FaPN: Feature-aligned Pyramid Network for Dense Image Prediction (Supplementary Material)

Shihua Huang Zhichao Lu Ran Cheng* Cheng He
Southern University of Science and Technology
{shihuahuang95, luzhichaocn, ranchengcn, chenghehust}@gmail.com

In this supplementary material we include (1) additional training details in Section 1; (2) more details on the real-time semantic segmentation experiment in Section 2; and (3) additional qualitative visualizations to demonstrate the effectiveness of the proposed FaPN in Section 3.

1. Training Settings

For all experiments shown in the main paper, we use SGD optimizer with 0.9 momentum and 0.0001 weight decay. The standard data augmentation of horizontal flipping and scaling are also applied. In addition, the weights of the batch normalization [3] layers derived from the ImageNet pre-trained models are kept frozen. To be consistent with prior works, we have not incorporated any testing time augmentation tricks. For semantic segmentation, the model is trained for 65K iterations starting with a learning rate of 0.01 that is reduced by a factor of 10 at 40K and 55K. For the other three dense prediction tasks, the model is trained for 90K or 270K iterations with the initial learning rate of 0.02 that is reduced to 0.002 at 60K/210K and 0.0002 at 80K/250K. Our implementation is based on the Detectron2 [5] with the default configurations, *i.e.* to maintain a fair comparison with prior works, neither have we tuned any training hyperparameters nor used advanced data augmentations.

2. Real-time Semantic Segmentation Continued

With a lightweight ResNet (*e.g.* ResNet18/34) as the bottom-up backbone, we denote the feature maps output by the last three stages, *i.e.* conv3, conv4, conv5, as $\{\mathbf{C}_3, \mathbf{C}_4, \mathbf{C}_5\}$, respectively. At the beginning, we simply attach an FSM layer on $\mathbf{C}_5$ to produce the coarsest resolution feature maps $\mathbf{P}_5 \in \mathbb{R}^{128\times\frac{H}{32}\times\frac{W}{32}}$ (*i.e.* the output channel of FSM is 128). With a coarser-resolution feature map $\mathbf{P}_i$ ($l \in [4, 5]$), we upsample its spatial resolution by a factor of 2 using nearest neighbor upsampling [4] to obtain $\mathbf{P}_i^{up} \in \mathbb{R}^{128\times\frac{H}{2^{i-1}}\times\frac{W}{2^{i-1}}}$. Afterwards, an FAM layer is used to align $\mathbf{P}_i^{up}$ to its corresponding bottom-up feature map $\hat{\mathbf{C}}_{i-1} \in \mathbb{R}^{128\times\frac{H}{2^{i-1}}\times\frac{W}{2^{i-1}}}$ deriving from $\mathbf{C}_{i-1}$ by undergoing an FSM layer for channel reduction. Instead of element-wise addition, the aligned $\hat{\mathbf{P}}_i^{up}$ is then merged with $\hat{\mathbf{C}}_{i-1}$ by concatenation along with channel, and the merged feature map $\mathbf{P}_{i-1} \in \mathbb{R}^{256\times\frac{H}{2^{i-1}}\times\frac{W}{2^{i-1}}}$ which has the identical spatial size to $\mathbf{C}_{i-1}$ is further input in a Conv 1×1 layer to reduce its channels to 128. This process is iterated until the finest resolution map $\mathbf{P}_3 \in \mathbb{R}^{128\times\frac{H}{8}\times\frac{W}{8}}$ is generated. Finally, we append a prediction layer on $\mathbf{P}_3$ to generate the final semantic mask.

We train our models using the SGD optimizer with momentum and weight decay set to 0.9 and 0.0005, respectively. During training, we apply random horizontal flip and scale to input images, followed by a crop to a fixed size. The scale is randomly selected from $\{0.75, 1, 1.25, 1.5, 1.75, 2.0\}$, and the cropped resolutions are 1536×768 and 640×640 for Cityscapes [2] and COCO-Stuff [1], respectively. For all datasets, we set the batch size and the initial learning rate to 16 and 0.01, while the learning rate decays following a "poly" strategy, specifically $0.01 \times (1 - \frac{iter}{maxiters})^{0.9}$. Following the prior works [6, 7], we train models for 40K and 20K training iterations on Cityscapes and COCO-Stuff, respectively. In the evaluation process, we compute the inference speed using one Titan RTX GPU without any speed-up trick (*e.g.*, Apex or TensorRT) or optimized depth-wise convolutions, and evaluate the accuracy without any testing augmentation technique (*e.g.*, multi-crop or multi-scale testing).

We first demonstrate the effectiveness of each module in our proposed real-time semantic segmentation framework separately and then study different feature fusion methods on the Cityscapes *val* set. We use ResNet18 pre-trained on ImageNet as our backbone in the following ablation analysis.

The ablation experimental results are given in Table 1. Basically, the incorporation of FAM into the baseline improves the performance from 68.6% to 73.8%. Further-

*Corresponding author.

Table 1: **Ablation Study on Real-time Semantic Segmentation:** Detailed comparisons of each component in our proposed real-time semantic segmentation FaPN over Citsycapes *val* set in terms of accuracy, parameters and FLOPs (computational complexity).

method	mIoU	#Params (M)	FLOPs (G)
FPN	68.6	11.4	44.5
+ FAM	73.8	12.2	51.0
+ FAM + FSM	74.2	12.6	51.0
+ FAM + FSM + Concat	75.6	12.6	51.8

more, FSM improves the performance to 74.2% with only $0.4M$ additional parameters. Besides, when we replace the element-wise sum operation with concatenation for fusing the detailed feature and aligned semantic feature, another 1.4% improvement is achieved with few extra FLOPs.

Figure 1 visualizes the semantic segmentation results of FaPN on Cityscapes under real-time settings (FPS $\geq$ 30)*. Noticeably, the proposed Feature Align Module (FAM; third column in Figure 1) significantly improves the segmentation quality from the baseline (*i.e*. FPN; second column in Figure 1). With the feature selection module and feature concatenation, our final approach FaPN further improves the performance on real-time semantic segmentation.

3. Additional Visualization

Figure 2 and Figure 3 visualize the dense prediction performance on MS COCO. Evidently, our method achieves a more accurate segmentation on object boundaries and small objects.

Acknowledgments: This work was supported by the National Natural Science Foundation of China (Grant No. 61903178, 61906081, and U20A20306).

References

[1] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. COCO-Stuff: Thing and stuff classes in context. In *CVPR*, 2018. 1

[2] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 1

[3] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167*, 2015. 1

[4] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 1

[5] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 1

[6] Changqian Yu, Changxin Gao, Jingbo Wang, Gang Yu, Chunhua Shen, and Nong Sang. Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation. *arXiv:2004.02147*, 2020. 1

[7] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *ECCV*, 2018. 1

*https://paperswithcode.com/sota/real-time-semantic-segmentation-on-cityscapes

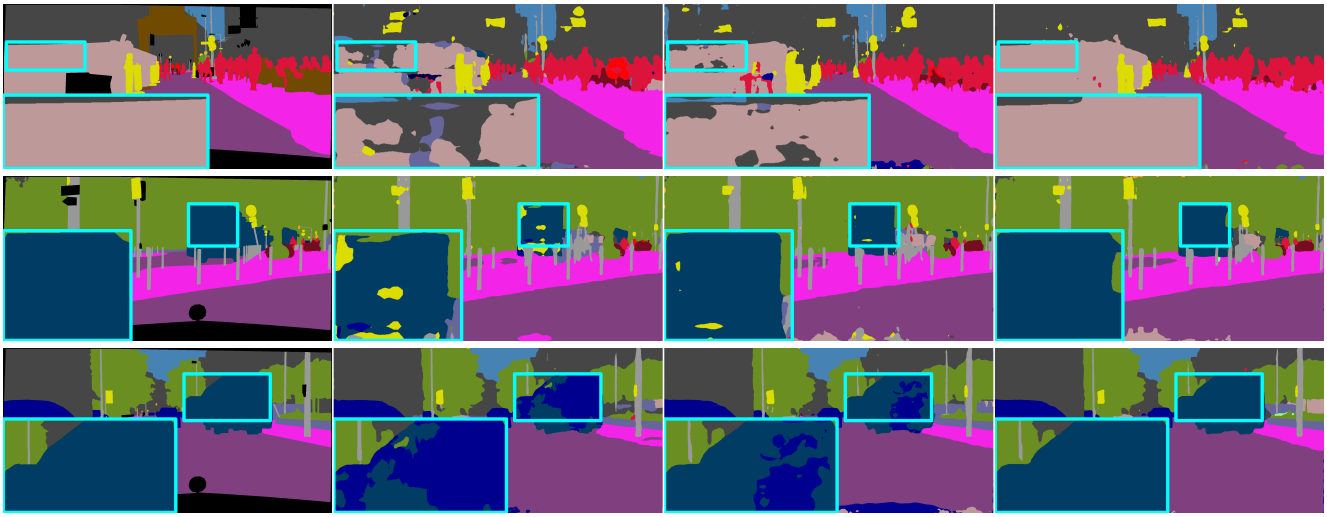

Figure 1: **Example visual comparisons among different approaches on the Cityscapes *val* set**. From left to right: ground truths, results from baseline, baseline with FAM and our FaPN, respectively. All models are using ResNet18.

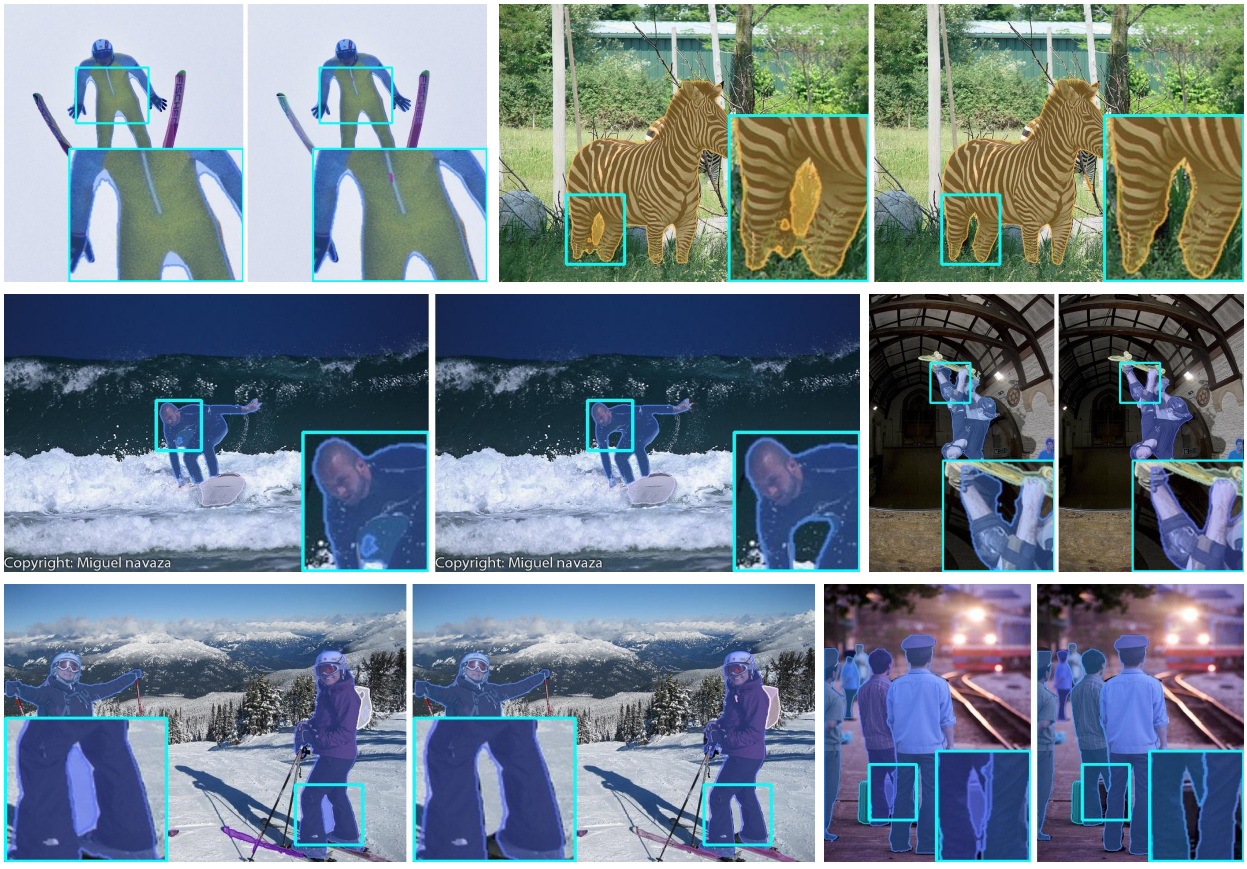

Figure 2: **Example pairs of instance segmentation results from FPN (*Left*) and our FaPN (*Right*) on object boundaries**. Both methods are implemented in Mask R-CNN with ResNet-50 being the backbone and PointRend as the mask head.

Figure 3: **Example results from methods with FPN or our FaPN on small objects**. In each group, from left to right, they are object detection, instance and panoptic segmentation, the top is achieved by FPN while the bottom by our FaPN. All models are based on ResNet50.