2021

인공지능세미나

김범채

AILab. 연구소장

2021.06.10.

주식회사 바스젠바이오

Hypercorrelation Squeeze for Few-Shot Segmentation

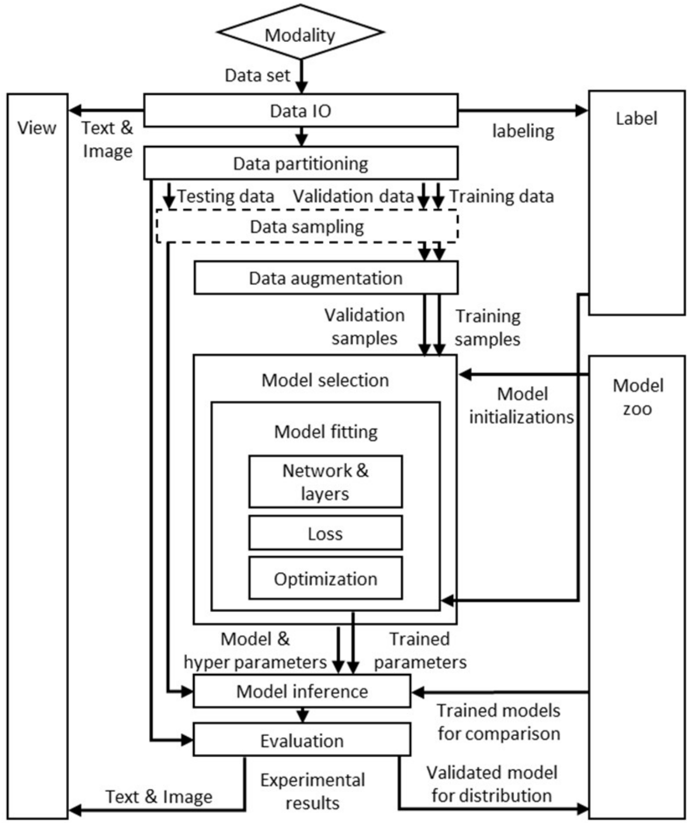
1. Intro
2. Related
3. Methodology
4. Experiment
5. Conclusion

INTRO

- Data Labeling
- Data Partitioning
- Data Sampling
- Data Augmentation

- Model Selection
- Model Fitting
    - Network & Layers
    - Loss
    - Optimization
- Model Inference

- Evaluation



구성	설명
Augmentation	Training & validation의 수가 부족하거나, 특정 조건을 부여할 때 사용
Model zoo	다양한 모델 저장된 모델 명 DB
Model fitting	모델에 맞는 하이퍼 파라미터를 선정
Model inference	학습된 모델의 예측 값 추출
Evaluation	모델 검증, Experiment results: accuracy, sensitivity, specificity, AUC

Problem

- To perform segmentation given only a few annotated examples.
- To avoid the risk of overfitting due to insufficient training data, used meta-learning approach called episodic training.

[그림 2] 퓨샷 러닝 모델의 일반화 성능을 높이기 위해 에피소딕 훈련에 기반을 둔 메타 러닝 방식이 널리 사용된다.

Problem

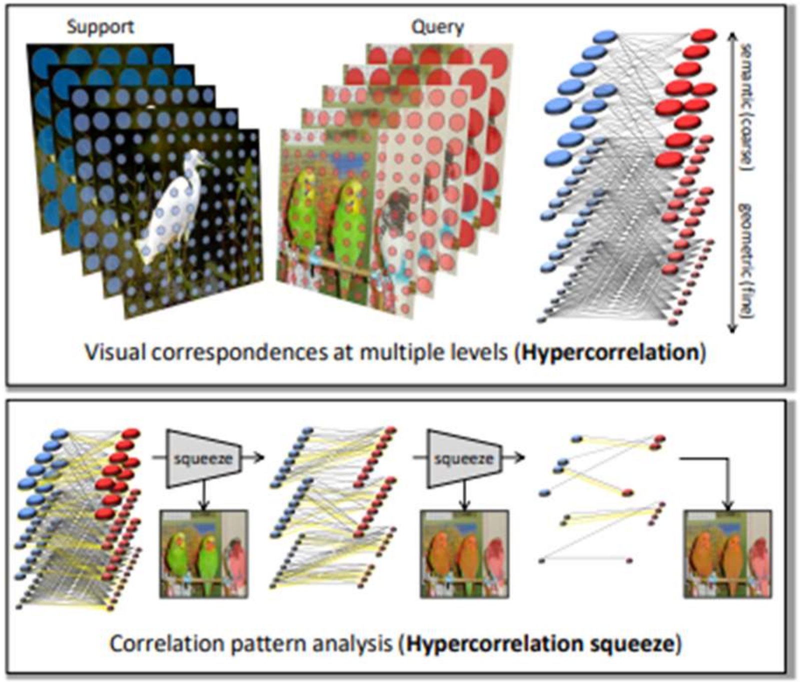


Figure 1: Our model performs visual reasoning in coarse-to-fine manner by gradually squeezing high-dimensional hypercorrelation to the target segmentation mask with efficient 4D convolutions.

Coarse-grained classification

Coarse-grained classification 은 Cifar10, Cifar100, MNIST 등의 데이터셋을 사용해 classification 하는 것이 Coarse-grained classification 의 예시입니다.

Fine-grained classification

Fine-grained classification 은 Coarse-grained classification 보다 더 세밀하게 classification 을 한다고 이해할 수 있습니다. Stanford dogs 가 가장 유명한 Fine-grained classification dataset 인데, 아래 이미지를 보시면

이미지 출처 [1]

Proposed Approach

- HSNet, which captures relevant patterns in multi-level feature correlations
- we adopt an encoder-decoder structure in our architecture
  - the encoder gradually squeezes dimension of the input hypercorrelations by aggregating their local information to a global context
  - the decoder processes the encoded context to predict a query mask.

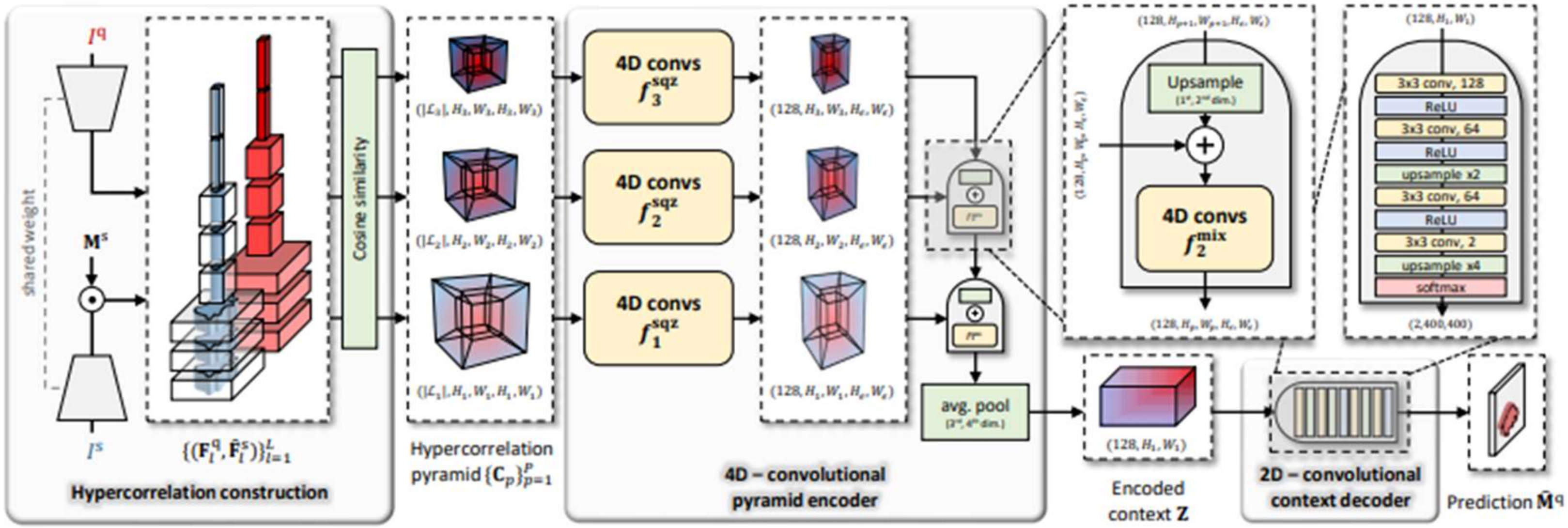
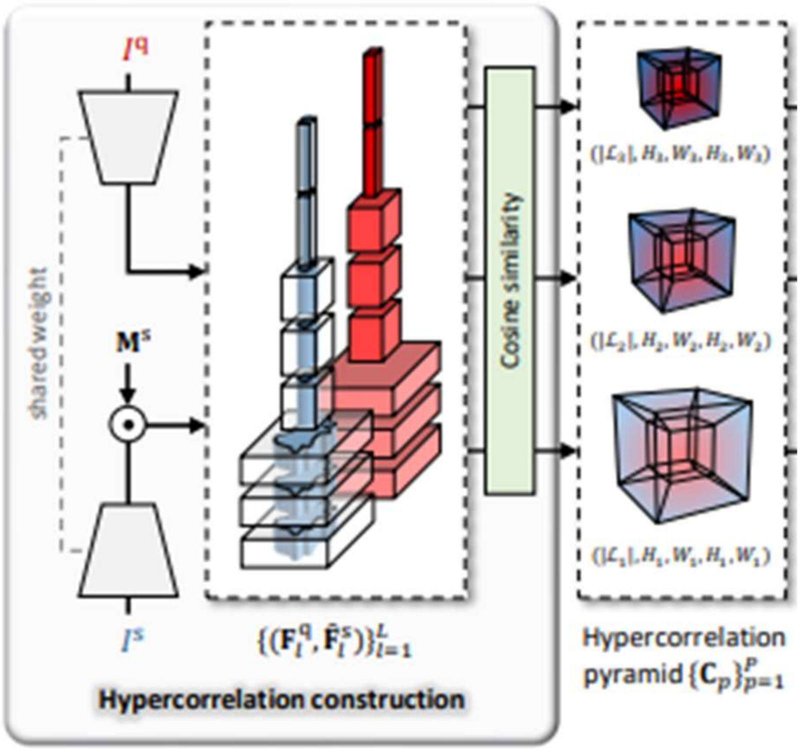


Figure 2: Overall architecture of the proposed network which consists of three main parts: hypercorrelation construction, 4D-convolutional pyramid encoder, and 2D-convolutional context decoder. We refer the readers to Sec. 4 for details of the architecture.



ilarities between the support and query images. Given a pair of query and support images, $I^q, I^s \in \mathbb{R}^{3\times H\times W}$, the backbone network produces a sequence of L pairs of intermediate feature maps $\{(\mathbf{F}_l^q, \mathbf{F}_l^s)\}_{l=1}^L$. We mask each support feature map $\mathbf{F}_l^s \in \mathbb{R}^{C_l\times H_l\times W_l}$† using the support mask $\mathbf{M}^s \in \{0,1\}^{H\times W}$ to discard irrelevant activations for reliable mask prediction:

$$\hat{\mathbf{F}}_l^s = \mathbf{F}_l^s \odot \zeta_l(\mathbf{M}^s), \quad (1)$$

where $\odot$ is Hadamard product and $\zeta_l(\cdot)$ is a function that bilinearly interpolates input tensor to the spatial size of the feature map $\mathbf{F}_l^s$ at layer l followed by expansion along channel dimension such that $\zeta_l : \mathbb{R}^{H\times W} \to \mathbb{R}^{C_l\times H_l\times W_l}$. For the subsequent hypercorrleation construction, a pair of query and masked support features at each layer forms a 4D correlation tensor $\hat{\mathbf{C}}_l \in \mathbb{R}^{H_l\times W_l\times H_l\times W_l}$ using cosine similarity:

$$\hat{\mathbf{C}}_l(\mathbf{x}^q, \mathbf{x}^s) = \mathrm{ReLU}\left(\frac{\mathbf{F}_l^q(\mathbf{x}^q)\cdot\hat{\mathbf{F}}_l^s(\mathbf{x}^s)}{\|\mathbf{F}_l^q(\mathbf{x}^q)\|\|\hat{\mathbf{F}}_l^s(\mathbf{x}^s)\|}\right), \quad (2)$$

where $\mathbf{x}^q$ and $\mathbf{x}^s$ denote 2-dimensional spatial positions of feature maps $\mathbf{F}_l^q$ and $\hat{\mathbf{F}}_l^s$ respectively, and ReLU suppresses noisy correlation scores. From the resultant set of 4D correlations $\{\hat{\mathbf{C}}_l\}_{l=1}^L$, we collect 4D tensors having the same spatial sizes‡ and denote the subset as $\{\hat{\mathbf{C}}_l\}_{l\in\mathcal{L}_p}$

Hadamard product: https://ko.wikipedia.org/wiki/%EC%95%84%EB%8B%A4%EB%A7%88%EB%A5%B4_%EA%B3%B1

4.2. 4D-convolutional pyramid encoder

Our encoder network takes the hypercorrelation pyramid $\mathcal{C} = \{\mathbf{C}_p\}_{p=1}^{P}$ to effectively squeeze it into a condensed feature map $\mathbf{Z} \in \mathbb{R}^{128 \times H_1 \times W_1}$. We achieve this correlation learning using two types of building blocks: a squeezing block f_p^{sqz} and a mixing block f_p^{mix}. Each block

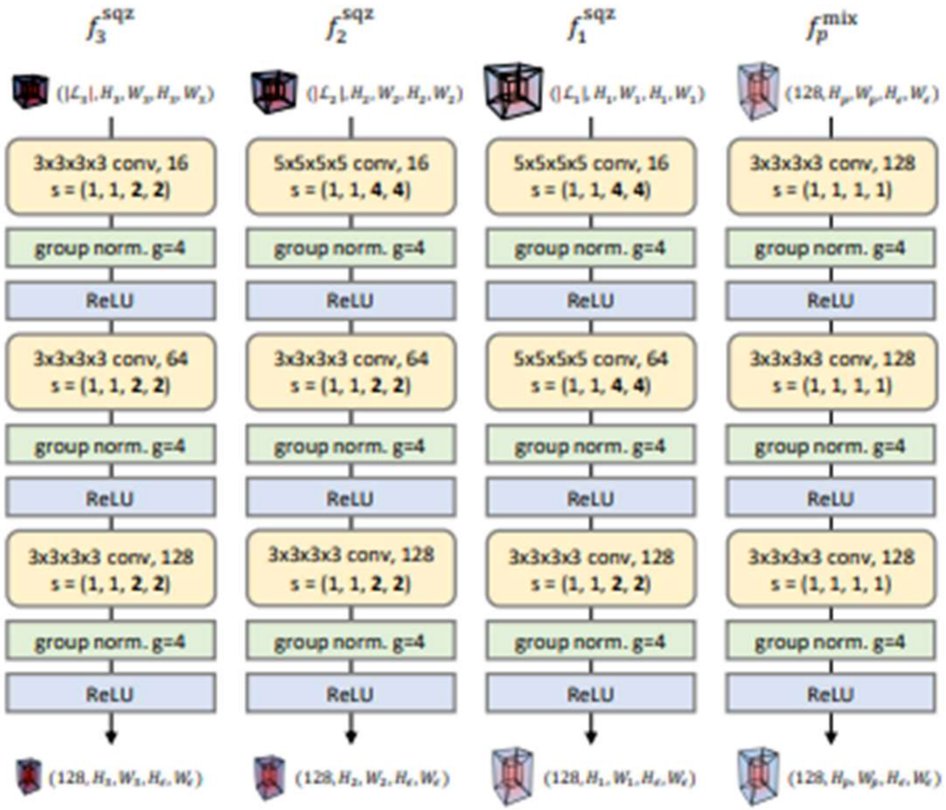
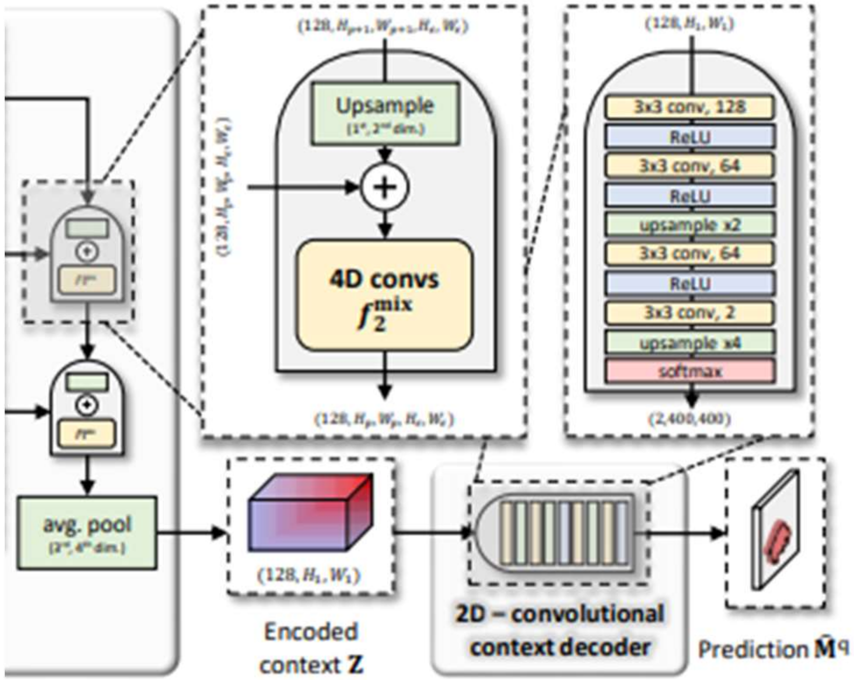


Figure 3: Building blocks in Hypercorrelation Squeeze Networks. s and g denotes strides of 4D conv and the number of groups in group normalization [75] respectively. Note $p \in \{1, 2\}$ for f_p^{mix}.



4.3. 2D-convolutional context decoder

The decoder network consists of a series of 2D convolutions, ReLU, and upsampling layers followed by softmax function as illustrated in Fig. 2. The network takes the context representation $\mathbf{Z}$ and predicts two-channel map $\hat{\mathbf{M}}^{\mathrm{q}} \in [0, 1]^{2\times H\times W}$ where two channel values indicate probabilities of foreground and background. During training, the network parameters are optimized using the mean of cross-entropy loss between the prediction $\hat{\mathbf{M}}^{\mathrm{q}}$ and the ground-truth $\mathbf{M}^{\mathrm{q}}$ over all pixel locations. During testing, we take the maximum channel value at each pixel to obtain final query mask prediction $\bar{\mathbf{M}}^{\mathrm{q}} \in \{0, 1\}^{H\times W}$ for evaluation.

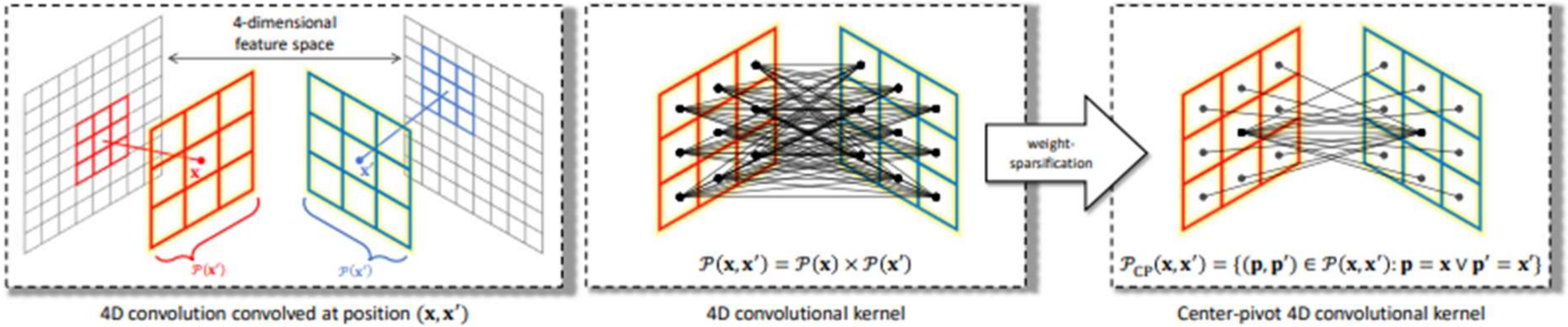


Figure 4: 4D convolution (left) and weights of 4D kernel [55, 77] (middle) and center-pivot 4D kernel (right). Each black wire that connects two different pixel locations represent a single weight of the 4D kernel. The kernel size used in this example is $(3, 3, 3, 3)$, *i.e.*, $\hat{k} = 3$.

Experimental results

1. Resuts on PASCAL-5^i dataset.

Backbone network	Methods	1-shot 5^0	5^1	5^2	5^3	mean	FB-IoU	5-shot 5^0	5^1	5^2	5^3	mean	FB-IoU	# learnable params
VGG16 [61]	OSLSM [58]	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 [51]	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 [60]	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 [72]	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 [67]	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 [72]	44.0	57.5	50.8	44.0	49.1	-	55.3	67.2	61.3	53.2	59.3	-	23.5M
	PGNet [82]	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 [35]	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 [67]	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 [43]	51.3	64.5	56.7	52.2	56.2	-	54.8	67.4	62.2	55.3	59.9	-	43.0M
	PPNet [35]	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 [71]	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 [67]	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 in mIoU and FB-IoU. Some results are from [4, 35, 67, 71, 76]. Superscript † denotes our model without support feature masking (Eqn. 1). Numbers in bold indicate the best performance and underlined ones are the second best.

2. Results on COCO-20^i and FSS-1000.

Backbone network	Methods	1-shot 20^0	20^1	20^2	20^3	mean	FB-IoU	5-shot 20^0	20^1	20^2	20^3	mean	FB-IoU
ResNet50 [17]	PPNet [35]	28.1	30.8	29.5	27.7	29.0	-	39.0	40.8	37.1	37.3	38.5	-
	PMM [76]	29.3	34.8	27.1	27.3	29.6	-	33.0	40.6	30.3	33.3	34.3	-
	RPMM [76]	29.5	36.8	28.9	27.0	30.6	-	33.8	42.0	33.0	33.3	35.5	-
	PFENet [67]	**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 [43]	17.0	18.0	21.0	28.9	21.2	-	19.1	21.5	23.9	30.1	23.7	-
	DAN [71]	-	-	-	-	24.4	62.3	-	-	-	-	29.6	63.9
	PFENet [67]	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**

Backbone network	Methods	mIoU 1-shot	mIoU 5-shot
VGG16 [61]	OSLSM [58]	70.3	73.0
	GNet [52]	71.9	74.3
	FSS [31]	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 [71]	85.2	88.1
	HSNet (ours)	**86.5**	**88.5**

Table 2. Performance on COCO-20^i (left) and FSS-1000 (right) in mIoU and FB-IoU. Some results are from [2, 4, 35, 67, 71, 76].

3. Qualitative results

Figure 4. Qualitative (1-shot) results on dataset in presence of large differences in object scales and extremely small objects.