Abstract

Few-shot semantic segmentation aims at learning to segment a target object from a query image using only a few annotated support images of the target class. This challenging task 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. It extracts diverse features from different levels of intermediate convolutional layers and constructs a collection of 4D correlation tensors, i.e., hypercorrelations. Using efficient center-pivot 4D convolutions in a pyramidal architecture, the method gradually squeezes high-level semantic and low-level geometric cues of the hypercorrelation into precise segmentation masks in coarse-to-fine manner. The significant performance improvements on standard few-shot segmentation benchmarks of PASCAL-50, COCO-2017, and FSS-1000 verify the efficacy of the proposed method.

1. Introduction

The advent of deep convolutional neural networks [17, 20, 64] has promoted dramatic advances in many computer vision tasks including object tracking [28, 29, 45], visual correspondence [22, 44, 48], and semantic segmentation. To cope with the challenge, there have been various attempts in semi- and weakly-supervised segmentation approaches [6, 26, 39, 66, 72, 77, 88] which in turn effectively alleviated the data-hunger issue. However, given only a few annotated training examples, the problem of poor generalization ability of the deep networks is yet the primary concern that many few-shot segmentation methods [10, 12, 13, 19, 33, 36, 37, 46, 54, 61, 63, 69, 70, 74, 75, 80, 83, 86, 87, 89] struggle to address.
framework, dubbed Hypercorrelation Squeeze Networks (HSNet), for the task of few-shot semantic segmentation. As illustrated in Fig. 1, our network exploits diverse geometric/semantic feature representations from many different intermediate CNN layers to construct a collection of 4D correlation tensors, i.e., hypercorrelations, which represent a rich set of correspondences in multiple visual aspects. Following the work of FPN [34], we adapt pyramidal design to capture both high-level semantic and low-level geometric cues for precise mask prediction in coarse-to-fine manner using deeply stacked 4D conv layers. To reduce computational burden caused by such heavy use of high-dimensional convs, we devise an efficient 4D kernel via reasonable weight-sparseification which enables real-time inference while being more effective and light-weight than the existing ones. The improvements on standard few-shot segmentation benchmarks of PASCAL-5 [61], COCO-20 [35], and FSS-1000 [33] verify the efficacy of the proposed method.

2. Related Work

Semantic segmentation. The goal of semantic segmentation is to classify each pixel of an image into one of the predefined object categories. Prevalent segmentation approaches [5, 7, 47, 49, 52, 62, 76] typically employ encoder-decoder structure in their architecture; the encoder aggregates features along deep convolutional pathways and provides high-dimensional feature map in low-resolution and the corresponding decoder takes the output to predict segmentation mask by reversing this process [49]. Although the methods clearly show the effectiveness of the encoder-decoder architecture in the task of semantic segmentation, offering useful insights to our study, they still suffer apparent disadvantages of data-driven nature of neural networks: lack of generalizibility under insufficient training data.

Few-shot learning. To resolve the generalization problem, many recent approaches to image classification made various attempts in training deep networks with a few annotated examples [1, 18, 25, 31, 50, 53, 59, 65, 67, 73, 79, 84, 85]. Vinyals et al. [73] propose matching networks for one-shot learning; the method utilizes a special kind of mini-batches called episodes to match training and testing environments, facilitating better generalization on novel classes. Snell et al. [65] introduce prototypical networks which compute distances between representative embeddings, i.e., prototypes, for few-shot classification. With the growing interests in few-shot learning in classification domain, the problem of few-shot segmentation has attracted a great deal of attention as well. Shaban et al. [61] propose one-shot semantic segmentation networks which (meta-) learns to generate parameters of FCN [62]. Inspired by the prototypical networks [65], utilizing prototype representations to guide mask prediction in a query image became a popular paradigm in few-shot segmentation literature [10, 36, 37, 46, 63, 75, 80, 87, 89].

Witnessing the limitation of prototypical approaches, e.g., loss of spatial structure due to masked average pooling [89], work of [74, 86] build pair-wise feature correlations, e.g., graph attention, to retain the spatial structure of the images for fine-grained mask prediction. Note that both prototypical and graph-based methods fundamentally focus on learning to find reliable correspondences between support and query images for accurate mask prediction. In this work, we advance this idea and focus on learning to analyze correspondences using adequately designed learnable layers, e.g., 4D convolutions [58], for effective semantic segmentation.

Learning visual correspondences. The task of visual correspondence aims to find reliable correspondences under challenging degree of variations [3, 14, 15, 43, 60]. Many methods [21, 22, 30, 38, 42, 44, 56, 58, 81] typically built upon convolutional features pretrained on classification task [9], showing they serve as good transferable representations. Recent approaches to semantic correspondence [21, 38, 42, 44] show that efficiently exploiting different levels of convolutional features distributed over all intermediate layers clearly benefits matching accuracy. In wide-baseline matching literature, a trending choice is to employ 4D convolutions [30, 41, 57, 58, 71] on dense feature matches to identify spatially consistent matches by analyzing local patterns in 4D space. The use of multi-level features and relational pattern analysis using 4D convs are the two widely adopted techniques in the field of visual correspondence.

In this paper we adapt the two most influential methodologies in visual correspondence to tackle few-shot segmentation: multi-level features and 4D convolutions. Inspired by the previous matching methods [42, 44, 27], which use multi-level features to build effective “appearance features”, we construct high-dimensional “relational features” using intermediate CNN features and process them with a series of 4D convolutions. However, their quadratic complexity still remains a major bottleneck in designing cost-effective deep networks, constraining many previous matching methods [30, 57, 58, 71] to use only a few 4D conv layers. To resolve the issue, we develop a light-weight 4D convolutional kernel by collecting only a small subset of vital parameters for effective pattern recognition, which eventually leads to an efficient decomposition into a pair of 2D conv kernels with a linear complexity. Our contributions can be summarized as follows:

- We present the Hypercorrelation Squeeze Networks that analyze dense feature matches of diverse visual aspects using deeply stacked 4D conv layers.
- We propose center-pivot 4D conv kernel which is more effective than the existing one in terms both accuracy and speed, achieving real-time inference.
- The proposed method sets a new state of the art on three standard few-shot segmentation benchmarks: PASCAL-5 [61], COCO-20 [35], and FSS-1000 [33].
3. Problem Setup

The goal of few-shot semantic segmentation is to perform segmentation given only a few annotated examples. To avoid the risk of overfitting due to insufficient training data, we adopt widely used meta-learning approach called episodic training [73]. Let us denote respective training and test set as \( D_{\text{train}} \) and \( D_{\text{test}} \), which are disjoint with respect to object classes. Both sets consist of multiple episodes, each of which is composed of a support set \( S = (I^*, M^*) \) and a query set \( Q = (I^0, M^0) \) where \( I^* \) and \( M^* \) are an image and its corresponding mask label respectively. During training, our model iteratively samples an episode from \( D_{\text{train}} \) to learn a mapping from \( (I^*, M^*, I^0) \) to query mask \( M^0 \). Once the model is trained, it uses the learned mapping for evaluation without further optimization, i.e., the model takes randomly sampled \( (I^0, M^0, I^0) \) from \( D_{\text{test}} \) to predict query mask.

4. Proposed Approach

In this section, we present a novel few-shot segmentation architecture, Hypercorrelation Squeeze Networks (HSNet), which capture relevant patterns in multi-level feature correlations between a pair of input images to predict fine-grained segmentation mask in a query image. As illustrated in Fig. 2, we adopt an encoder-decoder structure in our architecture: the encoder gradually squeezes dimension of the input hyper-correlations by aggregating their local information to a global context, and the decoder processes the encoded context to predict a query mask. In Sec. 4.1-4.3, we demonstrate each pipeline in one-shot setting, i.e., the model predicts the query mask given \( I^0 \) and \( S = (I^*, M^*) \). In Sec. 4.4, to mitigate large resource demands of 4D convs, we present a light-weight 4D kernel which greatly improves model efficiency in terms of both memory and time. In Sec. 4.5, we demonstrate how the model can be easily extended to \( K \)-shot setting, i.e., \( S = \{(I_k, M_k)\}_{k=1}^K \), without loss of generality.

4.1. Hypercorrelation construction

Inspired by recent semantic matching approaches [38, 42, 44], our model exploits a rich set of features from the intermediate layers of a convolutional neural network to capture multi-level semantic and geometric patterns of similarities between the support and query images. Given a pair of query and support images, \( I^0, I^* \in \mathbb{R}^{H \times W} \), the backbone network produces a sequence of \( L \) pairs of intermediate feature maps \( \{(F_{l}^0, F_{l}^*)\}_{l=1}^L \). We mask each support feature map \( F_{l}^s \in \mathbb{R}^{C_l \times H_l \times W_l} \) using the support mask \( M^s \in \{0, 1\}^{H \times W} \) to discard irrelevant activations for reliable mask prediction:

\[
\tilde{F}_l^0 = F_l^0 \odot \zeta(M^s),
\]

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

\[
\tilde{C}_{l}(x^0, x^s) = \text{ReLU} \left( \frac{F_l^0(x^0) \cdot \tilde{F}_{l}^0(x^s)}{||F_l^0(x^0)|| \cdot ||\tilde{F}_{l}^0(x^s)||} \right),
\]

where \( x^0 \) and \( x^s \) denote 2-dimensional spatial positions of feature maps \( F_{l}^0 \) and \( \tilde{F}_{l}^0 \) respectively, and ReLU suppresses noisy correlation scores. From the resultant set of 4D correlations \( \{\tilde{C}_{l}\}_{l=1}^L \), we collect 4D tensors if they have the same spatial sizes and denote the subset as \( \{\tilde{C}_{l}\}_{l \in L_p} \), where \( L_p \) is a subset of CNN layer indices \( \{1, \ldots, L\} \) at some pyramidal layer \( p \). Finally, all the 4D tensors in \( \{\tilde{C}_{l}\}_{l \in L_p} \) are concatenated along channel dimension to form a hypercorrelation \( C_{p} \in \mathbb{R}^{|L_p| \times H_p \times W_p \times H_p \times W_p} \), where \( (H_p, W_p) \), with abuse of notation, represents the spatial resolution of the hypercorrelation at pyramidal layer.
p. Given P pyramidal layers, we denote hypercorrelation pyramid as $C = \{C_p\}_{p=1}^P$, representing a rich collection of feature correlations from multiple visual aspects.

### 4.2. 4D-convolutional pyramid encoder

Our encoder network takes the hypercorrelation pyramid $C_0 = \{C_{p}\}_{p=1}^P$ to effectively squeeze it into a condensed feature map $Z \in \mathbb{R}^{128 \times H_1 \times W_1}$. 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 consists of three sequences of multi-channel 4D convolution, group normalization [78], and ReLU activation as illustrated in Fig. 3. In the squeezing block $f_p^{sqz}$, large strides periodically squeeze the last two (support) spatial dimensions of $C_p$ down to $(H_s, W_s)$ while the first two spatial (query) dimensions remain the same as $(H_q, W_q)$, i.e., $f_p^{sqz} : \mathbb{R}^{|L| \times H_q \times W_q \times H_p \times W_p} \rightarrow \mathbb{R}^{128 \times H_s \times W_s \times H_q \times W_q}$, where $H_s > H_q$ and $W_s > W_q$. Similar to FPN [34] structure, two outputs from adjacent pyramidal layers, $p$ and $p+1$, are merged by element-wise addition after upsampling the (query) spatial dimensions of the upper layer output by a factor of 2. The mixing block $f_p^{mix} : \mathbb{R}^{128 \times H_s \times W_s \times H_q \times W_q} \rightarrow \mathbb{R}^{128 \times H_s \times W_s \times H_q \times W_q}$ then processes this mixture with 4D convolutions to propagate relevant information to lower layers in a top-down fashion. After the iterative propagation, the output tensor of the lowest mixing block $f_1^{mix}$ is further compressed by average-pooling its last two (support) spatial dimensions, which in turn provides a 2-dimensional feature map $Z \in \mathbb{R}^{128 \times H_s \times W_s}$ that signifies a condensed representation of the hypercorrelation $C$.

### 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 $Z$ and predicts two-channel map $M^q \in [0, 1]^{2 \times H \times W}$ 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 $M^q$ and the ground-truth $M^q$ over all pixel locations. During testing, we take the maximum channel value at each pixel to obtain final query mask prediction $\hat{M}^q \in \{0, 1\}^{H \times W}$ for evaluation.

### 4.4. Center-pivot 4D convolution

Apparently, our network with such a large number of 4D convolutions demands a substantial amount of resources due to the curse of dimensionality, which constrained many visual correspondence methods [22, 30, 32, 58, 71] to use only a few 4D conv layers. To address the concern, we revisit the 4D convolution operation and delve into its limitations. Then we demonstrate how a unique weight-sparsification scheme effectively resolves the issues.

---

**Figure 3:** Building blocks in Hypercorrelation Squeeze Networks. $s$ and $g$ denotes strides of 4D conv and the number of groups in group normalization [78] respectively. Note $p \in \{1, 2\}$ for $f_p^{mix}$.

**4D convolution and its limitation.** Typical 4D convolution parameterized by a kernel $k \in \mathbb{R}^{k \times k \times k \times k}$ on a correlation tensor $c \in \mathbb{R}^{H \times W \times H \times W}$ at position $(x, x') \in \mathbb{R}^{4 \times 4}$ is formulated as

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

where $\mathcal{P}(x, x')$ denotes a set of neighbourhood regions within the local 4D window centered on position $(x, x')$, i.e., $\mathcal{P}(x, x') = \mathcal{P}(x) \times \mathcal{P}(x')$ as visualized in Fig. 4. Although the use of 4D convolutions on a correlation tensor has shown its efficacy with good empirical performance in correspondence-related domains [22, 30, 32, 58, 71], its quadratic complexity with respect to the size of input features still remains a primary bottleneck. Another limiting factor is over-parameterization of the high-dimensional kernel: Consider a single activation in an $n$D tensor convolved by $n$D conv kernel. The number of times that the kernel processes this activation is exponentially proportional to $n$. This implies some unreliable input activations with large magnitudes may entail some noise in capturing reliable patterns as a result of their excessive exposure to the high-dimensional kernel. The work of [81] resolves the former problem (quadratic complexity) using spatially separable 4D kernels to approximate the 4D conv with two separate 2D kernels along with additional batch normalization layers [23] that settle the latter problem (numerical instability). In this work we introduce a novel weight-sparsification scheme to address both issues at the same time.

*The correlation tensor $c$ is the output of cosine similarity (Eqn. 2) between a pair of feature maps, $F, F' \in \mathbb{R}^{H \times W}$, and $x$ and $x'$ denote 2-dimensional spatial positions of the respective feature maps.*
Center-pivot 4D convolution. Our goal is to design a lightweight 4D kernel that is efficient in terms of both memory and time while effectively approximating the existing ones [58, 81]. We achieve this via a reasonable weight-sparsification; from a set of neighborhood positions within a local 4D window of interest, our kernel 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. Specifically, we consider the activations at positions that pivots either one of 2-dimensional centers, e.g., x or x', as the foremost influential ones as illustrated in Fig. 4. Given 4D position (x, x'), we collect its neighbors if and only if they are adjacent to either x or x' in its corresponding 2D subspace and define two respective sets as \( P_c(x, x') = \{(p, p') \in P(x, x') : p = x\} \) and \( P_{c'}(x, x') = \{(p, p') \in P(x, x') : p' = x'\} \). The set of center-pivot neighbors is defined as \( P_{CP}(x, x') = P_c(x, x') \cup P_{c'}(x, x') \). Based on these two subsets of neighbors, center-pivot 4D convolution can be formulated as a union of two separate 4D convolutions:

\[
(c * k_{CP})(x, x') = (c * k_c)(x, x') + (c * k_{c'})(x, x')
\]

(4)

where \( k_c \) and \( k_{c'} \) are 4D kernels convolved on \( P_c(x, x') \) and \( P_{c'}(x, x') \) respectively. Note that \( (c * k_c)(x, x') \) is equivalent to convolutions with a 2D kernel \( k_2^{2D} = k(0,:) \in \mathbb{R}^{k \times k} \) performed on 2D slice of 4D tensor \( (c * :)(x, :) \). Similarly, with \( k_{c'}^{2D} = k(:, 0) \in \mathbb{R}^{k \times k} \), we reformulate Eqn. 4 as follows

\[
(c * k_{CP})(x, x') = \sum_{p' \in P(x')} c(x, p') k_c^{2D}(p' - x') + \sum_{p \in P(x)} c(p, x') k_{c'}^{2D}(p - x),
\]

(5)

which performs two different convolutions on separate 2D subspace, having a linear complexity. In Sec. 5.2, we experimentally demonstrate the superiority of the center-pivot 4D kernels over the existing ones [58, 81] in terms of accuracy, memory, and time. We refer the readers to the Appendix A for a complete derivation of Eqn. 5.

4.5. Extension to K-shot setting

Our network can be easily extended to K-shot setting: Given K support image-mask pairs \( S = \{(I^q_k, M^q_k)\}_{k=1}^K \) and a query image \( I^q \), model performs K forward passes to provide a set of K mask predictions \( \{\hat{M}^q_k\}_{k=1}^K \). We perform voting at every pixel location by summing all the K predictions and divide each output score by the maximum voting score. We assign foreground labels to pixels if their values are larger than some threshold \( \tau \) whereas the others are classified as background. We set \( \tau = 0.5 \) in our experiments.

5. Experiment

In this section we evaluate the proposed method, compare it with recent state of the arts, and provide in-depth analyses of the results with ablation study.

Implementation details. For the backbone network, we employ VGG [64] and ResNet [17] families pre-trained on ImageNet [9], e.g., VGG16, ResNet50, and ResNet101. For VGG16 backbone, we extract features after every conv layer in the last two building blocks: from conv4_x to conv5_x, and after the last maxpooling layer. For ResNet backbones, we extract features at the end of each bottleneck before ReLU activation: from conv3_x to conv5_x. This feature extracting scheme results in 3 pyramidal layers \( (P = 3) \) for each backbone. We set spatial sizes of both support and query images to 400 x 400, i.e., \( H, W = 400 \), thus having \( H_1, W_1 = 50, H_2, W_2 = 25 \), and \( H_3, W_3 = 13 \). The network is implemented in PyTorch [51] and optimized using Adam [24] with learning rate of 1e-3. We freeze the pre-trained backbone networks to prevent them from learning class-specific representations of the training data.

Datasets. We evaluate the proposed network on three standard few-shot segmentation datasets: PASCAL-5\(^i\) [61], COCO-20\(^i\) [35], and FSS-1000 [33]. PASCAL-5\(^i\) is created from PASCAL VOC 2012 [11] with extra mask annotations [16], consisting of 20 object classes that are evenly divided into 4 folds: \{5\(^i\) : i \in \{0, 1, 2, 3\}\}. COCO-20\(^i\) consists of mask-annotated images from 80 object classes divided into 4 folds: \{20\(^i\) : i \in \{0, 1, 2, 3\}\}. Following common training/evaluation scheme [37, 46, 70, 74, 80], we conduct cross-validation over all the folds; for each fold \( i \), samples from the other remaining folds are used for training and 1,000 episodes from the target fold \( i \) are randomly sampled for evaluation. For every fold, we use the same

\[
\begin{align*}
\mathbb{P}(x, x') &= \mathbb{P}(x) \times \mathbb{P}(x') \\
\mathbb{P}_{CP}(x, x') &= \{(p, p') \in \mathbb{P}(x, x') : p = x \vee p' = x'\}
\end{align*}
\]
Table 1: Performance on PASCAL-5′ [61] in mIoU and FB-IoU. Some results are from [4, 37, 70, 74, 80]. Superscript † denotes our model without support feature masking (Eqn. 1). Numbers in bold indicate the best performance and underlined ones are the second best.

model with the same hyperparameter setup following the standard cross-validation protocol. FSS-1000 contains mask-annotated images from 1,000 classes divided into training, validation and test splits having 520, 240, and 240 classes respectively.

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

5.1. Results and analysis

We evaluate the proposed model on PASCAL-5′, COCO-20′, and FSS-1000 and compare the results with recent methods [4, 37, 46, 54, 61, 63, 70, 74, 75, 86]. Table 1 summarizes 1-shot and 5-shot results on PASCAL-5′; all of our models with three different backbones clearly set new state of the arts with the smallest number of the learnable parameters. With ResNet101 backbone, our 1-shot and 5-shot results respectively achieve 6.1%p and 4.8%p of mIoU improvements over [70] and [4], verifying its superiority in few-shot segmentation task. As shown in Tab. 2, our model outperforms recent methods with a sizable margin on COCO-20′ as well, achieving 2.7%p (1-shot) and 6.8%p (5-shot) of mIoU improvements over [70] with ResNet101 backbone. Also on the last benchmark, FSS-1000, our method sets a new state of the art, outperforming [2, 74] as shown in Tab. 3.

We conduct additional experiments without support feature masking (Eqn. 1). Note that this setup is similar to co-segmentation problem [8, 68, 82] with stronger demands for generalizability since the model is evaluated on novel

| Backbone network | Methods | 50 | 50 | 1-shot mean | FB-IoU | 50 | 50 | 5-shot mean | FB-IoU | # learnable params |
|-------------------|---------|----|----|-------------|--------|----|----|-------------|--------|------------------|
| VGG16 [64]        | PANet [75] | 44.0 | 57.5 | 50.8 | 44.0 | 91.1 | - | 45.3 | 67.2 | 61.3 | 53.2 | 79.9 | 23.5 | 67.2 | 61.3 | 53.2 | 79.9 | - | 23.5M | 67.2 | 61.3 | 53.2 | 79.9 |
| ResNet50 [17]     | PNN [56] | 56.0 | 69.6 | 50.6 | 50.4 | 65.0 | 69.9 | - | 57.7 | 68.7 | 52.9 | 54.6 | 58.5 | 70.5 | - | 17.2M | 59.0 | 68.7 | 52.9 | 54.6 | 58.5 |
|                   | PPNet [37] | 48.6 | 60.6 | 55.7 | 46.5 | 52.8 | 69.2 | - | 58.9 | 68.3 | 56.8 | 50.8 | 63.0 | 75.8 | - | 31.5M | 54.7 | 68.3 | 56.8 | 50.8 | 63.0 |
|                   | 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 | 69.4 | 70.7 | 55.8 | 57.9 | 61.9 |
|                   | RePRI [1] | 59.8 | 68.3 | 62.1 | 48.5 | 59.7 | - | 64.6 | 71.4 | 71.1 | 59.3 | 66.6 | - | - | - | - | - | - | - | - | - | - |
| 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 | 59.6 | 65.7 | 59.6 | 54.0 | 59.7 | 73.4 |

| Backbone network | Methods | 20′ | 20′ | 20′ | mean | FB-IoU | 20′ | 20′ | 20′ | mean | FB-IoU |
|-------------------|---------|----|----|----|----|--------|----|----|----|----|--------|
| ResNet50 [17]     | PPNet [37] | 58.6 | 59.7 | 51.6 | 54.2 | 69.9 | - | 57.9 | 60.9 | 54.1 | 60.5 | 72.3 |
|                   | PFENet [70] | 60.6 | 69.4 | 54.4 | 55.9 | 60.1 | 72.9 | - | 62.8 | 70.4 | 54.9 | 57.6 | 61.4 | 73.5 |
|                   | RePRI [1] | 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 | 68.3 | 70.4 | 80.6 | 2.6M |
| ResNet101 [17]    | FBB [46] | 51.3 | 64.5 | 56.7 | 52.2 | 56.2 | - | 54.8 | 67.4 | 62.2 | 55.3 | 79.9 | - | - | - | - | - | - | - |
|                   | 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 | 69.4 | 70.0 | 69.4 | 60.7 | 65.1 |
|                   | DAN [74] | 54.7 | 68.6 | 57.8 | 51.6 | 56.1 | 72.7 | - | 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 | - | - | - | - | - | - |
|                   | RePRI [1] | 59.6 | 68.6 | 62.2 | 47.2 | 59.4 | - | 66.2 | 71.4 | 67.0 | 57.7 | 65.6 | - | - | - | - | - | - | - |
| 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 | 68.9 | 71.9 | 56.3 | 57.9 | 63.7 | 73.8 |
| 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 | 68.9 | 71.9 | 56.3 | 57.9 | 63.7 | 73.8 |

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

| Backbone network | Methods | mIoU 1-shot |
|-------------------|---------|-------------|
| ResNet50 [17]     | 82.3 | 85.5 |
| ResNet101 [17]    | 85.5 | 88.5 |

Note that this setup is similar to co-segmentation problem [8, 68, 82] with stronger demands for generalizability since the model is evaluated on novel

**Evaluation metrics.** We adopt mean intersection over union (mIoU) and foreground-background IoU (FB-IoU) as our evaluation metrics. The mIoU metric averages over IoU values of all classes in a fold: mIoU = \[ \frac{1}{C} \sum_{c=1}^{C} \text{IoU}_c \] where \( C \) is the number of classes in the target fold and \( \text{IoU}_c \) is the intersection over union of class \( c \). FB-IoU ignores object classes and computes average of foreground and background IoUs: FB-IoU = \[ \frac{1}{2} (\text{IoU}_P + \text{IoU}_B) \] where \( \text{IoU}_P \) and \( \text{IoU}_B \) are respectively foreground and background IoU values in the target fold. As mIoU better reflects model generalization capability and prediction quality than FB-IoU does, we mainly focus on mIoU in our experiments.
Table 4: Domain shift results. Subscripts denote backbone.

| Method            | COCO→PASCAL |   |   |   |
|-------------------|-------------|---|---|---|
|                   | 1-shot      | 5-shot | # params to train | data augmentation used during training |
| PFENet_{Res50} [70] | 61.1        | 63.4 | 10.8M         | flip, rotate, crop |
| RePRU_{ours} [1]  | 63.2        | 67.2 | 46.7M         | flip |
| HSNet_{ours}      | 61.4        | 68.7 | 2.6M          | none |
| HSNet_{ours}      | 64.1        | 70.3 | 2.6M          | none |

5.2. Ablation study

We conduct extensive ablation study to investigate the impacts of major components in our model: hypercorrelations, pyramidal architecture, and center-pivot 4D kernels. We also study how freezing backbone networks prevents overfitting and helps generalization on novel classes. All ablation study experiments are performed with ResNet101 backbone on PASCAL-5′ [61] dataset.

Ablation study on hypercorrelations. To study the effect of intermediate correlations \( \{C_p\}_{p \in \mathcal{L}_p}\) in hypercorrelation \( C_p \in \mathbb{R}^{[L_p] \times |H_p| \times |W_p| \times |H_p| \times |W_p|}\), 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 the hypercorrelations as \( C_p^{\text{shallow}} \) and \( C_p^{\text{deep}} \in \mathbb{R}^{[L_p] \times |H_p| \times |W_p| \times |H_p| \times |W_p|}\), and compare the results with ours \( C_p \) in Fig. 5. The large performance gaps between \( C_p\) 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 \( C_p^{\text{deep}} \) to \( C_p^{\text{shallow}} \) indicates that reliable feature representations typically appear at deeper layers of a CNN.

Figure 5: Ablation study on hypercorrelations on PASCAL-5′ [61] dataset in 1-shot (left) and 5-shot (right) mIoU results.

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

Ablation study on pyramid layers. To see the impact of hypercorrelation \( C_p\) at each layer \( p\), we perform experiments in absence of each pyramidal layer. We train and evaluate our model using two different hypercorrelation pyramids, \( C^{(2:3)} = \{C_2, C_3\}\) and \( C^{(3)} = \{C_3\}\), and compare the results with ours \( C = \{C_p\}_{p=1}^3\). Figure 6 summarizes the results; given hypercorrelation pyramid without geometric information \( C^{(2:3)}\), our model fails to refine object boundaries in the final mask prediction as visualized in Fig. 7. Given a single hypercorrelation that only encodes semantic relations \( C^{(3)}\), the model predictions are severely damaged, providing only rough localization of the target objects. These results indicate that capturing patterns of both semantic and geometric cues is essential for fine-grained localization.

Comparison between three different 4D kernels. We conduct ablation study on 4D kernel by replacing the proposed center-pivot 4D kernel with the original [58] and spatially separable [81] 4D kernels and compare their model size,
we use a stack of three 4D layers for both. (with a group norm and a ReLU activation) in the building blocks. The number of 4D layers in building blocks. Finetuning backbone networks. To investigate the significance of learning ‘feature correlations’ over learning ‘feature representation’ in few-shot regime, we finetune our backbone network and compare learning processes of the finetuned model and ours (frozen backbone). Figure 9 plots the training/validation curves of the finetuned model and ours on every fold of PASCAL-5\(^{i}\). The finetuned model rapidly overfits to the training data, losing generic, comprehensive visual representations learned from large-scale dataset [9]. Meanwhile, our model with frozen backbone provides better generalizability with large trade-offs between training and validation accuracies. The results reveal that learning new appearances under limited supervision requires understanding their ‘relations’ to diverse visual patterns acquired from a vast amount of past experiences, e.g., ImageNet classification. This is quite analogous to human vision perspective in the sense that we generalize novel concepts (what we see) by analyzing their relations to the past observations (what we know) [40].

For additional experimental details, results and analyses, we refer the readers to the Appendix.

6. Conclusion

We have presented a novel framework that analyzes complex feature correlations in a fully-convolutional manner using light-weight 4D convolutions. The significant performance improvements on three standard benchmarks demonstrate that learning patterns of feature relations from multiple visual aspects is effective in fine-grained segmentation under limited supervision. We also demonstrated a unique way of discarding insignificant weights leads to an efficient decomposition of a 4D kernel into a pair of 2D kernels, thus allowing extensive use of 4D conv layers at a significantly small cost. We believe our investigation will further facilitate the use of 4D convolutions in other domains that require learning to analyze high-dimensional correlations.

Acknowledgements. This work was supported by Samsung Advanced Institute of Technology (SAIT), the NRF grant (NRF-2017R1E1A1A01077999), and the IITP grant (No.2019-0-01906, AI Graduate School Program - POSTECH) funded by Ministry of Science and ICT, Korea.

| Kernel type                      | 5\(^{0}\) | 5\(^{1}\) | 5\(^{3}\) | mean | 5\(^{0}\) | 5\(^{1}\) | 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                  | 26.56     |

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.
References

[1] Kelsey Allen, Evan Shelhamer, Hanul Shin, and Joshua Tenenbaum. Infinite mixture prototypes for few-shot learning. In Proc. International Conference on Machine Learning (ICML), 2019. 2

[2] Reza Azad, Abdur R Fayjie, Claude Kauffman, Ismail Ben Ayed, Marco Pedersoli, and Jose Dolz. On the texture bias for few-shot cnn segmentation. In Proc. Winter Conference on Applications of Computer Vision (WACV), 2021. 6

[3] Vassileios Balntas, Karel Lenc, Andrea Vedaldi, and Kristian Mikolajczyk. Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors. In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 5173–5182, 2017. 2

[4] Malik Boudiaf, Hoel Kervadec, Ziko Imtiaz Masud, Pablo Piantanida, Ismail Ben Ayed, and Jose Dolz. Few-shot segmentation without meta-learning: A good transductive inference is all you need? In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2021. 6, 7, 14

[5] Fabio Cermelli, Massimiliano Mancini, Samuel Rota Bulo, Elisa Ricci, and Barbara Caputo. Modeling the background for incremental learning in semantic segmentation. In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020. 2

[6] Yu-Ting Chang, Qiaosong Wang, Wei-Chih Hung, Robinson Pirmuthu, Yi-Hsuan Tsai, and Ming-Hsuan Yang. Weakly-supervised semantic segmentation via sub-category exploration. In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020. 1

[7] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In Proc. European Conference on Computer Vision (ECCV), 2018. 1, 2

[8] Yun-Chun Chen, Yan-Yu Lin, Ming-Hsuan Yang, and Jia-Bin Huang. Show, match and segment: Joint weakly supervised learning of semantic matching and object co-segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2020. 6

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 248–255, 2009. 1, 2, 5, 8, 13

[10] Nanqing Dong and Eric P. Xing. Few-shot semantic segmentation with prototype learning. In Proc. British Machine Vision Conference (BMVC), 2018. 1, 2

[11] Mark Everingham, S. M. Ali Eslami, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. International Journal of Computer Vision (IJCV), 111(1):98–136, Jan 2015. 1, 5, 14

[12] Zhibo Fan, Jin-Gang Yu, Zhihao Liang, Jiarong Ou, Changxin Gao, Gui-Song Xia, and Yuanqing Li. Fgn: Fully guided network for few-shot instance segmentation. In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020. 1

[13] Siddhartha Gairola, Mayur Hemani, Ayush Chopra, and Balaji Krishnamurthy. Simpropnet: Improved similarity propagation for few-shot image segmentation. In Proc. International Joint Conference on Artificial Intelligence (IJCAI), 2020. 1

[14] Bumsun Ham, Minsu Cho, Cordelia Schmid, and Jean Ponce. Proposal flow. In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. 2

[15] Bumsun Ham, Minsu Cho, Cordelia Schmid, and Jean Ponce. Proposal flow: Semantic correspondences from object proposals. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2018. 2

[16] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Simultaneous detection and segmentation. In Proc. European Conference on Computer Vision (ECCV), 2014. 5

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, 2016. 1, 5, 6, 13, 14, 22, 23

[18] Ruibing Hou, Hong Chang, MA Bingpeng, Shiguang Shan, and Xinlin Chen. Cross attention network for few-shot classification. In Advances in Neural Information Processing Systems (NeurIPS), 2019. 2

[19] Tao Hu, Pengwan Yang, Chiliang Zhang, Gang Yu, Yadong Mu, and Cees G. M. Snoek. Attention-based multi-context guiding for few-shot semantic segmentation. In Proc. AAAI Conference on Artificial Intelligence (AAAI), 2019. 1

[20] Gao Huang*, Zhuang Liu*, Laurens van der Maaten, and Kilian Weinberger. Densely connected convolutional networks. In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017. 1

[21] Hao Huang, Jianchun Chen, Xiang Li, Lingjing Wang, and Yi Fang. Robust image matching by dynamic feature selection. In Proc. British Machine Vision Conference (BMVC), 2020. 2

[22] Shuaiyi Huang, Qiuyue Wang, Songyang Zhang, Shipeng Yan, and Xuming He. Dynamic context correspondence network for semantic alignment. In Proc. IEEE International Conference on Computer Vision (ICCV), 2019. 1, 2, 4

[23] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167. 4

[24] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In International Conference on Learning Representations (ICLR), 2015. 5, 13

[25] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In Proc. International Conference on Machine Learning (ICML), 2015. 2

[26] Jing Yu Koh, Duc Thanh Nguyen, Quang-Trung Truong, Sai-Kit Yeung, and Alexander Binder. Sideinfnet: A deep neural network for semi-automatic semantic segmentation with side information. In Proc. European Conference on Computer Vision (ECCV), 2020. 1

[27] Junghyup Lee, Dohyung Kim, Jean Ponce, and Bumsun Ham. Sfnet: Learning object-aware semantic correspondence. In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019. 2
[28] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. Fast online object tracking and segmentation: A unifying approach. In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[29] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. Siamrpn++: Evolution of siamese visual tracking with very deep networks. In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[30] Shuda Li, Kai Han, Theo W. Costain, Henry Howard-Jenkins, and Victor Prisacariu. Correspondence networks with adaptive neighbourhood consensus. In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[31] Wenbin Li, Lei Wang, Jinglin Xu, Jing Huo, Yang Gao, and Jiebo Luo. Revisiting local descriptor based image-to-class measure for few-shot learning. In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[32] Xinghui Li, Kai Han, Shuda Li, and Victor Prisacariu. Dual-resolution correspondence networks. In Advances in Neural Information Processing Systems (NeurIPS), 2020.

[33] Xiang Li, Tianhan Wei, Yau Pun Chen, Yu-Wing Tai, and Chi-Keung Tang. Fss-1000: A 1000-class dataset for few-shot segmentation. In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[34] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2117–2125, 2017.

[35] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context. arXiv preprint arXiv:1405.0312, 2015.

[36] Weide Liu, Chi Zhang, Guosheng Lin, and Fayao Liu. Cmnet: Cross-reference networks for few-shot segmentation. In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[37] Yongfei Liu, Xiangyi Zhang, Songyang Zhang, and Xuming He. Part-aware prototype network for few-shot semantic segmentation. In Proc. European Conference on Computer Vision (ECCV), 2020.

[38] Yanbin Liu, Linchao Zhu, Makoto Yamada, and Yi Yang. Semantic correspondence as an optimal transport problem. In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[39] Wenfeng Luo and Meng Yang. Semi-supervised semantic segmentation via strong-weak dual-branch network. In Proc. European Conference on Computer Vision (ECCV), 2020.

[40] Ellen M. Markman. Categorization and naming in children: Problems of induction. MIT Press, 1989.

[41] Juhong Min and Minsu Cho. Convolutional hough matching networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 2940–2950, June 2021.

[42] Juhong Min, Jongmin Lee, Jean Ponce, and Minsu Cho. Hyperpixel flow: Semantic correspondence with multi-layer neural features. In Proc. IEEE International Conference on Computer Vision (ICCV), 2019.

[43] Juhong Min, Jongmin Lee, Jean Ponce, and Minsu Cho. SPair-71k: A large-scale benchmark for semantic correspondence. arXiv preprint arXiv:1908.10543, 2019.

[44] Juhong Min, Jongmin Lee, Jean Ponce, and Minsu Cho. Learning to compose hypercolumns for visual correspondence. In Proc. European Conference on Computer Vision (ECCV), 2020.

[45] Hyeonsoeb Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[46] Khoi Nguyen and Sinisa Todorovic. Feature weighting and boosting for few-shot segmentation. In Proc. IEEE International Conference on Computer Vision (ICCV), 2019.

[47] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In Proc. IEEE International Conference on Computer Vision (ICCV), 2015.

[48] David Novotny, Diane Larlus, and Andrea Vedaldi. Anchornet: A weakly supervised network to learn geometry-sensitive features for semantic matching. In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[49] Thomas Brox Olaf Ronneberger, Philipp Fischer. U-net: Convolutional networks for biomedical image segmentation. Proc. Medical Image Computing and Computer-Assisted Intervention (MICCAI), 2015.

[50] Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. In Advances in Neural Information Processing Systems (NeurIPS), 2018.

[51] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In Advances in Neural Information Processing Systems (NeurIPS), 2019.

[52] Chao Peng, Xiangyu Zhang, Gang Yu, Guiming Luo, and Jian Sun. Large kernel matters – improve semantic segmentation by global convolutional network. In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[53] Limeng Qiao, Yemin Shi, Jia Li, Yaowei Wang, Tiejun Huang, and Sergey Levine. Few-shot segmentation propagation with guided networks. arXiv preprint arXiv:1806.07373, 2018.
tions. In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[85] Chi Zhang, Yujun Cai, Guosheng Lin, and Chunhua Shen. Deeperemd: Few-shot image classification with differentiable earth mover’s distance and structured classifiers. In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[86] Chi Zhang, Guosheng Lin, Fayao Liu, Jiushuang Guo, Qingyao Wu, and Rui Yao. Pyramid graph networks with connection attentions for region-based one-shot semantic segmentation. In Proc. IEEE International Conference on Computer Vision (ICCV), 2019.

[87] Chi Zhang, Guosheng Lin, Fayao Liu, Rui Yao, and Chunhua Shen. Canet: Class-agnostic segmentation networks with iterative refinement and attentive few-shot learning. In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[88] Tianyi Zhang, Guosheng Lin, Weide Liu, Jianfei Cai, and Alex Kot. Splitting vs. merging: Mining object regions with discrepancy and intersection loss for weakly supervised semantic segmentation. In Proc. European Conference on Computer Vision (ECCV), 2020.

[89] Xiaolin Zhang, Yunchao Wei, Yi Yang, and Thomas S Huang. Sg-one: Similarity guidance network for one-shot semantic segmentation. IEEE Transactions on Cybernetics, 2020.
Appendix A. Complete derivation of the center-pivot 4D convolution

In this section, we extend Sec. 4.4 to provide a complete derivation of the center-pivot 4D convolution. Note that a typical 4D convolution parameterized by a kernel \( k \in \mathbb{R}^{k \times k \times k \times k} \) on a correlation tensor \( c \in \mathbb{R}^{H \times W \times H \times W} \) at position \((x, x') \in \mathbb{R}^{4}\) is formulated as

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

where \( \mathcal{P}(x, x') \) denotes a set of neighbourhood regions within the local 4D window centered on position \((x, x')\), i.e., \( \mathcal{P}(x, x') = \mathcal{P}(x) \times \mathcal{P}(x') \) as visualized in Fig. 4. Now we design a light-weight, efficient 4D convolution via a reasonable weight-sparsification: from a set of neighborhood positions within a local 4D window of interest, our kernel aims to disregard a large number of activations located at fairly insignificant positions in the 4D window, thereby focusing only on a small subset of relevant activations for capturing complex patterns in the correlation tensor. Specifically, we consider activations at positions that pivots either one of 2-dimensional centers, e.g., \( x \) or \( x' \), as the foremost influential ones. Given 4D position \((x, x')\), we collect its neighbors if and only if they are adjacent to either \( x \) or \( x' \) in its corresponding 2D subspace and define two respective sets as

\[
\mathcal{P}_{c}(x, x') = \{(p, p') \in \mathcal{P}(x, x'): p = x\},
\]

and

\[
\mathcal{P}_{c'}(x, x') = \{(p, p') \in \mathcal{P}(x, x'): p' = x\}.
\]

The set of center-pivot neighbors \( \mathcal{P}_{CP}(x, x') \) is defined as a union of the two subsets:

\[
\mathcal{P}_{CP}(x, x') = \mathcal{P}_{c}(x, x') \cup \mathcal{P}_{c'}(x, x').
\]

Based on this small subset of neighbors, center-pivot 4D (CP 4D) convolution can be formulated as a union of two separate 4D convolutions:

\[
(c * k_{CP})(x, x') = (c * k_{c})(x, x') + (c * k_{c'})(x, x'),
\]

where \( k_{c} \) and \( k_{c'} \) are 4D kernels with their respective neighbors \( \mathcal{P}_{c}(x, x') \) and \( \mathcal{P}_{c'}(x, x') \). Now consider below

\[
(c * k_{c})(x, x') = \sum_{(p, p') \in \mathcal{P}_{c}(x, x')} c(p, p')k(p - x, p' - x') = \sum_{p' \in \mathcal{P}(x')} c(x, p')k(x - x, p' - x') = \sum_{p' \in \mathcal{P}(x')} c(x, p')k_{2D}(p' - x') = \sum_{p' \in \mathcal{P}(x')} c(x, p')k_{2D}(p' - x'),
\]

which is equivalent to convolution with kernel \( k_{c}^{2D} = k(0, :, \cdot) \in \mathbb{R}^{k \times k} \) performed on 2D slice of the 4D tensor \( c(x, :) \). Similarly,

\[
(c * k_{c'})(x, x') = \sum_{(p, p') \in \mathcal{P}_{c'}(x, x')} c(p, p')k(p - x, p' - x') = \sum_{p \in \mathcal{P}(x)} c(p, x')k(p - x, x' - x) = \sum_{p \in \mathcal{P}(x)} c(p, x')k(p - x, 0) = \sum_{p \in \mathcal{P}(x)} c(p, x')k_{2D}^{c'}(p - x),
\]

where \( k_{2D}^{c'} = k(:, 0) \in \mathbb{R}^{k \times k} \). Based on above derivations, we rewrite Eqn. 10 as follows

\[
(c * k_{CP})(x, x') = \sum_{p' \in \mathcal{P}(x')} c(x, p')k_{2D}^{c}(p' - x') + \sum_{p \in \mathcal{P}(x)} c(p, x')k_{2D}^{c'}(p - x),
\]

which performs two different convolutions on separate 2D subspaces, having a linear complexity.

Appendix B. Implementation details

For the backbone networks, we employ VGG [64] and ResNet [17] families pre-trained on ImageNet [9], e.g., VGG16, ResNet50, and ResNet101. For the VGG16 backbone, we extract features after every conv layer in the last two building blocks: from \texttt{conv4} to \texttt{conv5}, and after the last maxpooling layer. For the ResNet backbones, we extract features at the end of each bottleneck before ReLU activation: from \texttt{conv3} to \texttt{conv5}. This feature extracting scheme results in 3 pyramidal layers (\( P = 3 \)) for every backbone. We set spatial sizes of both support and query images to \( 400 \times 400 \), i.e., \( H, W = 400 \), thus having \( H_{1}, W_{1} = 50, H_{2}, W_{2} = 25, \) and \( H_{3}, W_{3} = 13 \) for both ResNet50 and ResNet101 backbones and \( H_{1}, W_{1} = 50, H_{2}, W_{2} = 25, \) and \( H_{3}, W_{3} = 12 \) for the VGG16 backbone. The network is implemented in PyTorch [51] and optimized using Adam [24] with learning rate of 1e-3. We train our model with batch size of 20, 40, and 20 for PASCAL-5$, COCO-20$, and FSS-1000 respectively. We freeze the pre-trained backbone networks to prevent them from learning class-specific representations of the training data. The intermediate tensor dimensions, the number of parameters of each layers and other additional details of the network are demonstrated in Tab. A5, A6, and A7 for respective backbones of VGG16, ResNet50, and ResNet101.
### Appendix C. Additional results and analyses

**Additional K-shot results.** Following the work of [4, 70, 80], we conduct K-shot experiments with K ∈ {1, 5, 10}. Table A2 compares our results with the recent methods [4, 70, 80] on PASCAL-5† and COCO-20†. The significant performance improvements on both datasets clearly indicate the effectiveness of our approach. Achieving 2.5%p and 4.6%p mIoU improvements over the previous best method [4] on respective PASCAL-5† and COCO-20†, our model again sets a new state of the art in 10-shot setting as well, showing notable improvements with larger K.

**Numerical comparisons of ablation study.** We tabularize Figures 5 and 6, e.g., ablation study on hypercorrelations and pyramidal layers, in Tables A3 and A4 respectively. Achieving 4.5%p mIoU improvements over C.pmean, our method clearly benefits from diverse feature correlations from multi-level CNN layers (C.p) as seen in Tab. A3. A large performance gap between C.(2:3) and C.(3) in Tab. A4 (63.9 vs. 55.5) reveals that the intermediary second pyramidal layer (p = 2) is especially effective in robust mask prediction compared to the first pyramidal layer (p = 1).

**Evaluation results without using ignore_label on PASCAL-5†.** The benchmarks of PASCAL-5† [61], COCO-20† [35], and FSS-1000 [33] consist of segmentation mask annotations in which each pixel is labeled with either background or one of the predefined object categories. As pixel-wise segmentation near object boundaries is ambiguous to perform even for human annotators, PASCAL-5† uses a special kind of label called ignore_label which marks pixel regions ignored during training and evaluation to mitigate the ambiguity†.

Most recent few-shot segmentation work [4, 37, 46, 54, 61, 63, 70, 74, 75, 86] adopt this evaluation criteria but we found that some methods [36, 80, 87, 89] do not utilize ignore_label in their evaluations. Therefore, the methods are unfairly evaluated as fine-grained mask prediction near object boundaries is one of the most challenging part in segmentation problem. For fair comparisons, we intentionally exclude the methods of [36, 80, 87, 89] from Tab. 1 and compare the results of our model evaluated without the use of ignore_label with those methods [36, 80, 87, 89]. The results are summarized in Tab. A1. Even without using ignore_label, the proposed method sets a new state of the art with ResNet50 backbone, outperforming the previous best methods of [80] and [36] by (1-shot) 2.8%p and (5-shot) 5.4%p respectively. With VGG16 backbone, our method performs comparably effective to the previous best method [4] while having the smallest learnable parameters.

---

Table A1: Evaluation results on PASCAL-5† [61] benchmark in mIoU and FB-IoU evaluation metrics without the use of ignore_label. The results of other methods are from [36, 70, 74, 80].

| Backbone network | Methods | PASCAL-5† | COCO-20† |
|------------------|---------|-----------|-----------|
|                  | 1-shot | 5-shot | 10-shot | 1-shot | 5-shot | 10-shot |
| VGG16 [64] | SG-One [89] | 40.2 | 58.4 | 48.4 | 38.4 | 46.3 | 63.1 | 41.9 | 58.6 | 48.6 | 39.4 | 47.1 | 65.9 | 19.0M |
|                  | CRNet [36] | - | - | - | 55.2 | 66.4 | | - | - | - | 58.5 | 71.0 | - | 55.7 | 66.8 |
|                  | HSNet (ours) | 53.6 | 61.7 | 55.0 | 50.3 | 55.2 | 70.5 | 58.3 | 64.7 | 58.9 | 54.6 | 59.1 | 73.5 | 2.6M |
| ResNet50 [17] | CANet [87] | 52.5 | 65.9 | 51.3 | 51.9 | 55.4 | 66.2 | 55.5 | 67.8 | 51.9 | 53.2 | 57.1 | 69.6 | 19.0M |
|                  | RPMM [80] | 55.2 | 66.9 | 52.6 | 50.7 | 56.3 | - | 56.3 | 67.3 | 54.5 | 51.0 | 57.3 | - | 19.7M |
|                  | CRNet [36] | - | - | - | 55.7 | 66.8 | | - | - | - | 58.8 | 71.5 | - | - |
|                  | HSNet (ours) | 57.4 | 66.8 | 55.8 | 56.5 | 59.1 | 73.8 | 62.6 | 69.2 | 62.5 | 62.4 | 64.2 | 77.4 | 2.6M |
| ResNet101 [17] | HSNet (ours) | 60.1 | 67.8 | 57.3 | 59.0 | 61.1 | 74.4 | 63.9 | 69.9 | 62.0 | 63.6 | 64.8 | 77.1 | 2.6M |

Table A2: Results on K-shot with ResNet50 backbone network where K ∈ {1, 5, 10}. The results of other methods are from [4].

| Method | PASCAL-5† | COCO-20† |
|--------|-----------|-----------|
|        | 1-shot | 5-shot | 10-shot | 1-shot | 5-shot | 10-shot |
| RPMM [80] | 56.3 | 57.3 | 57.6 | 30.6 | 35.5 | 33.1 |
| PFENet [70] | 60.8 | 61.9 | 62.1 | 35.8 | 39.0 | 39.7 |
| RepRI [4] | 59.7 | 66.6 | 68.1 | 34.1 | 41.6 | 44.1 |
| HSNet (ours) | 64.0 | 69.5 | 70.6 | 39.2 | 46.9 | 48.7 |

Table A3: Numerical results of Figure 5. All experiments are performed with ResNet101 backbone [17].

| Method | PASCAL-5† | COCO-20† |
|--------|-----------|-----------|
|        | 1-shot | 5-shot | 10-shot | 1-shot | 5-shot | 10-shot |
| C.pmean | 57.1 | 64.7 | 57.5 | 57.0 | 59.1 | 63.7 | 69.8 | 66.0 | 65.4 | 65.7 |
| C.p | 60.6 | 68.6 | 58.2 | 59.2 | 61.2 | 66.7 | 72.0 | 65.9 | 65.4 | 67.5 |
| C.(ours) | 67.3 | 72.3 | 62.0 | 63.1 | 66.2 | 71.8 | 74.4 | 67.0 | 68.3 | 70.4 |

Table A4: Numerical results of Figure 6. All experiments are performed with ResNet101 backbone [17].

†The use of ignore_label was originally adopted in PASCAL VOC dataset [11]. The same evaluation criteria is naturally transferred to PASCAL-5† [61] as it is created from PASCAL VOC.
Appendix D. Qualitative results

Results without support feature masking. As demonstrated in Sec. 5.1, we conduct experiments without support feature masking (Eqn. 1), similarly to co-segmentation problem with stronger demands for generalizibility. Figure A1 visualizes some example results on PASCAL-5\textsuperscript{i} dataset. Even without the use of support masks (in both training and testing), our model effectively segments target instances in query images. The results indicate that learning patterns of feature correlations from multiple visual aspects is effective in fine-grained segmentation as well as identifying ‘common’ instances in the support and query images.

Additional qualitative results. We present additional qualitative results on PASCAL-5\textsuperscript{i} [61], COCO-20\textsuperscript{i} [46], and FSS-1000 [33] benchmark datasets. All the qualitative results are best viewed in electronic forms. Example results in presence of large scale-differences, truncations, and occlusions are shown in Fig. A2, A3, and A4. Figure A5 visualizes model predictions under large illumination-changes in support and query images. Figure A6 visualizes some sample predictions given exceptionally small objects in either support or query images. As seen in Fig. A7, we found that our model sometimes predicts more reliable segmentation masks than ground-truth ones. Some qualitative results in presence of large intra-class variations and noisy clutters in background are shown in Fig. A8 and A9. Given only a single support image-annotation pair, our model effectively segments multiple instances in a query image as visualized in Fig. A10. Figure A11 shows representative failure cases; our model fails to localize target objects in presence of severe occlusions, intra-class variances and extremely tiny support (or query) objects. As seen in Fig. A12, the model predictions become much reliable given multiple support image-mask pairs, i.e., $K > 1$.

The code and data to reproduce all experiments in this paper is available at our project page: http://cvlab.postech.ac.kr/research/HSNet/.

Figure A1: Example results without support feature masking (Eqn. 1) on PASCAL-5\textsuperscript{i} dataset.
Figure A2: Qualitative (1-shot) results on PASCAL-5\textsuperscript{i} [61] dataset under large differences in object scales.

Figure A3: Qualitative (1-shot) results on PASCAL-5\textsuperscript{i} [61] and COCO-20\textsuperscript{i} [35] datasets under large truncations and occlusions.
Figure A4: Qualitative (1-shot) results on COCO-20$^i$ [35] dataset under large differences in object scales.

Figure A5: Qualitative (1-shot) results on COCO-20$^i$ [35] dataset under large illumination-changes in support and query images.

Figure A6: Qualitative (1-shot) results on COCO-20$^i$ [35] dataset with exceptionally small objects.

Figure A7: Our network occasionally predicts more accurate segmentation masks than human-annotated ground-truths.
Figure A8: Qualitative (1-shot) results on PASCAL-51 [61] and COCO-200 [35] datasets under large intra-class variations.

Figure A9: Qualitative (1-shot) results on PASCAL-51 [61] and COCO-200 [35] datasets in presence of noisy background clutters.
Figure A10: One-to-many and many-to-many (1-shot) results on PASCAL-5 \[^{[61]}\], COCO-20 \[^{[35]}\], and FSS-1000 \[^{[33]}\] datasets.

Figure A11: Representative failure cases in presence of severe occlusions, intra-class variances and extremely tiny objects.
Figure A12: Comparison between 1-shot and 5-shot results on PASCAL-5i dataset [61]. Multiple support images and mask annotations clearly help our model generate accurate mask predictions on query images in many challenging cases.
| Layer                | Input               | Output               | Operation                              | # params. |
|---------------------|---------------------|----------------------|----------------------------------------|-----------|
| VGG16 Backbone      | \(I^q\) (3, 400, 400) | \(\{F_q^i\}_{i=1}^7\) (512, 12, 12) x 1 | Series of 2D CONVS                  | 14.7M (frozen) |
|                     | \(I^a\) (3, 400, 400) | \(\{F_a^i\}_{i=1}^7\) (512, 25, 25) x 3 |                          |           |
| Masking Layer       | \(\{F_q^i\}_{i=1}^7\) (512, 12, 12) x 1 | \(\{F_a^i\}_{i=1}^7\) (512, 25, 25) x 3 | BILINEAR INTERPOLATION           |           |
|                     | \(\{F_q^i\}_{i=1}^7\) (512, 25, 25) x 3 | \(\{F_a^i\}_{i=1}^7\) (512, 50, 50) x 3 | HADAMARD PRODUCT                  |           |
| Correlation Layer   | \(\{F_q^i\}_{i=1}^7\) (512, 12, 12) x 1 | \(\{F_a^i\}_{i=1}^7\) (512, 25, 25) x 3 | COSINE SIMILARITY                |           |
|                     | \(\{F_q^i\}_{i=1}^7\) (512, 25, 25) x 3 | \(\{F_a^i\}_{i=1}^7\) (512, 50, 50) x 3 |                          |           |
| Squeezing Block \(f_{1}^{sqz}\) | \(C_3\) (1, 12, 12, 12) | \(C_{sqz}^3\) (128, 12, 12, 2, 2) | CP 4D conv GROUP NORM RELU       | 167K      |
| Squeezing Block \(f_{2}^{sqz}\) | \(C_2\) (3, 25, 25, 25) | \(C_{sqz}^2\) (128, 25, 25, 2, 2) | CP 4D conv GROUP NORM RELU       | 169K      |
| Squeezing Block \(f_{1}^{sqz}\) | \(C_1\) (3, 50, 50, 50) | \(C_{sqz}^1\) (128, 50, 50, 2, 2) | CP 4D conv GROUP NORM RELU       | 202K      |
| Mixing Block \(f_{2}^{mix}\) | \(C_{sqz}^3\) (128, 12, 12, 2, 2) | \(C_{mix}^2\) (128, 25, 25, 2, 2) | BILINEAR INTERPOLATION ELEMENT-WISE ADDITION | 886K      |
|                     | \(C_{sqz}^2\) (128, 25, 25, 2, 2) | \(C_{mix}^2\) (128, 25, 25, 2, 2) | CP 4D conv GROUP NORM RELU       |           |
| Mixing Block \(f_{1}^{mix}\) | \(C_{mix}^1\) (128, 50, 50, 2, 2) | \(C_{sqz}^1\) (128, 50, 50, 2, 2) | BILINEAR INTERPOLATION ELEMENT-WISE ADDITION | 886K      |
|                     | \(C_{mix}^1\) (128, 50, 50, 2, 2) | \(C_{mix}^1\) (128, 50, 50, 2, 2) | CP 4D conv GROUP NORM RELU       |           |
| Pooling Layer       | \(C_{mix}^1\) (128, 50, 50, 2, 2) | \(Z\) (128, 50, 50) | AVERAGE-POOLING                 |           |
| Decoder Layer       | \(Z\) (128, 50, 50) | \(\bar{M}^l\) (2, 400, 400) | SERIES OF 2D CONVS WITH BILINEAR INTERPOLATION | 259K      |

Table A5: Hypercorrelation Squeeze Networks (HSNet) with VGG16 [64] backbone network. The reported number of parameters in VGG16 backbone network (14.7M) excludes those in fully-connected layers (unused in our model). The total number of ‘learnable’ parameters amounts to 2.6M. The number of intermediate features extracted from backbone network amounts to 7, i.e., \(L = 7\). The CP 4D conv refers to the proposed center-pivot 4D convolution.
| Layer                  | Input          | Output                                      | Operation                            | # params.  |
|-----------------------|----------------|---------------------------------------------|---------------------------------------|-----------|
| ResNet50 Backbone     | \( I^q \)      | \{\( F^q_l \)\}_{l=1}^{13} \rightarrow \{\( F^q_l \)\}_{l=1}^{13} \rightarrow \) | SERIES OF 2D CONVS | 23.6M (frozen) |
|                       | \( I^s \)      | \{\( F^s_l \)\}_{l=1}^{13} \rightarrow \{\( F^s_l \)\}_{l=1}^{13} \rightarrow \) | SERIES OF 2D CONVS | -          |
| Masking Layer         | \{\( F^s_l \)\}_{l=1}^{13} \rightarrow \{\( F^s_l \)\}_{l=1}^{13} \rightarrow \) | \{\( F^s_l \)\}_{l=1}^{13} \rightarrow \) | COSINE SIMILARITY | -          |
| Correlation Layer     | \{\( F^s_l \)\}_{l=1}^{13} \rightarrow \{\( F^s_l \)\}_{l=1}^{13} \rightarrow \) | \{\( C_p \)\}_{p=1}^{3} \rightarrow \) | -          |
| Squeezing Block \( f^{sqz}_3 \) | \( C_3 \)      | \{\( C^{sqz}_3 \)\} \rightarrow \) | CP 4D CONV GROUP NORM RELU | 168K       |
| Squeezing Block \( f^{sqz}_2 \) | \( C_2 \)      | \{\( C^{sqz}_2 \)\} \rightarrow \) | CP 4D CONV GROUP NORM RELU | 172K       |
| Squeezing Block \( f^{sqz}_1 \) | \( C_1 \)      | \{\( C^{sqz}_1 \)\} \rightarrow \) | CP 4D CONV GROUP NORM RELU | 203K       |
| Mixing Block \( f^{mix}_2 \) | \( C^{mix}_1 \) | \{\( C^{mix}_2 \)\} \rightarrow \) | CP 4D CONV GROUP NORM RELU | 886K       |
| Mixing Block \( f^{mix}_1 \) | \{\( C^{mix}_1 \)\} \rightarrow \) | \{\( C^{mix}_2 \)\} \rightarrow \) | CP 4D CONV GROUP NORM RELU | 886K       |
| Pooling Layer         | \( C^{mix}_1 \) | \{\( Z \)\} \rightarrow \) | -          |
| Decoder Layer         | \{\( Z \)\} \rightarrow \) | \{\( M \)\} \rightarrow \) | SERIES OF 2D CONVS WITH BILINEAR INTERPOLATION | 259K       |

Table A6: Hypercorrelation Squeeze Networks (HSNet) with ResNet50 [17] backbone network. The reported number of parameters in ResNet50 backbone network (23.6M) excludes those in fully-connected layers (unused in our model). The total number of ‘learnable’ parameters amounts to 2.6M. The number of intermediate features extracted from backbone network amounts to 13, i.e., \( L = 13 \).
| Layer                        | Input                     | Output                     | Operation                                      | # params.  |
|-----------------------------|---------------------------|----------------------------|------------------------------------------------|-----------|
| **ResNet101 Backbone**      | *(3, 400, 400)*           | *(2048, 13, 13) × 3*       | SERIES OF 2D CONVS                             | 42.6M (frozen) |
|                             | *(1024, 25, 25) × 23*     | *(512, 50, 50) × 4*        |                                                |           |
| Masking Layer               | *(3, 400, 400)*           | *(2048, 13, 13) × 3*       | BILINEAR INTERPOLATION                        | -         |
|                             | *(1024, 25, 25) × 23*     | *(512, 50, 50) × 4*        | HADAMARD PRODUCT                              | -         |
| Correlation Layer           | *(3, 13, 13, 13) × 23*    | *(3, 13, 13, 13) × 3*      | COSINE SIMILARITY                              | -         |
|                             | *(512, 50, 50) × 4*       | *(23, 25, 25, 25) × 23*   |                                                |           |
| Squeezing Block $f_{3}^{sqz}$ | *(128, 13, 13, 2) × 3*    | *(128, 13, 13, 2) × 3*    | (CP 4D conv) GROUP NORM ReLU                  | 168K      |
| Squeezing Block $f_{2}^{sqz}$ | *(128, 25, 25, 2) × 3*    | *(128, 25, 25, 2) × 3*    | (CP 4D conv) GROUP NORM ReLU                  | 185K      |
| Squeezing Block $f_{1}^{sqz}$ | *(128, 50, 50, 2) × 3*    | *(128, 50, 50, 2) × 3*    | (CP 4D conv) GROUP NORM ReLU                  | 203K      |
| Mixing Block $f_{2}^{mix}$   | *(128, 13, 13, 2) × 3*    | *(128, 25, 25, 2) × 3*    | BILINEAR INTERPOLATION ELEMENT-WISE ADDITION  | 886K      |
|                             | *(128, 25, 25, 2) × 3*    | *(128, 25, 25, 2) × 3*    | (CP 4D conv) GROUP NORM ReLU                  |           |
| Mixing Block $f_{1}^{mix}$   | *(128, 25, 25, 2) × 3*    | *(128, 25, 25, 2) × 3*    | BILINEAR INTERPOLATION ELEMENT-WISE ADDITION  | 886K      |
|                             | *(128, 50, 50, 2) × 3*    | *(128, 50, 50, 2) × 3*    | (CP 4D conv) GROUP NORM ReLU                  |           |
| Pooling Layer               | *(128, 50, 50) × 3*       | *(128, 50, 50) × 3*       | AVERAGE-POOLING                               | -         |
| Decoder Layer               | *(2, 400, 400)            | *(2, 400, 400)             | SERIES OF 2D CONVS WITH BILINEAR INTERPOLATION | 259K      |

**Table A7:** Hypercorrelation Squeeze Networks (HSNet) with ResNet101 [17] backbone network. The reported number of parameters in ResNet101 backbone network (42.6M) excludes those in fully-connected layers (unused in our model). The total number of ‘learnable’ parameters amounts to 2.6M. The number of intermediate features extracted from backbone network amounts to 30, i.e., $L = 30$. 

23