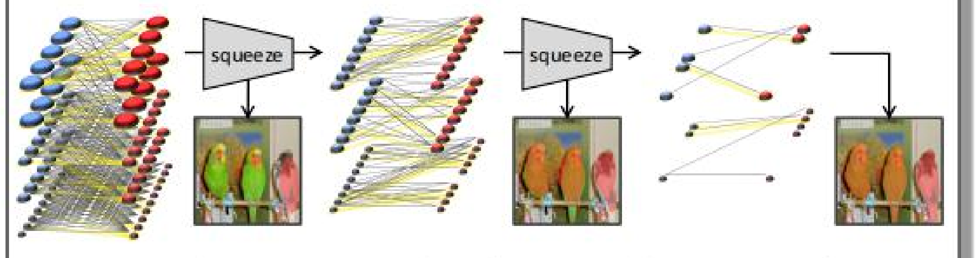
- **Motivation**



Few shot segmentation requires to understand diverse levels of visual cues and analyze fine-grained correspondence relations between the query and the support images. To address the problem, we propose Hypercorrelation Squeeze Networks (HSNet) that leverages multi-level feature correlation and efficient 4D convolutions.



Visual correspondences at multiple levels (**Hypercorrelation**)



Correlation pattern analysis (**Hypercorrelation squeeze**)

01 Hypercorrelation Squeeze for Few-Shot Segmentation

- **Framework of HSNet**

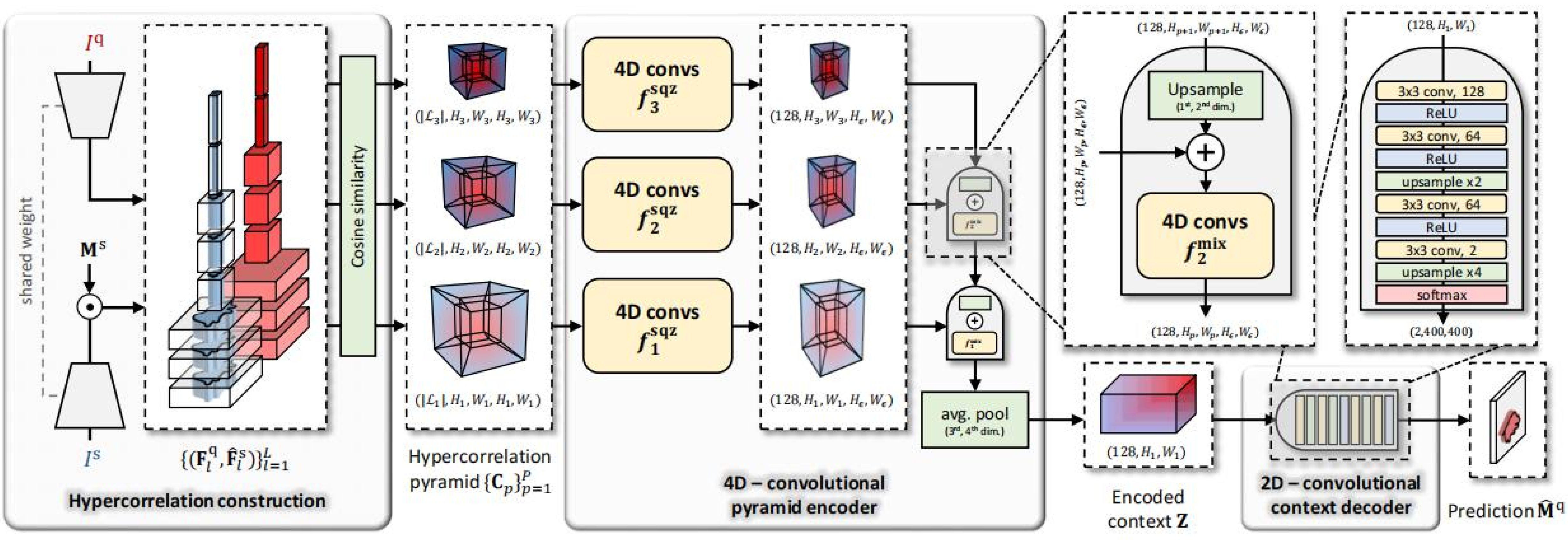


Figure 2: Overall architecture of the proposed network which consists of three main parts: hypercorrelation construction, 4D-convolutional pyramid encoder, and 2D-convolutional context decoder. We refer the readers to Sec. 4 for details of the architecture.

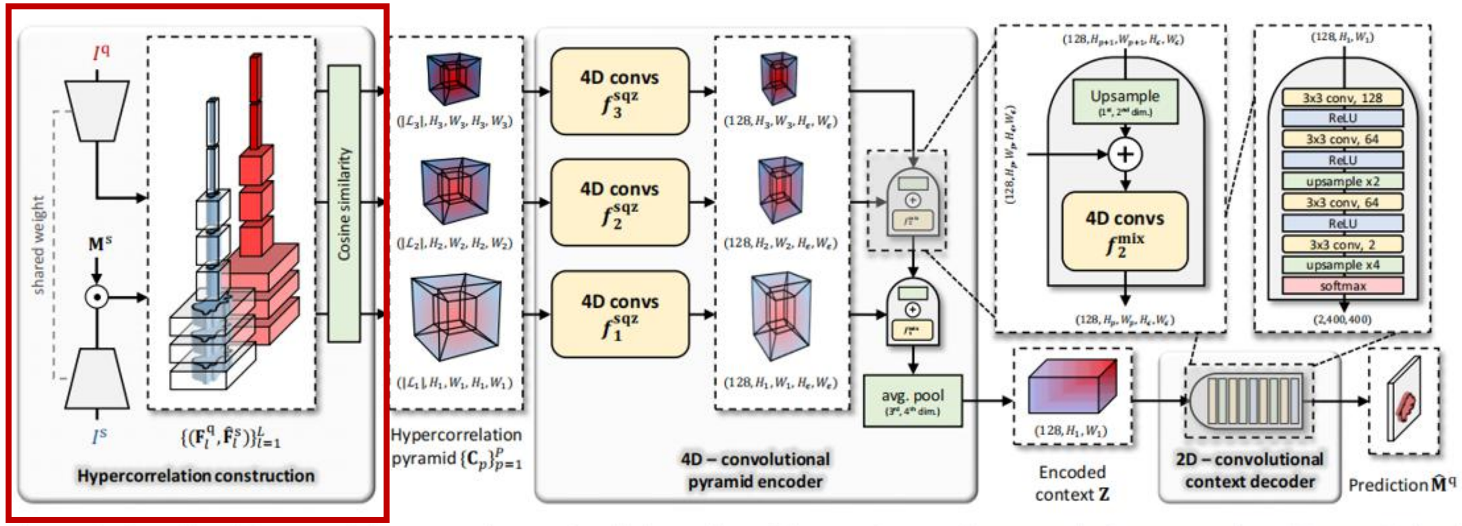
01 Hypercorrelation Squeeze for Few-Shot Segmentation

- **Hypercorrelation construction**
  - ReLU suppresses noisy correlation scores.

$$\hat{\mathbf{F}}_l^{\mathrm{s}} = \mathbf{F}_l^{\mathrm{s}} \odot \zeta_l(\mathbf{M}^{\mathrm{s}}),$$

$$\hat{\mathbf{C}}_l(\mathbf{x}^{\mathrm{q}}, \mathbf{x}^{\mathrm{s}}) = \mathrm{ReLU}\left(\frac{\mathbf{F}_l^{\mathrm{q}}(\mathbf{x}^{\mathrm{q}}) \cdot \hat{\mathbf{F}}_l^{\mathrm{s}}(\mathbf{x}^{\mathrm{s}})}{\|\mathbf{F}_l^{\mathrm{q}}(\mathbf{x}^{\mathrm{q}})\|\|\hat{\mathbf{F}}_l^{\mathrm{s}}(\mathbf{x}^{\mathrm{s}})\|}\right),$$

$$\mathbf{C}_p \in \mathbb{R}^{|\mathcal{L}_p| \times H_p \times W_p \times H_p \times W_p}$$

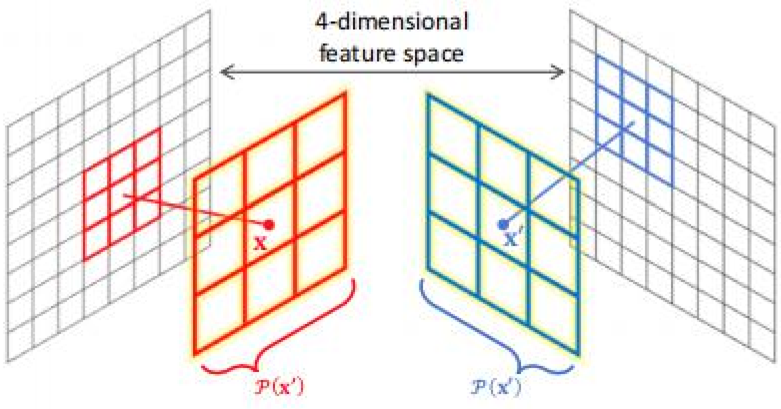


ResNet 50 $\rightarrow$ $\mathcal{L}p = 4, 6, 3$

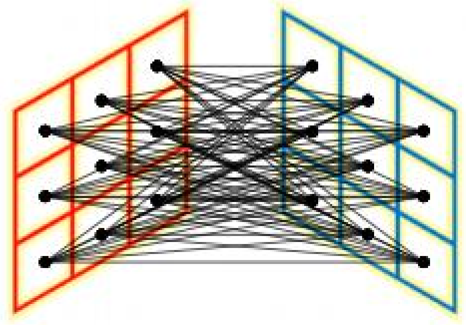
layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

01 Hypercorrelation Squeeze for Few-Shot Segmentation

- **4D conv**



4D convolution convolved at position $(\mathbf{x}, \mathbf{x}')$



$\mathcal{P}(\mathbf{x}, \mathbf{x}') = \mathcal{P}(\mathbf{x}) \times \mathcal{P}(\mathbf{x}')$

4D convolutional kernel

$$s(i,j) = (X * W)(i,j) = \sum_m \sum_n x(i-m, j-n) w(m,n)$$

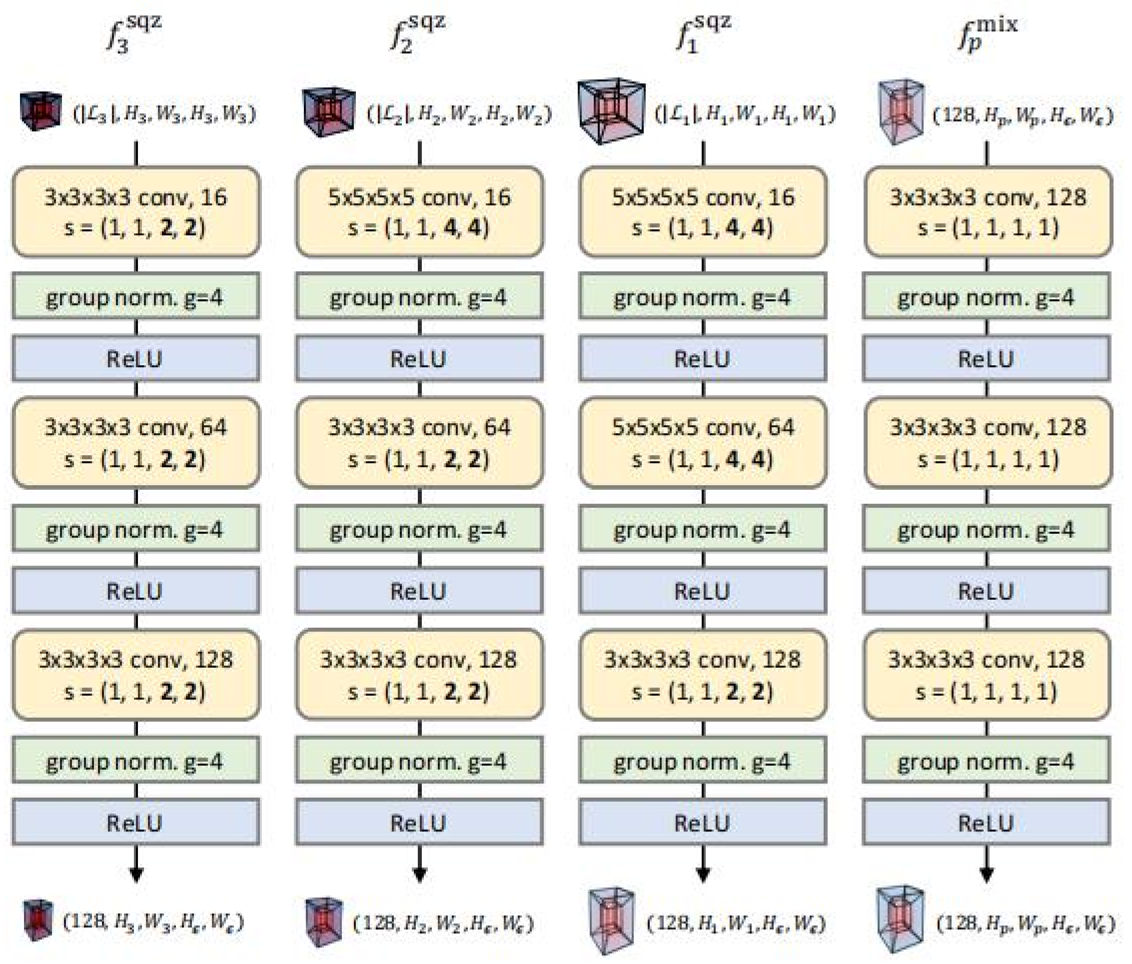
$$\boxed{s(i,j) = (X * W)(i,j) = \sum_m \sum_n x(i+m, j+n) w(m,n)}$$

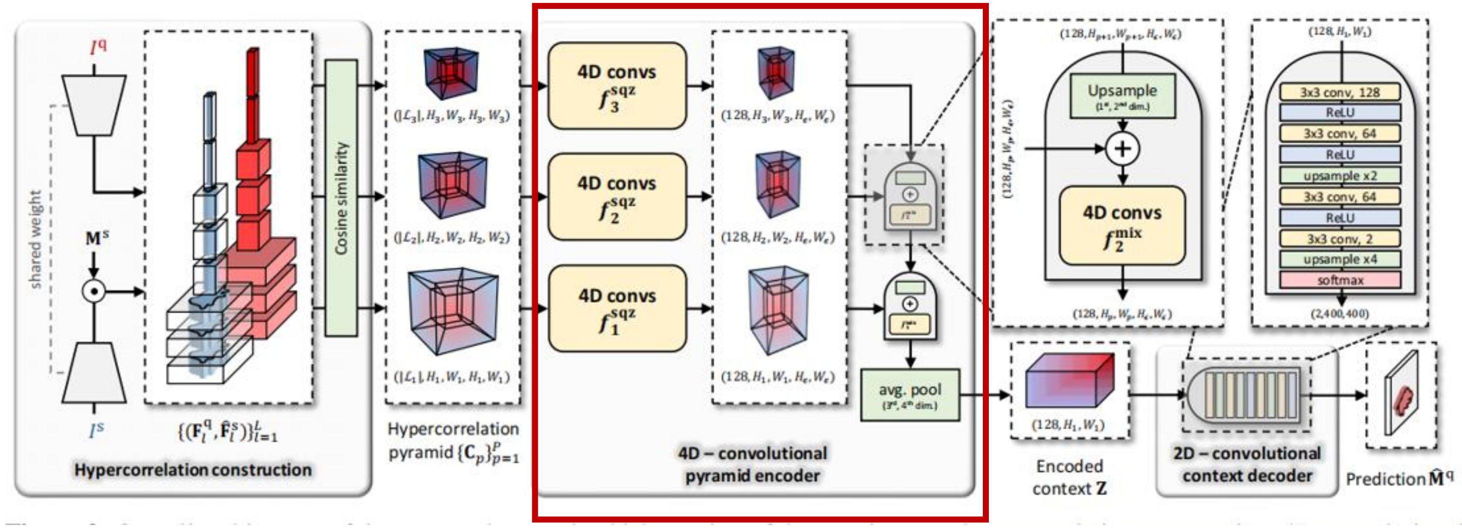
$$(c * k)(\mathbf{x}, \mathbf{x}') = \sum_{(\mathbf{p},\mathbf{p}') \in \mathcal{P}(\mathbf{x},\mathbf{x}')} c(\mathbf{p}, \mathbf{p}') k(\mathbf{p} - \mathbf{x}, \mathbf{p}' - \mathbf{x}'),$$

$$c \in \mathbb{R}^{H \times W \times H \times W}$$

01 Hypercorrelation Squeeze for Few-Shot Segmentation

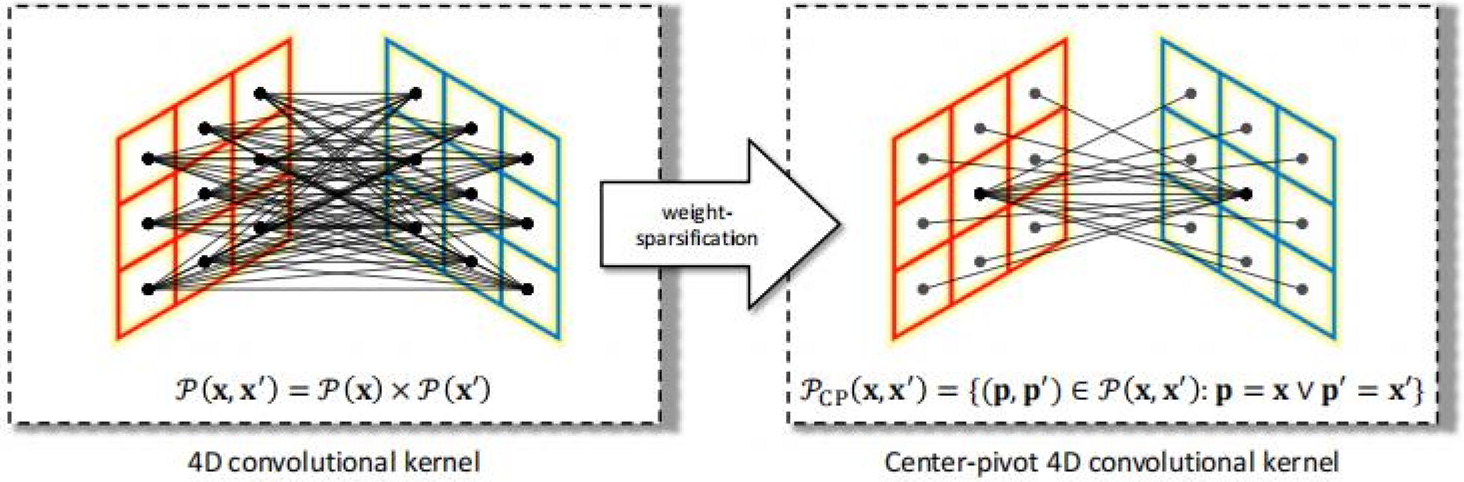
- **4D-convolutional pyramid encoder**
  - $$(c * k)(\mathbf{x}, \mathbf{x}') = \sum_{(\mathbf{p},\mathbf{p}') \in \mathcal{P}(\mathbf{x},\mathbf{x}')} c(\mathbf{p}, \mathbf{p}') k(\mathbf{p} - \mathbf{x}, \mathbf{p}' - \mathbf{x}'),$$





01 Hypercorrelation Squeeze for Few-Shot Segmentation

- **Center-pivot 4D convolutional**
  - Aims to disregard a large number of activations located at fairly insignificant positions in the 4D window, thereby focusing on a small subset of relevant activations only.



3*3 * 3*3

3*3 + 3*3

Backbone network	Methods	1-shot						5-shot						# learnable params
		5^0	5^1	5^2	5^3	mean	FB-IoU	5^0	5^1	5^2	5^3	mean	FB-IoU	
VGG16 [64]	OSLSM [61]	33.6	55.3	40.9	33.5	40.8	61.3	35.9	58.1	42.7	39.1	43.9	61.5	276.7M
	co-FCN [54]	36.7	50.6	44.9	32.4	41.1	60.1	37.5	50.0	44.1	33.9	41.4	60.2	34.2M
	AMP-2 [63]	41.9	50.2	46.7	34.7	43.4	61.9	40.3	55.3	49.9	40.1	46.4	62.1	15.8M
	PANet [75]	42.3	58.0	51.1	41.2	48.1	66.5	51.8	64.6	59.8	46.5	55.7	70.7	14.7M
	PFENet [70]	56.9	**68.2**	54.4	52.4	58.0	72.0	59.0	**69.1**	54.8	52.9	59.0	72.3	10.4M
	HSNet (ours)	**59.6**	65.7	**59.6**	**54.0**	**59.7**	**73.4**	**64.9**	69.0	**64.1**	**58.6**	**64.1**	**76.6**	**2.6M**
ResNet50 [17]	PANet [75]	44.0	57.5	50.8	44.0	49.1	-	55.3	67.2	61.3	53.2	59.3	-	23.5M
	PGNet [86]	56.0	66.9	50.6	50.4	56.0	69.9	57.7	68.7	52.9	54.6	58.5	70.5	17.2M
	PPNet [37]	48.6	60.6	55.7	46.5	52.8	69.2	58.9	68.3	66.8	58.0	63.0	75.8	31.5M
	PFENet [70]	61.7	69.5	55.4	56.3	60.8	73.3	63.1	70.7	55.8	57.9	61.9	73.9	10.8M
	RePRI [4]	59.8	68.3	**62.1**	48.5	59.7	-	64.6	71.4	**71.1**	59.3	66.6	-	-
	HSNet (ours)	**64.3**	**70.7**	60.3	**60.5**	**64.0**	**76.7**	**70.3**	**73.2**	67.4	**67.1**	**69.5**	**80.6**	**2.6M**
ResNet101 [17]	FWB [46]	51.3	64.5	56.7	52.2	56.2	-	54.8	67.4	62.2	55.3	59.9	-	43.0M
	PPNet [37]	52.7	62.8	57.4	47.7	55.2	70.9	60.3	70.0	**69.4**	60.7	65.1	77.5	50.5M
	DAN [74]	54.7	68.6	57.8	51.6	58.2	71.9	57.9	69.0	60.1	54.9	60.5	72.3	-
	PFENet [70]	60.5	69.4	54.4	55.9	60.1	72.9	62.8	70.4	54.9	57.6	61.4	73.5	10.8M
	RePRI [4]	59.6	68.6	**62.2**	47.2	59.4	-	66.2	71.4	67.0	57.7	65.6	-	-
	HSNet (ours)	**67.3**	**72.3**	62.0	**63.1**	**66.2**	**77.6**	**71.8**	**74.4**	67.0	**68.3**	**70.4**	**80.6**	**2.6M**
	HSNet† (ours)	66.2	69.5	53.9	56.2	61.5	72.5	68.9	71.9	56.3	57.9	63.7	73.8	**2.6M**

Table 1: Performance on PASCAL-5^i [61] in mIoU and FB-IoU. Some results are from [4, 37, 70, 74, 80]. Superscript † denotes our mode ***without*** support feature masking (Eqn. 1). Numbers in bold indicate the best performance and underlined ones are the second best.

Backbone network	Methods	1-shot 20^0	1-shot 20^1	1-shot 20^2	1-shot 20^3	1-shot mean	1-shot FB-IoU	5-shot 20^0	5-shot 20^1	5-shot 20^2	5-shot 20^3	5-shot mean	5-shot FB-IoU
ResNet50 [17]	PPNet [37]	28.1	30.8	29.5	27.7	29.0	-	39.0	40.8	37.1	37.3	38.5	-
	PMM [80]	29.3	34.8	27.1	27.3	29.6	-	33.0	40.6	30.3	33.3	34.3	-
	RPMM [80]	29.5	36.8	28.9	27.0	30.6	-	33.8	42.0	33.0	33.3	35.5	-
	PFENet [70]	**36.5**	38.6	34.5	33.8	35.8	-	36.5	43.3	37.8	38.4	39.0	-
	RePRI [4]	32.0	38.7	32.7	33.1	34.1	-	39.3	45.4	39.7	41.8	41.6	-
	HSNet (ours)	36.3	**43.1**	**38.7**	**38.7**	**39.2**	**68.2**	**43.3**	**51.3**	**48.2**	**45.0**	**46.9**	**70.7**
ResNet101 [17]	FWB [46]	17.0	18.0	21.0	28.9	21.2	-	19.1	21.5	23.9	30.1	23.7	-
	DAN [74]	-	-	-	-	24.4	62.3	-	-	-	-	29.6	63.9
	PFENet [70]	36.8	41.8	38.7	36.7	38.5	63.0	40.4	46.8	43.2	40.5	42.7	65.8
	HSNet (ours)	**37.2**	**44.1**	**42.4**	**41.3**	**41.2**	**69.1**	**45.9**	**53.0**	**51.8**	**47.1**	**49.5**	**72.4**

Table 2: Performance on COCO-20^i [46] in mIoU and FB-IoU. The results of other methods are from [4, 37, 70, 74, 80].

Backbone network	Methods	mIoU 1-shot	mIoU 5-shot
VGG16 [64]	OSLSM [61]	70.3	73.0
	GNet [55]	71.9	74.3
	FSS [33]	73.5	80.1
	DoG-LSTM [2]	80.8	83.4
	HSNet (ours)	**82.3**	**85.8**
ResNet50 [17]	HSNet (ours)	**85.5**	**87.8**
ResNet101 [17]	DAN [74]	85.2	88.1
	HSNet (ours)	**86.5**	**88.5**

Table 3: Mean IoU comparison on FSS-1000 [33]. Some results are from [2, 74].

01 Hypercorrelation Squeeze for Few-Shot Segmentation

- **Experiments**

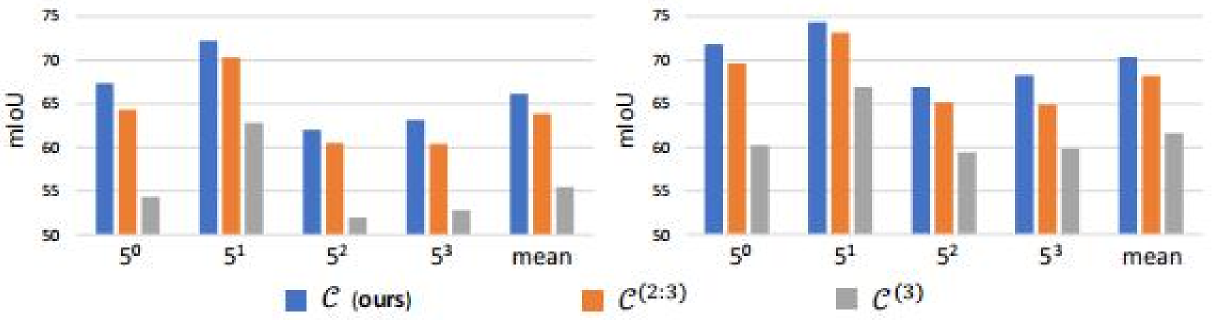


Figure 6: Ablation study on pyramid layers on PASCAL-5^i [61] dataset in 1-shot (left) and 5-shot (right) mIoU results.

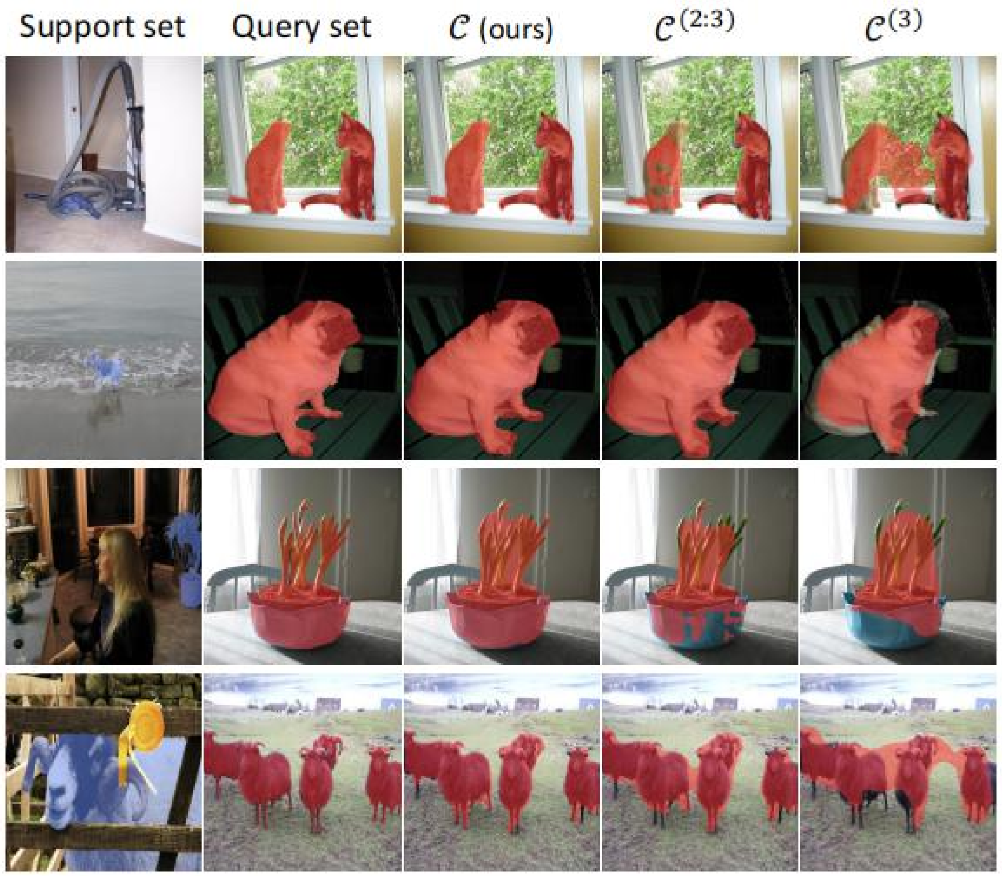
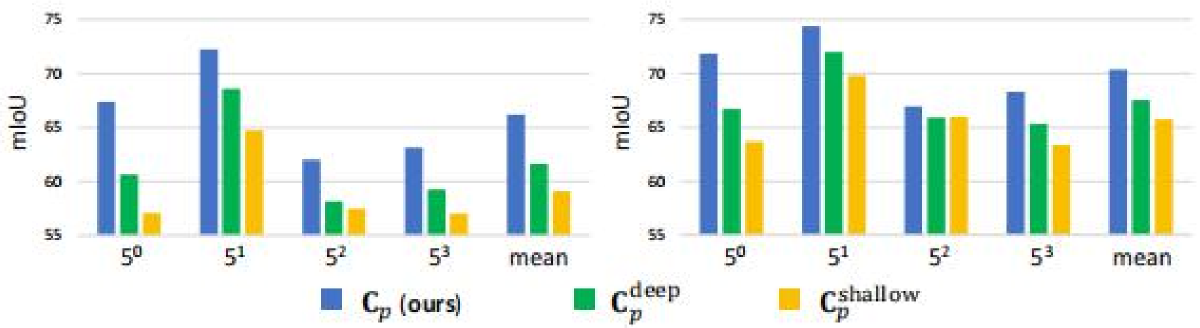


Figure 7: Ablation study on hypercorrelation pyramid layers.

01 Hypercorrelation Squeeze for Few-Shot Segmentation

- **Experiments**



Ablation study on hypercorrelations. To study the effect of intermediate correlations $\{\hat{\mathbf{C}}_l\}_{l \in \mathcal{L}_p}$ in hypercorrelation $\mathbf{C}_p \in \mathbb{R}^{|\mathcal{L}_p| \times H_p \times W_p \times H_p \times W_p}$, we form single-channel hypercorrelations using only *a single intermediate correlation*. Specifically, we form two different single-channel hypercorrelations using the smallest (shallow) and largest (deep) layer indices in $\mathcal{L}_p$ and denote the hypercorrelations as $\mathbf{C}_p^{\text{shallow}}, \mathbf{C}_p^{\text{deep}} \in \mathbb{R}^{1 \times H_p \times W_p \times H_p \times W_p}$, and compare the results with ours ($\mathbf{C}_p$) in Fig. 5. The large performance gaps between $\mathbf{C}_p$ and the single-channel hypercorrelations confirm that capturing diverse correlation patterns from dense intermediate CNN layers is crucial in effective pattern analyses. Performance degradation from $\mathbf{C}_p^{\text{deep}}$ to $\mathbf{C}_p^{\text{shallow}}$ indicates that reliable feature representations typically appear at deeper layers of a CNN.

01 Hypercorrelation Squeeze for Few-Shot Segmentation

- **Experiments**
  - We use the same training/test folds where object classes in training and testing do not overlap.

Method	COCO→PASCAL 1-shot	COCO→PASCAL 5-shot	# params to train	data augmentation used during training
$\text{PFENet}_{\text{res50}}$ [70]	61.1	63.4	10.8M	flip, rotate, crop
$\text{RePRI}_{\text{res50}}$ [4]	**63.2**	67.7	46.7M	flip
$\text{HSNet}_{\text{res50}}$(ours)	61.6	**68.7**	**2.6M**	**none**
$\text{HSNet}_{\text{res101}}$(ours)	**64.1**	**70.3**	**2.6M**	**none**

Table 4: Domain shift results. Subscripts denote backbone.

01 Hypercorrelation Squeeze for Few-Shot Segmentation

• **Experiments**

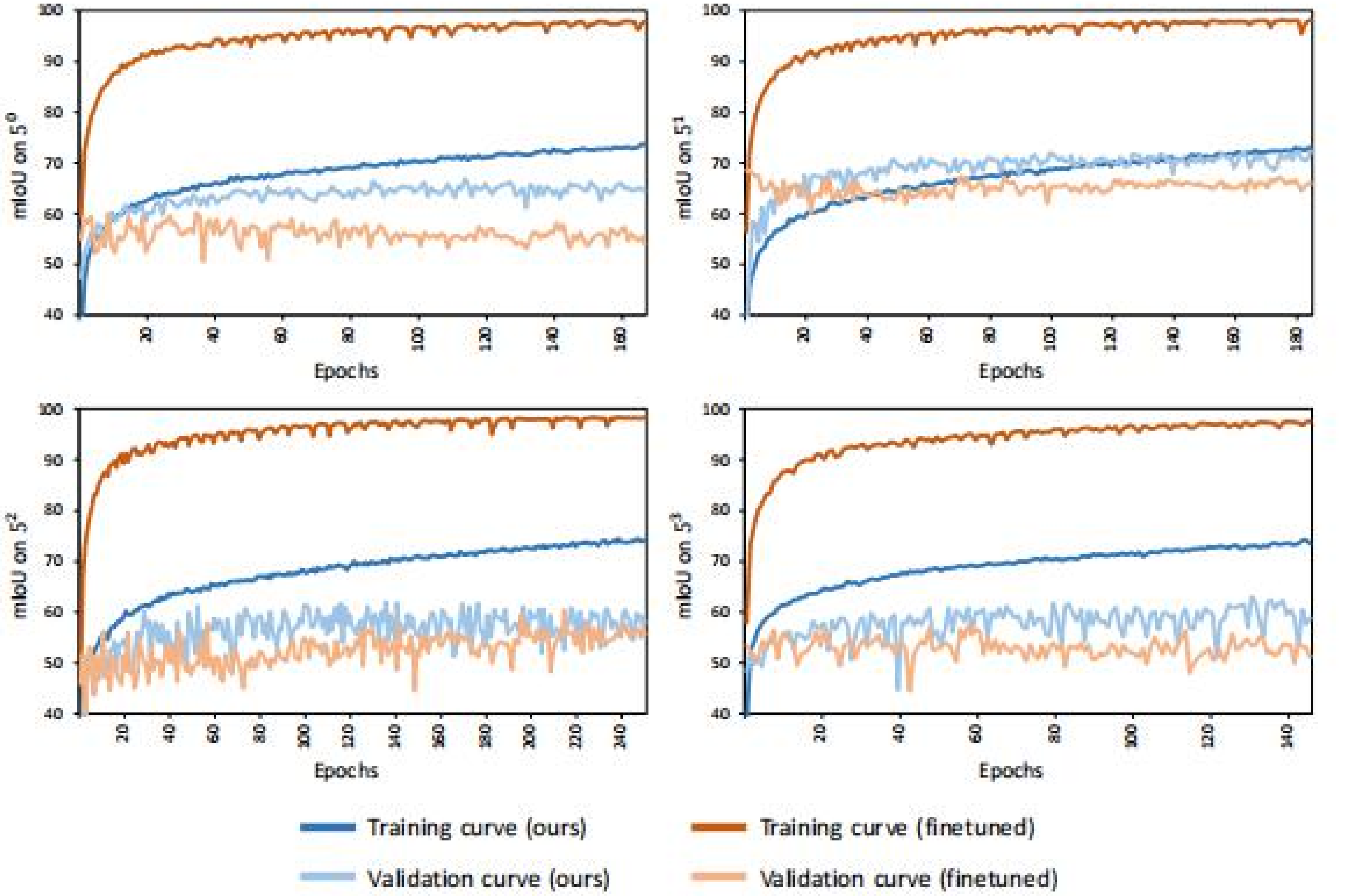


Figure 9: Learning curves (x-axis: epoch, y-axis: mIoU) on PASCAL-5^i. We carefully tuned the learning rate of the backbone and set it to 100 times smaller than the layers in HSNet (1e-5).

01 Hypercorrelation Squeeze for Few-Shot Segmentation

- **Experiments**

Kernel type	1-shot 5^0	5^1	5^2	5^3	mean	5-shot 5^0	5^1	5^2	5^3	mean	# learnable params	time (ms)	memory footprint (GB)	FLOPs (G)
Original 4D kernel [58]	64.5	71.4	62.3	61.7	64.9	70.8	**74.8**	**67.4**	67.5	70.1	11.3M	512.17	4.12	702.35
Separable 4D kernel [81]	66.1	72.0	**63.2**	62.6	65.9	71.2	74.1	67.2	68.1	70.2	4.4M	28.48	1.50	28.40
Center-pivot 4D kernel (ours)	**67.3**	**72.3**	62.0	**63.1**	**66.2**	**71.8**	74.4	67.0	**68.3**	**70.4**	**2.6M**	**25.51**	**1.39**	**20.56**

Table 5: Comparison between three different 4D conv kernels in model size, per-episode inference time, memory consumption and FLOPs. For fair comparison, the inference times of all the models are measured on a machine with an Intel i7-7820X and an NVIDIA Titan-XP.

[81] Volumetric correspondence networks for optical flow. (NeurIPS2019)