Few-Shot Semantic Segmentation with Prototype Learning

Nanqing Dong [1] [2] and Eric. P. Xing [2]

[1] Cornell University, [2] Petuum, Inc.

nd367@cornell.edu

Abstract

Semantic segmentation assigns a class label to each image pixel. This dense prediction problem requires large amounts of manually annotated data, which is often unavailable. Few-shot learning aims to learn the pattern of a new category with only a few annotated examples. In this paper, we formulate the few-shot semantic segmentation problem from 1-way (class) to N-way (classes). Inspired by few-shot classification, we propose a generalized framework for few-shot semantic segmentation with an alternative training scheme. The framework is based on prototype learning and metric learning. Our approach outperforms the baselines by a large margin.

Introduction

Few-shot learning aims to learn the pattern of new concepts unseen in the training data, given only a few labeled examples. In few-shot classification, each image only has one label. However, in few-shot semantic segmentation, each image can contain multiple semantic classes. In an N-way k-shot semantic segmentation task, there are a *support set* and a *query*. Each of N classes in the support set has k image and pixel-level annotation pairs. Given a support set, we want to predict the segmentation mask for N classes for a query image. Previous studies [4, 2] on few-shot semantic segmentation are special cases of $N = 1$. In this work, we propose our prototype-based few-shot semantic segmentation framework, which is based on cognitive *prototype theory* [3] and *prototypical networks* [5].
The main contributions of this work:

- We formulate the N-way k-shot semantic segmentation problem.
- We propose a prototype-based framework which is efficient for few-shot semantic segmentation.
- We explore a few techniques to address the overfitting problem in the training process.
- We demonstrate the effectiveness of distance metric learning and nearest neighbor classification in the few-shot semantic segmentation.

N-way k-shot Semantic Segmentation

Let $S = \{(x^i, y^i)\}_{i=1}^{N_S}$ denote the support set, where x^i represents an image with shape $[H^i, W^i, 3]$ and y^i represents the corresponding annotation for x^i. y^i is a binary mask with shape $[H^i, W^i, 1]$ for certain semantic class. x^q is the query image with shape $[H^q, W^q, 3]$, which is not in S. We allow $x^i = x^j$ as long as $y^i \neq y^j$.

The model is provided with k image-mask pairs for each of N classes (excluding the background) which are not seen in the training. We have $N_S = N \times k$ for the support set. Given a new image, the goal is to learn a segmentation model $\mathcal{F}_\Theta$ to predict the segmentation mask for the N classes. The goal can be interpreted as learning a mapping $S \rightarrow \mathcal{F}_\Theta(\cdot, S)$. Given x^q, the mapping will define a probability distribution over outputs $\mathcal{F}_\Theta(x^q, S)$. Here, different from the binary mask for single-class annotation in S, $\mathcal{F}_\Theta(x^q, S)$ and the ground truth y^q both have a shape $[H^q, W^q, N+1]$.

For each episode during the training, N classes are randomly selected at first. Then a support set S and a query image-annotation pair (x^q, y^q) are randomly selected based on chosen N classes. The training objective is thus to minimize the pixel-wise multi-class cross-entropy loss J_Θ,

$$J_\Theta(x^q, y^q) = -\frac{1}{H^q \times W^q} \sum_j \sum_c y^q_{j,c} \ln \mathcal{F}_\Theta(x^q, S)_{j,c} \tag{1}$$

, where j ranges over all the spatial positions and $c \in \{1, ..., N+1\}$.

Few-Shot Semantic Segmentation with Prototype Learning

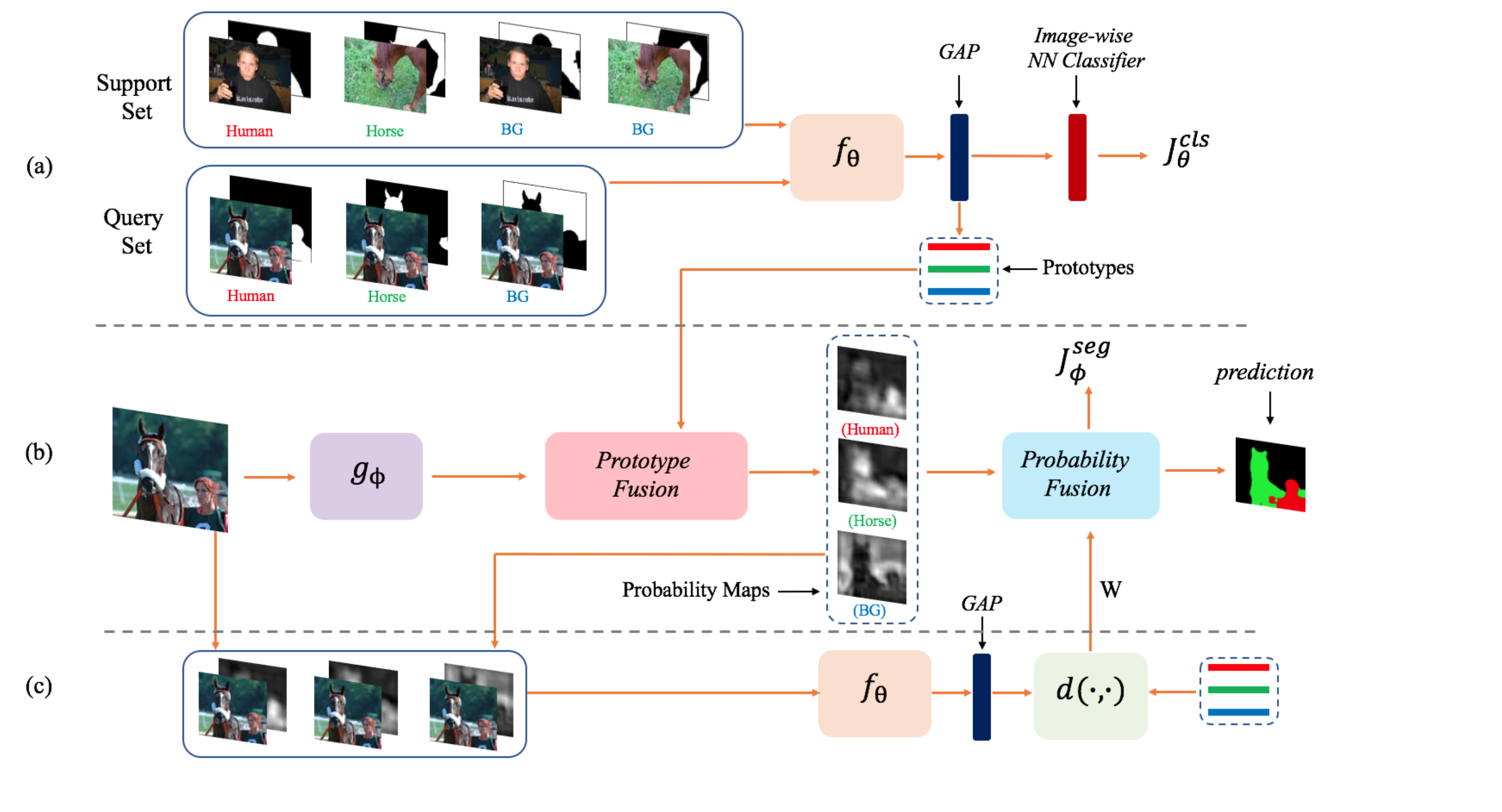


Figure 1: Illustration of the model architecture and main data flow for a 2-way 1-shot task. (a) is the prototype learner branch. The prototype learner classifies the query set given the support set and learns the prototypes. (b) is the segmentor branch. The segmentor takes the query image and the prototypes learned from (a) to output the prediction segmentation masks. (c) demonstrates how the weight W used in fusion the probability maps is calculated. Each probability map is concatenated with the query image, the same as the query set, then fed into the prototype learner to produce a feature vector. A similarity measure function $d(\cdot, \cdot)$ takes the feature vector and the prototype to output a similarity score.

Prototype Learner Branch Let f_θ denotes a feature extractor with parameters θ in the prototype learner branch. Following [4], the input to f_θ is x^i masked by y^i (element-wise multiplication), which has a shape $[H^i, W^i, 3]$. f_θ embeds the input into feature maps with M channels. We use a global average pooling layer (GAP) to filter out the spatial information from the feature maps. The output of GAP is a M-dimensional feature vector. Assume S_c is a subset of S which only contains semantic class c, we define the prototype of class c as the mean feature vectors for that class:

$$\boldsymbol{p}_c = \frac{1}{|S_c|} \sum_{(x^i, y^i) \in S_c} GAP(f_\theta(x^i, y^i)) \tag{2}$$

, where $|S_c| = k$. In addition to the prototypes of the N semantic classes, we introduce one more prototype for the background (BG). As illustrated in Figure 1, the raw images in S and the binary masks indicating the BG make a new set $S_{BG} = \{(x^i, y^i_{BG})\}$. The prototype of BG is calculated using Equation 2 where $|S_{BG}| = Nk$.

Distance Metric Learning The prototypes are defined in the projected space where the distance metric is learned through f_θ. We can learn more representative prototypes by learning a better distance metric in the prototype learner branch. After we get the prototypes $\boldsymbol{p}$, we use a non-parametric weighted nearest neighbor classifier to categorize the semantic class. For N-way learning task, y^q can be decomposed into $N+1$ binary masks y^q_c where $c \in \{1, ..., N+1\}$. The optimization goal is to maximize

$$p_\theta(y = c|(x^q, y^q_c)) = \frac{\exp(d(GAP(f_\theta(x^q, y^q_c)), \boldsymbol{p}_c))}{\sum_{c'=1}^{N+1} \exp(d(GAP(f_\theta(x^q, y^q_c)), \boldsymbol{p}_{c'}))} \tag{3}$$

, where $d(\cdot, \cdot)$ stands for a similarity measure function. Let J_θ^{cls} be the auxiliary loss for the prototype learner branch which is optimized alternatively with the $J_\phi^{seg}(x^q, y^q)$ from the segmentation branch.

$$J_\theta^{cls} = -\frac{1}{N+1} \sum_t \sum_c I_{c=t} \log(p_\theta(y = c|(x^q, y^q_t))) \tag{4}$$

, where $I_{c=t}$ is a binary indicator function.

Segmentor Branch Let g_ϕ denotes another feature extractor with parameters ϕ in the segmentation branch. We apply an unpooling layer (UP) [1] to restore the prototypes into feature maps with the same shape of $g_\phi(x^q)$. We fuse $g_\phi(x^q)$ with each of the $N+1$ restored prototypes by element-wise addition. Assume $\mathfrak{m}$ stands for the feature maps for a semantic class (including the BG), we have

$$\boldsymbol{m}_c = g_\phi(x^q) + UP(\boldsymbol{p}_c) \tag{5}$$

. The $\mathfrak{m}$ is compressed into a single-channel feature map with a 1×1 conv and an element-wise sigmoid operation.

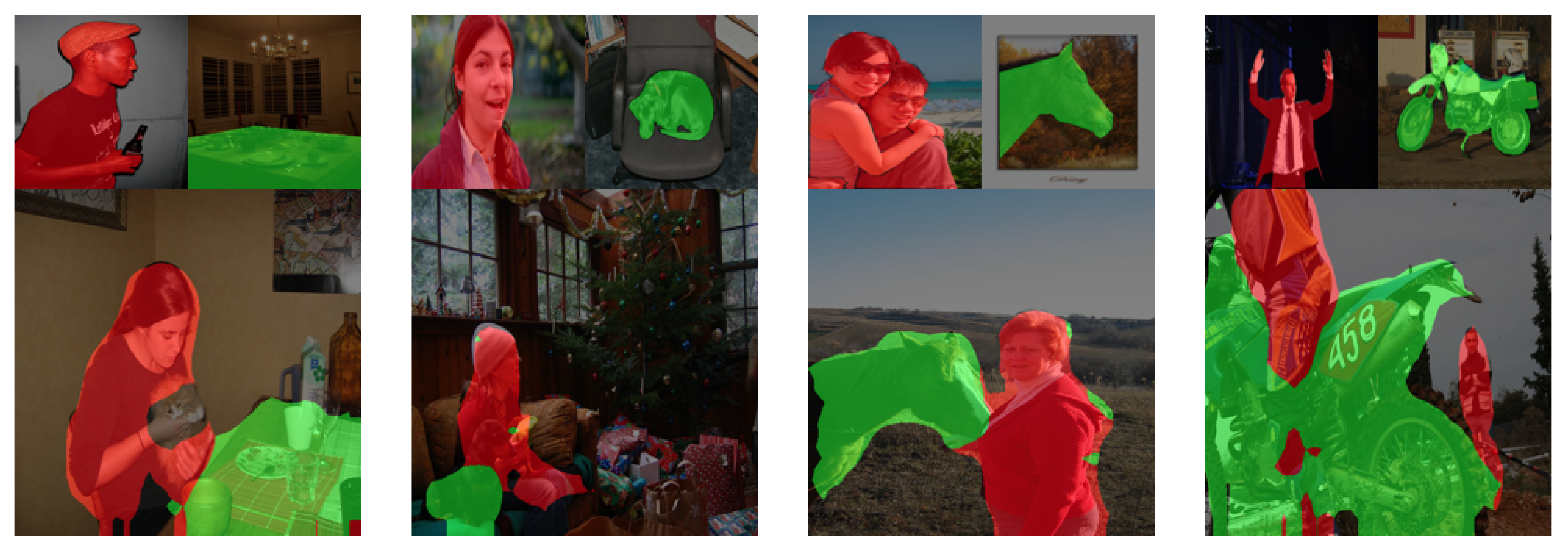
Non-parametric Fusion We propose to use a 1×1 conv followed by bilinear interpolation to produce the final logits, where the 1×1 conv is parameterized with a $[N+1, N+1]$ weight matrix W. Let α denotes the αth channel of logits before the softmax operation and β denotes the βth channel of feature maps after the sigmoid operation. With fixed θ,

$$W_{\beta,\alpha} = \frac{\exp(d(GAP(f_\theta(x^q, \hat{y}^q_\beta)), \boldsymbol{p}_\alpha))}{\sum_{c=1}^{N+1} \exp(d(GAP(f_\theta(x^q, \hat{y}^q_c)), \boldsymbol{p}_\alpha))} \tag{6}$$

, and $\hat{y}^q_\beta$ denotes the predicted probability map for class β. It is easy to show $\sum_\beta W_{\beta,\alpha} = 1$. The W is bounded with constraints and it does not require parameter tuning.

Permutation Training For an N-way learning task, there are $N+1$ classes. In the training, the concatenation of the single-channel feature maps have an order (which class comes first and which class comes last), which depends on the input order of the prototypes. There are $(N+1)!$ different orders in total. For an episode with S and (x^q, y^q), we train the model with different orders of the prototypes. In practice, we randomly select a subset of the whole orders when N is large. The core idea behind this technique is to make the network discriminative to the difference between prototypes, instead of memorizing all the semantic information.

Demo



(a) Dining Table (b) Dog (c) Horse (d) Motorbike

Figure 2: Qualitative results of our method for 2-way 1-shot semantic segmentation.

References

[1] Wei Liu, Andrew Rabinovich, and Alexander C Berg. Parsenet: Looking wider to see betters. In *International Conference on Learning Representations Workshop*, 2016.

[2] Kate Rakelly, Evan Shelhamer, Trevor Darrell, Alyosha Efros, and Sergey Levine. Conditional networks for few-shot semantic segmentation. In *International Conference on Learning Representations Workshop*, 2018.

[3] Eleanor H Rosch. Natural categories. *Cognitive Psychology*, 4(3):328–350, 1973.

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

[5] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, 2017.