Few-Shot Semantic Segmentation with Cyclic Memory Network

Guo-Sen Xie[†,‡*], Huan Xiong[§,‡*], Jie Liu[‡], Yazhou Yao[†], Ling Shao[♮]
[‡]Mohamed bin Zayed University of AI, UAE [♮]Inception Institute of AI, UAE
[†]Nanjing University of Science and Technology, China [§]Harbin Institute of Technology, China

Abstract

Few-shot semantic segmentation (FSS) is an important task for novel (unseen) object segmentation under the data-scarcity scenario. However, most FSS methods rely on uni-directional feature aggregation, e.g., from support prototypes to get the query prediction, and from high-resolution features to guide the low-resolution ones. This usually fails to fully capture the cross-resolution feature relationships and thus leads to inaccurate estimates of the query objects. To resolve the above dilemma, we propose a cyclic memory network (CMN) to directly learn to read abundant support information from all resolution features in a cyclic manner. Specifically, we first generate N pairs (key and value) of multi-resolution query features guided by the support feature and its mask. Next, we circularly take one pair of these features as the query to be segmented, and the rest N-1 pairs are written into an external memory accordingly, i.e., this leave-one-out process is conducted for N times. In each cycle, the query feature is updated by collaboratively matching its key and value with the memory, which can elegantly cover all the spatial locations from different resolutions. Furthermore, we incorporate the query feature re-adding and the query feature recursive updating mechanisms into the memory reading operation. CMN, equipped with these merits, can thus capture cross-resolution relationships and better handle the object appearance and scale variations in FSS. Experiments on PASCAL-5^i and COCO-20^i well validate the effectiveness of our model for FSS.

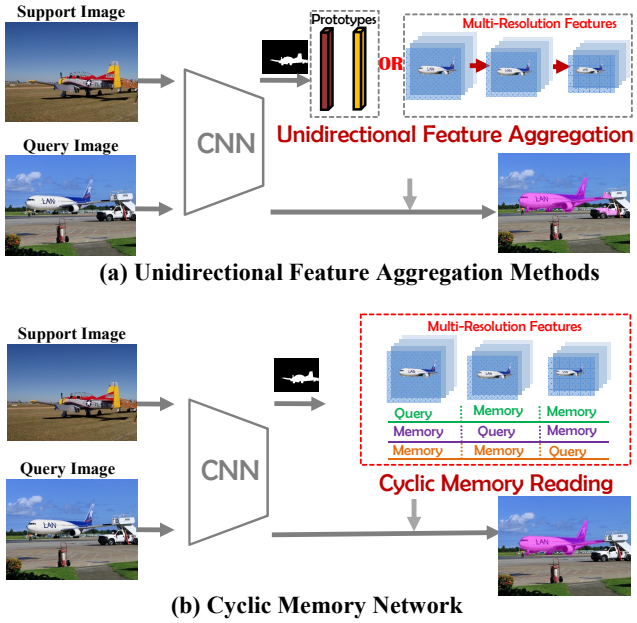

Figure 1. **Illustrations of existing FSS models and our CMN. (a)** Existing FSS methods usually rely on *unidirectional feature aggregation*, e.g., utilizing the support prototypes to get the query prediction or leveraging the high-resolution features to guide the low-resolution ones. However, some *car* regions are wrongly predicted as *airplane*, which is due to the large object variations in the support and query images (the airplanes have different scales and appearance colors). In this case, these *unidirectional* methods fail to capture and overcome the object variations. **(b)** CMN predicts the *airplane* in the query image pretty well, which benefits from our *cyclic memory reading* on the multi-resolution features.

1. Introduction

Training high performance semantic segmentation models [1, 3, 18, 24, 42], based on convolutional neural networks [12, 27, 38, 47], typically requires large amounts of human-annotated training data, e.g., pixel-level annotations are essential for training a desirable segmentation model. However, data annotating by humans is usually costly and labor-intensive. Moreover, these models, almost always, fail to segment novel (unseen) objects, when given very few (one) training images (image) with annotations. To this end, as in conventional zero- and few-shot classification models [28, 36, 37] that aim to mitigate data annotation and novel object recognition issues in the high-level semantic category space, few-shot semantic segmentation (FSS) [25] has become an active research topic for alleviating these issues in the low-level image pixel space, under the object segmentation scenario.

FSS leverages scarce support images with ground-truth

*Corresponding authors.

masks – i.e., labeled support set – to segment unseen objects from a query image, where the focused objects share a common class for both the support and query images. Besides the support images, a large-scale training set (base data) having disjoint classes with the support set is provided for learning transferable knowledge from the seen to the unseen domains. Typically, meta-training [28] is performed on this base data by episode sampling. Here, the constitution of each episode is the same as meta-testing scenario, i.e., a support image set and a query image set. As such, these support images are used as guidance information for the foreground prediction of the query image.

Extensive progresses have been achieved in FSS [7, 21, 26, 32], and most of them utilize a two-branch metric-based network architecture: one for coping with the support images and the other for the query image. For a considered class, the support branch outputs its global and/or local prototypes [9, 17, 39, 46], e.g., by masked average-pooling on the labeled support feature maps. Further, the query branch takes these support prototypes as guidance to segment the query objects by a location-wise matching between query feature in each location and these prototypes. In this way, we can achieve the support guided prediction maps for the query, which however, is an unidirectional feature aggregation and thus it is hard to fully capture the object variations based on these limited support guidance (Fig. 1(a)). Recently, some works [31, 41, 44] explore a dense matching scheme – i.e., non-local attention variants [34] – from support to query images, which can alleviate the object variation issue to some extent, however, these methods are essentially still unidirectional at the prediction layer or at most bi-directional for exchanging information between support-query features at the middle layers. As in common semantic segmentation task, the ideal utilizations of multi-resolution features [4, 16, 30] are the key for achieving accurate segmentation results. To achieve so, some works [30,44] adopt feature pyramid fusion in FSS for enhancing the original features, however, the interactions between these multi-resolution features are unidirectional and/or from high-resolution features to low-resolution ones (Fig. 1(a)), which fails to fully capture the cross-resolution feature relationships.

In this paper, to address the aforementioned issues, we propose a novel cyclic memory network (CMN) [14, 22] (§3.3 and Fig. 2) by directly learning to read abundant support information from all resolution features for tackling FSS. Specifically, we generate K pairs (key and value) of multi-resolution query features guided by the support feature and its mask. The intuitions under our framework lie in that (1) Given each pixel in a supposed query feature having a specific resolution, CMN can explicitly access and read all the other features with different resolutions, which are served as comprehensive guidances to distinguish the considered pixel, i.e., belonging to a foreground or a background. As shown in the second row of the allocation table in Fig. 1(b), when taking the medium resolution as query, the large and small resolutions (memory) are guidances. (2) Contrast to unidirectional feature aggregation of previous methods (Fig. 1(a)), CMN circularly takes each resolution feature as the query one and the rest features are fed into to the memory, which can thus fully exploit the cross-resolution relationships and handle the object appearance and scale changes much better than the previous methods. As such, a precise prediction of the *airplane* in the query image is achieved in Fig. 2(b). Since FSS is a pixel-level prediction task, to reserve and transfer more structural context for desirable unseen object segmentation, we further incorporate the query feature re-adding and the query feature recursive updating mechanisms (Fig. 3) into the memory reading operation. To sum up, our contributions are:

- We propose a cyclic memory network (CMN) which circularly takes the cross-resolution features as memory guidance to precisely segment the supposed query feature, for tackling the FSS task. To the best of our knowledge, we are the first to model FSS as a memory network framework.
- We introduce the query feature re-adding and the query feature recursive updating mechanisms into the memory reading operation of CMN, which have improved the performance of CMN significantly.
- We achieve state-of-the-art performances under mean-IoU and competitive results under FB-IoU on two FSS benchmarks.

2. Related Work

Memory Networks. Neural Turing Machine [10] is the pioneer work of using memory network to solve problems of sorting and copying. Modern end-to-end memory networks [14, 19, 29] are initially designed for tackling natural language processing tasks by additionally utilizing an external memory for information storing and reading. Later on, memory networks are gradually applied to vision tasks, such as video object segmentation [22], object tracking [40], and movie story understanding [20]. Typically, the encoded key of the query feature is used to measure its similarity with the stacked keys of the memory, and the value of the memory is updated by attending these similarities. Finally, the fused values are returned for different tasks. In this paper, we extend the standard memory network into a cyclic variant, i.e., CMN, which is suitable for FSS task and has its own advantages: (i) Based on the multi-resolution query features (implicitly containing information of the support features), we circularly take the feature of each resolution as query, and the rest as memory; this can thus fully capture the cross-resolution relationships, and it is a special design for FSS, which is greatly different from other memory-based tasks. (ii) Our framework is

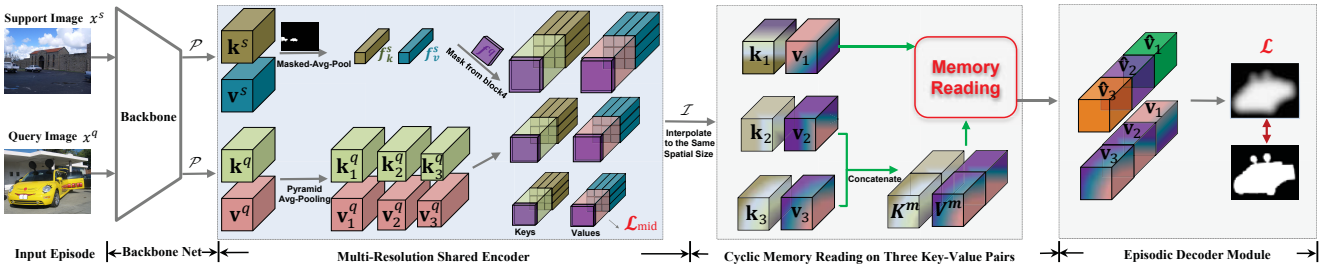

Figure 2. **CMN architecture by illustrating three resolutions.** An input episode, constituting of a support image x^s and a query image x^q, is first fed into CMN (§3.3). Then, the *Multi-Resolution Shared Encoder* is leveraged to generate three pairs of *keys* and *values*, which are interpolated to have the same spatial sizes, denoted as $\mathcal{V} = \{(\mathbf{k}_l, \mathbf{v}_l)\}_{l=1}^{3}$. Further, three times of *cyclic memory reading* –in the figure, one cycle is shown with $(\mathbf{k}_1, \mathbf{v}_1)$ and $\{(\mathbf{k}_l, \mathbf{v}_l)\}_{l=2}^{3}$ as key and memory, respectively – are performed on $\mathcal{V}$ for capturing the cross-resolution relationships, which return reinforced value features $\{\hat{\mathbf{v}}_l\}_{l=1}^{3}$. Finally, both $\{\hat{\mathbf{v}}_l\}_{l=1}^{3}$ and the original $\{\mathbf{v}_l\}_{l=1}^{3}$ are repectively concatenated and taken as inputs to the *episodic decoder module* for segmenting the query object. $\mathcal{L}$ and $\mathcal{L}_{\text{mid}}$ (§3.4) are BCE losses for training CMN.

performed in an episode-based meta-learning scheme. At each episode, the memory is updated as the corresponding multi-resolution features which are obtained by the current support-query features. By contrast, all previous methods leverage memory networks in a fully supervised setting.

Semantic Segmentation. Since the success of fully convolutional neural networks [18], great progress has been made in the semantic segmentation field [18], which aims to classify each pixel to a specific class. Prevailing segmentation models include Deeplab [3], SegNet [1], UNet [24], and PSPNet [47]. Besides, dilated convolutions [43], encoder-decoder structure [24], ASPP [3], and skip connection [3] are widely-used strategies for achieving good performances. Nevertheless, the above mentioned segmentation models are relying on large amounts of pixel-level annotations for training, and the trained models always fail to segment unseen class objects. We aim to tackle the unseen object segmentation problem by utilizing CMN.

Few-Shot Semantic Segmentation. Two-branch metric-based networks are usually adopted for FSS. For instance, OSLSM [25] consists of a conditional branch for generating classifier weights and a segmentation branch for outputting the prediction mask. Inevitably, under the metric-based learning paradigm, prototypes are obtained by squeezing the features in the foreground of the support images, which are taken as guidance to segment the query images. Other representative methods for FSS are PL [7], coFCN [23], SG-One [46], A-MCG [13], FWB [21], PANet [32], CANet [45], CRNet [16], PPN [17] and PMM [39]. The essential differences of these methods are the ways of obtaining these prototypes and the ways of using them. The common aspect of these approaches lies in the unidirectional feature aggregation process for achieving the query prediction mask, which thus cannot well capture the object appearance variances. Inspired by the conventional semantic segmentation, some works [30, 44] explore the utilizations of multi-resolution features, which, however, are intrinsically using high-resolution features to guide the low-resolution ones or using support feature to guide multi-resolution query ones. As such, the cross-resolution relationships among support-query images are still out of full use.

In the FSS literatures, existing works that are most related to ours are PFENet [30] and BriNet [41]. For preserving the maximum domain generalization capability on unseen classes, these two works (i) have fixed the backbone weights and (ii) utilize both mid- and high-level features to construct discriminative support/query features. To pursue a desirable generalizable model, we adopt the aforementioned experiences for building CMN. By contrast, CMN is the first memory network in FSS field to alleviate the object variations among support-query images in a cyclic manner.

3. Methodology

3.1. Definitions

In FSS, a base training set $\mathcal{D}_{\text{train}}$ and a test set $\mathcal{D}_{\text{test}}$ with non-overlapping classes are provided for meta-learning and testing, respectively. In the training stage, multiple episodes (sub-tasks) are sampled from $\mathcal{D}_{\text{train}}$. By contrast, we randomly sample episodes from $\mathcal{D}_{\text{test}}$ for meta-testing. The constitution of each episode are a support set $\mathcal{S}$ and a query set $\mathcal{Q}$. Specifically, under K-shot setting, we have $\mathcal{S} = \{(x_i^s, m_i^s)\}_{i=1}^{K}$ constituting of K image-mask pairs, where, (x_i^s, m_i^s) is the ith (support) image-mask pair for a certain class c. Let $\mathcal{Q} = \{(x^q, m^q)\}$ with x^q and m^q being the query image and corresponding ground-truth binary mask sampled from the same class c. In this way, each episode $(\mathcal{S}, \mathcal{Q})$ is focused on a specific class c and producing a prediction mask $\hat{m}^q$ for the contained query image, by feeding $\mathcal{S}$ and x^q to the model. Next, the binary cross-entropy loss $\text{BCE}(\hat{m}^q, m^q)$ is calculated for updating the model weights. Once the model is trained, we further perform meta-testing by sampling multiple episodes,

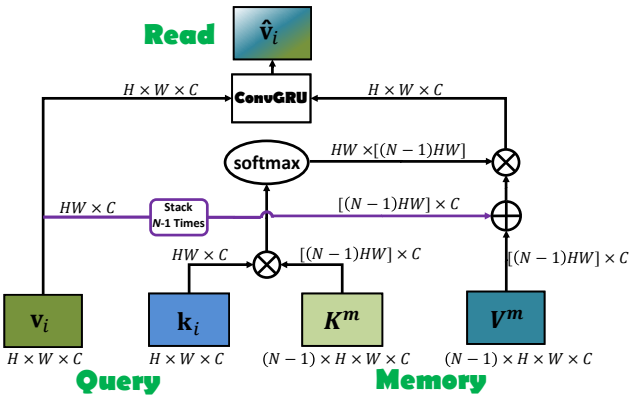

Figure 3. Implementation of one cycle memory reading, where $\otimes$ and $\oplus$ indicate matrix multiplication and addition, respectively.

$\{(\mathcal{S}_i^{\text{ts}}, \mathcal{Q}_i^{\text{ts}})\}_{i=1}^{N_{\text{ts}}}$, from $\mathcal{D}_{\text{test}}$. In the following, we take the 1-shot setting (K=1 in $\mathcal{S}$) for an easy illustration of CMN.

3.2. Overview

As in Fig. 2, CMN is built based on the memory network in a cyclic manner. Under the 1-shot setting, for each episode $(\mathcal{S}, \mathcal{Q})$ with $\mathcal{S} = \{(x^s, m^s)\}$, we first feed the support image-mask pair (x^s, m^s) and the query image x^q (to be segmented) to the shared encoder for obtaining N pairs of multi-resolution fused key and value feature maps, denoted as $\mathcal{V} = \{(\mathbf{k}_i, \mathbf{v}_i)\}_{i=1}^{N}$. Each resolution key-value feature in $\mathcal{V}$ is circularly considered as the query to be segmented, and the rest $N-1$ resolution features are written as memory. The keys and values of query and memory are further fed into our memory reading module with query feature re-adding and query feature recursive updating mechanisms. In this way, every pixel of the key feature map for the query is densely matched over the key of the memory to output the similarity scores with all the spatial locations from different resolutions. These scores in turn are used to summarize the long-range and cross-resolution information from the memory, and thus leading to a reinforced value feature $\hat{\mathbf{v}}_i$ for the query $\mathbf{v}_i$. The above memory reading and writing process is repeated for N times in a cyclic manner. As such, we finally obtain the reinforced features of N resolutions, termed as $\{\hat{\mathbf{v}}_i\}_{i=1}^{N}$, which are further taken by the episodic decoder and used to reconstruct the ground-truth mask m^q of the query image x^q.

3.3. Cyclic Memory Network

Multi-Resolution Shared Encoder. Different from previous memory networks, since we aim to explore the relationships among different resolution features, we generate multi-resolution (key and value) feature maps based on a shared encoder for facilitating the cyclic memory reading. Specifically, given the support image-mask pair (x^s, m^s) and the query image x^q as inputs, the encoder outputs N pairs of key-value feature maps w.r.t. N resolutions, by designing learnable convolutions, and pyramid pooling operations based on some specific backbones. In our implementation of the encoder, we mainly follow the golden routines validated by [30, 41]. Taking ResNet having four group layers (block1-4) as an example, we use feature maps from both block2 and 3 and compress them to key and value feature maps with a pre-defined channel dimension C by a 1×1 convolution. Meanwhile, the mask-pooled support feature and initial query feature from block4 are used to obtain a prediction mask, which is also fused into the multi-resolution key-value features. Further, as in previous work [46], a masked average-pooled global vector is further used to assist the construction of multi-resolution features. Finally, we fix the weights of the backbone during meta-training, which is also adopted by [30, 41, 45] for achieving a better generalization on unseen classes.

Specifically, taking the largest resolution as an example, and suppose we have achieved the initial key and value feature maps for (x^s, x^q) based on the compressed features of block2 and 3 (this operation is denoted as $\mathcal{P}$):

$$\mathbf{k}^s, \mathbf{v}^s = \mathcal{P}(x^s) \in \mathbb{R}^{H\times W\times C},\ \mathbf{k}^q, \mathbf{v}^q = \mathcal{P}(x^q) \in \mathbb{R}^{H\times W\times C}, \tag{1}$$

where $H \times W$ is the used largest resolution, meanwhile, H, W, and C are the height, width, and channel dimension, respectively. To get the final multi-resolution key and value features, pyramid average-pooling [47] is first conducted on $\mathbf{k}^q$, $\mathbf{v}^q$ to generate N new resolution features $\mathcal{V}^q = \{(\mathbf{k}_i^q, \mathbf{v}_i^q)\}_{i=1}^{N}$ with $\mathbf{k}_i^q$ $(\mathbf{v}_i^q) \in \mathbb{R}^{H_i\times W_i\times C}$. We suppose $(\mathbf{k}^q, \mathbf{v}^q)$ is included in $\mathcal{V}^q$, and $H_i{>}H_j$, $W_i{>}W_j$ when $i{<}j$. Furthermore, we denote the masked average-pooled global vector w.r.t. $(\mathbf{k}^s, \mathbf{v}^s)$ guided by m^s as follows:

$$(f_k^s, f_v^s) = \text{masked\_avg\_pool}((\mathbf{k}^s, \mathbf{v}^s), m^s) \in \mathbb{R}^{1\times1\times C}. \tag{2}$$

Notably, we only construct one pair of global vector under the support key-value features with the largest resolution. Inspired by [30], we further use the features of block4 $\mathbf{f}_{b4}^s$ and $\mathbf{f}_{b4}^q$ w.r.t. x^s and x^q to get a prediction mask f^q $(\in \mathbb{R}^{H\times W})$ for the key and value of the query image. The calculation of each element in $f^q(x, y)$ is obtained by finding the maximum cosine similarity value among the feature vectors in the same location considered in $\mathbf{f}_{b4}^q$ and all the foreground locations of $\mathbf{f}_{b4}^s$. Finally, we can construct the multi-resolution key and value feature maps $\mathbf{k}_i, \mathbf{v}_i \in \mathbb{R}^{H\times W\times C}$ $(i = 1, \cdots, N)$ as follows:

$$\begin{aligned}\mathbf{k}_i &= \mathcal{I}_{H\times W}(\mathcal{C}(\mathbf{k}_i^q \oplus \mathcal{E}_{H_i\times W_i}(f_k^s) \oplus \mathcal{I}_{H_i\times W_i}(f^q))),\\ \mathbf{v}_i &= \mathcal{I}_{H\times W}(\mathcal{C}(\mathbf{v}_i^q \oplus \mathcal{E}_{H_i\times W_i}(f_v^s) \oplus \mathcal{I}_{H_i\times W_i}(f^q))),\end{aligned} \tag{3}$$

where $\oplus$ is the concatenating, $\mathcal{I}_{x\times y}$ interpolates the input to a size of $x \times y$, $\mathcal{E}_{x\times y}$ expands the input vector to a size of $x\times y$, and $\mathcal{C}$ consists of one 1×1 and two 3×3 convolutions. Here, we use $\mathcal{I}$ to interpolate the key and value features to the same spatial and channel sizes, however, they still preserve different resolutions. This operation aims to facilitate the afterward memory reading operations.

Cyclic Memory Reading. Since we have obtained N pairs of key and value feature maps reflecting different resolution properties w.r.t. the same query image, we circularly take one pair of them as supposed query to be segmented, and the rest N-1 pairs are written into an external memory for each episode training. This cyclic memory reading and writing process is conducted for N times. Specifically, in one cycle, suppose $(\mathbf{k}_i, \mathbf{v}_i)$ is the key and value features serving as the query, and $\mathcal{V}_{i/N} = \{(\mathbf{k}_l, \mathbf{v}_l)\}_{l=1}^{i-1} \cup \{(\mathbf{k}_l, \mathbf{v}_l)\}_{l=i+1}^{N}$ are the keys and values in the memory. We first stack the keys and values of the rest $N-1$ resolutions from the memory, leading to $K^m \in \mathbb{R}^{(N-1)\times H\times W\times C}$ and $V^m \in \mathbb{R}^{(N-1)\times H\times W\times C}$, where N-1 can be seen as the frame number of the memory. Further, $(\mathbf{k}_i, \mathbf{v}_i)$ and K^m, V^m are taken as inputs to the memory reading module, and the outputs are the reinforced value $\hat{\mathbf{v}}_i$.

During this memory reading, we leverage the key maps $\mathbf{k}_i$ and K^m of the query and memory to calculate the similarities between all locations of them, and get the following similarity map:

$$\mathbf{e}_i = \bar{\mathbf{k}}_i \overline{K^m}^{\mathsf{T}} \in \mathbb{R}^{HW\times[(N-1)HW]}, \tag{4}$$

where $\bar{\mathbf{k}}_i \in \mathbb{R}^{HW\times C}$ and $\overline{K^m} \in \mathbb{R}^{[(N-1)HW]\times C}$ are the reshaped maps of $\mathbf{k}_i$ and K^m, respectively. Based on $\mathbf{e}_i$, the value of the memory is retrieved and initially returned as:

$$\mathbf{g}_i = \mathrm{softmax}(\mathbf{e}_i)\overline{V^m} \in \mathbb{R}^{HW\times C}, \tag{5}$$

where $\mathrm{softmax}(\cdot)$ is a row-wise softmax normalization and $\overline{V^m} \in \mathbb{R}^{[(N-1)HW]\times C}$ is the reshaped map of V^m. In this way, $\mathbf{g}_i$ will contain the long-range and cross-resolution information from the memory. Under the context of our FSS task, since the stacked value maps ($V^m = [\mathbf{v}_1; \cdots; \mathbf{v}_{i-1}; \mathbf{v}_{i+1}; \cdots; \mathbf{v}_N]$) in the memory are essentially different resolution variants compared with the current query value map $\mathbf{v}_i$, we propose a re-adding mechanism by adding a stacked $\mathbf{v}_i$ with N-1 times (denoted as $V_i^q \in \mathbb{R}^{(N-1)\times H\times W\times C}$) into Eq. (5). The final returned value from the memory is updated as:

$$\mathbf{g}_i = \mathrm{softmax}(\mathbf{e}_i)(\overline{V^m} + \overline{V_i^q}) \in \mathbb{R}^{HW\times C}, \tag{6}$$

where $\overline{V_i^q} \in \mathbb{R}^{[(N-1)HW]\times C}$ is reshaped by V_i^q. Experimental result shows performance gains by our simple re-adding mechanism (Table 5).

Finally, instead of concatenating $\mathbf{v}_i$ and the reshaped $\mathbf{g}_i$ (still denoted as $\mathbf{g}_i$) followed by the compressing operation, we propose the query feature recursive updating mechanism, which takes $\mathbf{v}_i$ and $\mathbf{g}_i$ as inputs and explores the recursive relationships among them. In our implementation, we adopt ConvGRU [2] to get the final output of the memory reading module, as following:

$$\hat{\mathbf{v}}_i = \mathcal{U}_{\mathrm{GRU}}(\mathbf{v}_i, \mathbf{g}_i) \in \mathbb{R}^{H\times W\times C}. \tag{7}$$

Experimental result shows significant performance gains by this query feature recursive updating mechanism (Table 5).

Episode Decoder. After N times of memory reading, we obtain the reinforced value features $\{\hat{\mathbf{v}}_i\}_{i=1}^N$ w.r.t. these N resolutions. Together with the original value features $\{\mathbf{v}_i\}_{i=1}^N$, we have two group of multi-resolution features in total. To reconstruct the GT mask m^q of the query image x^q in the current episode, we leverage the following sequential encoding to each group of the above features:

$$\begin{aligned}\tilde{m}_1^q &= \mathcal{P}_{\mathrm{cls}}(\mathcal{P}_{\mathrm{ASPP}}(\mathcal{P}_{\mathrm{res\_conv}}(\mathbf{v}_1 \oplus \mathbf{v}_2 \oplus \cdots \oplus \mathbf{v}_N))) \in \mathbb{R}^{h\times w\times 2},\\ \tilde{m}_2^q &= \mathcal{P}_{\mathrm{cls}}(\mathcal{P}_{\mathrm{ASPP}}(\mathcal{P}_{\mathrm{res\_conv}}(\hat{\mathbf{v}}_1 \oplus \hat{\mathbf{v}}_2 \oplus \cdots \oplus \hat{\mathbf{v}}_N))) \in \mathbb{R}^{h\times w\times 2},\end{aligned} \tag{8}$$

where h and w are the height and width of m^q; $\mathcal{P}_{\mathrm{res\_conv}}$, $\mathcal{P}_{\mathrm{ASPP}}$ and $\mathcal{P}_{\mathrm{cls}}$ are denoted as the episode decoder, and their detailed architectures are in §4.2. The final predicted mask is taken as the mean of $\tilde{m}_1^q$ and $\tilde{m}_2^q$, i.e., $\hat{m}^q = (\tilde{m}_1^q + \tilde{m}_2^q)/2$. Our CMN is trained in an episodic manner. As such, we term this decoding process as an episode decoder. The modules in Eq. (3)-(8) for CMN are all differentiable and we train CMN in an end-to-end manner.

3.4. Training Objective

Until now, §3.3 solely illustrates the paradigm of CMN training under one episode of a 1-shot case. For the batch-based training with N_b episodes ($\{(\mathcal{S}_i, \mathcal{Q}_i)\}_{i=1}^{N_b}$), where $\mathcal{Q}_i = (x_i^q, m_i^q)$. Using Eq. (8), we achieve the predicted average mask $\hat{m}_i^q \in \mathbb{R}^{h\times w\times 2}$ for every query x_i^q. Further, we adopt the binary cross entropy (BCE) loss among $\hat{m}_i^q$ and m_i^q for training CMN. This BCE loss is as follows:

$$\mathcal{L} = \frac{1}{N_b}\sum_{i=1}^{N_b} \mathrm{BCE}(\hat{m}_i^q, m_i^q). \tag{9}$$

Inspired by [30], given the multi-resolution value features (denoted as $\{\tilde{\mathbf{v}}_i\}_{i=0}^N$) before the $\mathcal{I}_{H\times W}$ operation (Eq. (3)), we use N additional branches to predict the query masks for these N resolutions. Notably, these N branches are implemented as one 3×3 and one 1×1 convolutions. As such, we have another mid-level BCE loss:

$$\mathcal{L}_{\mathrm{mid}} = \frac{1}{N_b \times N}\sum_{i=1}^{N_b}\sum_{k=1}^{N} \mathrm{BCE}(\mathcal{P}_{\mathrm{mid}}^k(\tilde{\mathbf{v}}_k), m_i^q), \tag{10}$$

where $\mathcal{P}_{\mathrm{mid}}^k(\cdot)$ indicates the encoding function for the kth branch. To this end, our final training objective is:

$$\mathcal{L}_{\mathrm{final}} = \mathcal{L} + \mathcal{L}_{\mathrm{mid}}. \tag{11}$$

4. Experiments

4.1. Settings

Datasets. Two golden FSS datasets – i.e., PASCAL-5^i [25] and COCO-20^i [21] – are used for the evaluation of

Methods	Backbone	mean-IoU (1-shot)					FB-IoU	mean-IoU (5-shot)					FB-IoU
		fold-0	fold-1	fold-2	fold-3	mean	(1-shot)	fold-0	fold-1	fold-2	fold-3	mean	(5-shot)
OSLSM BMVC'17 [25]	VGG-16	33.6	55.3	40.9	33.5	40.8	61.3	35.9	58.1	42.7	39.1	43.9	61.5
co-FCN ICLRW'18 [23]	VGG-16	31.7	50.6	44.9	32.4	41.1	60.1	37.5	50.0	44.1	33.9	41.4	60.2
AMP ICCV'19 [26]	VGG-16	41.9	50.2	46.7	34.7	43.4	62.2	41.8	55.5	50.3	39.9	46.9	63.8
SG-One TCYB'19 [46]	VGG-16	40.2	58.4	48.4	38.4	46.3	63.1	41.9	58.6	48.6	39.4	47.1	65.9
PANet ICCV'19 [32]	VGG-16	42.3	58.0	51.1	41.2	48.1	66.5	51.8	64.6	59.8	46.5	55.7	70.7
CANet CVPR'19 [45]	ResNet-50	52.5	65.9	51.3	51.9	55.4	66.2	55.5	67.8	51.9	53.2	57.1	69.6
PGNet ICCV'19 [44]	ResNet-50	56.0	66.9	50.6	50.4	56.0	69.9	57.7	68.7	52.9	54.6	58.5	70.5
FWB ICCV'19 [21]	ResNet-101	51.3	64.5	56.7	52.2	56.2	-	54.8	67.4	62.2	55.3	59.9	-
PMMs ECCV'20 [39]	ResNet-50	52.0	67.5	51.5	49.8	55.2	-	55.0	68.2	52.9	51.1	56.8	-
PPNet ECCV'20 [17]	ResNet-50	47.8	58.8	53.8	45.6	51.5	-	58.4	67.8	**64.9**	56.7	62.0	-
DAN ECCV'20 [31]	ResNet-101	54.7	68.6	**57.8**	51.6	58.2	71.9	57.9	69.0	60.1	54.9	60.5	72.3
SimPropNet IJCAI'20 [9]	ResNet-50	54.9	67.3	54.5	52.0	57.2	73.0	57.2	68.5	58.4	56.1	60.0	72.9
BriNet BMVC'20 [41]	ResNet-50	56.5	67.2	51.6	53.0	57.1	-	-	-	-	-	-	-
PFENet TPAMI'20 [30]	ResNet-50	61.7	69.5	55.4	56.3	60.8	73.3	63.1	**70.7**	55.8	57.9	61.9	73.9
Baseline	ResNet-50	62.1	68.2	55.3	53.8	59.9	71.7	63.3	68.7	55.1	55.3	60.6	71.8
CMN	ResNet-50	**64.3**	**70.0**	57.4	**59.4**	**62.8**	72.3	**65.8**	70.4	57.6	**60.8**	**63.7**	72.8

Table 1. Experimental comparisons under 1-shot and 5-shot settings on PASCAL-5^i. mean-IoU of each fold and the averaged mean-IoU (FB-IoU) of four folds are shown.

Methods	Backbone	mean-IoU (1-shot)					FB-IoU	mean-IoU (5-shot)					FB-IoU
		fold-0	fold-1	fold-2	fold-3	mean	(1-shot)	fold-0	fold-1	fold-2	fold-3	mean	(5-shot)
PANet ICCV'19 [32]	VGG-16	-	-	-	-	20.9	59.2	-	-	-	-	29.7	63.5
FWB ICCV'19 [21]	VGG-16	18.4	16.7	19.6	25.4	20.0	-	20.9	19.2	21.9	28.4	22.6	-
FWB ICCV'19 [21]	ResNet-101	17.0	18.0	21.0	28.9	21.2	-	19.1	21.5	23.9	30.1	23.7	-
PMMs ECCV'20 [39]	ResNet-101	29.3	34.8	27.1	27.3	29.6	-	33.0	40.6	30.1	33.3	34.3	-
DAN ECCV'20 [31]	ResNet-101	-	-	-	-	24.4	62.3	-	-	-	-	29.6	63.9
PPNet ECCV'20 [17]	ResNet-50	28.1	30.8	29.5	27.7	29.0	-	39.0	40.8	37.1	37.3	38.5	-
BriNet BMVC'20 [41]	ResNet-50	32.9	36.2	37.4	30.9	34.4	-	-	-	-	-	-	-
PFENet TPAMI'20 [30]	VGG-16	33.4	36.0	34.1	32.8	34.1	60.0	35.9	40.7	38.1	36.1	37.7	61.6
PFENet TPAMI'20 [30]	ResNet-101	34.3	33.0	32.3	30.1	32.4	58.6	38.5	38.6	38.2	34.3	37.4	61.9
Baseline	ResNet-50	34.2	39.6	35.8	34.3	36.0	60.6	37.7	45.7	38.0	37.5	39.7	62.7
CMN	ResNet-50	**37.9**	**44.8**	**38.7**	**35.6**	**39.3**	61.7	**42.0**	**50.5**	**41.0**	**38.9**	**43.1**	63.3

Table 2. Experimental comparisons under 1-shot and 5-shot settings on COCO-20^i. mean-IoU of each fold and the averaged mean-IoU (FB-IoU) of four folds are shown.

CMN. PASCAL-5^i is constructed based on PASCAL VOC 2012 [8, 11], and COCO-20^i is built from MSCOCO [15]. As in [25] and [21], we split the data into four folds based on the number of the total classes on the two datasets. Then, the cross-validation results on each fold are reported. Specifically, for each evaluation on the two datasets, 15 and 60 object classes are taken as the training set with the rest 5 and 20 object classes as the test set, respectively. For meta-testing, 1,000 episodes are randomly sampled from the test set for evaluating their metrics.

Metrics. As in [25,32,33,35], mean-IoU and FB-IoU are used metrics for evaluating CMN. Specifically, mean-IoU is achieved by taking the average of the intersection-over-unions (IoUs) over different foreground classes of the test set. Meanwhile, the mean-IoU of each fold and averaged mean-IoU of four folds are reported. FB-IoU indicates the foreground and background IoU, and all object classes are taken as one single foreground class in the test set. Further, FB-IoU is obtained by taking the average of the IoUs of the foreground and background classes. In the literatures, compared with FB-IoU, the mean-IoU is widely taken as the key metric for FSS, this is because the performance bias of some classes can be alleviated by considering the differences of all classes.

K-Shot Evaluation. For the K-shot case (K>1), like [5, 21, 41, 45], a feature-level early fusion strategy [23] is adopted by averaging the support image features to achieve a single fused support feature. By doing so, the afterward operations and evaluations are the same as the 1-shot case.

4.2. Implementation Details

CMN is implemented by using Pytorch and models are trained on a Tesla V100 GPU in a meta-learning manner. The batch size N_b in Eq. (10) for PASCAL-5^i and COCO-20^i is 4 and 8, respectively. For CMN training, the SGD optimizer is adopted, the model is trained for 100 epochs on two datasets, and the learning rate is 0.0025 and 0.005 for PASCAL-5^i and COCO-20^i, respectively. As in [30, 41], 'poly' strategy is also used for adjusting the lr.

VGG-16 [27], ResNet-50 [12] and ResNet-101 [12] are adopted to carry out the experiments on the two datasets used. As in [9, 30, 41, 45], the used backbones are initialized from the pre-trained models on ImageNet [6] and their

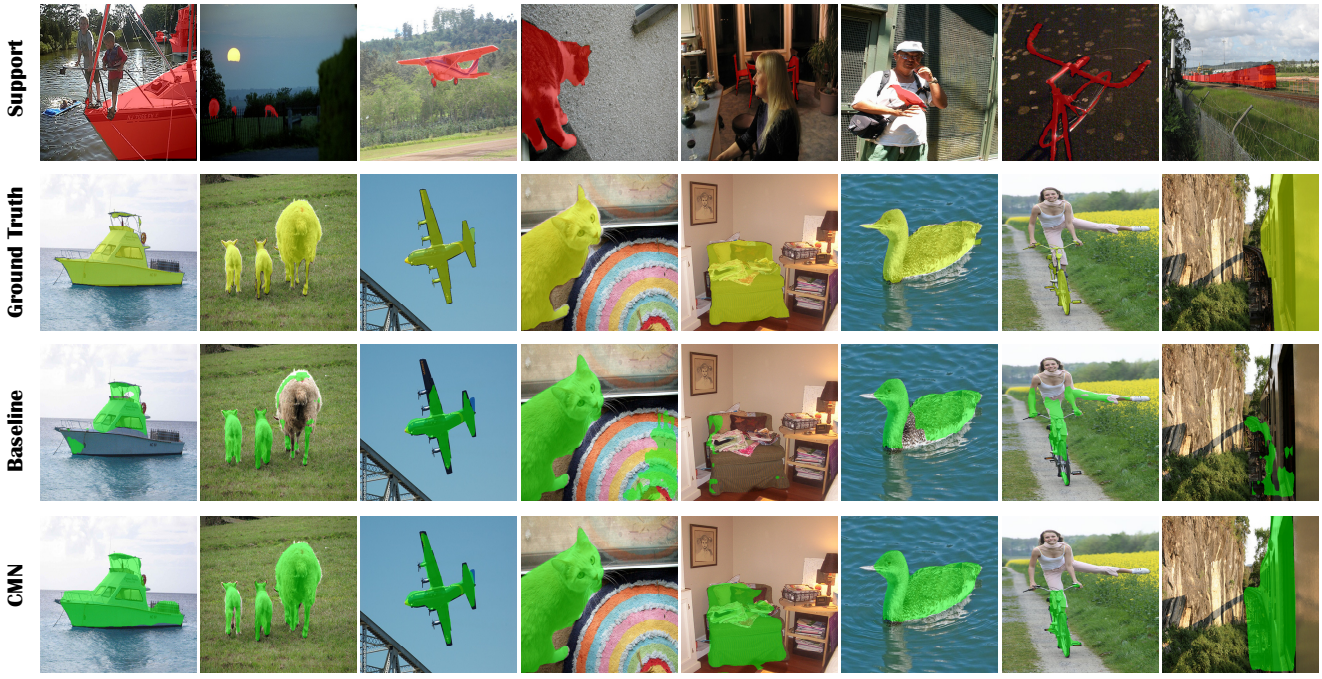

Figure 4. Segmentation examples on unseen classes for the PASCAL-5^i dataset. Specifically, the first row is the support images with ground-truths (GT) marked as red, the second row is the query images with GT (yellow masks), and the third and fourth rows are the predictions of the baseline and CMN respectively.

weights are fixed. In this way, the generalization can be preserved as much as possible. The channel dimension C in Eq. (1) is 256 for all experiments. In addition, dilated convolutions are used for ensuring the size of the key and value feature maps after block2 to be 1/8 of the original input. The input image size is set to 473×473 for all backbones, thus leading to the largest key and value feature maps with a size of 60×60, i.e., H=W=60 in Eq. (1). We use three resolutions of [60×60, 15×15, 8×8] in Eq. (3) for two datasets, which means N=3 in our experiments. Moreover, the number of output channel is set to 256 in the convolution components of $\mathcal{C}$ in Eq. (3). The architecture details of the decoder functions in Eq. (8) are: $\mathcal{P}_{\text{res\_conv}}$: two 1×1 conv of 256 output channels, followed by two 3×3 conv of 256 channel with residual connections, and finally another two 3×3 conv of 256 channels with residual connections. $\mathcal{P}_{\text{ASPP}}$: atrous spatial pyramid pooling with a 1×1 conv without dilation and 3×3 conv with dilation rates 6, 12, 18. $\mathcal{P}_{\text{cls}}$: a 3×3 conv of 256 channels and a 1×1 conv of two channels.

4.3. Comparison with State-of-the-Arts

Our CMN is compared with all the FSS methods under the metrics of mean-IoU and FB-IoU on the two datasets. Different from [44, 45] which leverage multi-scale testing, single scale meta-testing is conducted for CMN. In the used datasets, spatial regions of most foreground objects are small in the whole image. As such, the calculated number of FB-IoU usually benefits from the background regions, which makes it not a convincing metric to evaluate the performances of the models. Nevertheless, we also report the averaged FB-IoU over four folds for showing comprehensive references.

PASCAL-5^i. Table 1 presents the mean-IoU and FB-IoU under 1-shot and 5-shot for all the compared methods under the same testbed. It can be seen that **(i)** The averaged mean-IoUs under four folds are consistently better than the compared methods, which validate the effectiveness of CMN. **(ii)** CMN performs much better than the baseline method that uses the same multi-resolution features, yet without the cyclic memory reading module. **(iii)** Although, the FB-IoUs of CMN are not the best one, we still achieve competitive results compared with the counterpart methods.

COCO-20^i. From Table 2, we conclude that state-of-the-art results on averaged mean-IoUs under both 1-shot and 5-shot settings are achieved on COCO-20^i. For example, under ResNet-50 backbone and the same training/test protocols, **(i)** CMN is better than BriNet [41] by 3.3 on 1-shot setting under the averaged mean-IoU. **(ii)** CMN performs much better than the strong baseline methods. It means that our CMN model has captured the cross-resolution relations by the memory reading module, which further boosts the FSS performances.

4.4. Ablation Study

We take PASCAL-5^i as an example dataset. Unless specified, we adopt averaged mean-IoUs over the first two folds

5-shot testing	mean-IoU	FB-IoU
1-shot baseline	62.8	72.3
Feature-Avg	**63.7**	**72.8**
Mask-Avg	63.3	72.5
Mask-OR	62.9	71.9

Table 3. Comparisons of 5-shot feature fusion.

Backbone	mean-IoU	
	1-shot	5-shot
VGG-16	63.0	64.5
ResNet-50	**67.2**	**68.1**
ResNet-101	66.0	67.3

Table 4. Effects of different backbones.

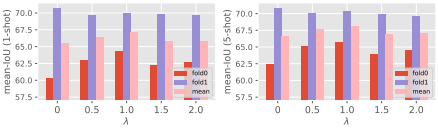

Figure 5. mean-IoUs under different coefficient of $\mathcal{L}_{\text{mid}}$.

Models	mean-IoU	
	1-shot	5-shot
CMN *w* `Memory Value`	66.2	66.7
CMN *w* `Query Value`	66.3	67.7
CMN *w* `Concatenate`	65.8	66.3
CMN *w* `Compress`	66.0	66.4
Full CMN	**67.2**	**68.1**

Table 5. Effects of query feature re-adding and recursive updating mechanisms.

N	mean-IoU	
	1-shot	5-shot
1	N/A	N/A
2	66.0	66.5
3	**67.2**	**68.1**
4	66.9	67.6

Table 6. Effects of feature resolution number N.

– i.e., fold-0 and fold-1 – under 1-shot and 5-shot settings for all the ablation experiments.

Feature Resolution Number N. The number N of feature resolutions is important for achieving good results. By varying N from 1 to 4, the performances of CMN are presented in Table 6, and N=3 performs best. This means that CMN with three resolutions is necessary to capture the cross-resolution relations. We set N=3 in all experiments.

Query Feature Re-adding Mechanism. Since Eq. (6) adds value feature from query to enrich value feature of memory during memory reading, we denote the CMN variants using Eq. (5) and Eq. (6) as CMN *w* `Memory Value` and the full CMN, respectively. The model variant using the value feature from query instead of that from the memory is further considered, i.e., $\mathbf{g}_i = \text{softmax}(\mathbf{e}_i)\overline{V_i^q}$ which is named as CMN *w* `Query Value`. Table 5 indicates that query feature re-adding performs better than the models using single value features.

Query Feature Recursive Updating Machanism. Without the query feature recursive updating, 1) we consider concatenating $\mathbf{v}_i$ and the reshaped $\mathbf{g}_i$ followed by the compressing operation, which is termed as CMN *w* `Concatenate`. 2) we compress the channel of all the key features to $C/2$ (instead of C) by 1×1 conv, which is denoted as CMN *w* `Compress`. Experimental results in Table 5 show superior performance gains by our query feature recursive updating mechanism.

Coefficient of $\mathcal{L}_{\text{mid}}$. We set the coefficient (λ) of mid-level loss in Eq. (11) as 1.0 for all experiments. By varying its values from $\{0.0, 0.5, 1.0, 1.5, 2.0\}$, we observe the mean-IoU under two folds and the averaged mean-IoU of them, under 1- and 5-shot settings (Fig. 5). The best performances are achieved when λ=1.0.

Effects of Backbones. VGG-16, ResNet-50, and ResNet-101 are used to conduct experiments for evaluating the effects of different backbones for CMN. Results are shown in Table 4, where ResNet-50 performs better than others.

Feature Fusion under 5-shot Setting. As shown in § 4.1, five support image features are fused by an early feature fusion strategy. We further evaluate two late fusion methods, i.e., OR fusion on masks [25] and average fusion on masks [45], to compare their results (Table 3). To achieve a precise comparison, we leverage averaged mean-IoU and FB-IoU over four folds for this ablation. Since feature fusion performs best, we use it to get the 5-shot results.

4.5. Visualized Results

We take testing episodes from PASCAL-5^i as examples to visualize the segmentation results under 1-shot setting. Specifically, the qualitative comparisons of our CMN model and the baseline model without the cyclic memory reading are illustrated in Fig. 4. We conclude that (i) CMN can well segment query objects with appearance variations, e.g., the *boad* and *train* objects are well segmented. (ii) CMN can well segment query objects with scale variations by the cross-resolution memory reading mechanism. Examples are *sofa* and *sheep*.

5. Conclusion

In this paper, we propose a cyclic memory network (CMN) to tackle the important few-shot semantic segmentation (FSS) task. In CMN, we generate N pairs of multi-resolution key and value features, which are circularly taken as query to be segmented with the rest features written as memory. In this way, the memory reading is conducted for N times to process all resolution queries. Moreover, query feature re-adding and query feature recursive updating mechanisms are proposed to enhance this memory reading module. CMN achieves new state-of-the-arts on the used PASCAL-5^i and COCO-20^i datasets.

Acknowledgments This work was supported by the National Natural Science Foundation of China (Nos. 61702163).

References

[1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. In *TPAMI*, 2017. 1, 3

[2] Nicolas Ballas, Li Yao, Chris Pal, and Aaron Courville. Delving deeper into convolutional networks for learning video representations. In *arXiv:1511.06432*, 2015. 5

[3] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. In *TPAMI*, 2017. 1, 3

[4] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. In *arXiv:1706.05587*, 2017. 2

[5] Tao Chen, Guosen Xie, Yazhou Yao, Qiong Wang, Fumin Shen, Zhenmin Tang, and Jian Zhang. Semantically meaningful class prototype learning for one-shot image segmentation. *TMM*, 2021. 6

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 6

[7] Nanqing Dong and Eric P Xing. Few-shot semantic segmentation with prototype learning. In *BMVC*, 2018. 2, 3

[8] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. In *IJCV*, 2010. 6

[9] Siddhartha Gairola, Mayur Hemani, Ayush Chopra, and Balaji Krishnamurthy. Simpropnet: Improved similarity propagation for few-shot image segmentation. In *IJCAI*, 2020. 2, 6

[10] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014. 2

[11] Bharath Hariharan, Pablo Arbeláez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *ICCV*, 2011. 6

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 6

[13] Tao Hu, Pengwan Yang, Chiliang Zhang, Gang Yu, Yadong Mu, and Cees GM Snoek. Attention-based multi-context guiding for few-shot semantic segmentation. In *AAAI*, 2019. 3

[14] Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. In *ICML*, 2016. 2

[15] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 6

[16] Weide Liu, Chi Zhang, Guosheng Lin, and Fayao Liu. Crnet: Cross-reference networks for few-shot segmentation. In *CVPR*, 2020. 2, 3

[17] Yongfei Liu, Xiangyi Zhang, Songyang Zhang, and Xuming He. Part-aware prototype network for few-shot semantic segmentation. In *ECCV*, 2020. 2, 3, 6

[18] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 1, 3

[19] Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. Key-value memory networks for directly reading documents. *EMNLP*, 2016. 2

[20] Seil Na, Sangho Lee, Jisung Kim, and Gunhee Kim. A read-write memory network for movie story understanding. In *ICCV*, 2017. 2

[21] Khoi Nguyen and Sinisa Todorovic. Feature weighting and boosting for few-shot segmentation. In *CVPR*, 2019. 2, 3, 5, 6

[22] Seoung Wug Oh, Joon-Young Lee, Ning Xu, and Seon Joo Kim. Video object segmentation using space-time memory networks. In *ICCV*, 2019. 2

[23] Kate Rakelly, Evan Shelhamer, Trevor Darrell, Alyosha Efros, and Sergey Levine. Conditional networks for few-shot semantic segmentation. In *ICLR Workshop*, 2018. 3, 6

[24] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 1, 3

[25] Amirreza Shaban, Shray Bansal, Zhen Liu, Irfan Essa, and Byron Boots. One-shot learning for semantic segmentation. In *BMVC*, 2017. 1, 3, 5, 6, 8

[26] Mennatullah Siam, Boris N Oreshkin, and Martin Jagersand. Amp: Adaptive masked proxies for few-shot segmentation. In *ICCV*, 2019. 2, 6

[27] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2014. 1, 6

[28] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *NeurIPS*, 2017. 1, 2

[29] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In *NIPS*, 2015. 2

[30] Zhuotao Tian, Hengshuang Zhao, Michelle Shu, Zhicheng Yang, Ruiyu Li, and Jiaya Jia. Prior guided feature enrichment network for few-shot segmentation. In *TPAMI*, 2020. 2, 3, 4, 5, 6

[31] Haochen Wang, Xudong Zhang, Yutao Hu, Yandan Yang, Xianbin Cao, and Xiantong Zhen. Few-shot semantic segmentation with democratic attention networks. In *ECCV*, 2020. 2, 6

[32] Kaixin Wang, Jun Hao Liew, Yingtian Zou, Daquan Zhou, and Jiashi Feng. Panet: Few-shot image semantic segmentation with prototype alignment. In *ICCV*, 2019. 2, 3, 6

[33] Wenguan Wang, Xiankai Lu, Jianbing Shen, David J Crandall, and Ling Shao. Zero-shot video object segmentation via attentive graph neural networks. In *CVPR*, 2019. 6

[34] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018. 2

[35] Guo-Sen Xie, Jie Liu, Huan Xiong, and Ling Shao. Scale-aware graph neural network for few-shot semantic segmentation. In *CVPR*, 2021. 6

[36] Guo-Sen Xie, Li Liu, Xiaobo Jin, Fan Zhu, Zheng Zhang, Jie Qin, Yazhou Yao, and Ling Shao. Attentive region embedding network for zero-shot learning. In *CVPR*, 2019. 1

[37] Guo-Sen Xie, Li Liu, Fan Zhu, Fang Zhao, Zheng Zhang, Yazhou Yao, Jie Qin, and Ling Shao. Region graph embedding network for zero-shot learning. In *ECCV*, 2020. 1

[38] Guo-Sen Xie, Xu-Yao Zhang, Shuicheng Yan, and Cheng-Lin Liu. Sde: A novel selective, discriminative and equalizing feature representation for visual recognition. *IJCV*, 2017. 1

[39] Boyu Yang, Chang Liu, Bohao Li, Jianbin Jiao, and Qixiang Ye. Prototype mixture models for few-shot semantic segmentation. In *ECCV*, 2020. 2, 3, 6

[40] Tianyu Yang and Antoni B Chan. Learning dynamic memory networks for object tracking. In *ECCV*, 2018. 2

[41] Xianghui Yang, Bairun Wang, Kaige Chen, Xinchi Zhou, Shuai Yi, Wanli Ouyang, and Luping Zhou. Brinet: Towards bridging the intra-class and inter-class gaps in one-shot segmentation. In *BMVC*, 2020. 2, 3, 4, 6, 7

[42] Yazhou Yao, Tao Chen, Guo-Sen Xie, Chuanyi Zhang, Fumin Shen, Qi Wu, Zhenmin Tang, and Jian Zhang. Non-salient region object mining for weakly supervised semantic segmentation. In *CVPR*, 2021. 1

[43] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *arXiv:1511.07122*, 2015. 3

[44] 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 *CVPR*, 2019. 2, 3, 6, 7

[45] 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 *CVPR*, 2019. 3, 4, 6, 7, 8

[46] Xiaolin Zhang, Yunchao Wei, Yi Yang, and Thomas S Huang. Sg-one: Similarity guidance network for one-shot semantic segmentation. In *TCYB*, 2020. 2, 3, 4, 6

[47] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017. 1, 3, 4