This ICCV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Feature Weighting and Boosting for Few-Shot Segmentation

Khoi Nguyen and Sinisa Todorovic
Oregon State University
Corvallis, OR 97330, USA
{nguyenkh,sinisa}@oregonstate.edu

Abstract

This paper is about few-shot segmentation of foreground objects in images. We train a CNN on small subsets of training images, each mimicking the few-shot setting. In each subset, one image serves as the query and the other(s) as support image(s) with ground-truth segmentation. The CNN first extracts feature maps from the query and support images. Then, a class feature vector is computed as an average of the support's feature maps over the known foreground. Finally, the target object is segmented in the query image by using a cosine similarity between the class feature vector and the query's feature map. We make two contributions by: (1) Improving discriminativeness of features so their activations are high on the foreground and low elsewhere; and (2) Boosting inference with an ensemble of experts guided with the gradient of loss incurred when segmenting the support images in testing. Our evaluations on the PASCAL-5^i and COCO-20^i datasets demonstrate that we significantly outperform existing approaches.

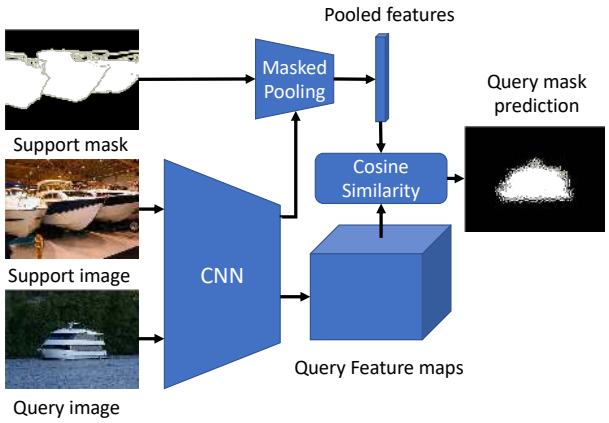

Figure 1. Our goal is to segment a foreground object class in a query image, given a single (few) support image(s) showing the same class with the known ground-truth segmentation mask(s). The target class may appear in the query and support images in different numbers of instances and 3D poses. Recent work uses cosine similarity to first estimate a similarity map between CNN features of the query and support images. The similarity map is then used for segmentation of the target class in the query image.

1. Introduction

This paper is about few-shot segmentation of foreground objects in images. As Fig. 1 shows, given only a few training examples – called support images – and their ground-truth segmentation of the target object class, our goal is to segment the target class in the query image. This problem is challenging, because the support and query images may significantly differ in the number of instances and 3D poses of the target class, as illustrated in Fig. 1. This important problem arises in many applications dealing with scarce training examples of target classes.

Recently, prior work has addressed this problem by training an object segmenter on a large training set, under the few-shot constraint [26, 6, 20]. The training set is split into many small subsets. In every subset, one image serves as the query and the other(s) as the support image(s) with known ground truth(s). As shown in Fig. 1, their framework uses a CNN – e.g., VGG [27] or ResNet [12] – for extracting feature maps from the support and query images. The support's feature maps are first pooled over the known ground-truth foreground. Then, the support's masked-pooled features are used to estimate a cosine similarity map with the query's features. The resulting similarity map and the query's features are finally passed to a few convolutional layers in order to segment the target object class in the query. The incurred loss between the prediction and the query's ground-truth is used for the CNN's training.

The above framework has two critical limitations which we address in this paper. First, we experimentally found that the CNN has a tendency to learn non-discriminative features with high activations for different classes. To address this issue, as Fig. 2 shows, our first contribution extends prior work by efficiently estimating feature relevance so as to encourage that their activations are high inside the ground-truth locations of the target class, and low elsewhere in the

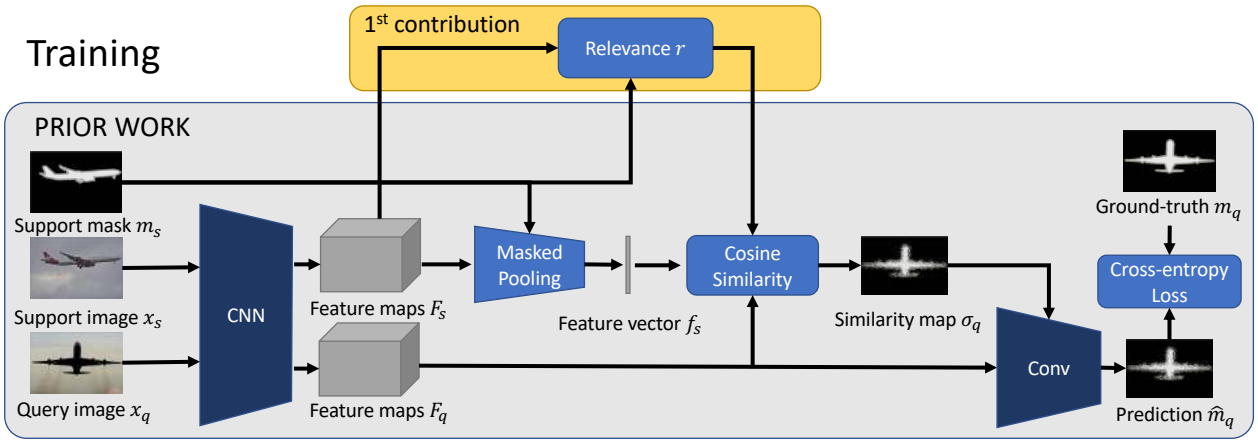

Figure 2. Our few-shot learning from a single support image: Prior work averages a class feature vector over the known foreground of the support image. A dot product of this class feature vector and feature maps of the query image gives a similarity map. The similarity map and features of the query image are used for segmenting the target class in the query. We extend prior work by additionally estimating feature relevance (contribution 1). Our contribution 2 is shown in Fig. 3.

image. This is formulated as an optimization problem, for which we derive a closed-form solution.

Second, learning from few support images is prone to overfitting and poor generalization to the query in the face of the aforementioned large variations of the target class. To address this issue, as Fig. 3 shows, our second contribution is a new boosted inference, motivated by the traditional ensemble learning methods which are robust to overfitting [9, 10]. We specify an ensemble of experts, where each expert adapts the features initially extracted from the support image. This feature adaptation is guided by the gradient of loss incurred when segmenting the support image relative to its provided ground truth. The ensemble of experts produce the corresponding ensemble of object segmentations of the query image, whose weighted average is taken as our final prediction. Importantly, while we use the first contribution in both training and testing, similar to the traditional ensemble learning methods, our second contribution is applied only in testing for boosting the performance of our CNN-based segmenter.

For K-shot setting, both contributions are naturally extended for segmenting the query image by jointly analyzing the provided support images and their ground-truths rather than treating support images independently as in prior work.

For evaluation, we compare with prior work on the benchmark PASCAL-5^i dataset [26]. Our results demonstrate that we significantly outperform the state of the art. In addition, we perform evaluation on the larger and more challenging COCO-20^i dataset [16]. To the best of our knowledge, we are the first to report results of few-shot object segmentation on COCO-20^i.

In the following, Sec. 2 reviews previous work, Sec. 3 specifies our two contributions, Sec. 4 describes our implementation details and complexity, and Sec. 5 presents our experimental results.

2. Related Work

This section reviews related work on few-shot image classification and semantic segmentation.

Few-shot classification predicts image class labels with access to few training examples. Prior work can be broadly divided into three groups: transfer learning of models trained on classes similar to the target classes [28, 22, 29, 30], meta-learning approaches that learn how to effectively learn new classes from small datasets [8, 18], and generative approaches aimed at data-augmentation [31, 25].

Semantic segmentation labels all pixels in the image. Recently, significant advances have been made by using fully convolutional network (FCN) [17] and its variants — including SegNet [1], UNet [23], RefineNet [15], PSPNet [33], DeepLab v1[2], v2 [3], v3 [4], v3+ [5] — all of which are usually evaluated on the PASCAL VOC 2012 [7] and MSCOCO [16] datasets. However, these approaches typically require very large training sets, which limits their application to a wide range of domains.

Few-shot semantic segmentation labels pixels of the query image that belong to a target object class, conditioned by the ground-truth segmentation masks of a few support images. Prior work typically draws from the above mentioned approaches to few-shot image classification and semantic segmentation. For example, the one-shot learning method OSLSM [26] and its extensions — namely, CoFCN [20, 21], PL+SEG [6], and SG-One [32] — consist of the conditioning and segmentation branches implemented as VGG [27] and FCN-32s [17], respectively. The condi-

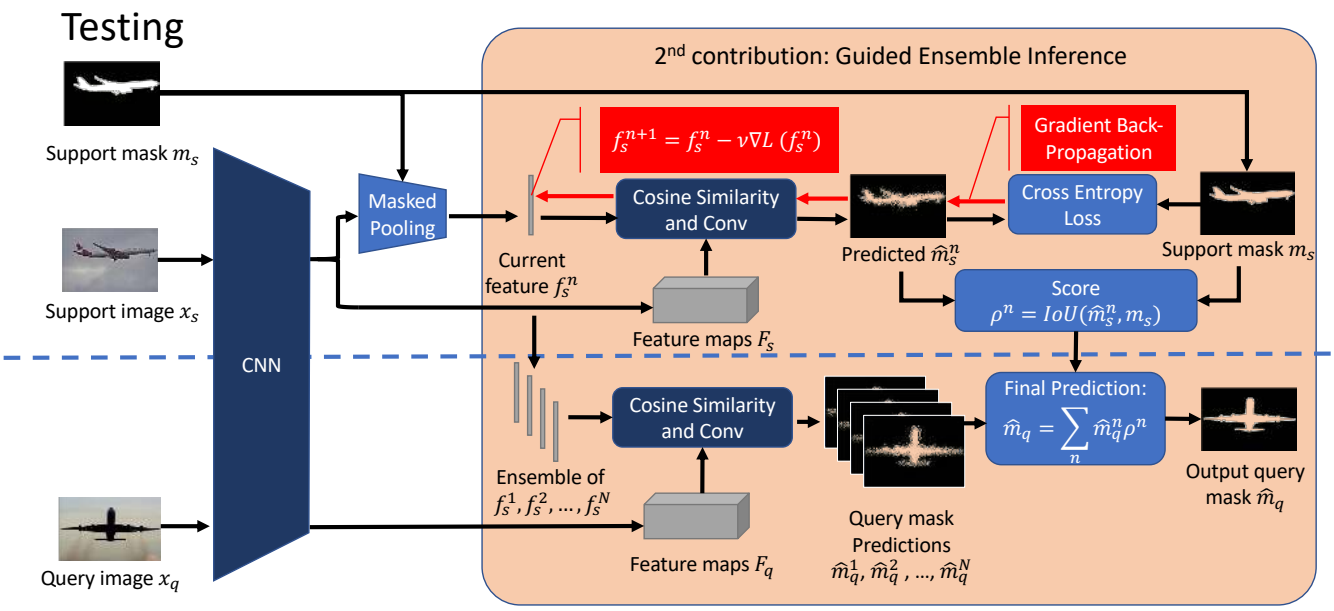

Figure 3. Our new boosted inference: We generate an ensemble of the support's features guided by the gradient of loss, incurred when segmenting the target object in the support image. A dot product between the query's feature maps and the ensemble of the support's features gives the corresponding ensemble of similarity maps, whose weighted average is taken as our final segmentation of the query.

tioning branch analyzes the target class in the support image, and conditions the segmentation branch for object segmentation in the query image. Co-FCN improves OSLSM by segmenting the query image based on a concatenation of pooled features from the support image and feature maps from the query image. PL+SEG first estimates a distance between the query's feature maps and prototypes predicted from the support image, and then labels pixels in the query image with the same class as their nearest neighbor prototypes. SG-One also estimates similarity between a pooled feature from the support image and feature maps of the query for predicting the query's segmentation. Our approach extends SG-One with the two contributions specified in the next section.

3. Our Approach

An object class is represented by K support images with ground-truth segmentation masks. Given a query image showing the same object class in the foreground, our goal is predict the foreground segmentation mask. For $K = 1$, this problem is called one-shot semantic segmentation. Below, and in the following Sec. 3.1 and Sec. 3.2, we consider the one-shot setting, for simplicity. Then, we discuss the K-shot setting, $K > 1$, in Sec. 3.3

Given a large set of training images showing various object classes and the associated ground-truth segmentation masks, our approach follows the common episodic training strategy. In each training episode, we randomly sample a pair of support and query images, x_s and x_q, with binary segmentation masks, m_s and m_q, of the target object class in the foreground. Elements of the mask are set to 1, $m_{s,i} = 1$, for pixels i occupied by the target class; otherwise, $m_{s,i} = 0$. The same holds for m_q. We use m_s to condition the target class in the query image.

The standard cross-entropy loss, $L(\hat{m}_q, m_q)$, between the binary ground-truth m_q and predicted query mask $\hat{m}_q = f_\theta(x_s, m_s, x_q)$, is used for the end-to-end training of parameters θ of our deep architecture, $\theta^* = \arg\min_\theta L(\hat{m}_q, m_q)$.

3.1. Training of Our Deep Architecture

Fig. 2 shows the episodic training of a part of our deep architecture (without our second contribution which is not trained) on a pair of support x_s and query x_q images. We first use a CNN to extract feature maps $F_s \in \mathbb{R}^{d\times w\times h}$ from x_s, and feature maps $F_q \in \mathbb{R}^{d\times w\times h}$ from x_q, where d is the feature dimensionality, and w and h denote the width and height of the feature map. Then, we average F_s over the known foreground locations in m_s, resulting in the average class feature vector f_s of the support image. For this masked feature averaging, m_s is down-sampled to $\tilde{m}_s \in \{0,1\}^{w\times h}$ with the size $w \times h$ of the feature maps, and f_s is estimated as

$$f_s = \frac{1}{|\tilde{m}_s|} \sum_{i=1}^{wh} F_{s,i}\, \tilde{m}_{s,i}. \quad (1)$$

where $|\tilde{m}_s| = \sum_i \tilde{m}_{s,i}$ is the number of foreground locations in $\tilde{m}_s$. Next, we compute the cosine similarity between f_s and every feature vector $F_{q,i}$ from the query feature maps F_q. This gives a similarity map, $\sigma_q \in [0,1]^{w \times h}$, between the support and query image:

$$\sigma_{q,i} = \cos(f_s, F_{q,i}) = \frac{f_s^T F_{q,i}}{\|f_s\|_2 \cdot \|F_{q,i}\|_2}, \; i = 1, \ldots, w\ h. \tag{2}$$

We expect that σ_q provides informative cues for object segmentation, as high values of $\sigma_{q,i}$ indicate likely locations of the target class in the query image.

Before we explain how to finally predict $\hat{m}_q$ from σ_q and F_q, in the following, we specify our first technical contribution aimed at extending the above framework.

Contribution 1: Feature Weighting. For learning more discriminative features of the target class from a single (or very few) support image(s), we introduce a regularization that encourages high feature activations on the foreground and simultaneoulsy low feature activations on the background of the support image. This is formalized as an optimization problem for maximizing a sum of *relevant* differences between feature activations. Let $\phi_s \in \mathbb{R}^{d \times 1}$ denote a vector of feature differences normalized over the foreground and background areas in the segmentation map of the support image as

$$\phi_s = \sum_{i=1}^{wh} F_{s,i} \left[\frac{\tilde{m}_{s,i}}{|\tilde{m}_s|} - \frac{1 - \tilde{m}_{s,i}}{wh - |\tilde{m}_s|} \right], \tag{3}$$

The relevance $r \in \mathbb{R}^{d \times 1}$ of features in ϕ_s is estimated by maximizing a sum of the feature differences:

$$\underset{r}{\text{maximize}} \quad \phi_s^\top \, r, \quad \text{s.t.} \quad \|r\|_2 = 1 \, . \tag{4}$$

The problem in (4) has a closed-form solution:

$$r^* = \frac{\phi_s}{\|\phi_s\|_2}. \tag{5}$$

We use the estimated feature relevance r^* when computing the similarity map between the support and query images. Specifically, we modify the cosine similarity between f_s and F_q, given by (2), as

$$\sigma^*_{q,i} = \cos(f_s, F_{q,i}, r^*) = \frac{(f_s \odot r^*)^T (F_{q,i} \odot r^*)}{\|f_s \odot r^*\|_2 \cdot \|F_{q,i} \odot r^*\|_2}, \tag{6}$$

where $\odot$ is the element-wise product between two vectors.

Note that we account for feature relevance in both training and testing. As r^* has a closed-form solution, it can be computed very efficiently. Also, the modification of the similarity map in (6) is quite simple and cheap to implement in modern deep learning frameworks.

As shown in Fig. 2, in the final step of our processing, we concatenate σ^*_q and F_q together, and pass them to a network with only two convolutional layers for predicting $\hat{m}_q$.

3.2. Contribution 2: Feature Boosting

In testing, the CNN is supposed to address a new object class which has not been seen in training. To improve generalization to the new class, in testing, we use a boosted inference – our second contribution – inspired by the gradient boosting [10]. Alg.1 summarizes our boosted inference in testing. Note that in testing parameters of the CNN and convolutional layers remain fixed to the trained values.

As shown in Fig. 3, given a support image with ground-truth m_s and a query image, in testing, we predict not only the query mask $\hat{m}_q$, but also the support mask $\hat{m}_s$ using the same deep architecture as specified in Sec. 3.1. $\hat{m}_s$ is estimated in two steps. First, we compute a similarity map, σ^*_s, as a dot product between $f_s \odot r^*$ and $F_{s,i} \odot r^*$, as in (6). Second, we pass σ^*_s and F_s to the two-layer convolutional network for predicting $\hat{m}_s$. Third, we estimate the standard cross-entropy loss $L(\hat{m}_s, m_s)$, and iteratively update the average class features as

$$f_s^{n+1} = f_s^n - \nu \, \partial L(\hat{m}_s^n, m_s)/\partial f_s^n, \tag{7}$$

where ν is the learning rate. f_s^n, $n = 1, \ldots, N$, are experts that we use for predicting the corresponding query masks $\hat{m}_q^n$, by first estimating the similarity map $\sigma^*_{q,i} = \cos(f_s^n, F_{q,i}, r^*)$, $i = 1, ...wh$, as in (6), and then passing σ^*_q and F_q to the two-layer network for computing $\hat{m}_q^n$.

Finally, we fuse the ensemble $\{\hat{m}_q^n : n = 1, \ldots, N\}$ into the final segmentation, $\hat{m}_q$, as

$$\hat{m}_q = \sum_{n=1}^{N} \hat{m}_q^n \, \rho^n, \tag{8}$$

Algorithm 1: Guided Ensemble Inference in Testing

Input: F_s, f_s, F_q, m_s, ν
Output: $\hat{m}_q$
// Guided ensemble of experts
1. Initialize: $f_s^1 = f_s, E = \{\}, R = \{\}$;
2. **for** $n = 1, \ldots, N$ **do**
   a. $\sigma^*_{s,i} = \cos(f_s^n, F^n_{s,i}, r^*)$, $i = 1, ...wh$, as in (6) ;
   b. $\hat{m}_s^n = \text{Conv}(\sigma^*_s, F_s^n)$;
   c. $\rho^n = \text{IoU}(\hat{m}_s^n, m_s)$;
   d. Compute cross-entropy loss: $L(\hat{m}_s^n, m_s)$;
   e. $f_s^{n+1} = f_s^n - \nu \, \partial L(\hat{m}_s^n, m_s)/\partial f_s^n$;
   f. $E = E \cup \{f_s^n\}, R = R \cup \{\rho^n\}$,
end
// Inference using E and R
4. **for** $n = 1, \ldots, N$ **do**
   a. $\sigma^*_{q,i} = \cos(f_s^n, F_{q,i}, r^*)$, $i = 1, ...wh$, as in (6) ;
   b. $\hat{m}_q^n = \text{Conv}(\sigma^*_q, F_q)$;
end
5. $\hat{m}_q = \sum_{n=1}^{N} \hat{m}_q^n \, \rho^n$

where ρ^n denotes our estimate of the expert's confidence in correctly segmenting the target class, computed as the intersection-over-union score between $\hat{m}_s^n$ and m_s:

$$\rho^n = \text{IoU}(\hat{m}_s^n, m_s). \tag{9}$$

3.3. K-shot Setting

When the number of support images $K > 1$, prior work [20, 21, 6, 32] predicts $\hat{m}_q^k$ for each support image independently, and then estimates $\hat{m}_q$ as an average over these predictions, $\hat{m}_q = \frac{1}{K}\sum_{k=1}^{K} \hat{m}_q^k$. In contrast, our two contributions can be conveniently extended to the K-shot setting so as to further improve our robustness, beyond the standard averaging over K independent segmentations of the query.

Our contribution 1 is extended by estimating relevance $r \in \mathbb{R}^{d\times 1}$ of a more general difference vector of feature activations defined as

$$\phi_s = \sum_{k=1}^{K}\sum_{i}^{wh} F_{s,i}^k \left[\frac{\tilde{m}_{s,i}^k}{|\tilde{m}_s^k|} - \frac{1-\tilde{m}_{s,i}^k}{wh-|\tilde{m}_s^k|}\right] \tag{10}$$

Similar to (4) and (5), the optimal feature relevance has a closed-form sulution $r^* = \frac{\phi_s}{\|\phi_s\|_2}$. Note that we estimate r^* jointly over all K support images, rather than as an average of independently estimated feature relevances for each support image. We expect the former (i.e., our approach) to be more robust than the latter.

Our contribution 2 is extended by having a more robust update of f_s^{n+1} than in (7):

$$f_s^{n+1} = f_s^n - \nu\,\partial \sum_{k=1}^{K} L(\hat{m}_s^k, m_s^k)/\partial f_s^n, \tag{11}$$

where $L(\hat{m}_s^k, m_s^k)$ is the cross entropy loss incurred for predicting the segmentation mask $\hat{m}_s^k$ using the unique vector f_s^n given by (11) for every support image k, as explained in Sec. 3.2. Importantly, we do not generate K independent ensembles of experts $\{f_s^n : n = 1, \ldots, N\}_{k=1,K}$ for each of the K support images. Rather, we estimate a single ensemble of experts more robustly over all K support images, starting with the initial expert $f_s^1 = \frac{1}{K}\sum_{k=1}^{K} f_s^k$.

4. Implementation Details and Complexity

Implementation. The CNN we use is a modified version of either VGG-16 [27] or ResNet-101 [12]. Our CNN has the last two convolutional layers modified so as to have the stride equal to 1 instead of 2 in the original networks. This is combined with a dilated convolution to enlarge the receptive field with rates 2 and 4, respectively. So the final feature maps of our network have stride $= 8$, which is 1/8 of the input image size. For the two-layer convolutional network (Conv) aimed at producing the final segmentation, we use a 3×3 convolution with ReLU and 128 channels, and a 1×1 convolution with 2 output channels – background and foreground. It is worth noting that we do not use a CRF as a common post-processing step [14].

For implementation, we use Pytorch [19]. Following the baselines [32, 20, 21], we pretrain the CNN on ImageNet [24]. Training images are resized to 512×512, while keeping the original aspect ratio. All test images keep their original size. Training is done with the SGD, learning rate $7e^{-3}$, batch size 8, and in 10,000 iterations. For the contribution 2, the number of experts N is analyzed in Sec. 5. For updating f_s^n in (7), we use Adam optimizer [13] with $\nu = 1e^{-2}$.

Complexity. In training, prior work [32, 20, 21]: (1) Uses a CNN with complexity $O(\text{CNN})$ for extracting features from the support and query images, (2) Computes the similarity map with complexity $O(d\ w\ h)$, and (3) [32] additionally uses a convolutional network for segmenting the query with complexity $O(\text{Conv})$. Note that $O(\text{Conv}) = O(d\ w\ h)$ as both the similarity map and convolutions in the Conv network are computed over feature maps with the size $d \times w \times h$. Also, note that $O(d\ w\ h)$ is significantly smaller than $O(\text{CNN})$. Our contribution 1 additionally computes the feature relevance using the closed-form solution with a linear complexity in the size of the feature maps $O(d\ w\ h)$. Therefore, our total training complexity is equal to that of prior work [32, 20, 21]: $O(\text{Train}) = O(\text{CNN}) + O(d\ w\ h)$.

In testing, complexity of prior work [32, 20, 21] is the same as $O(\text{Train})$. Our contribution 2 increases complexity in testing by additionally estimating the ensemble of N segmentations of the query image. Therefore, in testing, our complexity is $O(\text{Test}) = O(\text{CNN}) + O(N\ d\ w\ h)$. Thus, in testing, we increase only the smaller term of the total complexity. For small N, we have that the first term $O(\text{CNN})$ still dominates the total complexity. As we show in Sec. 5, for $N = 10$, we significantly outperform the state of the art, which justifies our slight increase in testing complexity.

5. Experiments

Datasets. For evaluation, we use two datasets: (a) PASCAL-5^i which combines images from the PASCAL VOC 2012 [7] and Extended SDS [11] datasets; and (b) COCO-20^i which is based on the MSCOCO dataset [16]. For PASCAL-5^i, we use the same 4-fold cross-validation setup as prior work [26, 20, 6]. Specifically, from the 20 object classes in PASCAL VOC 2012, for each fold $i = 0, ..., 3$, we sample five as test classes, and use the remaining 15 classes for training. Tab. 1 specifies our test classes for each fold of PASCAL-5^i. As in [26], in each fold i, we use 1000 support-query pairs of test images sampled from the selected five test classes.

We create COCO-20^i for evaluation on a more challenging dataset than PASCAL-5^i, since MSCOCO has 80 object classes and its ground-truth segmentation masks have

Dataset	Test classes
PASCAL-5^0	aeroplane, bicycle, bird, boat, bottle
PASCAL-5^1	bus, car, cat, chair, cow
PASCAL-5^2	diningtable, dog, horse, motorbike, person
PASCAL-5^3	potted plant, sheep, sofa, train, tv/monitor

Table 1. Evaluation on PASCAL-5^i uses the 4-fold cross-validation. The table specifies 5 test classes used in each fold $i = 0, ..., 3$. The remaining 15 classes are used for training.

lower quality than those in PASCAL VOC 2012. To the best of our knowledge, no related work has reported one-shot object segmentation on MSCOCO. For evaluation on COCO-20^i, we use 4-fold cross-validation. From the 80 object classes in MSCOCO, for each fold $i = 0, ..., 3$, we sample 20 as test classes, and use the remaining 60 classes for training. Tab. 2 specifies our test classes for each fold of COCO-20^i. In each fold, we sample 1000 support-query pairs of test images from the selected 20 test classes.

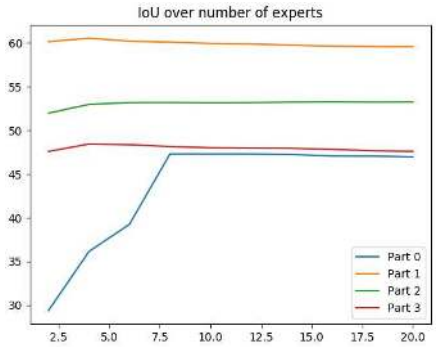

Figure 4. The mIoU of B+C1+C2 as a function of the number of experts N in the one-shot setting on PASCAL-5^i. Part 0 – Part 3 denote the four folds of cross-validation.

COCO-20^0		COCO-20^1		COCO-20^2		COCO-20^3	
1	Person	2	Bicycle	3	Car	4	Motorcycle
5	Airplane	6	Bus	7	Train	8	Truck
9	Boat	10	T.light	11	Fire H.	12	Stop
13	Park meter	14	Bench	15	Bird	16	Cat
17	Dog	18	Horse	19	Sheep	20	Cow
21	Elephant	22	Bear	23	Zebra	24	Giraffe
25	Backpack	26	Umbrella	27	Handbag	28	Tie
29	Suitcase	30	Frisbee	31	Skis	32	Snowboard
33	Sports ball	34	Kite	35	B. bat	36	B. glove
37	Skateboard	38	Surfboard	39	T. racket	40	Bottle
41	W. glass	42	Cup	43	Fork	44	Knife
45	Spoon	46	Bowl	47	Banana	48	Apple
49	Sandwich	50	Orange	51	Broccoli	52	Carrot
53	Hot dog	54	Pizza	55	Donut	56	Cake
57	Chair	58	Couch	59	P. plant	60	Bed
61	D. table	62	Toilet	63	TV	64	Laptop
65	Mouse	66	Remote	67	Keyboard	68	Cellphone
69	Microwave	70	Oven	71	Toaster	72	Sink
73	Fridge	74	Book	75	Clock	76	Vase
77	Scissors	78	Teddy	79	Hairdrier	80	Toothbrush

Table 2. Evaluation on COCO-20^i uses the 4-fold cross-validation. The table specifies 20 test classes used in each fold $i = 0, ..., 3$. The remaining 60 classes are used for training.

	Phase	B	B+C1	B+C2	B+C1+C2
VGG	Train	171	172	171	172
	Test	148	150	211	211
ResNet	Train	386	388	386	388
	Test	268	280	360	360

Table 3. Training and test times per example in milliseconds, with 1 Nvidia 1080 Ti GPU on PASCAL-5^i, for different ablations indicated in the top row. Using C2 increases only the test time. Specifically, we use $K = 10$ in our experiments

Metrics. As in [26, 20, 6], we use the mean intersection-over-union (mIoU) for quantitative evaluation. IoU of class l is defined as $\text{IoU}_l = \frac{TP_l}{TP_l+FP_l+FN_l}$, where TP, FP and FN are the number of pixels that are true positives, false positives and false negatives of the predicted segmentation masks, respectively. The mIoU is an average of the IoUs of different classes, $\text{mIoU} = \frac{1}{n_l}\sum_l \text{IoU}_l$, where n_l is the number of test classes. We report the mIoU averaged over the four folds of cross-validation.

Baselines, Ablations, and Variants of Our Approach. As a baseline B, we consider the approach depicted in the gray box "PRIOR WORK" in Fig. 2 and specified in Sec. 3.1 before our contribution 1. We also consider several ablations of our approach: B+C1 – extends the baseline B with contribution 1 only; B+C2 – extends the baseline B with contribution 2 only; and B+C1+C2 – represents our full approach. These ablations are aimed at testing the effect of each of our contributions on performance. In addition, we consider two alternative neural networks – VGG 16 and ResNet 101 – as the CNN for extracting image features. VGG 16 has also been used in prior work [26, 20, 6]. We also compare with an approach called Upper-bound that represents a variant of our full approach B+C1+C2 trained such that both training and testing datasets consist of the same classes. As Upper-bound does not encounter new classes in testing, it represents an upper bound of our full approach. Finally, in the K-shot setting, we consider another baseline called Average. It represents our full approach B+C1+C2 that first independently predicts segmentations of the query image for each of the $K > 1$ support images, and then averages all of the predictions. Our approach for the K-shot setting is called Our-K-shot, and differs from Average in that we rather jointly analyze all of the K support images than treat them independently, as explained in Sec. 3.3.

Training/testing time. The training/testing time is reported in Tab. 3. We can see that the contribution 1 just adds very small computational overhead over the baseline but significantly outperforms the baseline. Additionally, although contribution 2 has substantially larger testing time (about 40% with VGG backbone and 35% with ResNet backbone compare to the baseline), but it yields more significant performance gain than contribution 1 does.

Backbone	Methods	PASCAL-5^0	PASCAL-5^1	PASCAL-5^2	PASCAL-5^3	Mean
VGG 16	OSLSM [26]	33.60	55.30	40.90	33.50	40.80
	co-FCN [20]	36.70	50.60	44.90	32.40	41.10
	PL+SEG+PT [6]	-	-	-	-	42.70
	SG-One [32]	40.20	58.40	48.40	38.40	46.30
VGG 16	B	42.39	57.43	47.84	42.55	47.55
	B + C1	43.34	56.72	50.64	44.01	48.68
	B + C2	46.49	**60.27**	51.45	46.67	51.22
	B + C1 + C2	**47.04**	59.64	**52.61**	**48.27**	**51.90**
	Upper-bound	*51.33*	*64.20*	*61.72*	*58.02*	*58.82*
ResNet 101	B	43.24	60.82	52.58	48.57	51.30
	B + C1	46.06	61.22	54.90	48.65	52.71
	B + C2	47.46	63.76	54.11	51.50	54.21
	B + C1 + C2	**51.30**	**64.49**	**56.71**	**52.24**	**56.19**
	Upper-bound	*61.34*	*72.85*	*70.20*	*67.90*	*68.07*

Table 4. Mean IoU of one-shot segmentation on PASCAL-5^i. The best results are in bold.

Backbone	Methods	PASCAL-5^0	PASCAL-5^1	PASCAL-5^2	PASCAL-5^3	Mean
VGG 16	OSLSM [26]	35.90	58.10	42.70	39.10	43.95
	co-FCN [20]	37.50	50.00	44.10	33.90	41.38
	PL+SEG+PT [6]	-	-	-	-	43.70
	SG-One [32]	41.90	58.60	48.60	39.40	47.10
VGG 16	Average	48.01	59.43	54.53	48.50	52.62
ResNet 101	Average	51.72	64.99	61.05	53.34	57.78
VGG 16	Our-K-shot	**50.87**	**62.86**	**56.48**	**50.09**	**55.08**
ResNet 101	Our-K-shot	**54.84**	**67.38**	**62.16**	**55.30**	**59.92**

Table 5. Mean IoU of five-shot segmentation on PASCAL-5^i. The best results are in bold.

One-shot Segmentation. Tab. 4 compares our B+C1+C2 with the state of the art, ablations, and aforementioned variants in the one-shot setting on PASCAL-5^i. B+C1+C2 gives the best performance for both VGG 16 and ResNet 101, where the latter configuration significantly outperforms the state of the art with the increase in the mIoU averaged over the four folds of cross-validation by 13.49%. Relative to B, our first contribution evaluated with B+C1 gives relatively modest performance improvements. From the results for B+C2, our second contribution produces larger gains in performance relative to B and B+C1, suggesting that contribution 2 in and of itself is more critical than contribution 1. Interestingly, combining both contribution 1 and contribution 2 significantly improves the results relative to using either contribution only. We also observe that performance of our B+C1+C2 for some folds of cross-validation (e.g., PASCAL-5^2 and PASCAL-5^3) comes very close to that of Upper-bound, suggesting that our approach is very effective in generalizing to new classes in testing. Fig. 4 shows the mIoU of B+C1+C2 as a function of the number of experts N in the one-shot setting on PASCAL-5^i. As can be seen, for $N \geq 10$ our approach is not sensitive to a particular choice of N. We use $N = 10$ as a good trade-off between complexity and accuracy.

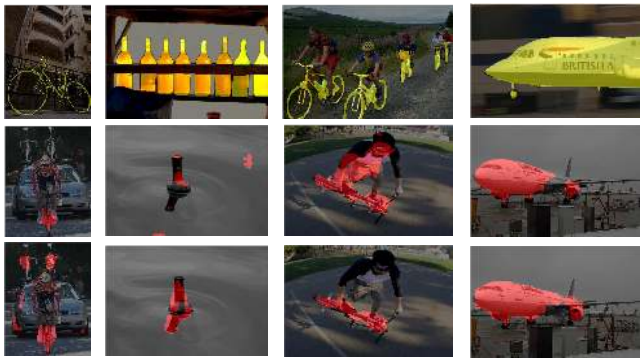

Figure 5. Examples from PASCAL-5^0. Top row: the support images with ground-truth segmentations in yellow. Middle row: the query images and segmentations in red predicted by B+C1+C2 in the one-shot setting. Bottom row: the query images and segmentations in red predicted by Our-K-shot in the five-shot setting (the remaining four support images are not shown). Our-K-shot in general improves performance over B+C1+C2, as Our-K-shot effectively uses the higher level of supervision in the five-shot setting. Best viewed in color.

Five-shot Segmentation. Tab. 5 compares Our-K-shot with the state of the art and Average in the five-shot setting

Backbone	Methods	COCO-20^0	COCO-20^1	COCO-20^2	COCO-20^3	Mean
VGG 16	B	12.13	11.65	14.86	22.09	13.26
	B+C1	15.65	13.91	15.36	23.50	17.11
	B+C2	13.28	14.91	19.50	23.22	17.73
	B+C1+C2	**18.35**	**16.72**	**19.59**	**25.43**	**20.02**
ResNet 101	B	13.11	14.80	14.54	26.48	17.23
	B+C1	15.48	14.76	16.85	28.35	18.86
	B+C2	14.09	17.82	18.46	27.85	19.56
	B+C1+C2	**16.98**	**17.98**	**20.96**	**28.85**	**21.19**

Table 6. The mIoU of ablations in the one-shot setting on COCO-20^i.

Backbone	Methods	COCO-20^0	COCO-20^1	COCO-20^2	COCO-20^3	Mean
VGG 16	Average	20.76	16.87	20.55	27.61	21.45
ResNet 101	Average	18.73	18.46	21.27	29.20	21.92
VGG 16	Our-K-shot	**20.94**	**19.24**	**21.94**	**28.39**	**22.63**
ResNet 101	Our-K-shot	**19.13**	**21.46**	**23.93**	**30.08**	**23.65**

Table 7. The mIoU of B+C1+C2 in the five-shot setting on COCO-20^i.

on PASCAL-5^i. Our-K-shot gives the best performance for both VGG 16 and ResNet 101, where the latter configuration significantly outperforms the state of the art with the increase in the mIoU averaged over the four folds of cross-validation by 15.97%. In comparison with Average, the joint analysis of K support images by Our-K-shot appears to be more effective, as Our-K-shot gives superior performance in every fold of cross-validation.

Results on COCO-20^i. Tab. 6 and Tab. 7 shows our ablations' results in the one-shot and five-shot settings on COCO-20^i. The former results are obtained with B+C1+C2 and the latter, with Our-K-shot. The lower values of mIoU relative to those in Tab. 4 and Tab. 5 indicate that COCO-20^i is more challenging than PASCAL-5^i. Surprisingly, in fold COCO-20^0, B+C1+C2 with VGG 16 outperforms its counterpart with ResNet 101 in the one-shot setting. The same holds for Our-K-shot in the five-shot setting. On average, using ResNet 101 gives higher results. As expected, the increased supervision in the five-shot setting in general gives higher accuracy than the one-shot setting.

Qualitative Results. Fig. 5 shows challenging examples from PASCAL-5^0, and our segmentation results obtained with B+C1+C2 with ResNet 101 for the one-shot setting, and Our-K-shot with ResNet 101 for the five-shot setting. In the leftmost column, the bike in the support image has different pose from the bike in the query image. While this example is challenging for B+C1+C2, our performance improves when using Our-K-shot. In the second column from left, the query image shows a partially occluded target – a part of the bottle. With five support images, Our-K-shot improves performance by capturing the bottle's shadow. The third column from left shows that the bike's features in the support image are insufficiently discriminative as the person also gets segmented along with the bike. With more examples, the bike is successfully segmented by Our-K-shot. In the rightmost column, the plane in the support image is partially occluded, and thus in the query image B+C1+C2 can only predict the head of the airplane while Our-K-shot's predicted segment covers most of the airplane.

6. Conclusion

We have addressed one-shot and few-shot object segmentation, where the goal is to segment a query image, given a support image and the support's ground-truth segmentation. We have made two contributions. First, we have formulated an optimization problem that encourages high feature responses on the foreground and low feature activations on the background for more accurate object segmentation. Second, we have specified the gradient boosting of our model for fine-tuning to new classes in testing. Both contributions have been extended to the few-shot setting for segmenting the query by jointly analyzing the provided support images and their ground truths, rather than treating the support images independently. For evaluation, we have compared with prior work, strong baselines, ablations and variants of our approach on the PASCAL-5^i and COCO-20^i datasets. We significantly outperform the state of the art on both datasets and in both one-shot and five-shot settings. Using only the second contribution gives better results than using only the first contribution. Our integration of both contributions gives a significant gain in performance over each.

Acknowledgement. This work was supported in part by DARPA XAI Award N66001-17-2-4029 and AFRL STTR AF18B-T002.

References

[1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[2] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014.

[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. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2018.

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

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

[6] Nanqing Dong and Eric P Xing. Few-shot semantic segmentation with prototype learning. In *BMVC*, volume 3, page 4, 2018.

[7] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

[8] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70 (ICML)*, pages 1126–1135. JMLR. org, 2017.

[9] Yoav Freund, Robert Schapire, and Naoki Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

[10] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[11] Bharath Hariharan, Pablo Arbeláez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *2011 International Conference on Computer Vision*, pages 991–998. IEEE, 2011.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[14] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 109–117. Curran Associates, Inc., 2011.

[15] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[16] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollr. Microsoft coco: Common objects in context. In *European conference on computer vision (ECCV)*, June 2016.

[17] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[18] Alex Nichol and John Schulman. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2, 2018.

[19] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

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

[21] Kate Rakelly, Evan Shelhamer, Trevor Darrell, Alexei A Efros, and Sergey Levine. Few-shot segmentation propagation with guided networks. *arXiv preprint arXiv:1806.07373*, 2018.

[22] Mengye Ren, Sachin Ravi, Eleni Triantafillou, Jake Snell, Kevin Swersky, Josh B. Tenenbaum, Hugo Larochelle, and Richard S. Zemel. Meta-learning for semi-supervised few-shot classification. In *International Conference on Learning Representations*, 2018.

[23] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *International Conference on Medical image computing and computer-assisted intervention*, 2015.

[24] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[25] Eli Schwartz, Leonid Karlinsky, Joseph Shtok, Sivan Harary, Mattias Marder, Abhishek Kumar, Rogerio Feris, Raja Giryes, and Alex Bronstein. Delta-encoder: an effective sample synthesis method for few-shot object recognition. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2850–2860, 2018.

[26] Amirreza Shaban, Shray Bansal, Zhen Liu, Irfan Essa, and Byron Boots. One-shot learning for semantic segmentation. In *Proceedings of the 28th British Machine Vision Conference (BMVC)*, 2017.

[27] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[28] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 4077–4087, 2017.

[29] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1199–1208, 2018.

[30] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems (NIPS)*, pages 3630–3638, 2016.

[31] Yu-Xiong Wang, Ross Girshick, Martial Hebert, and Bharath Hariharan. Low-shot learning from imaginary data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7278–7286, 2018.

[32] Xiaolin Zhang, Yunchao Wei, Yi Yang, and Thomas Huang. Sg-one: Similarity guidance network for one-shot semantic segmentation. *arXiv preprint arXiv:1810.09091*, 2018.

[33] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.