Supplementary Material for Pixel Matching Network

1. Overview

We provide the implementation details, detailed analyses, and additional experiments. In particular:

1. In Section 2, we give the Implementation Details of our model.

2. In Section 3, we provide the details of the Conv1, Conv2, and Classifier Head in decoder module.

3. We conduct experiments on FSS-1000 dataset [2] in Section 4 and visualize some segmentation samples.

4. We provide the ablation analysis of both linear head and low-level representations in Section 5.

2. Implementation Details

Our model consists of two parts, backbone and segmentation head. Following the previous works, we conduct experiments on two backbones, *i.e.*, ResNet50 and ResNet101 [1], which are pre-trained on ImageNet and frozen during the training stage. The segmentation head is trained with the SGD optimizer in a 0.001 learning rate and 0.9 momentum. We train 100 epochs for PASCAL and FSS datasets, and 50 epochs for the COCO dataset. The batch size is set to 24 for both datasets. The images from both datasets are resized to 384 $\times$ 384. Our model is trained on the PyTorch with two NVIDIA Tesla A100 GPUs.

3. The Detailed Modules

3.1. Conv1 and Conv2

Conv1 and Conv2 are two convolutional blocks in the decoder module. As shown in Fig. 1, two multi-scale and multi-receptive-field convolutional networks *(in Fig. 1 (a), (b) right)* are introduced to replace the ordinary convolutional networks *(in Fig. 1 (a), (b) left)*. Each *conv block* contains a 2d convolution layer, a GroupNorm layer, and a ReLU layer. For *Conv1*, the feature map is enhanced with a 1×1 *conv block* and a 3×3 *conv block*. And the outputs of two *conv blocks* are concatenated in the channel dimension for multi-receptive-field features fusion, which is

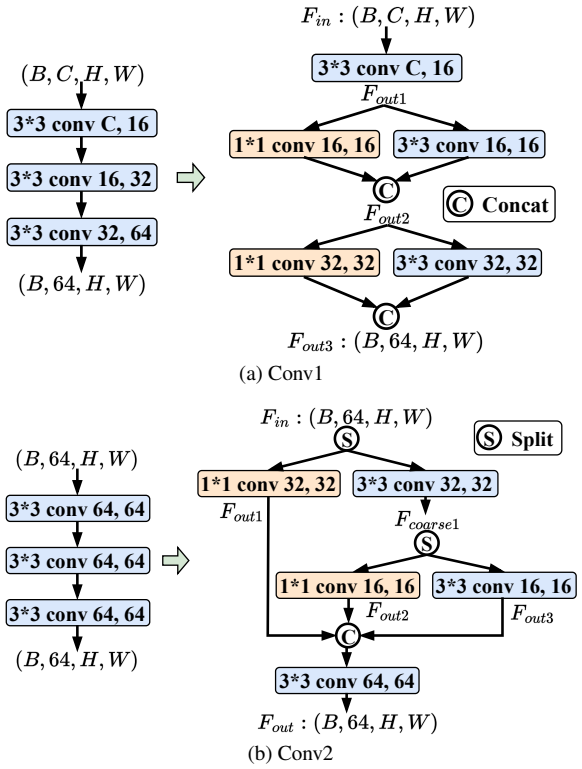

Figure 1. Illustration of the proposed Conv1 and Conv2 modules in the decoder. The right convolutional networks are strengthened from the left ones, which contain more convolutional layers and fewer parameters.

formulated as:

$$\begin{aligned} F_{out1} &= C_{3\times3}(F_{in}), \\ F_{out2} &= Concat(C_{1\times1}(F_{out1}), C_{3\times3}(F_{out1})), \\ F_{out3} &= Concat(C_{1\times1}(F_{out2}), C_{3\times3}(F_{out2})), \end{aligned} \tag{1}$$

where $Concat$ denotes concatenating two feature maps in the channel dimension, $C_{1\times1}$ and $C_{3\times3}$ denote a convlutional layer with the 1×1 and 3×3 kernels, separately.

For *Conv2*, the feature map F_{in} is split into two parts in the channel dimension, which are processed by a 1×1 *conv block* and a 3×3 *conv block* respectively, obtaining F_{out1} and $F_{coarse1}$. And the feature map $F_{coarse1}$ is split and processed by the mentioned two *conv blocks* again, ob-

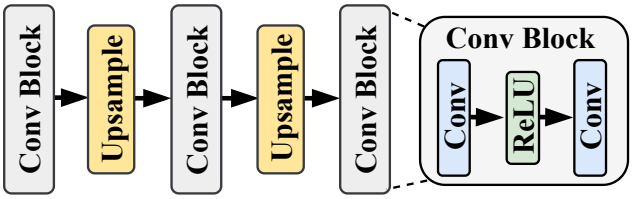

Figure 2. Illustration of the Classifier Head in the decoder.

taining F_{out2} and F_{out3}. Then, all the output features are concatenated in the channel dimension to match the size of the input feature F_{in}. The process is formulated as:

$$\begin{aligned} F_{out1}, F_{coarse1} &= Split(F_{in}), \\ F_{out1}, F_{coarse1} &= C_{1\times1}(F_{out1}), C_{3\times3}(F_{coarse1}), \\ F_{out2}, F_{out3} &= Split(F_{coarse1}), \\ F_{out2}, F_{out3} &= C_{1\times1}(F_{out2}), C_{3\times3}(F_{out3}), \\ F_{out} &= Concat(F_{out1}, F_{out2}, F_{out3}), \end{aligned} \tag{2}$$

where $Split$ denotes splitting the feature map in the channel dimension, $Concat$ denotes concatenating feature maps in the channel dimension, $C_{1\times1}$ and $C_{3\times3}$ denote a convlutional layer with the 1×1 and 3×3 kernels, separately.

As shown in Fig. 1, the *right* convolutional network, with fewer parameters, consists of more convolutional layers than the *left* one. With $Conv1$ and $Conv2$, the multi-scale and multi-receptive-field features are fused, which is important for the pixel-level segmentation task. Moreover, the increased convolutional layers provide the learning space for the affinity matrix to eliminate the noises brought by the freeze backbone, which benefits segmenting the query samples.

3.2. Classifier Head

The classifier head is designed to enhance the query foreground information in decoder module. As shown in Fig. 2, Classifier Head contains a series of convolutional blocks and two upsampling operations between two convolutional blocks. Each convolutional block consists of two convolutional layers connected with a ReLU operation.

4. Results on the FSS1000

As shown in Tab. 1, our method obtains the best performance with Resnet101 backbone on the FSS-1000 dataset, achieving 1.2% and 1.7% improvements under 1-shot and 5-shot settings, respectively. With Resnet50 backbone, our method achieves 87.9% under the 1-shot setting, as the second-best competitor, while the 5-shot result achieves 89.2%, surpassing the best competitor DCAMA by 0.4%. As shown in Fig. 3, HSNet is underfitting in many samples while our method performs better in all shown samples.

FSS1000					
backbone	methods	1-shot		5-shot	
		mIoU	FB-IoU	mIoU	FB-IoU
ResNet50	HSNet [4]	85.5	-	86.5	-
	DCAMA [5]	**88.2**	**92.5**	88.8	92.9
	CMNet [3]	82.5	-	83.8	-
	Our	87.9	92.3	**89.2**	**93.3**
ResNet101	DAN [6]	85.2	-	88.1	-
	HSNet [4]	86.5	-	88.5	-
	DCAMA [5]	88.3	92.4	89.1	93.1
	Our	**89.5**	**93.4**	**90.8**	**94.4**

Table 1. FSS performances (%) on FSS-1000 with different backbones (ResNet50 and ResNet101). The best results are marked in **bold**.

Benchmarks	SOTA	Our	Our + Linear Head
Cross-Category			
PASCAL5^i	66.8	65.4	66.2(+0.6)
COCO5^i	43.3	40.4	44.2(+3.8)
FSS1000	88.2	87.9	88.7(+0.8)
Cross-Dataset			
COCO5^i-PASCAL5^i	65.6	66.6	71.4(+4.8)
COCO5^i-FSS1000	81.9	84.3	82.6(-1.7)
PASCAL-FSS1000	78.6	84.6	84.1(-0.5)
Cross-Domain			
PASCAL-Deepglobe	37.9	37.1	36.5(-0.6)
PASCAL-ISIC2018	41.2	51.2	45.5(-5.7)
PASCAL-Chest Xray	66.6	70.4	58.3(-12.1)
PASCAL5^i-SUIM	34.7	34.8	34.7(-0.1)
Parameters(M)	2.6	0.68	11.2

Table 2. Impacts (%) of 'Linear Head' under 1-shot setting with Resnet50 backbone in 10 benchmarks.

Sources	Fold-0	Fold-1	Fold-2	Fold-3	Mean
None	66.1	68.1	60.6	54.9	62.4
Query and Support	67.3	70.5	62.3	57.5	64.4
Support	64.2	67.4	60.2	55.1	61.7
Query	67.3	72.0	62.4	59.9	65.4

Table 3. Impact (%) of low-level feature representations from different sources under 1-Shot setting with Resnet50 backbone on the PASCAL dataset. 'None' denotes that no low-level feature representations are fused into the decoder.

5. Ablation Study

Linear Head. Typically, a linear head is added after each block of the backbone to fit the segmentation task. In this experiment, we conduct experiments to evaluate its impacts on the FSS performances. The experimental re-

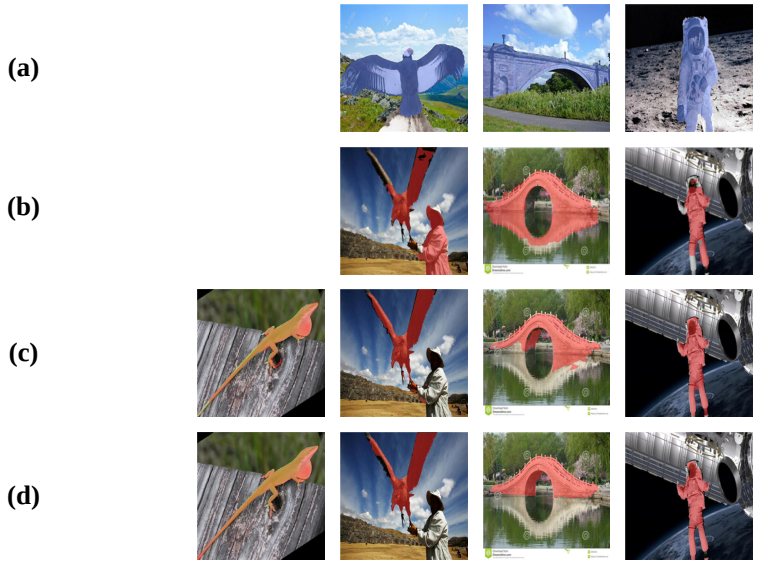

Figure 3. Some 1-Shot visualization on the FSS-1000 dataset with ResNet50 backbone. (a) Support image, (b) HSNet [4], (c) PMNet (Ours), (d) Ground-Truth.

sults in Tab. 2 show that adding the Linear Head would improve the FSS performance in most cross-class and cross-dataset tasks. Especially in COCO5^i, Linear Head takes 3.8% improvement. However, the model without Linear Head achieves better performance in all cross-domain tasks. We speculate that the linear head facilitates the training of both the cross-category and cross-dataset tasks but may cause the overfitting issue for the cross-domain tasks. To this end, we conclude that adding the linear head on each block of the backbone will boost the FSS performances under both cross-category and cross-dataset scenarios but hurt the performance under the cross-domain scenario. Note that adding the linear head would introduce a large number of parameters.

Impacts of low-level representations from different sources. The low-level feature representations are fused into the decoder for query segmentation. In this experiment, we conduct experiments to evaluate the impacts of low-level feature representations from different sources on the final performance. As shown in Tab. 3, we observe that fusing the low-level feature presentations from the support image would hurt the performance while the low-level presentations from the query image are beneficial for the segmentation results of the query image.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 1

[2] Xiang Li, Tianhan Wei, Yau Pun Chen, Yu-Wing Tai, and Chi-Keung Tang. Fss-1000: A 1000-class dataset for few-shot segmentation. In *CVPR*, pages 2869–2878, 2020. 1

[3] Weide Liu, Chi Zhang, Henghui Ding, Tzu-Yi Hung, and Guosheng Lin. Few-shot segmentation with optimal transport matching and message flow. *TMM*, 2022. 2

[4] Juhong Min, Dahyun Kang, and Minsu Cho. Hypercorrelation squeeze for few-shot segmentation. In *ICCV*, pages 6941–6952, 2021. 2, 3

[5] Xinyu Shi, Dong Wei, Yu Zhang, Donghuan Lu, Munan Ning, Jiashun Chen, Kai Ma, and Yefeng Zheng. Dense cross-query-and-support attention weighted mask aggregation for few-shot segmentation. In *ECCV*, pages 151–168, 2022. 2

[6] Haochen Wang, Xudong Zhang, Yutao Hu, Yandan Yang, Xianbin Cao, and Xiantong Zhen. Few-shot semantic segmentation with democratic attention networks. In *ECCV*, pages 730–746, 2020. 2