A Self-Distillation Embedded Supervised Affinity Attention Model for Few-Shot Segmentation

Qi Zhao, *Member, IEEE*, Binghao Liu, Shuchang Lyu, *Graduate Student Member, IEEE*, and Huojin Chen

***Abstract*—Few-shot segmentation focuses on the generalization of models to segment unseen object with limited annotated samples. However, existing approaches still face two main challenges. First, a huge feature distinction between support and query images a causes knowledge transferring barrier, which harms the segmentation performance. Second, limited support prototypes cannot adequately represent features of support objects, hard to guide high-quality query segmentation. To deal with the above two issues, we propose a self-distillation embedded supervised affinity attention model to improve the performance of few-shot segmentation task. Specifically, the self-distillation guided prototype module uses self-distillation to align the features of support and query. The supervised affinity attention module generates a high-quality query attention map to provide sufficient object information. Extensive experiments prove that our model significantly improves the performance compared to existing methods. Comprehensive ablation experiments and visualization studies also show the significant effect of our method on the few-shot segmentation task. On COCO-20^i data set, we achieve new state-of-the-art results. Training code and pretrained models are available at https://github.com/cv516Buaa/SD-AANet.**

***Index Terms*—Attention mechanism, few-shot learning, few-shot segmentation, self-distillation.**

I. Introduction

SEMANTIC segmentation, as a significant computer vision task, aims to assign a class label to each pixel in the image. Fully convolutional networks (FCNs) [1] have been the pioneer to handle this task in an end-to-end manner, then various of deep neural network models [2], [3], [4], [5], [6], [7] have made great improvements recently. However, massive pixel-level annotated data required in semantic segmentation leads to expensive annotation cost. In addition, these methods have a dramatically performance decline when meeting unseen classes. In contrast, the human cognitive ability can easily accurately complete complex tasks, such as recognition based on the existing knowledge and a small amount of new labeled data.

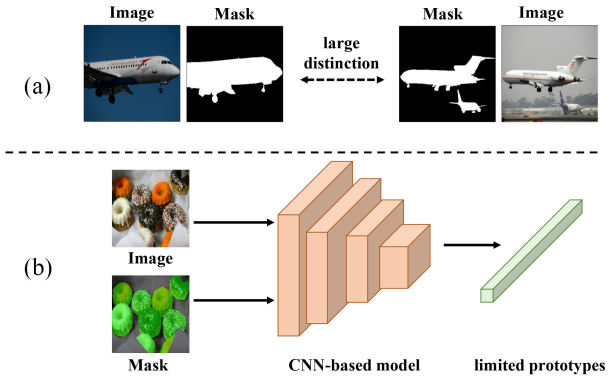


Fig. 1. Two main challenges in few-shot segmentation task. (a) Huge feature distinction between support and query caused by differences on shooting angle, lighting conditions, shape, and color of objects with same category. (b) Limited support prototypes cannot adequately represent features of support objects.

To address the above-mentioned issues, few-shot segmentation [8] is proposed, using limited annotated data to segment unseen classes. Different from fully supervised semantic segmentation, few-shot segmentation splits the whole data into support set and query set. The support set in this task provides meaningful and critical features of a certain class to guide the method extracting target with same class in query set.

Current few-shot segmentation methods are mainly based on metric learning, containing two main technical routes: 1) affinity learning [9], [10], [11] and 2) prototypical learning [12], [13], [14]. Affinity learning acquires feature of support object with the help of support mask. Then, each pixel-wise support feature is matched to query feature through various correlation measure operations, guiding query segmentation.

Prototypical learning methods usually use one or few prototypes to represent support object feature and guide segmentation of query target. These methods adopt masked global average pooling (masked GAP) [12] to obtain a support prototype. Generally, the support prototype is combined with the query feature through correlation metrics to realize high-quality segmentation.

However, there are still two major challenges need to be solved in few-shot segmentation task as illustrated in Fig. 1. First, the appearance of objects in images may have different quantities, perspectives, illumination intensities, etc. The feature distinction between support and query dramatically reduces the segmentation performance. Second, limited

Manuscript received 15 October 2022; revised 14 February 2023; accepted 26 February 2023. Date of publication 2 March 2023; date of current version 12 February 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62072021. *(Corresponding author: Huojin Chen.)*

Qi Zhao, Binghao Liu, and Shuchang Lyu are with the Department of Electronic and Information Engineering, Beihang University, Beijing 100191, China (e-mail: zhaoqi@buaa.edu.cn; liubinghao@buaa.edu.cn; lyushuchang@buaa.edu.cn).

Huojin Chen is with the College of Art and Design, Beijing University of Technology, Beijing 100124, China (e-mail: chenhuojin@bjut.edu.cn).

Digital Object Identifier 10.1109/TCDS.2023.3251371

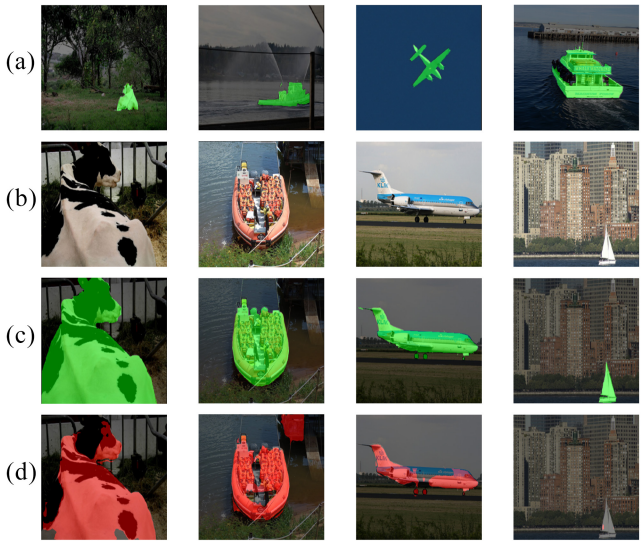


Fig. 2. Some failure cases of current methods. From top to bottom: (a) support image and its ground truth, (b) query image, (c) ground truth of query image, and (d) prediction of current methods.

support prototypes are incapable of providing sufficient representative information. Some typical failure cases of existing methods can be seen in Fig. 2.

To deal with issue caused by the image distinctions, we attempt to creatively introduce the self-distillation method into few-shot segmentation task. We propose a novel module named self-distillation guided prototype generating module (SDPM), which adopts self-distillation approach to bridge the gap between support and query features, finding commonalities between two features. SDPM takes support label, support feature and query feature as inputs, outputting a channel reweighting query feature, and a support prototype with an intrinsic class feature.

In the few-shot segmentation task, one or few support prototypes can not provide sufficient representative information to segment the query target. So we design the supervised affinity attention module (SAAM), a CNN-based end-to-end module which can be simply embedded in deep CNN models and introduces negligible computation cost. SAAM has the same inputs as SDPM, and aims to generate an affinity attention map to give a prior prediction of query target.

Based on the two modules mentioned above, we propose the self-distillation embedded affinity attention network (SD-AANet) to produce intrinsic prototype and affinity attention map efficiently. Extensive experiments show that our SD-AANet achieves state-of-the-art performance on COCO-20^i and comparable state-of-the-art results on Pascal-5^i.

Our contributions are summarized as follows.

1) We propose the SDPM to generate an intrinsic prototype by self-distillation approach, which can efficiently align the features of support and query. Otherwise, our SAAM helps to produce a query attention map to teach decoder where to focus. Through combining SDPM and SAAM, SD-AANet can better address the two challenges mentioned above.
2) SD-AANet achieves new state-of-the-art results on COCO-20^i [15], [16] data sets (mIoU for 1-shot: 40.9%, 5-shot: 48.7%) and comparable state-of-the-art results on PASCAL-5^i [8] (mIoU for 1-shot: 62.9%) for few-shot segmentation task.

II. Related Work

A. Semantic Segmentation

Semantic Segmentation aims to predict a semantic category for each pixel in image. Convolutional neural network (CNN)-based methods have made great progress in the semantic segmentation field. FCN [1] replaces fully connected layers (FC layers) with convolutional layers, achieving semantic segmentation in an end-to-end manner. SegNet [2] and UNet [3] employ symmetric "Encode–Decoder" architectures to map the original image to the same-size predictions. PSPNet [6] integrates the pyramid pooling module (PPM) into several baseline architectures like ResNet [17], [18]) to obtain contextual information from different scales by using different kernel-sized pooling layers. Chen et al. [19], [20] employed dilated convolution to expand the receptive field. In addition, some works focus on the attention mechanism. PSANet [21] proposes a point-wise spatial attention to explore better connection information between pixels. DANet [22] adopts position attention module and channel attention module to learn position and channel interdependencies. CCNet [23] adopts a criss-cross attention module to capture contextual information from full-image dependencies. However, well-performed semantic segmentation networks need a large amount of annotated data as training samples which are expensive to obtain.

B. Few-Shot Learning

Few-shot Learning seeks to recognize new objects with only few annotated samples. In this field, as an interpretable approach, metric learning [24], [25], [26] is widely used. Koch et al. [24] proposed a siamese architecture which shows great performance on K-shot image classification tasks. This architecture can also be extended to deal with K-shot semantic segmentation. Meta learning [27] enables machine to quickly acquire useful prior information from limited labeled samples. Meta-learning LSTM [28] and Model-Agnostic [29] methods apply recurrent neural network (RNN) to represent and store the prior information to handle the few-shot problem. To own the advantage of both two methods, ProtoMAML [30] combines the complementary strengths of metric-learning and gradient-based meta-learning methods.

C. Few-Shot Segmentation

Few-shot semantic segmentation aims at performing dense pixel-wise classification for unseen classes. Shaban et al. [8] are the pioneers to officially define the few-shot semantic segmentation problem. They propose a two-branch architecture (OSLSM) to produce a binary mask for the new semantic class with a dot-similarity manner. SG-One [12], which is now a benchmark architecture in the one-shot segmentation task, proposes an architecture that consists of a guidance branch and a segmentation branch. Based on two branchs design of SG-One [12], [9], [13], [14], [16], [31], [32], [33] further promote

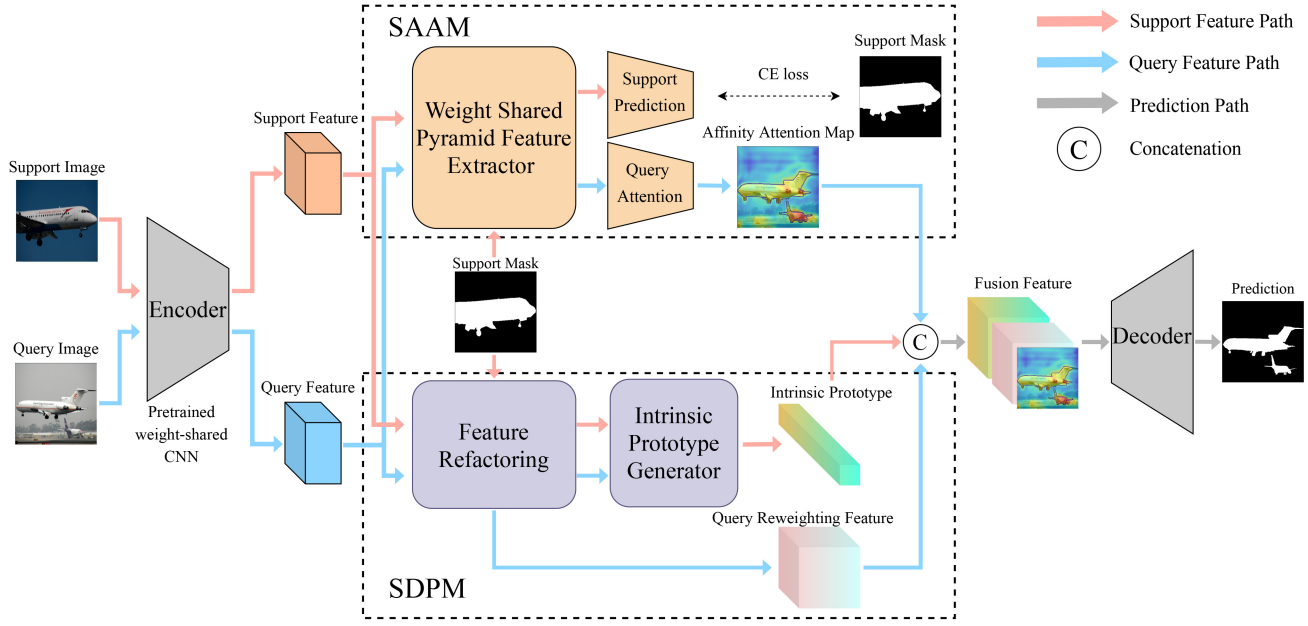


Fig. 3. Architecture of SD-AANet. In SD-AANet, middle-level features of CNN backbone and support mask are input to SDPM and SAAM. SDPM uses support prototype to realize channel reweighting on support and query features. Then self-distillation approach in SDPM produces an intrinsic support prototype. SAAM introduces support ground truth supervision to a learnable pyramid feature extractor, producing high-quality query attention map. Finally, the fusion of intrinsic prototype, query feature, and query attention goes through the decoder to predict the final result.

the few-shot segmentation performance. PFENet [34] proposes a training-free prior generation process to produce prior segmentation attention for the model, and a feature enrichment module to enrich query features with the support features. ASR [35] reformulates few-shot segmentation as a semantic reconstruction problem and converts base class features into a series of basic vectors. HSNet [36] introduces 4-D convolutions to extract diverse features from different levels of intermediate convolutional layers. ASNet [37] trains a learner to construct class-wise foreground maps for multilabel classification and pixel-wise segmentation. NTRENet [38] explicitly mines and eliminates background and distracting objects regions for better segmentation. MSANet [39] exploits multiple feature-maps of support images and query images to estimate accurate semantic relationships. However, the feature distinction and the weak representation of limited support prototypes still hinder performance. Our SD-AANet utilizes self-distillation, prototypical learning and affinity learning, solving problems above and achieving performance improvements.

III. Proposed Method

In this section, we first briefly describe the definition of the few-shot segmentation task in Section III-A. Then in Sections III-B and III-C, we introduce our SDPM and SAAM in details, respectively. Finally, in Section III-D, we discuss optimization details and multiclass segmentation application of our proposed self-distillation embedded affinity attention model (SD-AANet).

A. Problem Setting

Few-shot segmentation is proposed to segment target of unseen classes under the guidance of limited annotated samples of the same classes. The data set is split into two sets, training set D_{train} and test set D_{test}, taking the class as the split standard. Defining the classes in D_{train} as C_{train} and the classes in D_{test} as C_{test}, the two sets do not intersect, which means $C_{\text{train}} \bigcap C_{\text{test}} = \phi$.

Training and testing processes of few-shot segmentation can be seen as episodes. The episode paradigm was proposed in [25], and Shaban et al. [8] first introduced it to few-shot segmentation. Each episode is consist of a support set S and a query set Q with the same class C. There are K samples in the support set S, which is formulated as $S = \{(I_s^1, M_s^1), (I_s^2, M_s^2), \ldots, (I_s^K, M_s^K)\}$. Each image-label pair (I_s^i, M_s^i) represents a sample in S, where I_s^i and M_s^i are the support image and its ground truth, respectively. Similar to the support set S, query set Q has one sample (I_q, M_q), where I_q and M_q are the query image and its ground truth, respectively, having the same class C with the support set S. The input of the model is a pair of query image I_q and support set S, formulated as $\{I_q, S\} = \{I_q, (I_s^1, M_s^1), (I_s^2, M_s^2), \ldots, (I_s^K, M_s^K)\}$. Query ground truth M_q is invisible in the training stage and it is used to evaluate the performance of methods.

B. Self-Distillation Guided Prototype Generating Module

Current prototypical learning methods, such as PFENet [34], approach a great performance on PASCAL-5^i and COCO-20^i, outperforming previous works by a large margin. Prototypes generated by these methods can efficiently guide the segmentation of query target. However, there are large feature differences between support and query targets. So we need to align the features of support and query. Objects always have two types of features, intrinsic features which commonly exist in all objects of this class and unique features which may distinct in different objects. Take airplane as an example,

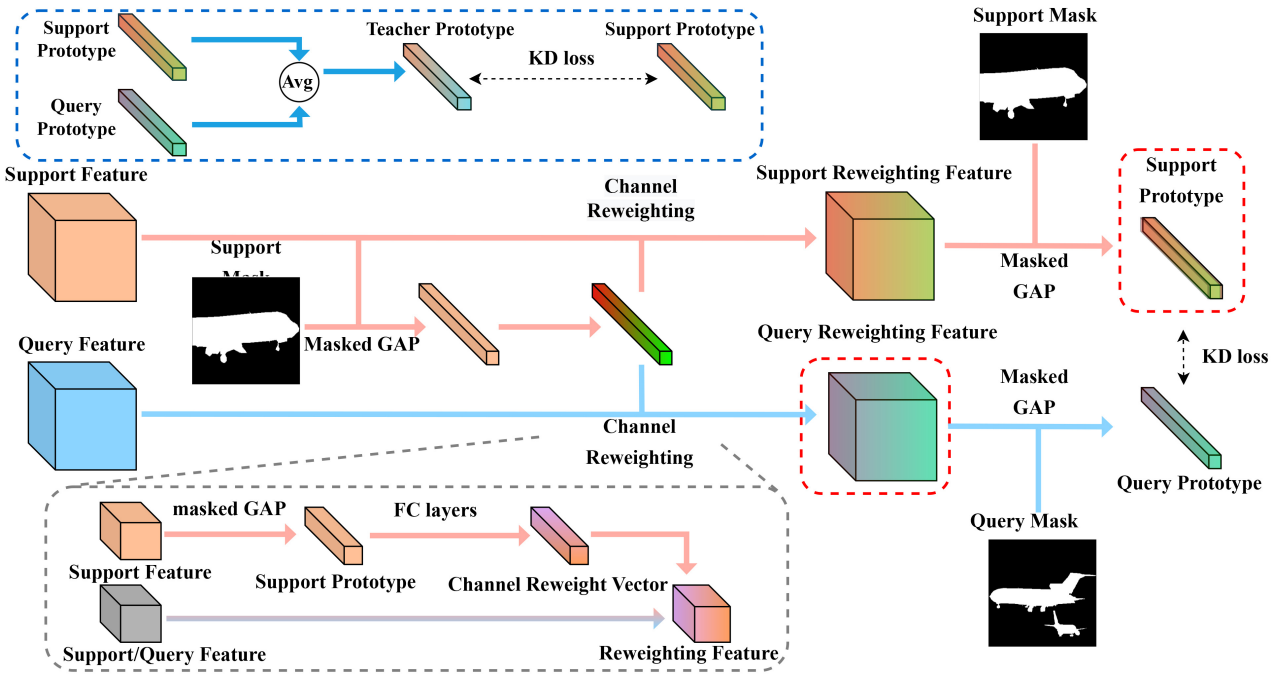


Fig. 4. SDPM first applies masked GAP to generate support prototype, then it uses support prototype to produce channel reweighting vector. Channels of both support feature and query feature are reweighting by above-mentioned reweighting vector. After that, new support prototype and query prototype are generated by masked GAP, then self-distillation approach is used between two prototypes to produce intrinsic support prototype. In order to promote the learning process of model, teacher vector in self-distillation approach is the average of support prototype and query prototype, as shown in blue dotted box. The outputs of SDPM are query channel reweighting feature and support prototype shown in red dotted boxes.

all airplane are made by metal and have wings. These features existing in all airplanes can be seen as intrinsic features. As the differences of shooting angle and lighting conditions, the shape and color of airplanes can be different, so they are unique features. Normally, humans have the cognitive ability to easily spot intrinsic features and apply them to subsequent tasks. For similar purposes, in few-shot segmentation, we need to find representative features of support and query images containing abundant intrinsic features.

The knowledge distillation approach proposed by Hinton et al. [40] greatly inspires us to transfer the knowledge between support and query prototypes. Zagoruyko and Komodakis [41] and He et al. [42] expanded the knowledge distillation technique by distilling attentions in middle layers. Fukuda et al. [43] proposed integrating multiple teacher networks to teach the student network. Zhao et al. [44] realized the knowledge distillation in a single deep neural network, where student network is a part of the teacher network.

Inspired by the above methods, we introduce self-distillation approach to the prototypical learning method, aiming to extract intrinsic features and aligning the features of support prototype and query prototype.

1) Support Guided Channel Reweighting: Different from the SENet [45] which uses the global feature to reweight channels of the feature map, we alternatively adopt support prototype with better suitability for obtaining object-related information. As shown in Fig. 4, architecture in a gray dotted box is the variation of SE Module. Support image I_s and query image I_q with the same class go through a shared backbone CNN, denoted as $\mathcal{F}(\cdot)$. M_s and M_q denote the ground truths of support and query, F_s and F_q denote the support feature and query feature which are outputs of middle-level layers of the backbone CNN

$$F_s = \mathcal{F}(I_s), F_q = \mathcal{F}(I_q) \tag{1}$$

note that I_s and I_q have the same shape $n \times c \times h \times w$, in which n, c, h, and w represent batch size, number of channels, height, and width of the feature map.

Then, the support prototype is generated by masked GAP by calculating the average vector of the features in object area in the feature map

$$p_s = \mathcal{F}_p(F_s, M_s) = \frac{\sum_{i=1}^{h}\sum_{j=1}^{w} F_s^{(i,j)}\left[M_s^{(i,j)} == 1\right]}{\sum_{i=1}^{h}\sum_{j=1}^{w}\left[M_s^{(i,j)} == 1\right]} \tag{2}$$

where i and j denote the index of row and column, $F_s^{(i,j)}$ denotes the position at row i, column j in support feature map, and $M_s^{(i,j)}$ denotes the position at row i, column j in support ground truth. $\mathcal{F}_p(\cdot)$ denotes the masked GAP operation. To guarantee the correctness of (2), M_s is resized to the same height and width with the support feature map. $[\cdot]$ denotes the Iverson bracket, a notation that signifies a number that is 1 if the condition in square brackets is satisfied, and 0 otherwise.

Acquired support prototype is then input to a series of FC layers to learn contributions of each feature channel, and there are ReLU functions between FC layers. Output of the FC layers is also a vector having the same number of channel with support feature and query feature. We have $v_s = \mathcal{FC}(p_s)$, where $\mathcal{FC}(\cdot)$ and v_s denote the FC layers and output channel reweighting vector, respectively.

The channel reweighting vector v_s scales channels of support feature and query feature, according to each channel's importance. Instead of using SE Block directly, we adopt a feature fusion strategy by using the average of scaled feature and input feature as the final channel reweighting feature

$$\tilde{F}_s = \frac{\mathcal{F}_{\text{scale}}(v_s, F_s) + F_s}{2}, \tilde{F}_q = \frac{\mathcal{F}_{\text{scale}}\left(v_s, F_q\right) + F_q}{2} \quad (3)$$

where $\mathcal{F}_{\text{scale}}(v_s, F_s)$ denotes the channel scale function and $\tilde{F}_s$ denotes the final channel reweighting feature, so do the $\mathcal{F}_{\text{scale}}(v_s, F_q)$ and $\tilde{F}_q$.

2) Self-Distillation Embedded Method: After Hinton et al. [40] first proposing the knowledge distillation in deep learning, many studies [44], [46], [47] have been conducted to let models learning from themselves. These approaches are named as self-distillation which aims to promote performance of model without external knowledge input.

Inspired by the above works, we introduce self-distillation approach to the prototypical neural network, which can significantly improve few-shot segmentation performance by extracting intrinsic support feature.

To generate the intrinsic support prototype with the help of the self-distillation approach, the average of support and query prototypes is used as the teacher. Masked GAP is employed to obtain both support prototype and query prototype from channel reweighting features as $p'_s = \mathcal{F}_p(\tilde{F}_s, M_s), p'_q = \mathcal{F}_p(\tilde{F}_q, M_q)$, where p'_s and p'_q denote the support prototype and query prototype after channel reweighting.

The query prototype and support prototype are adopted in self-distillation process. Both two prototypes can be seen as a combination of two parts feature, intrinsic feature and unique feature. So p'_s can be represented as $p'_s(f_i, f_s)$, where f_i and f_s denote the intrinsic feature and support unique feature, respectively. Similarly, query prototype p'_q can be represented as $p'_q(f_i, f_q)$, and f_q is the query unique feature. Following the knowledge distillation method, we apply the Kullback–Leibler (KL) divergence loss to realize the supervision of support prototype

$$d_s = \text{Softmax}\left(p'_s\right), \ d_q = \text{Softmax}\left(p'_q\right) \quad (4)$$

$$L_{\text{KD}} = \text{KL}(d_t \| d_s) = \sum_{i=1}^{c} d_t^i \log \frac{d_s^i}{d_t^i} \quad (5)$$

where Softmax$(\cdot)$ denotes the softmax function, d_s and d_q denote the outputs of the softmax function while inputs are p'_s and p'_q. d_t denotes the teacher prototype in knowledge distillation operation and it is equals to $d_t = [(d_s + d_q)/2]$. L_{KD} in (5) denotes the loss of self-distillation between support prototype and teacher prototype, and KL$(\cdot)$ denotes the KL divergence function.

Self-distillation approach enhances the consistency of support prototype and query prototype. Because the two prototypes are combinations of intrinsic feature and unique feature, the approach results in the lessen of unique feature and strengthen of intrinsic feature, which means $p'_s(f_i, f_s) \rightarrow p'_s(f_i)$. The $p'_s(f_i)$ in formula denotes the support prototype with only intrinsic feature, more suitable for guiding the query segmentation.

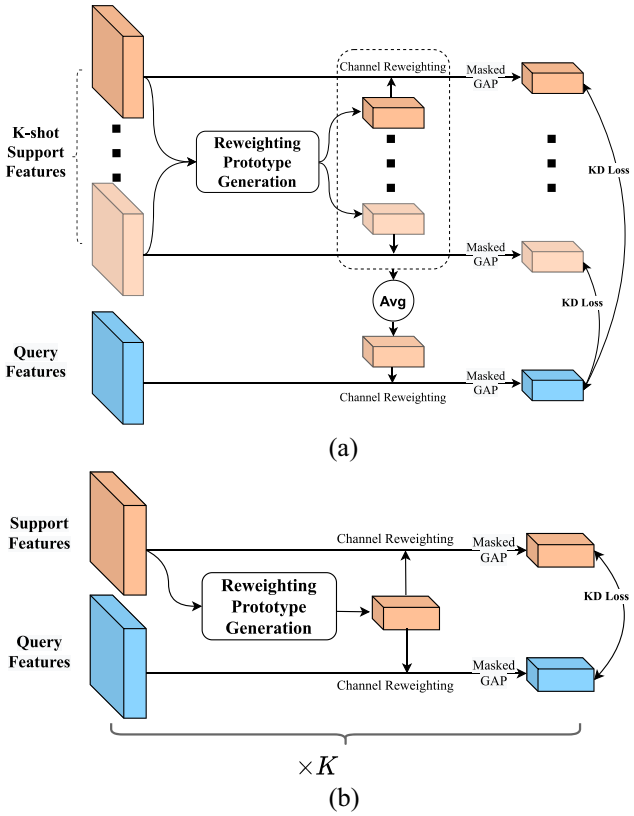


Fig. 5. Two strategies of SDPM in k-shot task, integral teacher prototype strategy (a) use the average of K reweighting vectors and masked GAP to produce the teacher prototype, while separate teacher prototype strategy (b) produces an exclusive teacher prototype for each support prototype.

SDPM can efficiently reduce the unique feature in support prototype and significantly ease the gap between support and query features. Extensive experiments show the improvement of performance more intuitively.

3) K-Shot Setting: In addition to 1-shot segmentation, segmenting the query target under the guidance of K (greater than 1) support images is defined as K-shot segmentation. To extend SDPM to K-shot segmentation, this module needs to be modified appropriately. Because of the distinctions between K support images, teacher prototype extracted from query feature should supervise each support prototype separately. Depending on whether the teacher prototypes of K support prototypes are same, we design two strategies of SDPM for K-shot segmentation task, Integral Teacher Prototype Strategy, and Separate Teacher Prototype Strategy. Details of two strategies are shown in Fig. 5.

The core idea of the Integral Teacher Prototype Strategy is applying the average of K reweighting vectors to scale each channel of query feature. Then masked GAP is adopted to extract teacher prototype, and K knowledge distillation losses are calculated between teacher prototype and each support prototype. The final self-distillation loss is the average of K losses

$$\tilde{F}_q = \frac{\mathcal{F}_{\text{scale}}\left(\sum_{i=1}^{K} v_s^i, F_q\right) + F_q}{2} \quad (6)$$

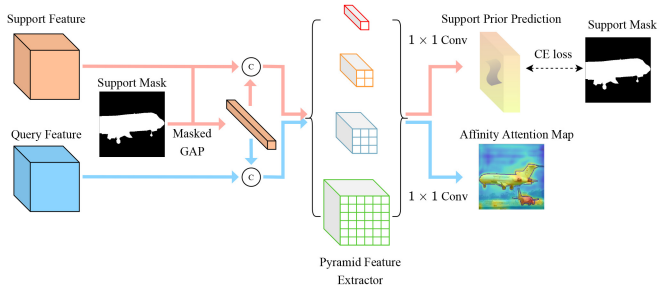


Fig. 6. SAAM produces support prototype using masked GAP and concatenates support prototype to both support feature and query feature. PPM [6] is adopted to extract features of two concatenated results. Then, two 1×1 convolutions with a channel of 1 and 2 are applied to generate support prediction and query attention map. Support ground truth is used to supervise the support prediction, and the output of SAAM is query attention map.

$$L_{\mathrm{KD}}^{I} = \frac{1}{K}\sum_{i=1}^{K} \mathrm{KL}\left(p_q' \| v_s^i\right) \tag{7}$$

where v_s^i denotes the prototype of ith support sample, p_q' denotes the query prototype generated from $\tilde{F}_q$, and L_{KD}^{I} denotes the knowledge distillation loss of Integral Teacher Prototype Strategy.

Different from the Integral Teacher Prototype Strategy, as shown in Fig. 5(b), Separate Teacher Prototype Strategy produces an exclusive teacher prototype for each support prototype, via applying each reweighting vector to scale the query feature separately

$$\tilde{F}_q^i = \frac{\mathcal{F}_{\mathrm{scale}}\left(v_s^i, F_q\right) + F_q}{2} \tag{8}$$

$$L_{\mathrm{KD}}^{S} = \frac{1}{K}\sum_{i=1}^{K} \mathrm{KL}\left(p_q^i \| v_s^i\right) \tag{9}$$

where $\tilde{F}_q^i$ and p_q^i denote the query reweighting feature and teacher prototype produced by ith support reweighting vector. L_{KD}^{S} denotes the knowledge distillation loss of Separate Teacher Prototype Strategy. Final output query feature $\tilde{F}_q$ of SDPM is the average of K query reweighting features, formulated as $\tilde{F}_q = (1/K)\sum_{i=1}^{K} \tilde{F}_q^i$.

No matter which strategy is applied, the final output support prototype is the average of K support prototypes. The ablation experiments in Section IV-C show the performances of two strategies. In the end, we use Separate Teacher Prototype Strategy in our overall model.

C. Supervised Affinity Attention Mechanism

Limited support prototypes can not provide enough target features, so we consider using the attention mechanism to provide more adequate prior information about the target for the task. The attention mechanism can effectively capture the location of object and let deep neural network models know where to focus, some previous works also utilize the attention mechanism. PFENet [34] uses high-level features of both support and query to generate query attention map. By employing ImageNet [48] pretrained model as backbone and fixing its weights, the prior attention mechanism is training-free.

However, the ImageNet pretrained model is produced to tackle the classification task, which is not suitable to generate attention map in a few-shot segmentation task straightly. So we propose a supervised affinity attention mechanism (SAAM), which can solve the problem caused by unrepresentative of limited support features. The architecture of our SAAM is shown in Fig. 6.

1) Supervised Attention: We first utilize masked GAP to obtain a support prototype and expand it to the same spatial shape with a support feature. Then the expanded prototype is concatenated to both support feature and query feature, we define the results as $F_{C,s}$ and $F_{C,q}$, respectively. Following, $F_{C,s}$ and $F_{C,q}$ are input to a pyramid feature extractor severally and outputs are defined as $F_{P,s}$ and $F_{P,q}$. We use PPM [6] as the pyramid feature extractor.

On the head of the PPM, there are two 1×1 convolution layers (Convs) to generate support prediction and query attention map, respectively. Support prediction is generated by 1×1 Conv with two output channels. The 1×1 Conv for query attention map generation has only one output channel. Then, a support label is applied to supervise the SAAM

$$L_{ce,s} = -\sum_{i=1}^{h}\sum_{j=1}^{w}\left(M_s^{(i,j)} \cdot \log\left(P_s^{(i,j)}\right)\right) \tag{10}$$

where $L_{ce,s}$ is the cross entropy loss of support prediction in SAAM. $M_s^{(i,j)}$ and $P_s^{(i,j)}$ are (i, j) location of support mask and support prediction in SAAM.

The SAAM uses the whole support feature to guide the generation of a query attention map, so the missing intrinsic features of support are learned under supervision. Combined with SDPM, the SD-AANet makes a tradeoff during training and obtains richer intrinsic features to achieve better segmentation performance.

2) K-Shot Setting: K-shot setting of SAAM is intuitive, because the only different part is support path. K support features go through SAAM severally, then each support prediction is supervised by its own label. Loss of K-shot SAAM is the average of K losses

$$L_{ce,s} = \sum_{i=1}^{K} L_{ce,s}^{i} \tag{11}$$

where $L_{ce,s}^{i}$ is the cross entropy loss of ith support image.

D. Optimization

Based on the SDPM and SAAM, we propose the self-distillation embedded supervised affinity attention network (SD-AANet), as shown in Fig. 3. For the whole model, we choose cross-entropy loss L_{ce} for the final segmentation prediction. Counting losses of SDPM and SAAM in, the total loss of SD-AANet is the combination of L_{ce}, L_{KD}, and $L_{ce,s}$ as follows:

$$L = L_{ce} + \alpha \cdot L_{\mathrm{KD}} + \beta \cdot L_{ce,s} \tag{12}$$

where α and β are coefficients of L_{KD} and $L_{ce,s}$, used to balance three loss compositions.

Algorithm 1: Learning Process of Multiclass 1-Shot Segmentation

Input: support set $S = \{(I_s^1, M_s^1), \cdots, (I_s^5, M_s^5)\}$, query image I_q and mask M_q

Output: learnable parameters p of SD-AANet

for *each batch* $i = 0$ *to total batch* **do**
- Obtain support prototypes v_s^i ($i = 1, \ldots, 5$);
- Get attention maps m^i ($i = 1, \ldots, 5$) and query feature F_q;
- Concatenate v_s^i to v_c;
- $v_{new} = MLP(v_c)$;
- Concatenate v_{new}, F_q and m^i as F_{new};
- $p = Decoder(F_{new})$;
- Calculate L_{ce}, L_{KD} and $L_{ce,s}$;
- Train model with $L = L_{ce} + \alpha \cdot L_{KD} + \beta \cdot L_{ce,s}$;

end

E. Multiclass Few-Shot Segmentation

To further explore the potential of SD-AANet, we propose a new pipeline for segmenting multiclass objects simultaneously under few-shot setting. Because the fore-mentioned method and pipeline can not applied to segmenting multiclass objects straightly, we modify the decoder and design new training and testing pipeline. We describe the modification and the new pipeline under 1-shot setting as follows.

The input support images and masks are come from five different classes, at least one of which has same class with objects in query image. Then after going through encoder, SDPM and SAAM, there will be one query feature, five support prototypes, and five attention maps. Before decoder, we concatenate these prototypes to one vector and add an MLP to reduce its dimension. Then we concatenate the obtained vector, query feature, and five attention map as input of decoder. To make decoder segment objects from five classes simultaneously, we change the final output channel from 2 to 5. The output prediction has five channels, whose order is the same as the order of concatenation of five support prototypes. The whole learning process of multiclass 1-shot segmentation can be seen in Algorithm 1.

IV. Experiments

A. Implementation Details

1) Data Sets: The PASCAL-5^i [8] and COCO-20^i [15], [16] data sets are used in experiments to evaluate our method. PASCAL-5^i consists of two parts, PASCAL VOC 2012 [57] and extended annotations from SDS data sets [58]. There are 20 classes in original PASCAL VOC 2012 and SDS, and they are evenly divided into four folds, defined as Fold-i, $i \in \{0, 1, 2, 3\}$. Each fold contains five classes following settings in OSLSM [8], and 1000 pairs of support-query are used in our test.

Following [16], COCO data set, owning 80 classes totally, is also splited into four folds with 20 classes in each fold. The set of class indexes contained in fold-i is written as $\{4x - 3 + i\}$, $i \in \{1, 2, \ldots, 20\}$. Due to the number of images in COCO validation is 40137, which is much more than the PASCAL-5^i. So we randomly choose 4000 support-query pairs each fold during testing following [16], which can provide more reliable and stable results for 20 classes than 1000 pairs.

To realize the few-shot segmentation, we use three folds to train and test the model on the last fold for cross-validation. We alternatively choose different folds in testing to evaluate performance of our model, and we carry out five rounds of experiment and take the average to get the final experimental results.

2) Experimental Setting: We use PyTorch to construct our framework, and we apply ResNet50 and ResNet101 [17] as our backbones for PASCAL-5^i and COCO-20^i, respectively. We choose the ResNet with atrous convolutions as the same with previous works [13], [16], [59]. The ImageNet [48] pretrained weights provided by PyTorch are used to initialize backbone networks. We use SGD as our optimizer. We set the momentum and weight decay to 0.9 and 0.0001, respectively. The "poly" policy [5] is adopted in our experiments to decay the learning rate by multiplying $(1 - [\text{current}_{\text{iter}} \text{max}_{\text{iter}}])^{\text{power}}$ where *power* is set to 0.9. α and β are set to 50 and 0.5, respectively. We use PFENet [34] as our baseline.

The experiments on PASCAL-5^i train models for 200 epochs as [34], while the initial learning rate and batch size are set to 0.0025 and 4. Because there are more images in training set of COCO-20^i, we train models for 50 epochs with 0.0005 and 8 for the initial learning rate and batch size, respectively. We fix the parameters of backbone networks and update other parameters during training. Each example is processed with mirror operation and random rotation from -10 to 10 degrees. Finally, limited by equipment performance, we randomly crop 321×321 patches from the processed images as training samples, which significantly reduces storage consumption and runtime. During evaluation, we resize the processed images to 321×321 and pad zero to maintain the original aspect ratio of images. Then the prediction is resized back to the original label sizes to evaluate performance. Following [34], for COCO-20^i, we also resize the prediction to 473×473 with respect to its original aspect ratio to make another evaluation. The single-scale results are output without multiscale testing and any other post-processing. Our experiments are conducted on an NVIDIA GeForce RTX3090 GPU and Intel Xeon CPU 10900K.

3) Evaluation Metrics: Following [13], [16], [34], we adopt class means intersection over union (mIoU) as our evaluation metric, because the class mIoU is more reasonable than the foreground-background IoU (FB-IoU) [13]. The formulation of class mIoU is $(1/C) \sum_{i=1}^{C} \text{IoU}_i$, where C is the number of classes belong to each fold. So $C = 5$ for PASCAL-5^i and $C = 20$ for COCO-20^i. The IoU_i is intersection over union of ith class.

B. Results

As shown in Tables I and II, we adopt ResNet50 and ResNet101 to build our models for PASCAL-5^i and COCO-20^i, respectively. And, we report the class mIoU results to prove the performance of our proposed models. By

TABLE I
COMPARISON WITH STATE-OF-THE-ART METHODS ON PASCAL-5^i WITH CLASS MIOU METRIC. BASELINE IN TABLE FOLLOWS THE PFENET [34]

Methods	Backbone	1-shot					5-shot				
		Fold-0	Fold-1	Fold-2	Fold-3	Mean	Fold-0	Fold-1	Fold-2	Fold-3	Mean
OSLSM [8]	VGG-16	33.6	55.3	40.9	33.5	40.8	35.9	58.1	42.7	39.1	44.0
co-FCN [49]	VGG-16	36.7	50.6	44.9	32.4	41.1	37.5	50.0	44.1	33.9	41.4
SG-One [12]	VGG-16	40.2	58.4	48.4	38.4	46.3	41.9	58.6	48.6	39.4	47.1
AMP [50]	VGG-16	41.9	50.2	46.7	34.7	43.4	41.8	55.5	50.3	39.9	46.9
PANet [31]	VGG-16	42.3	58.0	51.1	41.2	48.1	51.8	64.6	59.8	46.5	55.7
PGNet [9]	ResNet50	56.0	66.9	50.6	50.4	56.0	57.7	68.7	52.9	54.6	58.5
FWB [16]	ResNet101	51.3	64.5	56.7	52.2	56.2	54.8	67.4	62.2	55.3	59.9
CANet [13]	ResNet50	52.5	65.9	51.3	51.9	55.4	55.5	67.8	51.9	53.2	57.1
CRNet [51]	ResNet50	-	-	-	-	55.7	-	-	-	-	58.8
RPMMs [14]	ResNet50	55.2	65.9	52.6	50.7	56.3	56.3	67.3	54.5	51.0	57.3
SimPropNet [52]	ResNet50	54.8	67.3	54.5	52.0	57.2	57.2	68.5	58.4	56.1	60.0
PPNet [32]	ResNet50	47.8	58.8	53.8	45.6	51.5	58.4	67.8	64.9	56.7	62.0
DAN [11]	ResNet101	54.7	68.6	57.8	51.6	58.2	57.9	69.0	60.1	54.9	60.5
PFENet [34]	ResNet50	61.7	69.5	55.4	56.3	60.8	63.1	70.7	55.8	57.9	61.9
ASGNet [53]	ResNet101	59.8	67.4	55.6	54.4	59.3	64.6	71.3	64.2	57.3	64.4
SCL [33]	ResNet50	63.0	70.0	56.5	57.7	61.8	64.5	70.9	57.3	58.7	62.9
MMNet [54]	ResNet50 v2	62.7	70.2	57.3	57.0	61.8	62.2	71.5	57.5	62.4	63.4
CWT [55]	ResNet50	56.3	62.0	59.9	47.2	56.4	61.3	68.5	68.5	56.6	63.7
CMN [56]	ResNet50	64.3	70.0	57.4	59.4	62.8	65.8	70.4	57.6	60.8	63.7
ASR [35]	ResNet50	55.2	70.4	53.4	53.7	58.2	59.4	71.9	56.9	55.7	61.0
HSNet [36]	ResNet50	64.3	70.7	60.3	60.5	64.0	70.3	73.2	**67.4**	**67.1**	69.5
ASNet [37]	ResNet50	**68.9**	71.7	**61.1**	**62.7**	**66.1**	**72.6**	**74.3**	65.3	**67.1**	**70.8**
NTRENet [38]	ResNet50	**68.9**	71.7	59.4	59.8	64.2	66.2	72.8	61.7	62.2	65.7
Baseline(ours)	ResNet50	59.5	70.0	55.9	56.1	60.4	63.7	70.5	57.2	57.5	62.2
SD-AANet(ours)	ResNet50	62.7	**71.8**	58.8	58.1	62.9	65.5	71.6	62.5	62.3	65.5

TABLE II
COMPARISON WITH STATE-OF-THE-ART METHODS ON COCO-20^i WITH CLASS MIOU METRIC. BASELINE IN TABLE FOLLOWS THE PFENET [34]. MODELS WITH † ARE EVALUATED ON THE LABELS RESIZED TO 473×473 SIZE. MODELS WITHOUT† ARE TESTED ON LABELS WITH THE ORIGINAL SIZES

Methods	Bcakbone	1-shot					5-shot				
		Fold-0	Fold-1	Fold-2	Fold-3	Mean	Fold-0	Fold-1	Fold-2	Fold-3	Mean
FWB [16]	ResNet101	19.9	18.0	21.0	28.9	21.2	19.1	21.5	23.9	30.1	23.7
PANet [31]	VGG-16	-	-	-	-	20.9	-	-	-	-	29.7
DAN [11]	ResNet101	-	-	-	-	24.4	-	-	-	-	29.6
RPMMs [14]	ResNet50	-	-	-	-	30.6	-	-	-	-	35.5
PPNet [32]	ResNet50	28.1	30.8	29.5	27.7	29.0	39.0	40.8	37.1	37.3	38.5
PFENet [34]	ResNet101	34.3	33.0	32.3	30.1	32.4	38.5	38.6	38.2	34.3	37.4
PFENet† [34]	ResNet101	36.8	41.8	38.7	36.7	38.5	40.4	46.8	43.2	40.5	42.7
ASGNet [53]	ResNet50	-	-	-	-	34.6	-	-	-	-	42.5
SCL [33]	ResNet101	36.4	38.6	37.5	35.4	37.0	38.9	40.5	41.5	38.7	39.9
MMNet [54]	ResNet50 v2	34.9	41.0	37.2	37.0	37.5	37.0	40.3	39.3	36.0	38.2
CWT [55]	ResNet101	30.3	36.6	30.5	32.2	32.4	38.5	46.7	39.4	43.2	42.0
CMN [56]	ResNet50	37.9	**44.8**	38.7	35.6	39.3	42.0	50.5	41.0	38.9	43.1
Baseline(ours)	ResNet101	35.5	41.5	37.6	36.1	37.7	41.5	47.3	45.2	41.6	43.9
Baseline(ours)†	ResNet101	36.7	41.9	38.1	37.2	38.5	41.7	48.6	46.8	42.9	45.0
SD-AANet(ours)	ResNet101	39.3	43.9	39.8	39.5	40.6	43.1	51.4	52.7	46.3	48.4
SD-AANet(ours)†	ResNet101	**39.6**	44.3	**39.9**	**39.9**	**40.9**	**43.2**	**51.9**	**52.9**	**46.6**	**48.7**

incorporating the SDPM and SAAM, with the 321×321 size of input images which is smaller than 473×473 used in previous works, our SD-AANet still achieves comparable state-of-the-art results on PASCAL-5^i and reaches new state-of-the-art results on COCO-20^i for class mIoU metric.

In Table I, we compare our model with other state-of-the-art methods on PASCAL-5^i. On this data set, ASNet [37] and HSNet [36] achieve better results which are significantly higher than other methods. However, HSNet introduces 4-D convolutions to integrate multilevel features, although center-pivot is used to decrease the space and time complexities, the costs are still huge. ASNet computes the correlations between each point in support feature and those in query feature, which can introduce nonnegligible computational cost. Otherwise, our SD-AANet still shows some advantages in some categories, such as the result on Fold-1 for 1-shot task.

In Table II, our SD-AANet achieves new state-of-the-art performance on COCO-20i for both 1-shot task and 5-shot

TABLE III
COMPLEXITY AND COMPUTATIONAL EFFICIENCY OF BASELINE AND SD-AANET

Methods	GPU memory	FPS	# learnable params
Baseline	2216M	18.75	10.8M
SD-AANet	2483M	17.65	14.1M

TABLE IV
ABLATION STUDY OF OUR PROPOSED SDPM AND SAAM ON PASCAL-5^i FOR 1-SHOT SEGMENTATION. "SD-AANET" MEANS THE MODEL WITH BOTH SDPM AND SAAM

Methods	Fold-0	Fold-1	Fold-2	Fold-3	Mean
Baseline	59.5	70.0	55.9	56.1	60.4
Baseline + SAAM	60.5	70.8	57.4	57.6	61.6
Baseline + SDPM	61.1	71.0	58.0	57.8	62.0
SD-AANet	**62.7**	**71.8**	**58.8**	**58.1**	**62.9**

TABLE V
ABLATION STUDY OF USING SINGLE-SCALE INFERENCE OR MULTISCALE INFERENCE ON PASCAL-5^i FOR 1-SHOT SEGMENTATION

Methods	Fold-0	Fold-1	Fold-2	Fold-3	Mean
Single-scale	**62.7**	71.8	58.8	58.1	62.9
Multi-scale	62.5	**72.2**	**59.3**	**58.9**	**63.2**

task, and surpasses CMN [56] by 1.6% and 5.6%. Besides, the SDPM and SAAM improve the performance by 2.4% and 3.7% than our baseline.

We analyze the complexity and computational efficiency of the baseline model and SD-AANet in Table III. Compared to the baseline model, our SD-AANet increases the GPU memory cost during the inference phase and the number of learnable parameters by 12% and 30%, respectively. Otherwise, SD-AANet slightly reduces the inference speed from 18.75 to 17.65 on frames per second (FPS).

From what has been discussed above, due to the introduction of SDPM and SAAM, SD-AANet achieves superior results on few-shot segmentation task.

C. Ablation Study

1) Ablation Study of SDPM and SAAM: To quantitatively analyze the influence of SDPM and SAAM, we conduct an experiment about the performance of model w/ and w/o the SDPM and SAAM. Table IV shows the class mIoU results of each model on PASCAL-5^i for 1-shot task.

It can be seen in Table IV that, compared to baseline, using only SAAM or SDPM improve the performance with class mIoU increases of 1.2% and 1.6%, respectively. Adopting both SAAM and SDPM can further improve the performance, with 2.5% class mIoU gain.

2) Ablation Study of Multiscale Inference: Table V shows a comparison experiment between single-scale inference and multiscale inference of SD-AANet. The experiment is conducted on PASCAL-5^i for 1-shot task. In this experiment, we resize the logits before classify layer to 321×321 and 473×473, and adopt softmax operation to get two predictions with different size. Then, we resize the 321×321 prediction to 473×473 by using bilinear interpolation. Finally, the two 473×473 predictions are added to get final prediction.

TABLE VI
ABLATION STUDY OF SDPM WITH INTEGRAL TEACHER PROTOTYPE STRATEGY OF SEPARATE TEACHER PROTOTYPE STRATEGY ON PASCAL-5^i FOR 5-SHOT SEGMENTATION

Methods	Fold-0	Fold-1	Fold-2	Fold-3	Mean
Integral Strategy	64.9	70.8	61.7	61.1	64.6
Separate Strategy	**65.5**	**71.6**	**62.5**	**62.3**	**65.5**

TABLE VII
ABLATION STUDY OF BASELINE MODEL AND SD-AANET FOR MULTICLASS 1-SHOT SEGMENTATION

Methods	Fold-0	Fold-1	Fold-2	Fold-3	Mean
Baseline	39.2	45.0	36.0	36.9	39.3
SD-AANet	42.3	49.2	41.2	40.1	43.2

The result shows that multiscale inference can improve the performance slightly with 0.3% class mIoU gain.

3) Ablation Study of 5-Shot Strategy: Table VI studies the influence of different 5-shot strategy of SDPM. We design two strategies for SDPM in 5-shot task, Integral Teacher Prototype Strategy and Separate Teacher Prototype Strategy, named Integral Strategy and Separate Strategy later. Comparison experiment shown in Table VI indicates the Separate Strategy achieves more outstanding result than Integral Strategy. It means that assign a unique teacher prototype to each support prototype can facilitate the production of intrinsic support prototype, which promotes the segmentation performance of query target.

4) Ablation Study of Multiclass Segmentation: To further explore the potential of SD-AANet, we propose a new pipeline for segmenting multiclass objects simultaneously under few-shot setting. We conduct experiments on Pascal-5^i using the baseline model and our SD-AANet. The experiments are based on 1-shot setting.

As show in Table VII, due to the difficulty of segmenting objects from multiply classes simultaneously, the results of multiclass 1-shot segmentation experiments are significantly lower than current few-shot segmentation results. However, Table VII still clearly shows the advantages of SD-AANet over the baseline model. On each fold, our SD-AANet can improve the performance of the baseline model by at least 3.2 mIoU. For average result on all four folds, SD-AANet gets 3.9 mIoU increase compared to baseline.

We also analyze results of two models on each classes. As show in Table VIII, "Class1" to "Class 5" denote five classes in each folds orderly. We can see that for some classes, SD-AANet only gets slight improvement such as the "Class 5" of Fold-3, which is "tv/monitor." However, on some hard classes such as the "Class 4" of Fold-2, "motorbike," SD-AANet achieves remarkable progress.

D. Visualization Analysis

1) Qualitative Visualization of Segmentation Results: To show the performance of our proposed architectures intuitively, we visualize final prediction masks produced by our SD-AANet in Fig. 7. Meanwhile, we compare the segmentation results between baseline and SD-AANet to evaluate the performance improvement realized by SD-AANet.

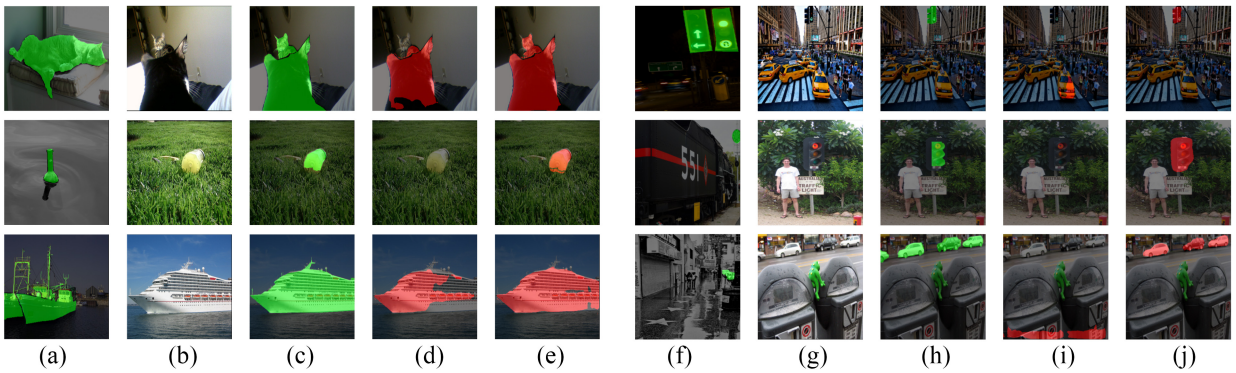


Fig. 7. Qualitative results of the proposed SD-AANet and the baseline for 1-shot task. The columns from (a)–(e) is results on PASCAL-5^i, and the columns from (f)–(j) are results on COCO-20^i. (a) Support and GT. (b) Query. (c) GT. (d) Baseline. (e) SD-AANet. (f) Support and GT. (g) Query. (h) GT. (i) Baseline. (j) SD-AANet.

TABLE VIII
MULTICLASS 1-SHOT SEGMENTATION EXPERIMENT RESULTS ON EACH CLASS

Fold index	Method	Class 1	Class 2	Class 3	Class 4	Class 5	Mean
Fold-0	Baseline	25.0	52.2	34.1	31.7	53.0	39.2
	SD-AANet	26.1	55.3	46.9	27.0	56.1	42.3
Fold-1	Baseline	38.4	56.9	15.1	58.0	56.6	45.0
	SD-AANet	42.0	61.7	16.3	64.1	61.9	49.2
Fold-2	Baseline	55.5	57.0	51.9	5.4	10.3	36.0
	SD-AANet	56.4	59.2	52.2	16.5	21.8	41.2
Fold-3	Baseline	57.9	35.9	52.3	19.8	18.9	36.9
	SD-AANet	61.5	40.7	57.3	21.8	19.1	40.1

As shown in Fig. 7, columns (a) and (f) are support images and their ground truths, which are marked in green in figures. Columns (b) and (g) are query images and columns (c) and (h) are the ground truths of them, which are also marked in green. Columns (d) and (i) are the prediction results of baseline, and columns (e) and (j) are the predictions of SD-AANet, marked in red.

As we can see, the second row in Fig. 7 shows cases which have tremendous differences between support objects and query target. Taking (a) to (e) columns as an example, the bottles in support image and query image has totally different colors, shapes, and perspectives, which leads to the segmentation failure of baseline. Relying upon the intrinsic feature extracted by SD-AANet, we can capture intrinsic features of the class and ignore the interference of other factors, so we successfully segment the bottle with negligible error. Other samples also confirm this point.

2) t-SNE Visualization of Support Prototypes: We conduct a *t*-distributed stochastic neighbor embedding (*t*-SNE) visualization experiment for support prototypes in Fig. 8. In Fig. 8, four columns in turn represent results of baseline, baseline with SAAM, baseline with SDPM and SD-AANet, and four rows in turn represent four folds from Fold-0 to Fold-3. We use models to process 5000 samples of five novel classes and get 5000 support prototypes output from SDPM or backbone. Then t-SNE is adopted to embed prototypes to 2-D space to visualize, and the operation is repeated for four methods on four folds. As shown in Fig. 8, SDPM can significantly expand the distance between prototypes of different classes and make prototypes of same classes more compact, so the results in columns (c) and (d) are more distinct. Columns (b) and (d) show SAAM can also make support prototypes more discriminative to a certain extent.

Taking Fold-1 and Fold-3 as examples, figures about two folds in the first column show prototypes of five classes mix together and some classes are split at both ends of figures. The second column, which represents baseline with SAAM can distinguish classes slightly clearer, such as gray points in Fold-1 and orange points in Fold-3. In the third column, baseline with SDPM has the greater performance to produce intrinsic prototype, so the points with same color seemed more compact, such as gray points in Fold-1 and blue points in Fold-3. Combine the SAAM and SDPM, SD-AANet achieves the best performance which can obviously seen in figures. Clear dividing lines can be seen in the fourth column figures of Fold-1 and Fold-3.

3) Visualization of Performance on Small Target: To further analyze the segmentation performance of SD-AANet for small targets, we choose the samples whose targets have less than 5000 pixels to conduct comparison experiment between SD-AANet and baseline. The samples are split to ten parts, each part has a span of 500 pixels. We calculate the average class mIoU of each part produced by two models to draw a histogram, shown in Fig. 9.

The results shown in Fig. 9 illustrate that our SD-AANet achieves more class mIoU in all ten parts, so SD-AANet has greater segmentation performance than baseline for small target segmentation.

4) Visualization of Affinity Attention: To intuitively analyze the quality of affinity attention produced by SAAM, we visualize the attentsion map as shown in Fig. 10. The first two rows are query image and its ground truth label, respectively, and the third row is the affinity attention map where close to red (warm-toned) means more attention, vice versa.

In the first three columns, we can see that SAAM can effectively capture the spatial information of targets, even if they are small or there are multiple targets in one image. The last two columns show the SAAM can focus on large-scale targets and capture key information of separate parts, such as the rear-view mirror of the bus and wheels of the airplane, with only one support sample.

5) Visualization of Prototype's Representative: To discuss the representative of support prototype, we calculate the cosine

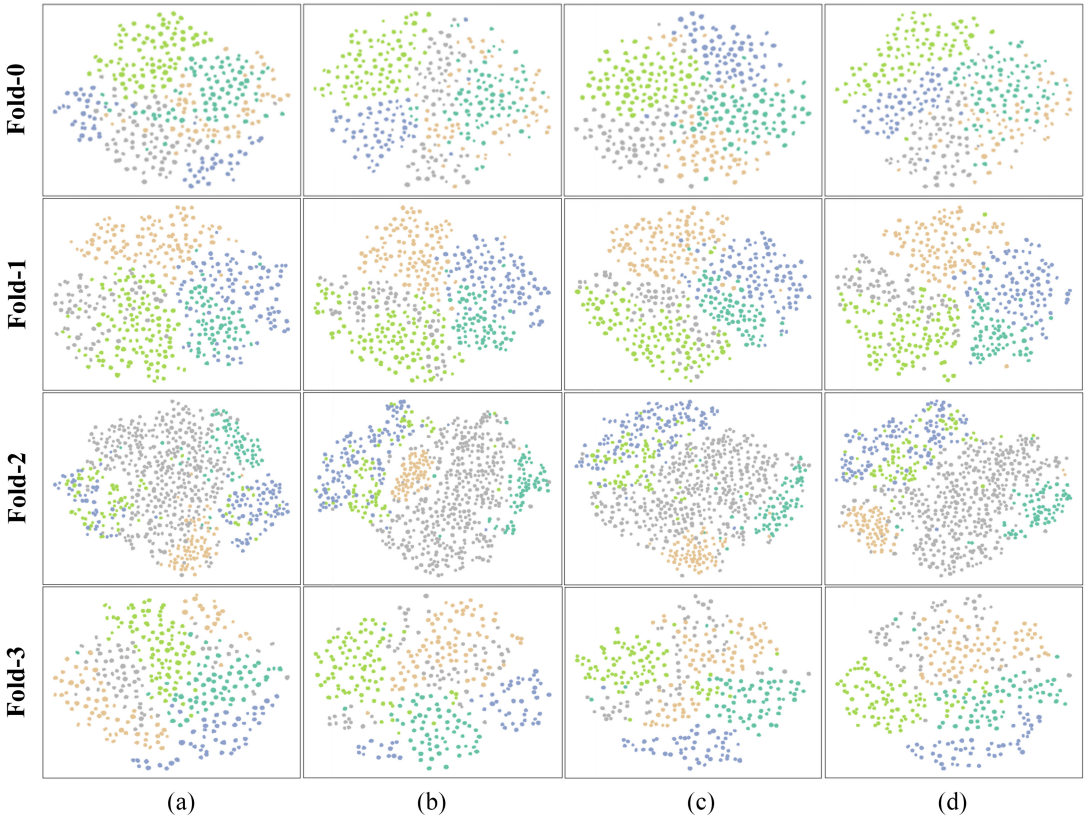


Fig. 8. Visualization comparison study of support prototypes between t-SNE results. Each figure contains 5000 support prototypes generated from the same 5000 image pairs. (a) Baseline. (b) Baseline + SAAM. (c) Baseline + SDPM. (d) SD-AANet.

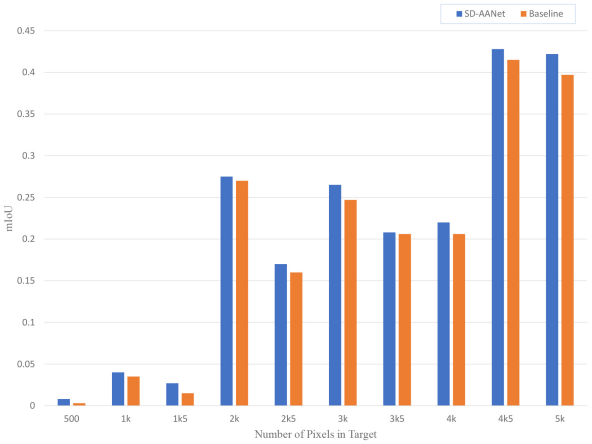


Fig. 9. Quantitative comparative experiment of segmentation performance on small-scale target between baseline and SD-AANet on PASCAL-5^i for 1-shot segmentation.

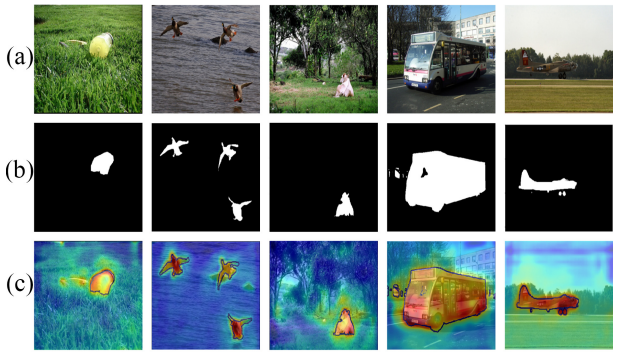


Fig. 10. Visualization of affinity attention map generated by SAAM on PASCAL-5^i for 1-shot segmentation. (a) Query. (b) GT. (c) Attention.

similarity between support prototype and its feature map

$$\cos\left(x_{i,j}, p'_s\right) = \frac{x_{i,j}^{\mathrm{T}} \cdot p'_s}{\left\|x_{i,j}\right\| \cdot \left\|p'_s\right\|} \tag{13}$$

where $x_{i,j}$ denotes the vector in support feature at (i, j) location and $x_{i,j}^{\mathrm{T}}$ denotes its transpose. p'_s denotes the support prototype output from SDPM. Finally, the $\|\cdot\|$ denotes the norm of the vector.

Similar to attention maps in Fig. 10, the similarity map shown in Fig. 11 use warm-toned color to represent high similarity. Three rows from top to bottom in turn illustrate support image and its ground truth, similarity map produced by baseline, and similarity map produced by SA-AANet. In first four columns we can see the prototype produced by SD-AANet can filter more irrelevant background compared to baseline, which means the prototype focus on intrinsic feature to capture the target regardless of other environmental factors. In the fifth column, baseline fails to found part of sheep which is mixed up with the fence, while SD-AANet finds the whole spatial area of the sheep.

Fig. 11. Visualization of support similarity map generated by support prototype and support feature using cosine similarity. The comparison visualization study is process on PASCAL-5^i for 1-shot segmentation. (a) Support and GT. (b) Baseline. (c) SD-AANet.

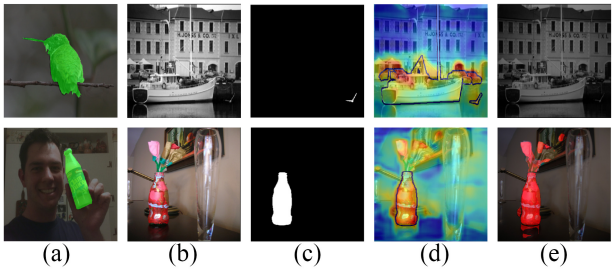


Fig. 12. Visualization study of failure cases of SD-AANet on PASCAL-5^i for 1-shot segmentation. (a) Support and GT. (b) Query. (c) GT. (d) Attention. (e) Prediction.

6) Visualization of Failure Cases: As shown in Fig. 12, SD-AANet still fails to segment some targets because the target size is too small or the target is very similar to the background. The first column is support image and its ground truth, and the next two columns are query image and its ground truth, respectively. The fourth column is the attention map produced by SAAM and the last column is prediction of SD-AANet.

We can see in the first row that the target is a bird of very small size, and there are great differences between support object and query target. In the second row, the bottle is very similar to the flower which is hard to discriminate.

V. Conclusion

In this article, we propose a novel few-shot segmentation method named SD-AANet. Our method significantly differs from existing methods by combining self-distillation, prototypical learning, and affinity learning. To address the problem of large intraclass variation, we propose the SDPM to extract intrinsic prototype, which can efficiently align the features of support and query. We further construct an SAAM for generating high-quality prior attention map of query image. Extensive experiments on two standard benchmarks verify the performance superiority of our method. Besides, future work may focus on utilizing self-distillation to zero-shot segmentation task.

References

[1] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3431–3440.

[2] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017.

[3] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent.*, 2015, pp. 234–241.

[4] G. Lin, A. Milan, C. Shen, and I. Reid, "RefineNet: Multi-path refinement networks for high-resolution semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1925–1934.

[5] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2018.

[6] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2881–2890.

[7] Y. Lu, Y. Chen, D. Zhao, B. Liu, Z. Lai, and J. Chen, "CNN-G: Convolutional neural network combined with graph for image segmentation with theoretical analysis," *IEEE Trans. Cogn. Develop. Syst.*, vol. 13, no. 3, pp. 631–644, Sep. 2021.

[8] A. Shaban, S. Bansal, Z. Liu, I. Essa, and B. Boots, "One-shot learning for semantic segmentation," in *Proc. Brit. Mach. Vis. Conf.*, 2017.

[9] C. Zhang, G. Lin, F. Liu, J. Guo, Q. Wu, and R. Yao, "Pyramid graph networks with connection attentions for region-based one-shot semantic segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 9587–9595.

[10] X. Yang et al., "BriNet: Towards bridging the intra-class and inter-class gaps in one-shot segmentation," in *Proc. Brit. Mach. Vis. Conf.*, 2020.

[11] H. Wang, X. Zhang, Y. Hu, Y. Yang, X. Cao, and X. Zhen, "Few-Shot semantic segmentation with democratic attention networks," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 730–746.

[12] X. Zhang, Y. Wei, Y. Yang, and T. S. Huang, "SG-one: Similarity guidance network for one-shot semantic segmentation," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3855–3865, Sep. 2020.

[13] C. Zhang, G. Lin, F. Liu, R. Yao, and C. Shen, "CANet: Class-agnostic segmentation networks with iterative refinement and attentive few-shot learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 5217–5226.

[14] B. Yang, C. Liu, B. Li, J. Jiao, and Q. Ye, "Prototype mixture models for few-shot semantic segmentation," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 763–778.

[15] T.-Y. Lin et al., "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 740–755.

[16] K. Nguyen and S. Todorovic, "Feature weighting and boosting for few-shot segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 622–631.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 630–645.

[19] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected CRFs," in *Proc. Int. Conf. Learn. Represent.*, 2015.

[20] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proc. European Conf. Comput. Vis. (ECCV)*, 2018, pp. 801–818.

[21] H. Zhao et al., "PSANet: Point-wise spatial attention network for scene parsing," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 267–283.

[22] J. Fu et al., "Dual attention network for scene segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 3146–3154.

[23] Z. Huang, X. Wang, L. Huang, C. Huang, Y. Wei, and W. Liu, "CCNET: Criss-cross attention for semantic segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 603–612.

[24] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *Proc. ICML Deep Learn. Workshop*, 2015.

[25] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 3630–3638.

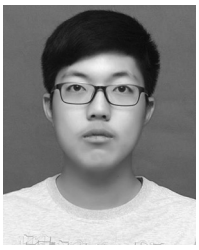
[26] B. Zhang et al., "Mixture distribution graph network for few shot learning," *IEEE Trans. Cogn. Develop. Syst.*, vol. 14, no. 3, pp. 892–901, Sep. 2022.

[27] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-learning with memory-augmented neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1842–1850.

[28] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *Proc. Int. Conf. Learn. Represent.*, 2017.
[29] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1126–1135.
[30] E. Triantafillou et al. "Meta-dataset: A dataset of datasets for learning to learn from few examples," in *Proc. Int. Conf. Learn. Represent.*, 2020.
[31] K. Wang, J. H. Liew, Y. Zou, D. Zhou, and J. Feng, "PANet: Few-shot image semantic segmentation with prototype alignment," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 9197–9206.
[32] Y. Liu, X. Zhang, S. Zhang, and X. He, "Part-aware prototype network for few-shot semantic segmentation," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 142–158.
[33] B. Zhang, J. Xiao, and T. Qin, "Self-guided and cross-guided learning for few-shot segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 8312–8321.
[34] Z. Tian, H. Zhao, M. Shu, Z. Yang, R. Li, and J. Jia, "Prior guided feature enrichment network for few-shot segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 2, pp. 1050–1065, Feb. 2022.
[35] B. Liu, Y. Ding, J. Jiao, X. Ji, and Q. Ye, "Anti-aliasing semantic reconstruction for few-shot semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 9747–9756.
[36] J. Min, D. Kang, and M. Cho, "Hypercorrelation squeeze for few-shot segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 6941–6952.
[37] D. Kang and M. Cho, "Integrative few-shot learning for classification and segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 9979–9990.
[38] Y. Liu, N. Liu, Q. Cao, X. Yao, J. Han, and L. Shao, "Learning non-target knowledge for few-shot semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 11573–11582.
[39] E. Iqbal, S. Safarov, and S. Bang, "MSANet: Multi-similarity and attention guidance for boosting few-shot segmentation," 2022, *arXiv:2206.09667*.
[40] G. E. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.
[41] S. Zagoruyko and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," in *Proc. Int. Conf. Learn. Represent.*, 2017.
[42] T. He, C. Shen, Z. Tian, D. Gong, C. Sun, and Y. Yan, "Knowledge adaptation for efficient semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 578–587.
[43] T. Fukuda, M. Suzuki, G. Kurata, S. Thomas, J. Cui, and B. Ramabhadran, "Efficient knowledge distillation from an ensemble of teachers," in *Proc. Interspeech*, 2017, pp. 3697–3701.
[44] Q. Zhao, S. Lyu, Z. Zhang, T.-B. Xu, and G. Cheng, "Embedded knowledge distillation in depth-level dynamic neural network," 2021, *arXiv:2103.00793*.
[45] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7132–7141.
[46] K. Kim, B. Ji, D. Yoon, and S. Hwang, "Self-knowledge distillation: A simple way for better generalization," 2020, *arXiv:2006.12000*.
[47] L. Zhang, J. Song, A. Gao, J. Chen, C. Bao, and K. Ma, "Be your own teacher: Improve the performance of convolutional neural networks via self distillation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 3713–3722.
[48] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.
[49] K. Rakelly, E. Shelhamer, T. Darrell, A. Efros, and S. Levine, "Conditional networks for few-shot semantic segmentation," in *Proc. Int. Conf. Learn. Represent.*, 2018.
[50] M. Siam, B. N. Oreshkin, and M. Jägersand, "AMP: Adaptive masked proxies for few-shot segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 5248–5257.
[51] W. Liu, C. Zhang, G. Lin, and F. Liu, "CRNet: Cross-reference networks for few-shot segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 4164–4172.
[52] S. Gairola, M. Hemani, A. Chopra, and B. Krishnamurthy, "SimPropNet: Improved similarity propagation for few-shot image segmentation," in *Proc. 29th Int. Joint Conf. Artif. Intell. (IJCAI)*, 2020, pp. 573–579.
[53] G. Li, V. Jampani, L. Sevilla-Lara, D. Sun, J. Kim, and J. Kim "Adaptive prototype learning and allocation for few-shot segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 8334–8343.
[54] Z. Wu, X. Shi, G. Lin, and J. Cai, "Learning meta-class memory for few-shot semantic segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 517–526.
[55] Z. Lu, S. He, X. Zhu, L. Zhang, Y.-Z. Song, and T. Xiang, "Simpler is better: Few-shot semantic segmentation with classifier weight transformer," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 8721–8730.
[56] G.-S. Xie, H. Xiong, J. Liu, Y. Yao, and L. Shao, "Few-shot semantic segmentation with cyclic memory network," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 7273–7282.
[57] M. Everingham, S. M. Eslami, L. Van. Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *Int. J. Comput. Vis.*, vol. 111, pp. 98–136, Jan. 2015.
[58] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Simultaneous detection and segmentation," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 297–312.
[59] T. Hu, P. Yang, C. Zhang, G. Yu, Y. Mu, and C. G. M. Snoek, "Attention-based multi-context guiding for few-shot semantic segmentation," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 8441–8448.

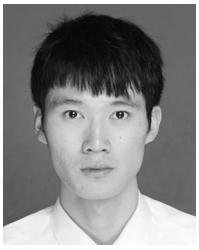
Qi Zhao (Member, IEEE) was born in China, in 1966. She received the Ph.D. degree in communication and information system from Beihang University, Beijing, China, in 2002.

She is a Professor and works with Beihang University. She was with the Department of Electrical and Computer Engineering, University of Pittsburgh, Pittsburgh, PA, USA, as a Visiting Scholar from 2014 to 2015. Since 2016, she has been working on wearable device based first-view image processing and deep learning based image recognition. Her current research interests include few-shot semantic segmentation, medical image processing, object detection, and target tracking.



Binghao Liu received the B.E. degree in electronics and information engineering from Beihang University, Beijing, China, in 2019, where he is currently pursuing the Ph.D. degree with the School of Electronic and Information Engineering.

His research interests include weakly supervised learning, semantic segmentation, and few-shot segmentation.



Shuchang Lyu (Graduate Student Member, IEEE) received the B.E. degree in communication and information from Shanghai University, Shanghai, China, in 2016, and the M.E. degree in communication and information system from the School of Electronic and Information Engineering, Beihang University, Beijing, China, in 2019, where he is currently pursuing the Ph.D. degree with the School of Electronic and Information Engineering.

His research interests include deep learning, image classification, few-shot semantic segmentation, and object detection.

Huojin Chen was born in China, in 1967. He received the bachelor's degree in testing technology and instruments from the Jilin University of Technology, Changchun, China, in 1989, and the master's degree in system engineering from Tianjin University, Tianjin, China, in 1992.

He is an Associate Professor with the College of Art and Design, Beijing University of Technology, Beijing, China. His main research interests focus on computer vision, system engineering, and design management.