Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.059709

ARTICLE

CAMSNet: Few-Shot Semantic Segmentation via Class Activation Map and Self-Cross Attention Block

Jingjing Yan[1], **Xuyang Zhuang**[2,*], **Xuezhuan Zhao**[1,2], **Xiaoyan Shao**[1,*] **and Jiaqi Han**[1]

[1]School of Computer Science, Zhengzhou University of Aeronautics, Zhengzhou, 450046, China
[2]National Key Laboratory of Air-Based Information Perception and Fusion, China Airborne Missile Academy, Luoyang, 471000, China

*Corresponding Authors: Xuyang Zhuang. Email: zhuangxuyang68@163.com; Xiaoyan Shao. Email: shaoxiaoyan@zua.edu.cn

Received: 15 October 2024; Accepted: 30 December 2024; Published: 06 March 2025

ABSTRACT: The key to the success of few-shot semantic segmentation (FSS) depends on the efficient use of limited annotated support set to accurately segment novel classes in the query set. Due to the few samples in the support set, FSS faces challenges such as intra-class differences, background (BG) mismatches between query and support sets, and ambiguous segmentation between the foreground (FG) and BG in the query set. To address these issues, The paper propose a multi-module network called CAMSNet, which includes four modules: the General Information Module (GIM), the Class Activation Map Aggregation (CAMA) module, the Self-Cross Attention (SCA) Block, and the Feature Fusion Module (FFM). In CAMSNet, The GIM employs an improved triplet loss, which concatenates word embedding vectors and support prototypes as anchors, and uses local support features of FG and BG as positive and negative samples to help solve the problem of intra-class differences. Then for the first time, the Class Activation Map (CAM) from the Weakly Supervised Semantic Segmentation (WSSS) is applied to FSS within the CAMA module. This method replaces the traditional use of cosine similarity to locate query information. Subsequently, the SCA Block processes the support and query features aggregated by the CAMA module, significantly enhancing the understanding of input information, leading to more accurate predictions and effectively addressing BG mismatch and ambiguous FG-BG segmentation. Finally, The FFM combines general class information with the enhanced query information to achieve accurate segmentation of the query image. Extensive Experiments on PASCAL $-$ 5^i and COCO $-$ 20^i demonstrate that the CAMSNet yields superior performance and set a state-of-the-art.

KEYWORDS: Few-shot semantic segmentation; semantic segmentation; meta learning

1 Introduction

Deep learning has rapidly advanced [1], resulting in significant improvements in semantic segmentation techniques [2]. However, accurate segmentation heavily relies on annotated data, necessitating extensive pixel-level labeling to achieve generalization. To mitigate this data dependency, semi-supervised and WSSS methods have been proposed. Despite their benefits, these methods struggle to generalize to novel class. Addressing the challenge of predicting numerous novel classes using limited base general class information has become a critical issue in the field of deep learning. Since Sanban et al. introduced the task of FSS, it has quickly become a prominent research area [3]. Currently, metric-based meta-learning strategies are the dominant approaches in this field [4]. These methods generally consist of two stages: meta-training, where feature representations of known classes are learned, and meta-testing, which enables fast inference and segmentation of unseen classes. However, these approaches still encounter substantial challenges, particularly in handling intra-class differences and BG mismatch.

Intra-class differences refer to instances within the same category that are difficult for the model to distinguish due to variations in appearance or form. For example, as shown in Fig. 1, both the oriole and white dove are "birds," or the cow and cat, viewed from different perspectives, exhibit significant visual differences despite belonging to the same category. The existing methods have made improvements by adopting various strategies to address this issue. Reference [5] used an L2L similarity metric to measure the similarity between aligned local features in the embedding space. By learning a distinct metric for each category [6], the model can more effectively differentiate between samples within the same class. By minimizing cross-entropy loss and maximizing mutual information [7], the model can better handle query set uncertainty, reducing intra-class differences. However, these methods still face some challenges, especially when dealing with targets that exhibit significant intra-class differences, where their effectiveness is limited. This paper proposes the GIM that concatenates embedding vectors of dataset category words with support prototypes as anchor. It introduces the Local Feature Generator (LFG) to sample local support features as positive and negative pairs, calculating triplet loss to minimize the distance between positive samples and maximize the distance between negative ones. The General Information Genertor (GIG) produces embedding vectors for category words, aiding in the extraction of general feature information and addressing the challenge of intra-class differences in FSS.

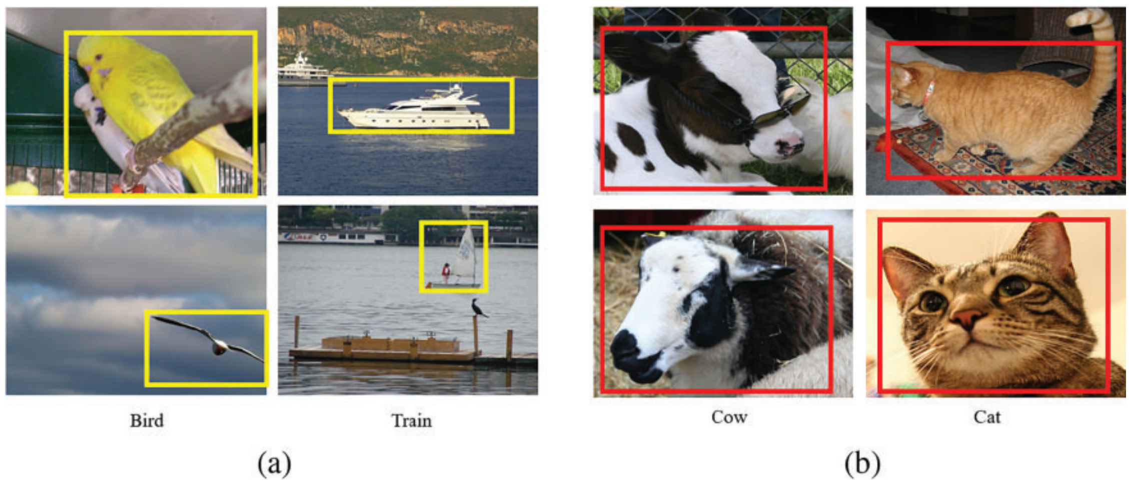

Figure 1: Intra-class differences. (a) The object has the same semantic label but belongs to different categories. (b) The same object appearing differently from various perspectives is referred to as perspective distortion

BG mismatch occurs when the similarity between the BG in the query image and the FG features in the support set is excessively high, causing the query BG to incorrectly match the support FG, which ultimately affects the model's segmentation accuracy. As shown in Fig. 2, the excessive similarity between the query BG (e.g., grassland) and the supported FG leads to an incorrect match. To alleviate this issue, existing studies have proposed various strategies. By fine-tuning the parameters in the Prototype Adaptive Module (PAM) module [8], the basic segmentation model can quickly adapt to new categories, thereby mitigating the impact of BG mismatch. Reference [9] proposed a novel NTRENet to effectively mine and eliminate BG and distracting object (DO) regions from the query images. Reference [10] introduced the FS-PCS, which calibrates the correlation of BG classes through the Basic Prototype Calibration (BPC) module. However, these methods still face certain limitations, especially in complex BG or when there is a high similarity between the FG and BG, which significantly constrains the performance of existing FSS methods. To address these challenges, the paper proposes an enhanced SCA Block, which enables the model to better distinguish between FG and BG features. This module helps the model to more effectively process input information, improve prediction accuracy, and resolve issues related to BG mismatch and unclear segmentation between FG and BG.

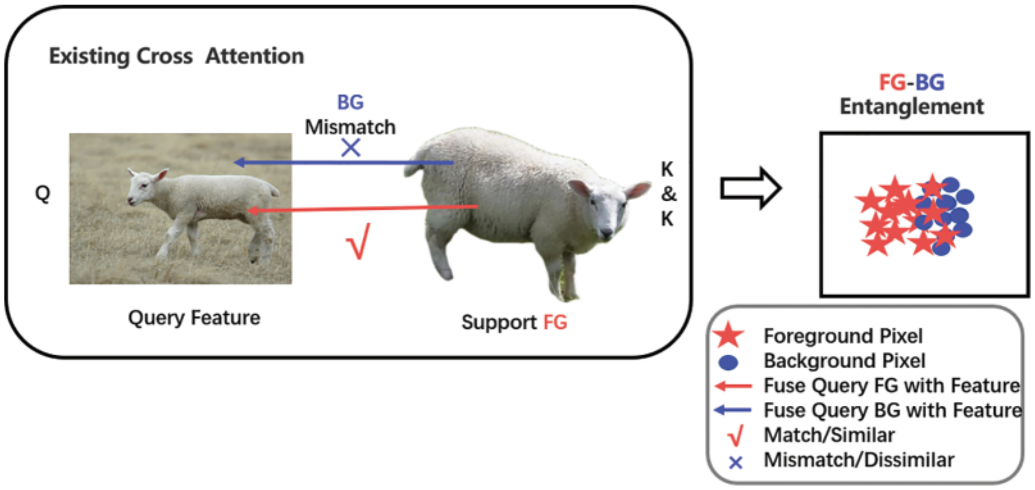

Figure 2: BG mismatch and ambiguous segmentation of BG and FG. The query FG matches correctly with the support FG, but the query BG incorrectly matches with the support FG

This paper proposes CAMSNet for accurate few-shot semantic segmentation. The model consists of the GIM, the CAMA module, the SCA Block, and the FFM. This method demonstrated superior performance on the PASCAL $- 5^i$ [3] and COCO $- 20^i$ [11] datasets. The main contributions are summarized as follows:

- The paper proposes four modules, GIM, CAMA, SCA, and FFM, to achieve more accurate FSS, with a performance improvement of 1.9% compared to state-of-the-art models.
- The paper propose a GIM that incorporates word embedding vectors and triplet loss to better extract general class information and address intra-class differences.
- For the first time, the CAM from the WSSS domain has been applied to FSS. This enhances the model's ability to locate query set images and improves overall performance.
- The introduction of an improved SCA Block effectively learns support and query features, models global image information, and captures long-distance dependencies between different regions. This approach addresses the issues of BG mismatch and ambiguous FG and BG segmentation.

Through these methods, the CAMSNet model excels in FSS tasks by effectively addressing challenges such as intra-class differences, BG mismatch, and ambiguous FG and BG segmentation. This contribution not only resolves current issues in FSS but also introduces innovative ideas and methodologies for future research in this field.

The structure of this paper is as follows: Section 2 reviews the key technologies and developments in semantic segmentation, FSS, category activation maps, cross-attention mechanisms and feature fusion. Section 3 provides an overview of the paper's overall framework, highlighting the key modules and technologies. Section 4 presents a detailed analysis of the experimental setup and results. Finally, Section 5 summarizes the proposed methods, discusses potential future developments, and identifies areas for further improvement in the model.

2 Related Work

Semantic Segmentation. Semantic segmentation [12] is a fundamental computer vision task that aims to assign a specific category to each pixel in an image. Since the proposal of FCN [13], semantic segmentation has made significant progress and is widely used in fields such as medical image recognition, autonomous driving, and geological exploration [14]. AMF-SparseInst [15] is a real-time instance segmentation model that effectively emphasizes key features of small objects within complex backgrounds. It captures multi-scale contextual information and enhances the effectiveness of semantic fusion features using a pyramid pooling module (SimAM-ASPP) combined with depthwise separable convolution and a 3D attention mechanism (SimAM). The model also incorporates a Lite-BiFPN module and a feature enhancement module for further refinement. Mobile-Deep [16] built upon the DeepLabv3+ framework, utilizes MobileNetv2 to reduce parameters. It addresses sample imbalance by combining Focal Loss and Dice Loss, introduces the Efficient Atrous Spatial Pyramid Pooling (E-ASPP) module and Roberts operator to enhance accuracy, and leverages multi-scale feature fusion to improve model performance, thereby achieving rapid and accurate segmentation of Printed Circuit Boards (PCB) images. However, traditional semantic segmentation networks are incapable of handling novel categories. During training, pixel-level annotation of large-scale data is required, resulting in high labor costs and computational expenses, which hinder practical applications. Table 1 summarizes the advantages and disadvantages of each algorithm.

Few-Shot Semantic Segmentation. The key to FSS is to use limited labeled samples of known categories to segment images of unseen categories. Categorized by the availability of supervisory information, it falls into unsupervised (no additional info) and supervised learning (with additional info). CWT [17] addresses the distribution mismatch problem of pre-trained models through a two-stage training method. RePRI [18] improves supervision by applying an enhanced cross-entropy loss on top of traditional semantic segmentation training. BAM [19] combines a base learner and a meta-learner to separately handle segmentation tasks for base classes and novel categories. MIANet [20] improves model performance by interactively integrating information from different scales, utilizing multi-level contextual information. PI-CLIP [21] leverages CLIP to introduce textual information, enhancing the model's generalization ability in the case of unseen categories and limited labeled samples. LLaFS [22] applies large language models to FSS, imporving the model's performance in segmenting unseen categories and complex scenes. However, most of these methods focus on improvements in model architecture and training strategies, with relatively less attention given to the application of attention mechanisms in FSS. In this paper, we propose an effective SCA Block to address key issues in FSS. By introducing the self-cross attention mechanism, SCA Block can more effectively capture contextual information in the image, thereby improving the model's generalization ability and segmentation accuracy.

Class Activation Map. Reference [23] introduced a gradient-independent class activation mapping method called Score-CAM.This method calculates linear weights by using the model's global confidence score for the feature map. In [24], an integrated system is proposed, consisting of two modules: classification and segmentation. Multiple network architectures are independently trained to handle various anomalies, and their components are then combined. The final structure includes two branches: one for anomaly classification and the other for anomaly segmentation. To prevent information loss during deep Convolutional Neural Network (CNN) training on high-resolution images, guided GradCAM (GCAM) adjustment patch neural networks are employed for anomaly localization. The Layer-wise Relevance Propagation (LRP) mechanism provides a deeper explanation for class activation maps [25], further enhancing the understanding of the model's internal reasoning process. However, these methods still suffer from inaccurate localization. The paper propose the CAMA module, which convolves high-level feature maps with the weights of the classification layer to calculate the activation map for each category. This is the first time that CAM, used in weakly supervised semantic segmentation, is applied to FSS to help precisely locate information in the query set.

Table 1: Comparison of algorithms

Algorithm name	Year	Advantages	Disadvantages
AMF-SparseInst [15]	2024	Multi-scale feature fusion improves segmentation accuracy	Implementation complexity may be high
Mobile-Deep [16]	2023	By integrating a lightweight model with efficient modules, it achieves rapid and accurate segmentation of PCB images	The segmentation effect may need further improvement in specific complex scenarios or extreme cases
CWT [17]	2021	Learns to adapt classifier weights for better generalization	May still face challenges when novel categories are highly dissimilar to training classes
BAM [19]	2022	Addresses FSS from a new perspective	Specific implementation and performance need further exploration
EGGCAM [24]	2024	Improves the accuracy of anomaly detection	Primarily targeted at anomaly detection, not semantic segmentation
WSSS-Transformer [25]	2022	Achieves end-to-end weakly-supervised semantic segmentation	Weakly-supervised learning may have lower performance compared to fully-supervised learning
Swin Transformer [26]	2021	Improves performance on vision tasks with a hierarchical structure	May have higher model complexity compared to other methods
PCNN-AGA [27]	2023	Combines pulse coupled neural network and adaptive glowworm algorithm	Specific implementation and performance in segmentation tasks need further verification
Paper [28]	2021	Adaptability to Few-Shot Scenarios and Attention to Semantic Consistency	Limited Exploration of Dataset Diversity and Potential Overfitting in Few-Shot Learning
CA-FSS [29]	2022	Introduces cross-attention mechanism, improving few-shot segmentation results	Specific implementation and performance may vary across different datasets
HFF [30]	2023	Hierarchical feature fusion improves the accuracy of semantic segmentation	Implementation complexity may be high, requiring more computational resources
MIANet [20]	2023	Effectively integrates multi-scale contextual information	May require more computational resources due to multi-scale integration
PI-CLIP [21]	2024	Leverages CLIP to incorporate textual information for better generalization	Relies on large pre-trained models (CLIP), which may be resource-intensive

(Continued)

Table 1 (continued)

Algorithm name	Year	Advantages	Disadvantages
LLaFS [22]	2024	Applies large language models to improve segmentation in unseen categories	Large language models can be computationally expensive

Cross-Attention Mechanism. CNN excel at capturing local features in images due to their local convolution operations but struggle to effectively model global information and long-range dependencies between different regions. This is a significant drawback in FSS tasks. Recent works [26,31,32] have shown that Transformer [33] architectures can achieve outstanding results in computer vision tasks. Specifically, Swin Transformer (ST) [26] can compute efficient self-attention within small windows to reduce computational burden and achieve good segmentation results. In [27], an image segmentation algorithm is proposed that combines Pulse Coupled Neural Network (PCNN) and the adaptive firefly algorithm. This approach retains the advantages of the glowworm algorithm while introducing adaptive movement step size and overall optimal value as adjustment factors, enhancing the ability to find the global optimal solution. However, the aforementioned methods face challenges with invalid and misaligned patches. Based on this, the paper proposes an improved SCA mechanism for FSS tasks, which better captures the overall semantic information of images and helps understand the global context, demonstrating superior performance.

Feature Fusion. Feature fusion has become a key technique for improving the performance of FSS models, aiming to effectively combine features from different layers or sources to enhance the model's representational power. Reference [28] proposed a method that fuses multi-level features from the network to integrate global context and local details, helping the model capture finer-grained information. Reference [29] implemented feature fusion between the support and query sets using a cross-attention module, enabling the model to dynamically adapt to different target instances, which effectively improved segmentation accuracy in few-shot learning scenarios. Reference [30] introduced a hierarchical feature fusion method that combines features from different layers to preserve both global and local information, thereby enhancing feature representation capability. In this paper, we adopt the FFM structure as the feature fusion module, which combines general class information and query features through layer-wise convolution operations and skip connections, resulting in more accurate feature fusion. This design helps the model better capture key features in few-shot segmentation tasks and effectively improves segmentation accuracy.

3 Methodology

3.1 Problem Definition

The objective of FSS is to leverage k-shot annotated support set$S$ to predict the segmentation of novel classes in the query set$Q$. The training set D_{train} and test set D_{test} are completely separate, with D_{test} containing novel classes. Thus, the classes in the training samples C_{train} and those in the test samples C_{test} satisfy the following formula:

$$C_{train} \cap C_{test} = \varnothing \tag{1}$$

The training set D_{train} and testing set D_{test} are divided into support set S and query set Q, as shown in the following formula:

$$D_{train} = \left\{\left(S_i^{train}, Q_i^{train}\right)\right\}_{i=1}^{N_{train}}, D_{test} = \left\{\left(S_i^{test}, Q_i^{test}\right)\right\}_{i=1}^{N_{test}} \tag{2}$$

where N_{train} and N_{test} represent the number of episodes in the training set S and the test set Q.

For support set S is defined as $S = \{x_i^s, m_i^s\}_{i=1}^k$, where k represents the number of support images in an episode, typically 1 or 5. $x_i^s \in R^{3\times H\times W}$ denotes the RGB format support images, where $H \times W$ represent the image dimensions. $m_i^s \in (0,1)^{H\times W}$ is the binary segmentation mask corresponding to the support images. For query set Q is defined as $Q = \{x^q, m^q\}$, which represents the information of one query image. $x^q \in R^{3\times H\times W}$ denotes the RGB format query image, and $m^q \in (0,1)^{H\times W}$ is the binary segmentation mask corresponding to the query image. Typically, the support images and the query images belong to the same category.

The main process of FSS network involves feeding the support-query image pairs x^s and x^q into the encoder to extract support features f_s and query features f_q. The support features f_s are then combined with their corresponding binary segmentation masks m^s to guide the upsampling of the query features f_q, resulting in the predicted segmentation mask $\hat{m}_q$ for the query image. This predicted mask $\hat{m}_q$ is compared with the ground truth segmentation mask m^q of the query image to evaluate the prediction results. After multiple rounds of training on the training set D_{train}, the network's weights are fixed, and the performance is assessed on test set D_{test} to evaluate the network's effectiveness.

3.2 Method Overview

The CAMSNet network consists of four modules: the General Information Module (GIM), the Class Activation Map Aggregation (CAMA) module, the Self-Cross Attention Block (SCA Block), and the Feature Fusion Module (FFM). The network architecture is shown in Fig. 3.

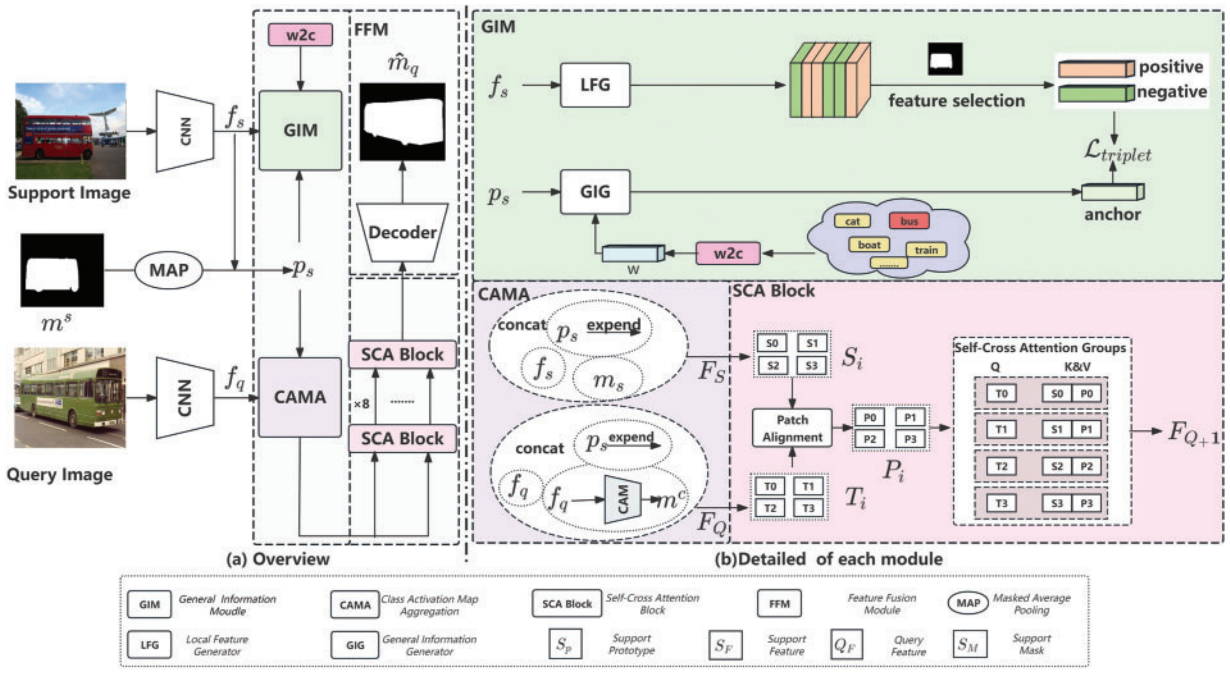

Figure 3: Overall architecture of (a) CAMSNet and (b) detailed of each module. The CAMSNet network consists of four modules: the General Information Module (GIM), the Class Activation Map Aggregation (CAMA) module, the Self-Cross Attention Block (SCA Block), and the Feature Fusion Module (FFM)

First, the task is to use a pre-trained backbone network, ResNet50, to extract support features f_s and query features f_q from the given support set S and query set Q, respectively. Support masks m^s are used to obtain the support prototype p_s through average mask pooling. The support features f_s, support prototype p_s, and corresponding semantic labels are input into the GIM. The support features f_s are processed by the LFG to obtain local support features. By using the binary masks of the support set, FG and BG are separated, and the class words of the dataset are converted into word embedding vectors using the Word2Vec model, ultimately generating general class information. To further address the class-internal differences of the FG in both the support and query sets and to precisely locate the target, the CAMA module is proposed. In this module, the support features f_s, support prototype p_s, and support mask m^s are first aggregated to obtain the fused support features F_S. Then, CAM is used to obtain the pseudo-mask m^c of the query features f_q. The pseudo-mask m^c, query features f_q, and support prototype p_s are aggregated to obtain the fused query features F_Q. To alleviate the issues of ineffective support patches and unaligned patches, the SCA Block aligns the outputs F_Q and F_S from the CAMA module. The aligned support patches and query patches are then processed using self-cross attention mechanisms to enhance the mutual interaction between features. Specifically, the query patch is used as Q, and the aligned support patches and query patches are split into K and V. Self-attention scores are calculated using self-attention, and cross-attention scores are computed using cosine similarity. Finally, the FFM fuses the general feature information and enhanced query information to obtain the final prediction result.

3.3 *General Information Module*

In the field of FSS, a key challenge is the inherent category differences within target objects. Existing methods [34] typically rely on aligning the query image with support samples by matching local image features. While these methods are effective in some cases, they often perform poorly when there are significant visual differences between the query and support images, as shown in Fig. 1. To address these limitations, the GIM is proposed. The goal of GIM is to generate a more powerful, category-based representation by integrating local region features with global semantic information, thereby improving segmentation performance. This method includes LFG and GIG, which are designed to comprehensively capture fine-grained spatial features and high-level semantic context. The specific algorithm flow is shown in Algorithm 1.

Algorithm 1: GIM for Few-shot semantic segmentation

```
1: **Input:** Support Features $f_s$, Support Prototype $P_s$, Semantic Labels
2: **Output:** Compute Triplet Loss $\mathscr{L}_{\text{triplet}}$
3: **Initialize Local Feature Generator (LFG) with $f_s$**
4: **for** each Convolution Block $i \in \{1,2,3\}$ in LFG **do**
5:     $f_{\text{local}}^{(i)} \leftarrow \text{Conv}_i(f_s)$
6:     Downsample $f_{\text{local}}^{(i)}$ to $\frac{1}{4}$ of original size
7: **end for**
8: Generate Local Features $f_{\text{local}} \leftarrow \text{LFG}(f_s)$
9: **Initialize General Information Generator (GIG) with $p_s$ and $w$**
10: **for** each Semantic Label $c \in C$ **do**
11:     $w_c \leftarrow \text{Word2Vec}(c)$
12: **end for**
13: Generate General Class Information $p_{\text{class}} \leftarrow \text{GIG}(P_{\text{support}}, w_c)$
```

(Continued)

Algorithm 1 (continued)

```
14: **Compute Triplet Loss:**
15: for each Anchor Feature $\in F_{\text{local}}$ do
16:     Sample Positive $F_{\text{positive}}$ and Negative $F_{\text{negative}}$ Pairs
17:     Compute Triplet Loss $\mathscr{L}_{\text{triplet}}$ using:
        $\mathscr{L}_{\text{triplet}} \leftarrow \max(0, d(p_{class}, F_{\text{positive}}) - d(p_{class}, F_{\text{negative}}) + \mathscr{T})$
18: end for
19: Update General Class Information $P_{\text{class}}$ using Triplet Loss
20: Output: Compute Triplet Loss $\mathscr{L}_{\text{triplet}}$
```

The LFG focuses on capturing features from specific regions in the support image. The support features f_s are passed through a set of convolutional layers, which include three convolution blocks, gradually downsampling the feature maps. The goal is to extract the most relevant local regions from the support image while eliminating less important background details.

$$f_{init} = \mathscr{F}_{conv}(\mathscr{F}_{downsample}(f_s)) \tag{3}$$

where $\mathscr{F}_{conv}$ denotes the convolution operation applied to the support features f_s, and $\mathscr{F}_{downsample}$ denotes the downsampling operation, which reduces the spatial dimensions and aggregates relevant features. The resulting f_{init} focuses on the most important regions, making it easier to distinguish the target in the query image. The next step is then performed:

$$f_{local} = \mathscr{F}_{LFG}(f_{init}) \tag{4}$$

where $\mathscr{F}_{LFG}(\cdot)$ represents the processing of the obtained f_{init} through the LFG, resulting in the extracted local features f_{local}.

The goal of GIG is to capture the overall semantic information of the target category. By utilizing semantic embeddings associated with category labels, it achieves a semantic representation of the category. The semantic label embedding is generated using a pre-trained word embedding model, converting the category label into a high-dimensional vector representation w. This semantic vector w is then fused with the support prototype, ultimately forming a unified category prototype p_{class}.

$$p_{class} = \mathscr{F}_{GIG}\left(w \oplus \mathrm{p}_s\right) \tag{5}$$

where w represents the word embedding vector of the category label, p_s is the support prototype, and $\oplus$ describes the process of concatenating the two to form a unified vector, which is then input into GIG for processing to obtain the category prototype p_{class}, which contains both visual features and semantic information.

To ensure that the features generated by LFG and GIG remain semantically consistent, a triplet loss is introduced to compare the generated category prototype p_{class} with the local features from the support image. The goal of the triplet loss is to pull the anchor (category prototype p_{class}) closer to the FG features (positive sample) while pushing it away from the BG features(negative sample). The selection of positive and negative samples can be seen as the FG-BG segmentation process.

To extract FG features, the features in the FG region where the mask is 1 are considered as f_{fg}, while the features in the BG region where the mask is 0 are considered as f_{bg}.

$$f_{fg} = \mathscr{F}_{select}(m_s = 1, f_{local}), f_{bg} = \mathscr{F}_{select}(m_s = 0, f_{local}) \tag{6}$$

where m_s is the mask of the support set, and f_{fg} and f_{bg} represent the FG and BG feature sets, respectively.

The BG samples f_{bg} are averaged to obtain the negative sample. The FG sample f_{fg} that is the farthest from the anchor is selected as the positive sample. In the end, the triplet loss encourages the model to pull the FG features closer to the anchor (category prototype p_{class}) while pushing the BG features farther away, thereby improving the model's ability to accurately segment the target category. The formula for the triplet loss is as follows:

$$\mathscr{L}_{triplet} = max(d(p_{class}, positivate) - d(p_{class}, negativate) + \mathscr{T}, 0) \tag{7}$$

where $d(\cdot)$ represents the Euclidean distance, and $\mathscr{T}$ is a hyperparameter set to 0.5, used to ensure sufficient distance between positive and negative samples, thereby helping the model effectively distinguish between FG and BG.

The GIM integrates local feature extraction with global semantic context information. By using the LFG and the GIG modules, region-specific and category-level features are generated and aligned through triplet loss. This approach demonstrates stronger robustness when handling intra-class variations, significantly improving segmentation accuracy even with a limited number of labeled support images.

3.4 Class Activation Map Aggregation

The CAMA module can further address the intra-class differences of FG in the support set and query set, and accurately locate the targets. This module introduces the CAM from WSSS into few-shot semantic segmentation task for the first time. By using CAM to aggregate query information, it achieves accurate localization of query information, compensating for the inability of few-shot data to cover all semantic concepts and helping to address intra-class differences. The algorithm implementation process is shown in Algorithm 2.

Algorithm 2: CAMA for Few-shot semantic segmentation

```
1: **Input:** Query Features $f_q$, Support Features $f_s$, Support Prototype $p_s$, Support Mask $m_s$, Category $c$
2: **Output:** Aggregated Query Features $F_Q$, Aggregated Support Features $F_S$
3: **Step 1:** Flatten and linearly project $f_q$ into Token representation
4: $f_q^{\text{tokens}} \leftarrow \text{Flatten}(f_q)$
5: $f_q^{\text{tokens}} \leftarrow \text{LinearProjection}(f_q^{\text{tokens}})$
6: **Step 2:** Apply Multi-Head Self-Attention in each Transformer block to capture global dependencies
7: **for** each Transformer Block $i \in \{1, 2, ..., N\}$ **do**
8:     $Q, K, V \leftarrow \text{MLP}(f_q^{\text{tokens}})$
9:     $f_q^{\text{block}} \leftarrow \text{SelfAttention}(Q, K, V)$
10:    $f_q^{\text{block}} \leftarrow \text{Concat}(f_q^{\text{block}})$
11:    $f_q^{\text{block}} \leftarrow \text{LayerNorm}(\text{MLP}(f_q^{\text{block}}))$
12: **end for**
13: **Step 3:** Generate Class Activation Map (CAM)
14: $W \leftarrow$ Classification Layer Weights
15: $m^c \leftarrow \text{ReLU}(W \cdot f_q^{\text{block}})$
16: $m^c \leftarrow \text{Normalize}(m^c)$
17: **Step 4:** Aggregate Support Features using $m^c$
18: $F_S \leftarrow \text{Aggregation}(f_s, p_s, m^s)$
```

(Continued)

Algorithm 2 (continued)

```
19: Step 5: Aggregate Query Features with Support Prototype
20: $F_Q \leftarrow \text{Aggregation}(m^c, f_s, p_s)$
21: Output: $F_Q, F_S$
```

CNN are good at extracting local low-level features but fail to capture more global and semantic features. In contrast, Vision Transformer (ViT) can effectively model long-range dependencies between different regions, thus obtaining richer and more discriminative feature representations. In few-shot learning, where sample data is scarce, introducing ViT based on the self-attention mechanism to extract query features can capture the correlations between samples, thereby better generalizing to new categories.

CAMA first flattens the query features f_q and linearly projects them into tokens. In each Transformer block, it uses multi-head self-attention to capture global feature dependencies. Specifically, for the i-th head, the patch token is projected through a Multi-Layer Perceptron (MLP) to obtain the query $Q_i \in R^{h \times w \times d_k}$, key $K_i \in R^{h \times w \times d_k}$, and value $V_i \in R^{h \times w \times d_k}$. Based on Q_i, K_i, and V_i, the output X_i is computed. Through this self-attention mechanism, ViT can effectively capture the global dependencies between image patches, extracting more discriminative feature representations.

$$S_i = \frac{Q_i (K_i)^T}{\sqrt{d_k}}, X_i = softmax(S_i) V_i, i \in 1, 2, 3, \dots, n \tag{8}$$

The output X of ViT block is obtained by concatenating the outputs of each attention head X_i and then processing the concatenated result through normalization and MLP layers. By stacking multiple such blocks, a feature map $F \in R^{h \times w \times d}$ for subsequent modules is generated.

For the extracted feature map $F \in R^{h \times w \times d}$ and the class c, a class activation map m^c is generated by weighting the contribution of features in F to the class. The weight matrix m comes from the weight parameters of the classification layer, as shown in Fig. 4.

$$m^c = Relu\left(\sum_{d}^{i=1} m^{i,c} F^i\right) \tag{9}$$

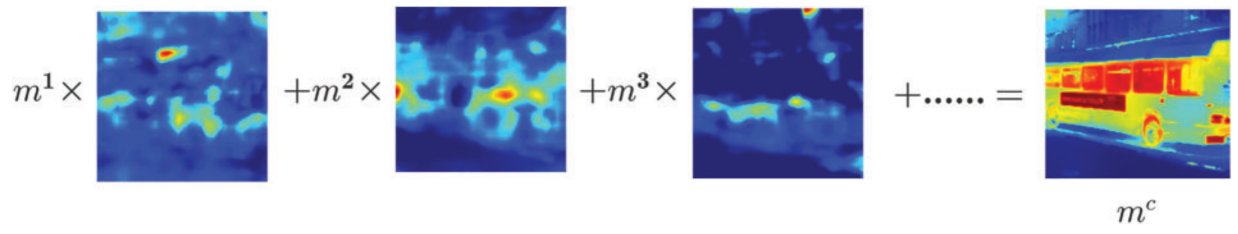

Figure 4: Calculation process of class activation maps. Each class contributes its required attention parts

During the generation of the class activation map m^c, the ReLu function is used to remove negative activations, ensuring that m^c contains only non-negative values. Then, m^c is normalized to scale it to the range $[0, 1]$. This simple and efficient CAM technique can highlight the most discriminative regions for class c in the image, helping to improve the performance of FSS tasks. Class activation map enable the model to better perceive and distinguish fine-grained semantic information in images, aiding in learning more general semantic features and enhancing generalization to new samples.

CAMA aggregates support features f_s, support prototype p_s, and support masks m_s to obtain the fused support features F_S. Then, using CAM, it obtains a pseudo-mask m^c for the query features f_q. The pseudo-mask m^c, query features f_q, and support prototype p_s are aggregated to obtain the fused query features F_Q. The main purpose of including support prototype p_s here is to narrow the gap between support features f_s and query features f_q.

3.5 Self-Cross Attention Block

The SCA Block combines the strengths of Self-Attention and Cross-Attention, specifically tailored to tackle the challenges posed by BG mismatch and ambiguous FG-BG. This module comprises two essential components: Patch Alignment (PA) and Self-Cross Attention (SCA), both aimed at reinforcing the relationship between query and support features in the context of BG discrepancies and FG-BG ambiguities. The algorithm implementation process is shown in Algorithm 3.

Algorithm 3: SCA Block for few-shot semantic segmentation

```
1: Input: Query Features $F_Q$, Support Features $F_S$
2: Output: Enhanced Query Features $F_Q^{\text{enhanced}}$
3: Step 1: Patch Alignment (PA) for Query and Support Features
4: $T_q \leftarrow \text{PAP}(F_Q)$
5: $T_s \leftarrow \text{PAP}(F_S)$
6: $\text{Sim} \leftarrow \text{CosineSimilarity}(T_q, T_s)$
7: $Indices \leftarrow \text{argmax}(\text{Sim})$
8: Step 2: Self-Attention and Cross-Attention for Query-Feature Enhancement
9: Self-Attention:
10: $A_{QQ} \leftarrow \text{ScaledDotProductAttention}(T_q, T_q)$
11: Cross-Attention:
12: $A_{QS} \leftarrow \text{CosineSimilarity}(T_q, T_s)$
13: Step 3: Attention Score Calculation
14: $A \leftarrow \text{softmax}(A_{QQ} + A_{QS})$
15: Step 4: Feature Aggregation
16: $F_Q^{\text{enhanced}} \leftarrow \text{FFN}(F_Q)$
17: Step 5: Iterative Processing
18: for each $i \in \{1, 2, \ldots, N\}$ do
19:     $F_Q^{\text{enhanced}} \leftarrow \text{SCA Block}(F_Q^{\text{enhanced}}, F_{\text{support}})$
20: end for
21: Output: Enhanced Query Features $F_Q^{\text{enhanced}}$
```

The PA aims to mitigate the issues of ineffective support patches and misaligned support patches by aligning each query patch with the support patches. The specific process is as follows: Firstly, the prototype representations of the query patches and support patches are calculated through Patch Average Pooling (PAP). The prototype of each patch only contains foreground information, ensuring that the focus is on the foreground part while ignoring the background. Then, the cosine similarity is used to compute the similarity between the query patch prototypes and support patch prototypes. This similarity measure quantifies the degree of matching between the FG features.

$$T_i^p = PAP(T_i), S_i^p = PAP(S_i), i \in 1, 2, 3, \ldots, n \tag{10}$$

where $S_i^p \in R^{N^2 \times C \times 1 \times 1}$ and $T_i^p \in R^{N^2 \times C \times K \times K}$ are the support and query patch prototypes, and PAP stands for patch average pooling. Based on the cosine similarity, the argmax operation is utilized to select the most similar support patch for alignment with each query patch, ensuring that each query patch is matched with a support patch containing foreground pixels. Through these steps, the PA module successfully aligns each query patch with its most relevant support patch, thereby addressing the issues of ineffective support patches and misaligned support patches. SCA further enhances the relationship between query patches and aligned support patches. The specific details are illustrated in Fig. 5. It combines self-attention and cross-attention to improve the quality of feature fusion, ensuring that query FG and BG are effectively distinguished.

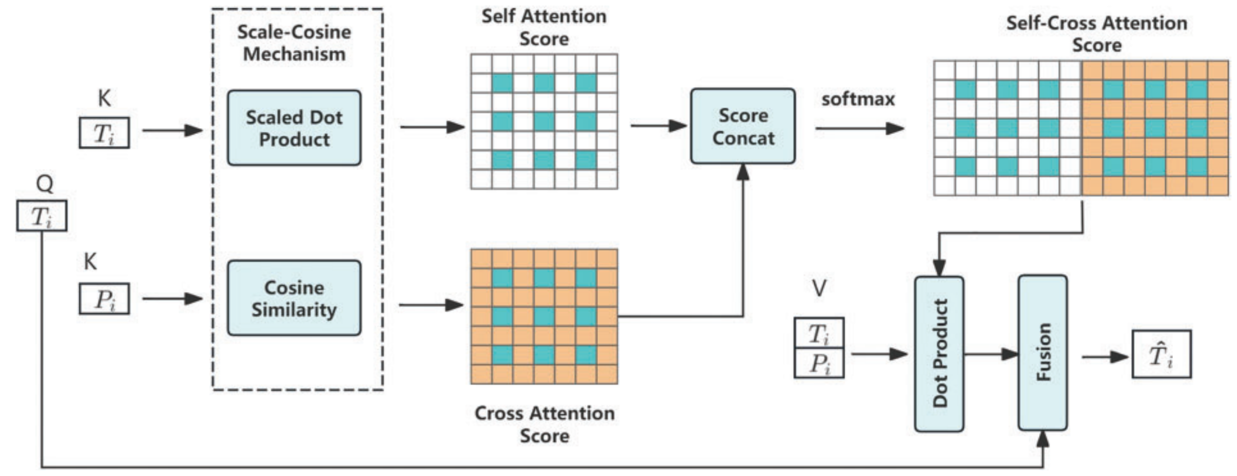

Figure 5: Details of self-cross attention (SCA). Query patch calculates self and cross attention scores with itself and the aligned support patch

The query patches are fused through self-attention and cross-attention. The query patches serve as Q, interacting with the K & V of the support patches to compute attention scores. Self-Attention calculates the relationship between the query patches Q and the query patches K & V using Scaled Dot Product. Cross-Attention computes the relationship between the query patches Q and the aligned support patches K & V using cosine similarity.

To prevent self-attention from overly focusing on the query foreground and ignoring support features, a Scaled Cosine (SC) mechanism is designed. By using Scaled Dot Product to calculate the attention within the query patches themselves and cosine similarity to compute the attention between query patches and aligned support patches, the model is encouraged to better integrate information from the support FG.

$$A_{QQ} = \frac{T_i \cdot T_i}{\sqrt{d_k}}, \; A_{QS} = \frac{T_i \cdot S_i^A}{\|T_i\| \, \|S_i^A\|} \tag{11}$$

The scores are represented as $A_{QQ} \in R^{N^2 \times K^2 \times K^2}$ and $A_{QS} \in R^{N^2 \times K^2 \times K^2}$, where N^2 and K^2 denote the number of patches and pixels, respectively.

The attention scores are normalized through a softmax operation, resulting in weighted fused features between the query patches and the support patches.

$$A = Softmax(Concat(A_{QQ}, A_{QS})) \tag{12}$$

where $A \in R^{N^2 \times K^2 \times K^2}$ represent the attention scores A.

A aggregates the query patches Q_i, the aligned patches P_i and fuses them with the query patches to enhance the query features.

$$\hat{Q}_i = FFN(A \cdot Concat(Q_i, S_i^p) + Q_i) \tag{13}$$

where FFN stands for Feed-Forward Network. The enhanced query patches are fed into the next SCA Block. Through eight iterative layers of SCA Blocks, the SCA module continuously enhances the similarity and difference between query and support features, thereby improving the feature representation capability in few-shot segmentation tasks. The output of the last SCA Block is then fed into the Feature Fusion Module (FFM) for segmentation.

As shown in the Fig. 6, the SCA module effectively addresses the issues of BG mismatch and ambiguou FG-BG segmentation in FSS by combining PA and SCA. By precisely aligning the query patches with the support patches and enhancing the information interaction between features using self-attention and cross-attention mechanisms, the SCA module significantly boosts the performance of few-shot segmentation tasks. Especially when dealing with complex scenes, it can better capture the similarities and differences between support and query features.

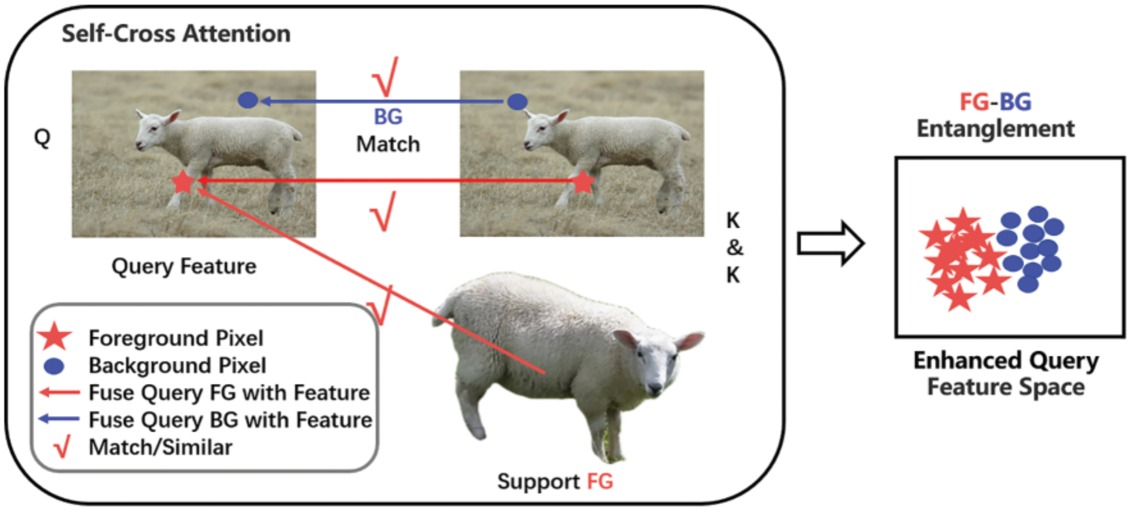

Figure 6: Our proposed self-cross attention. The BG of the query and support sets correctly matches, and the segmentation of the FG and BG in the query set is effective

3.6 Feature Fusion Module

To aggregate general class information p_{class} and query features processed F_Q through the self-cross attention mechanism, the paper adopts the FFM. The algorithm implementation process is shown in Algorithm 4. The specific algorithm implementation process is illustrated in the figure. To further enhance the fusion effect of features at different scales, the FEM structure is selected as the feature fusion module. In the specific implementation, the FFM combines general class information and query features through layer-by-layer convolution operations and skip connections. The ReLU activation function is used after each convolution operation to introduce nonlinear properties. During feature fusion, Batch Normalization is applied to accelerate convergence and improve model stability. In the FEM structure, to achieve efficient fusion of features at different scales, multi-scale convolution and pooling operations are employed. Specifically, 1×1, 3×3 and 5×5 convolution kernels are used to capture local features of different sizes, and a max pooling layer is utilized to extract contextual information at different scales.

Algorithm 4: FFM for few-shot semantic segmentation

```
1: **Input:** Query Features $F_Q^{\text{enhanced}}$, General Class Information $P_{class}$
2: **Output:** Final Fused Features $F_{\text{final}}$
3: **Step 1: Initial Feature Aggregation**
4: $F_{\text{init}} \leftarrow F_Q^{\text{enhanced}} \oplus P_{class}$
5: **Step 2: Multi-Scale Convolution and Pooling**
6: **For each scale $i \in \{1, 3, 5\}$**
7: $F_{\text{conv}}^i \leftarrow \text{Conv2D}(F_{\text{init}}, \text{kernel size} = i, \text{channels} = C)$
8: $F_{\text{pool}}^i \leftarrow \text{MaxPooling}(F_{\text{conv}}^i, \text{kernel size} = 2)$
9: **Step 3: Skip Connections and Aggregation**
10: $F_{\text{skip}} \leftarrow \text{Concat}(F_{\text{conv}}^1, F_{\text{conv}}^3, F_{\text{conv}}^5, F_{\text{pool}}^1, F_{\text{pool}}^3, F_{\text{pool}}^5)$
11: **Step 4: Batch Normalization and Non-Linearity**
12: $F_{\text{norm}} \leftarrow \text{BatchNorm}(F_{\text{skip}})$
13: $F_{\text{nonlinear}} \leftarrow \text{ReLU}(F_{\text{norm}})$
14: **Step 5: Final Fusion**
15: $F_{\text{fused}} \leftarrow \text{Conv2D}(F_{\text{nonlinear}}, \text{kernel size} = 3, \text{channels} = C)$
16: **Output:** $F_{\text{fused}}$
```

3.7 Prediction and Training Loss

The loss during the training process consists of segmentation loss and triplet loss. The segmentation loss includes intermediate prediction loss $\mathscr{L}_{seg1}$ and final prediction loss $\mathscr{L}_{seg2}$. The intermediate prediction loss $\mathscr{L}_{seg1}$ is calculated by segmenting the features extracted by ViT and comparing them with the ground truth mask of the query set. The final prediction loss $\mathscr{L}_{seg2}$ is derived by comparing the predicted mask $\hat{m}^q$ of the query set with the ground truth mask m^q.

The core idea of the triplet loss is to learn an embedding space by comparing triplets (anchor, positive, negative) such that similar samples are closer together and dissimilar samples are further apart. The general class information p_{class} is used as the anchor, and positive and negative pairs are extracted from local features in the support set. The final loss $\mathscr{L}$ function is:

$$\mathscr{L} = \mathscr{L}_{seg1} + \mathscr{L}_{seg2} + \mathscr{L}_{triplet} \tag{14}$$

4 Experiments

4.1 Setup

Datasets. Experiments are conducted on two commonly used few-shot segmentation datasets, $\text{PASCAL} - 5^{\text{i}}$ and $\text{COCO} - 20^{\text{i}}$. In the experiments on FSS, the commonly used dataset $\text{PASCAL} - 5^{\text{i}}$ is created from the additional annotations of PASCAL VOC2012 and SBD. The 20 classes in the dataset are evenly divided into 4 splits, with each split containing 5 classes. $\text{COCO} - 20^{\text{i}}$ created by MSCOCO, 80 categories are divided into 4 equal parts, each containing 20 categories. Specify specific training and validation categories based on the different split settings.

Data Preprocessing and Splitting. The images are uniformly resized to 473 × 473 to meet the input requirements of the model. The image pixel values are normalized to map them into the range [0, 1]. Based on the predefined split, the dataset is divided into training, validation, and test sets. Each subset contains a fixed proportion of samples, with the training set accounting for 70%, the validation set for 15%, and the test set for 15%. In each experiment, the class distribution within the training, validation, and test sets is ensured to be consistent to avoid the impact of class imbalance on the experimental results.

Data Augmentation. During the training process, data augmentation strategies such as random cropping and random rotation are employed. Random cropping involves randomly selecting a 473 × 473 pixel region from the original image for cropping. Random cropping can simulate variations in object scale and position, enhancing the model's adaptability to object detection and segmentation, especially in small-sample scenarios, where it mitigates overfitting by increasing the diversity of training samples. Random rotation involves rotating the image within a range of −30 to 30, enabling the model to learn spatial relationships and features of the image at different angles.

Evaluation. The current few-shot segmentation algorithms are mostly evaluated using Foreground-Background Intersection over Union (FB-IoU) and mean Intersection over Union (mIoU). The mIoU is calculated by computing the IoU for each class and then averaging over the total number of classes. The formula for Intersection over Union is as follows:

$$IoU = \frac{TP}{TP + FN + FP} \tag{15}$$

where *TP*, *FP* and *FN* represent true positives, false positives, and false negatives, respectively. Assuming there are classes, the calculation formulas for FB-IoU and mIoU are:

$$mIoU = \frac{1}{n}\sum_{i=1}^{n} IoU_i \tag{16}$$

When calculating FB-IoU, only FG and BG classes are considered, so n is 2. When calculating mIoU, BG classes are not considered. For the experimental dataset, n is 5. Since the number of images per class in the validation test is inconsistent, mIoU considers the multiple class situations, providing a more accurate performance evaluation of the model.

Implementation Details. The PolyWarmupAdamW optimizer is used to optimize the model separately, with an initial learning rate set to 6e-5. During training, the batch size is fixed at 4 to ensure a consistent number of samples processed in each training iteration. In terms of training epochs, the model is trained for 200 epochs on the PASCAL-5i dataset and 50 epochs on the COCO-20i dataset to fully train and test the model's performance on different datasets. To optimize the model's segmentation performance, the Dice loss is used as the loss function. Eight SCCA modules are designed, with each module's window size set to 8 (as specified in the ablation experiments). For category embeddings, the Word2Vec model trained on Google News data is used, and the generated 300-dimensional word vectors are employed to represent categories. In the K-shot learning setting, when k > 1, the features of multiple support images are averaged to enhance the model's ability to recognize minority classes.

The hyperparameter settings in Table 2 were optimized using a variety of strategies. First, the optimizer incorporates a combination of polynomial learning rate decay and the Warmup strategy to ensure stability during the early stages of training. The momentum and batch size settings were based on references [29] and [35], to meet the task requirements and hardware limitations. The number of training epochs was determined by the size and complexity of the datasets. For the smaller PASCAL-5i dataset, more epochs were required to fully optimize model performance. In contrast, for the larger and more complex COCO-20i dataset, fewer epochs were sufficient to achieve satisfactory results. The number and window size of SCCA modules were determined through extensive experimentation, achieving a practical trade-off between computational cost and performance. For category embeddings, 300-dimensional word vectors generated by Word2Vec were adopted as the default setting, providing a good balance between semantic representation capability and computational efficiency.

Table 2: Experimental parameter table

Parameter name	Value
Optimizer	PolyWarmupAdamW
Momentum	9×10^{-1}
Weight decay	1×10^{-4}
Learning rate	1×10^{-6}
Batch size	4
Epoch ($\text{PASCAL} - 5^{\text{i}}$)	150
Epoch ($\text{COCO} - 20^{\text{i}}$)	50
Dimensional word vector	300
Number of SCA Blocks	8

4.2 Comparision with State-of-the-Arts

$\text{PASCAL} - 5^{\text{i}}$: The results shown in Table 3 significantly surpass the state-of-the-art under the 1-shot and 5-shot settings on $\text{PASCAL} - 5^{\text{i}}$. Using ResNet50 as the backbone network, it can be seen that in the 5-shot setting, the model's stable and efficient performance with more support samples. The model achieves mIoU improvements of 0.9% in the Fold-3 settings, surpassing the previous advanced semantic segmentation model.

Table 3: Performance comparison on $\text{PASCAL} - 5^{\text{i}}$ in terms of mIoU. The best and second best results are highlighted with bold and underline

Backbone	***Methods***	**1-shot**					**5-shot**				
		5^0	5^1	5^2	5^3	***Mean***	5^0	5^1	5^2	5^3	***Mean***
ResNet50	PGNet [36]	56.0	66.9	50.6	50.4	50.6	57.7	68.7	52.9	54.6	58.5
	CANet [37]	52.5	65.9	51.3	51.9	55.4	55.5	67.8	51.9	53.2	57.1
	PANet [38]	44.0	57.5	50.8	44.0	49.1	55.3	67.2	61.3	53.2	59.3
	PPNet [39]	47.8	58.8	53.8	45.6	51.5	58.4	67.8	64.9	56.7	62.0
	RPMM [40]	55.2	66.9	52.6	50.7	56.3	56.3	67.3	54.5	51.0	57.3
	ASGNet [41]	58.8	67.9	56.8	53.7	59.3	63.7	70.6	64.2	57.4	63.9
	RePRI [18]	59.8	68.3	62.1	48.5	59.7	64.6	71.4	71.1	59.3	66.6
	MLC [42]	59.2	<u>71.2</u>	<u>65.6</u>	52.5	62.1	63.5	71.6	71.2	58.1	66.1
	SSP [35]	60.5	67.8	**66.4**	51.0	61.4	<u>67.5</u>	<u>72.3</u>	**75.2**	<u>62.1</u>	<u>69.3</u>
	QSCMNet [43]	62.9	69.8	56.7	<u>55.6</u>	61.3	65.3	70.8	57.6	56.5	62.6
	SRPNet [44]	62.8	69.3	55.8	58.1	61.5	64.3	70.3	55.1	60.5	62.6
	PFENet+QSR [45]	**63.1**	69.9	58.7	58.9	<u>62.7</u>	68.3	71.7	63.1	63.6	66.7

(Continued)

Table 3 (continued)

Backbone	***Methods***	**1-shot**					**5-shot**				
		5^0	5^1	5^2	5^3	***Mean***	5^0	5^1	5^2	5^3	***Mean***
	CaSLNet [46]	62.0	69.7	57.6	57.3	61.6	63.8	70.3	57.8	59.3	62.8
	DFBNet [47]	<u>63.0</u>	69.1	59.5	54.9	61.6	70.2	**74.6**	66.0	62.2	69.3
	CAMSNet	61.9	**71.3**	63.2	**55.7**	**63.0**	**68.9**	<u>72.9</u>	<u>74.0</u>	**63.0**	**69.7**

COCO – 20^i: Table 4 shows the mIoU performance comparison on COCO – 20^i between our method and several models. It has a wide variety of data types and a huge quantity, making it difficult to segment. Using ResNet50 as the backbone network, it can be seen that (1) Under the 1-shot setting and Fold-0 and Fold-1 settings, the model outperforms previous advanced methods with outstanding performance. MIoU achieved improvements of 1.9% and 1.0%, respectively. (2) Under the 5-shot setting, the model achieved 0.9% and 0.9% improvement in mIoU under the Fold-1 and Fold-2 settings, respectively, surpassing the previous advanced semantic segmentation model QSCMNet.

Table 4: Performance comparison on COCO – 20^i in terms of mIoU. The best and second best results are highlighted with bold and underline

Backbone	***Methods***	**1-shot**					**5-shot**				
		5^0	5^1	5^2	5^3	***Mean***	5^0	5^1	5^2	5^3	***Mean***
	PPNet [39]	28.1	30.8	29.5	27.7	29.0	39.0	40.8	37.1	37.3	38.5
	RPMM [40]	29.5	36.8	28.9	27.0	30.6	33.8	42.0	33.0	33.3	35.5
	ASGNet [41]	–	–	–	–	34.6	–	–	–	–	42.5
	ReRPI [18]	31.2	38.1	33.3	33.0	34.0	38.5	46.2	40.0	43.6	42.1
ResNet50	MMNet [48]	34.9	<u>41.0</u>	37.2	**37.0**	<u>37.5</u>	37.0	40.3	39.3	36.0	38.2
	SSP [35]	35.5	39.6	<u>37.9</u>	36.7	37.4	40.6	<u>47.0</u>	<u>45.1</u>	<u>43.9</u>	<u>44.1</u>
	QSCMNet [43]	<u>35.1</u>	37.0	37.7	35.6	36.4	41.8	42.7	42.6	**43.9**	42.8
	PFENet+QSR [45]	–	–	–	–	36.9	–	–	–	–	41.2
	MANet [49]	33.9	40.6	35.7	35.2	36.4	<u>41.9</u>	49.1	43.2	42.7	44.2
	CAMSNet	**37.0**	**42.0**	**38.2**	<u>36.9</u>	**38.4**	**42.0**	**47.9**	**46.0**	41.6	**44.3**

4.3 Experimental Analysis

An extensive ablation study was conducted on the dataset in the 1-shot setting to verify the effectiveness of the proposed key modules. The experiments in this section were performed on the dataset using ResNet50 as the backbone network.

Table 5 illustrates the impact of each component on the model. The three proposed components resulted in an mIoU of 62.5%. Using only the GIM, the mIoU reached 54.7%. With only the CAMA module, the mIoU improved by 3.8% over the former. Using both the GIM and SCA modules resulted in a 6.1% improvement in mIoU compared to using only the GIM. This is because the GIM generates general information, while the SCA module better separates the FG and BG. When all three modules were used together, the mIoU increased by 1.7% compared to using only the GIM and SCA modules. This is because the CAMA module can precisely locate the target, thereby improving accurate segmentation.

Table 5: Ablation studies of main model components. **GIM:** General Information Module; **CAM:** Class Activation Map; **SCA:** Self-Cross Attention

GIM	CAM	SCA	mIoU
–	–	–	54.7
–	✓	–	58.5
✓	–	✓	60.8
✓	✓	✓	62.5

Table 6 shows the impact of different scale CAMs and ViT on the model. Using a single scale CAM resulted in an mIoU of 58.4%. When using multiple scales, the mIoU reached 58.6%. With the addition of ViT and using multiple scales CAM, the mIoU increased to 60.9%. When ViT was added with a single scale CAM, the mIoU reached 62.5%. This is because ViT can capture global features, providing the network with strong global modeling capabilities. However, combining ViT with multiple scale CAMs can lead to overfitting, so using a single scale CAM improved the model's segmentation ability.

Table 6: Ablation studies of generating CAM. **ViT:** Vision Transformer; **CAM:** Class Activation Map; **CAMS:** Different scale CAM

ViT	CAM	CAMS	mIoU
–	✓	–	58.4
–	–	✓	58.6
✓	–	✓	60.9
✓	✓	–	62.5

Table 7 demonstrates the impact of the number of SCA Blocks on model performance. When using fewer blocks, the model struggles to capture sufficiently complex features, resulting in lower accuracy and IoU values. As the number of blocks increases, the model's performance improves, but it still does not reach the optimal level. Eight SCA Blocks represent the optimal configuration, balancing feature extraction and computational efficiency for the best performance. While increasing the number of blocks may further enhance the model's complexity, setting it to 16 blocks may lead to excessive computational burden and performance degradation, potentially causing overfitting.

Table 7: Ablation studies of SCA block number.block_4: use 4 SCA block.block_6: use 6 SCA block.block_8: use 8 SCA block.block_16: use 16 SCA block

Block_4	Block_6	Block_8	Block_16	mIoU
✓	–	–	–	56.2
–	–	–	✓	57.4
–	–	✓	–	60.6
–	✓	–	–	58.7

To verify the efficiency of ViT in generating CAMs, we conducted a *t*-test to compare CAMs generated by CNN and ViT. Fig. 7 displays the changes in mIoU. CNNs exhibit a slower initial improvement rate and subsequently plateau, whereas ViTs demonstrate faster growth and surpass CNNs in performance. This result confirms significant differences in mIoU between the two, favoring ViT's overall superior performance. Therefore, ViT is more suitable for CAM generation due to its faster training speed and higher accuracy.

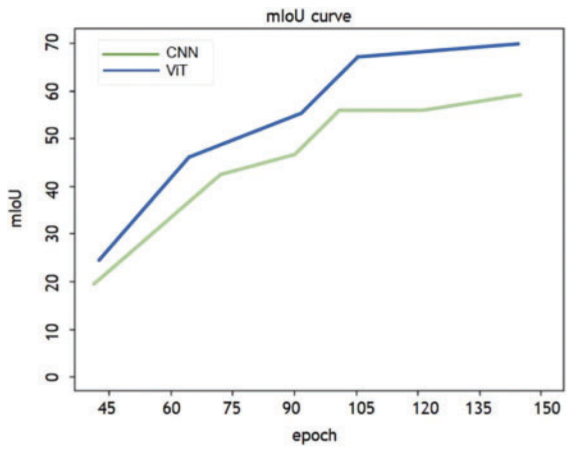

Figure 7: Comparison of the MIoU curves between the models generating CAMs via CNN and ViT

4.4 *Qualitative Comparsion*

Fig. 8 presents the visualization results of the proposed CAMSNet in the 1-shot setting. The figure demonstrates accurate segmentation for various categories, including sheep, airplane, bird, cat, and bicycle. It shows that even with significant intra-class differences between query samples and the support set (columns 3 and 4), CAMSNet can successfully segment the target objects, highlighting its effectiveness. Additionally, CAMSNet effectively mitigates issues of BG mismatch and ambiguous separation between FG and BG (column 1).

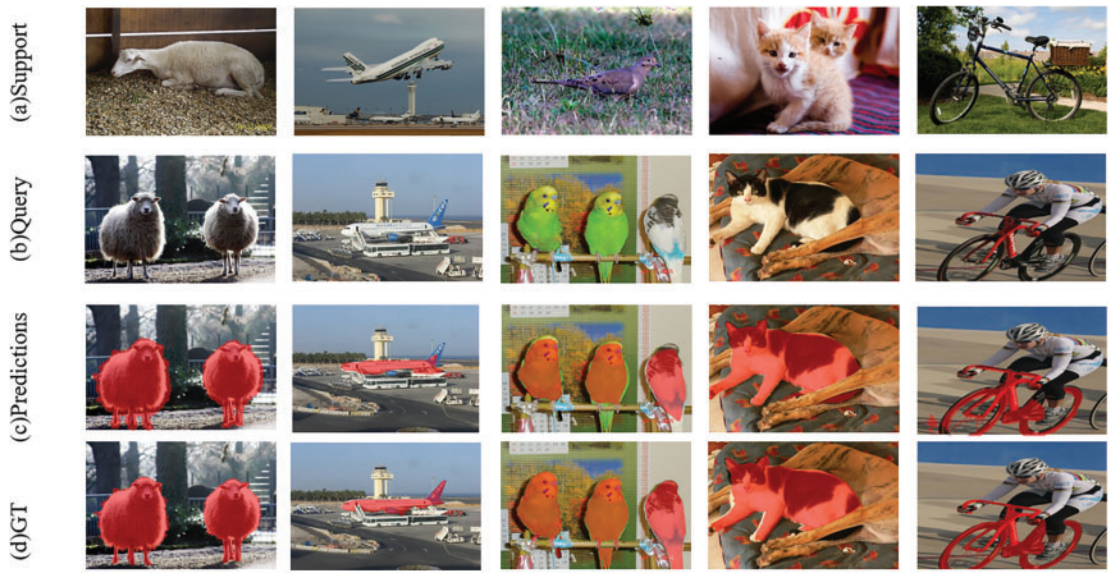

Figure 8: Qualitative results of CAMSNet and baseline on PASCAL – 5^i and COCO – 20^i

Fig. 9 illustrates the segmentation effects of different modules. It shows the incremental improvements to the network made by the GIM, CAMA, and SCA Blocks. Specifically, GIM enhances the ability to handle intra-class variance, the addition of CAMA helps in locating the query target, and SCA addresses BG mismatch and ambiguous separation between FG and BG. These results clearly demonstrate the crucial role each component plays in improving overall segmentation performance.

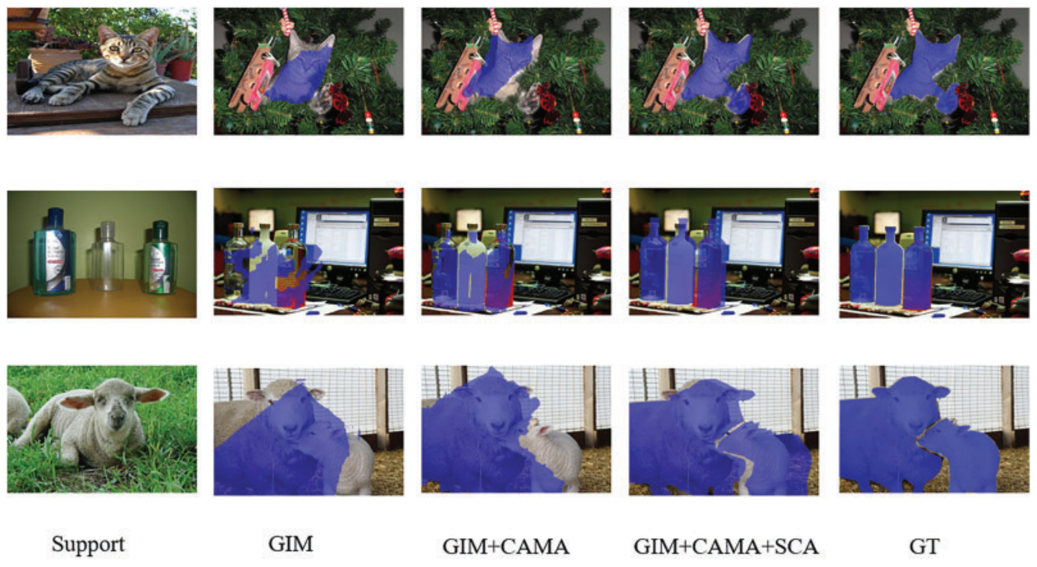

Figure 9: Segmentation effects of different modules

5 Conclusion

The paper propose a network called CAMSNet, which is composed of four main components: GIM, CAMA, SCA Block, and FFM, designed for FSS. The GIM generates general class information, helping to address intra-class differences. The CAMA utilizes CAM to create pseudo-masks for precise localization, deviating from the traditional use of cosine similarity for pseudo-mask generation. The SCA employs an improved self-cross attention to resolve issues of BG mismatch and ambiguous FG-BG segmentation. The FFM aggregates the general class information from the GIM and the enhanced query information from the SCA to facilitate precise segmentation. Extensive experiments on the dataset $\text{PASCAL}-5^i$ and $\text{COCO}-20^i$ demonstrate significant advancements in the field of FSS, effectively integrating word embeddings, class activation maps, and cross-attention mechanisms to provide an innovative solution to overcome existing limitations. A well-trained network will focus more on the key features of the training categories, which can lead to a decrease in the model's generalization ability. How to reduce the bias of the model on novel classes in new tasks is a problem that needs to be further studied and explored in the future.

Acknowledgement: We sincerely thank the National Natural Science Foundation of China, the Research Programs of Henan Provincial Science and Technology Department, the Chongqing Natural Science Foundation Committee, the Henan Provincial Key Research and Development Projects, the Aviation Science Foundation Committee, and the Henan Center for Outstanding Overseas Scientists for their financial support.

Funding Statement: The study has been supported by funding from the following sources: National Natural Science Foundation of China (U1904119); Research Programs of Henan Science and Technology Department (232102210033; 232102210054); Chongqing Natural Science Foundation (CSTB2023NSCQ-MSX0070); Henan Province Key Research and Development Project (231111212000); Aviation Science Foundation (20230001055002); supported by Henan Center for Outstanding Overseas Scientists (GZS2022011).

Author Contributions: The authors confirm contribution to the paper as follows: Jingjing Yan: Designed algorithms for research, wrote the paper. Xuyang Zhuang: Revised the manuscript, supervised the project. Xuezhuan Zhao: Research methodology, set the progress schedule. Xiaoyan Shao: Provided algorithmic support. Jiaqi Han: Revised the manuscript. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data that support the findings of this study are available from the corresponding author upon reasonable request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Li G, Wang X, Li Y, Li Z. Adaptive clustering object detection method for UAV images under long-tailed distributions. Inform Technol Cont. 2023;52(4):1025–44. doi:10.5755/j01.itc.52.4.33460.
2. Zhou T, Wang W, Konukoglu E, Van Gool L. Rethinking semantic segmentation: a prototype view. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2022. p. 2582–93.
3. Shaban A, Bansal S, Liu Z, Essa I, Boots B. One-shot learning for semantic segmentation. arXiv:170903410. 2017.
4. Catalano N, Matteucci M. Few shot semantic segmentation: a review of methodologies and open challenges. arXiv:230405832. 2023.
5. Zha Z, Tang H, Sun Y, Tang J. Boosting few-shot fine-grained recognition with background suppression and foreground alignment. IEEE Transact Circ Syst Video Technol. 2023;33(8):3947–61. doi:10.1109/TCSVT.2023.3236636.
6. Liu G, Zhao L, Li W, Guo D, Fang X. Class-wise metric scaling for improved few-shot classification. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV); 2021. p. 586–95.
7. Boudiaf M, Ziko I, Rony J, Dolz J, Piantanida P, Ben Ayed I. Information maximization for few-shot learning. Adv Neural Inform Process Syst. 2020;33:2445–57.
8. Wang J, Li J, Chen C, Zhang Y, Shen H, Zhang T. Adaptive FSS: a novel few-shot segmentation framework via prototype enhancement. Proc AAAI Conf Artif Intell. 2024;38:5463–71.
9. Liu Y, Liu N, Cao Q, Yao X, Han J, Shao L. Learning non-target knowledge for few-shot semantic segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2022. p. 11573–82.
10. An Z, Sun G, Liu Y, Liu F, Wu Z, Wang D, et al. Rethinking few-shot 3D point cloud semantic segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2024. p. 3996–4006.
11. Nguyen K, Todorovic S. Feature weighting and boosting for few-shot segmentation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV); 2019. p. 622–31.
12. Thisanke H, Deshan C, Chamith K, Seneviratne S, Vidanaarachchi R, Herath D. Semantic segmentation using vision transformers: a survey. Eng Appl Artif Intell. 2023;126:106669. doi:10.1016/j.engappai.2023.106669.
13. Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2015. p. 3431–40.
14. Tian Z, Zhao H, Shu M, Yang Z, Li R, Jia J. Prior guided feature enrichment network for few-shot segmentation. IEEE Transact Pattern Anal Mach Intell. 2020;44(2):1050–65. doi:10.1109/TPAMI.2020.3013717.
15. Chen Y, Wan L, Li S, Liao L. AMF-SparseInst: attention-guided multi-scale feature fusion network based on sparseInst. Inform Technol Control. 2024;53(3):675–94. doi:10.5755/j01.itc.53.3.35588.
16. Liu L, Ke C, Lin H. Mobile-deep based PCB image segmentation algorithm research. Comput Mater Contin. 2023;77(2):2443–61. doi:10.32604/cmc.2023.042582.

17. Lu Z, He S, Zhu X, Zhang L, Song YZ, Xiang T. Simpler is better: few-shot semantic segmentation with classifier weight transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV); 2021. p. 8741–50.
18. Boudiaf M, Kervadec H, Masud ZI, Piantanida P, Ben Ayed I, Dolz J. Few-shot segmentation without meta-learning: a good transductive inference is all you need?. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2021. p. 13979–88.
19. Lang C, Cheng G, Tu B, Han J. Learning what not to segment: a new perspective on few-shot segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2022. p. 8057–67.
20. Yang Y, Chen Q, Feng Y, Huang T. MIANet: aggregating unbiased instance and general information for few-shot semantic segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2023. p. 7131–40.
21. Wang J, Zhang B, Pang J, Chen H, Liu W. Rethinking prior information generation with CLIP for few-shot segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2024. p. 3941–51.
22. Zhu L, Chen T, Ji D, Ye J, Liu J. LLaFS: when large language models meet few-shot segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2024. p. 3065–75.
23. Wang H, Wang Z, Du M, Yang F, Zhang Z, Ding S, et al. Score-CAM: score-weighted visual explanations for convolutional neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPR); 2020. p. 24–5.
24. Rajkumar R, Shanthi D, Manivannan K. Efficient guided grad-CAM tuned patch neural network for accurate anomaly detection in full images. Inform Technol Control. 2024;53(2):355–71. doi:10.5755/j01.itc.53.2.34525.
25. Zhou B, Khosla A, Lapedriza A, Oliva A, Torralba A. Interpreting deep neural networks with layer-wise relevance propagation. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV); 2020. p. 147–56.
26. Liu Z, Lin Y, Cao Y, Hu H, Wei Y, Zhang Z, et al. Swin transformer: hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV); 2021. p. 10012–22.
27. Zhu J, Ma Y, Huang J, Wang L. Image segmentation combining pulse coupled neural network and adaptive glowworm algorithm. Inform Technol Control. 2023;52(2):487–99. doi:10.5755/j01.itc.52.2.33415.
28. Xie X, Chen X, Zhang W, Xu Y. Feature fusion for few-shot semantic segmentation. IEEE Transact Pattern Anal Mach Intell. 2021;43(7):1607–17.
29. Wang H, Liu Y, Li Z. Cross-attention for few-shot segmentation. IEEE Transact Image Process. 2022;31:1121–32.
30. Zhang Y, Wei L, Luo W. Hierarchical feature fusion for semantic segmentation. Comput Vis Image Underst. 2023;195:103432.
31. Ru L, Zhan Y, Yu B, Du B. Learning affinity from attention: end-to-end weakly-supervised semantic segmentation with transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2022. p. 16846–55.
32. Zhang G, Kang G, Yang Y, Wei Y. Few-shot segmentation via cycle-consistent transformer. Adv Neural Inform Process Syst. 2021;34:21984–96.
33. Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, et al. An image is worth 16x16 words: transformers for image recognition at scale. arXiv:201011929. 2020.
34. Zhang B, Xiao J, Qin T. Self-guided and cross-guided learning for few-shot segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2021. p. 8312–21.
35. Fan Q, Pei W, Tai YW, Tang CK. Self-support few-shot semantic segmentation. In: European Conference on Computer Vision; 2022; Cham: Springer Nature Switzerland. p. 701–19.
36. Zhang C, Lin G, Liu F, Guo J, Wu Q, Yao R. Pyramid graph networks with connection attentions for region-based one-shot semantic segmentation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV); 2019. p. 9587–95.

37. Zhang C, Lin G, Liu F, Yao R, Shen C. Canet: class-agnostic segmentation networks with iterative refinement and attentive few-shot learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2019. p. 5217–26.
38. Wang K, Liew JH, Zou Y, Zhou D, Feng J. Panet: few-shot image semantic segmentation with prototype alignment. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV); 2019. p. 9197–206.
39. Liu Y, Zhang X, Zhang S, He X. Part-aware prototype network for few-shot semantic segmentation. In: Computer Vision–ECCV 2020: 16th European Conference; 2020 Aug 23–28; Glasgow, UK: Springer; p. 142–58.
40. Yang B, Liu C, Li B, Jiao J, Ye Q. Prototype mixture models for few-shot semantic segmentation. In: Computer Vision–ECCV 2020: 16th European Conference; 2020 Aug 23–28; Glasgow, UK: Springer; p. 763–78.
41. Li G, Jampani V, Sevilla-Lara L, Sun D, Kim J, Kim J. Adaptive prototype learning and allocation for few-shot segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2021. p. 8334–43.
42. Yang L, Zhuo W, Qi L, Shi Y, Gao Y. Mining latent classes for few-shot segmentation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV); 2021. p. 8721–30.
43. Shao J, Gong B, Chen JY. Query-support semantic correlation mining for few-shot segmentation. Eng Appl Artif Intell: Int J Intell Real-Time Automat. 2023;126:106797. doi:10.1016/j.engappai.2023.106797.
44. Ding H, Zhang H, Jiang X. Self-regularized prototypical network for few-shot semantic segmentation. Pattern Recognit. 2023;133(9):109018. doi:10.1016/j.patcog.2022.109018.
45. Guan H, Spratling M. Query semantic reconstruction for background in few-shot segmentation. Visual Comput. 2024;40(2):799–810. doi:10.1007/s00371-023-02817-x.
46. Sun H, Zhang Z, Huang L, Jiang B, Luo B. Category-aware siamese learning network for few-shot segmentation. Cognit Comput. 2024;16(3):924–35. doi:10.1007/s12559-024-10273-5.
47. Jiang C, Zhou Y, Liu Z, Feng C, Li W, Yang J. Learning discriminative foreground-and-background features for few-shot segmentation. Multimed Tools Appl. 2024;83(18):55999–6019. doi:10.1007/s11042-023-17708-5.
48. Wu Z, Shi X, Lin G, Cai J. Learning meta-class memory for few-shot semantic segmentation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV); 2021. p. 517–26.
49. Ao W, Zheng S, Meng Y, Yang Y. Few-shot semantic segmentation via mask aggregation. Neural Process Lett. 2024;56(2):56. doi:10.1007/s11063-024-11511-5.