SegFix: Model-Agnostic Boundary Refinement for Segmentation

Yuhui Yuan[1,2,4⋆], Jingyi Xie[3⋆], Xilin Chen[1,2], and Jingdong Wang[4]

[1] Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS
[2] University of Chinese Academy of Sciences
[3] University of Science and Technology of China
[4] Microsoft Research Asia
{yuhui.yuan, jingdw}@microsoft.com, hsfzxjy@mail.ustc.edu.cn, xlchen@ict.ac.cn

Abstract. We present a model-agnostic post-processing scheme to improve the boundary quality for the segmentation result that is generated by any existing segmentation model. Motivated by the empirical observation that the label predictions of interior pixels are more reliable, we propose to replace the originally unreliable predictions of boundary pixels by the predictions of interior pixels. Our approach processes only the input image through two steps: (i) localize the boundary pixels and (ii) identify the corresponding interior pixel for each boundary pixel. We build the correspondence by learning a direction away from the boundary pixel to an interior pixel. Our method requires no prior information of the segmentation models and achieves nearly real-time speed. We empirically verify that our SegFix consistently reduces the boundary errors for segmentation results generated from various state-of-the-art models on Cityscapes, ADE20K and GTA5. Code is available at: https://github.com/openseg-group/openseg.pytorch.

Keywords: Semantic Segmentation; Instance Segmentation; Boundary Refinement; Model Agnostic

1 Introduction

The task of semantic segmentation is formatted as predicting the semantic category for each pixel in an image. Based on the pioneering fully convolutional network [43], previous studies have achieved great success as reflected by increasing the performance on various challenging semantic segmentation benchmarks [7,15,64].

Most of the existing works mainly addressed semantic segmentation through (i) increasing the resolution of feature maps [12,13,51], (ii) constructing more reliable context information [61,59,22,26,60] and (iii) exploiting boundary information [5,9,41,52]. In this work, we follow the 3[rd] line of work and focus on

⋆ Equal contribution.

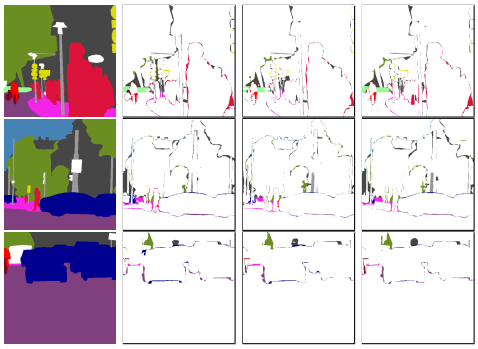


Fig. 1: **Qualitative analysis of the segmentation error maps.** The 1^{st} column presents the ground-truth segmentation maps, and the $2^{nd}/3^{rd}/4^{th}$ column presents the the error maps of DeepLabv3/HRNet/Gated-SCNN separately. These examples are cropped from Cityscapes `val` set. We can see that there exist many errors along the thin boundary for all three methods.

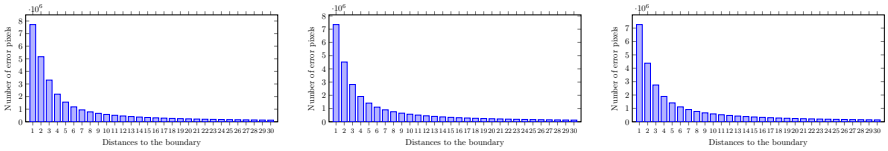
improving segmentation result on the pixels located within the thinning boundary[5] via a simple but effective model-agnostic boundary refinement mechanism.

Our work is mainly motivated by the observation that *most of the existing state-of-the-art segmentation models fail to deal well with the error predictions along the boundary.* We illustrate some examples of the segmentation error maps with DeepLabv3 [12], Gated-SCNN [52] and HRNet [51] in Figure 1. More specifically, we illustrate the statistics on the numbers of the error pixels *vs.* the distances to the object boundaries in Figure 2. We can observe that, for all three methods, the number of error pixels significantly decrease with larger distances to the boundary. In other words, predictions of the interior pixels are more reliable.

We propose a novel model-agnostic post-processing mechanism to reduce boundary errors by replacing labels of boundary pixels with the labels of corresponding interior pixels for a segmentation result. We estimate the pixel correspondences by processing the input image (without exploring the segmentation result) with two steps. The first step aims to localize the pixels along the object boundaries. We follow the contour detection methods [4,2,20] and simply use a convolutional network to predict a binary mask indicating the boundary pixels. In the second step, we learn a direction away from the boundary pixel to an interior pixel and identify the corresponding interior pixel by moving from the boundary pixel along the direction by a certain distance. Especially, our SegFix can reach nearly real-time speed with high resolution inputs.

Our SegFix is a general scheme that consistently improves the performance of various segmentation models across multiple benchmarks without any prior information. We evaluate the effectiveness of SegFix on multiple semantic segmentation benchmarks including Cityscapes, ADE20K and GTA5. We also extend SegFix to instance segmentation task on Cityscapes. According to the Cityscapes leaderboard, "HRNet + OCR + SegFix" and "PolyTransform + SegFix" achieve 84.5% and 41.2%, which rank the 1^{st} and 2^{nd} place on the semantic and instance segmentation track separately by the ECCV 2020 submission deadline.

[5] In this paper, we treat the pixels with neighboring pixels belonging to different categories as the boundary pixels. We use the distance transform to generate the ground-truth boundary map with any given width in our implementation.



(a) DeepLabv3 [12] (b) HRNet [51] (c) Gated-SCNN [52]

Fig. 2: **Histogram statistics of errors:** the number of error pixels *vs.* their (Euclidean) distances to the boundaries on Cityscapes `val` based on DeepLabv3/HRNet/Gated-SCNN. We can see that pixels with larger distance tend to be well-classified with higher probability and there exist many errors distributing within ~ 5 pixels width along the boundary.

2 Related Work

Distance/Direction Map for Segmentation: Some recent work [3,24,53] performed distance transform to compute distance maps for instance segmentation task. For example, [3,24] proposed to train the model to predict the truncated distance maps within each cropped instance. The other work [16,6,10,48] proposed to regularize the semantic or instance segmentation predictions with distance map or direction map in a multi-task mechanism. Compared with the above work, the key difference is that our approach does not perform any segmentation predictions and instead predicts the direction map from only the image, and then we refine the segmentation results of the existing approaches.

Level Set for Segmentation: Many previous efforts [47,8,29] have used the level set approach to address the semantic segmentation problem before the era of deep learning. The most popular formulation of level set is the signed distance function, with all the zero values corresponding to predicted boundary positions. Recent work [1,14,53,31] extended the conventional level-set scheme to deep network for regularizing the boundaries of predicted segmentation map. Instead of representing the boundary with a level set function directly, we implicitly encode the relative distance information of the boundary pixels with a boundary map and a direction map.

DenseCRF for Segmentation: Previous work [11,55,63,38] improved their segmentation results with the DenseCRF [34]. Our approach is also a kind of general post processing scheme while being simpler and more efficient for usage. We empirically show that our approach not only outperforms but also is complementary with the DenseCRF.

Refinement for Segmentation: Extensive studies [21,23,27,35,36] have proposed various mechanisms to refine the segmentation maps from coarse to fine. Different from most of the existing refinement approaches that depend on the segmentation models, to the best of our knowledge, our approach is the first model-agnostic segmentation refinement mechanism that can be applied to refine the segmentation results of any approach without any prior information.

Boundary for Segmentation: Some previous efforts [1,56,42,57] focused on localizing semantic boundaries. Other studies [5,52,17,41,40,30,18,19] also ex-

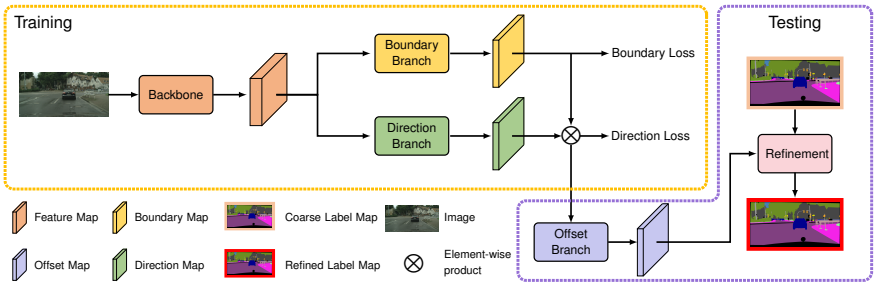


Fig. 3: **Illustrating the SegFix framework:** In the training stage, we first send the input image into a backbone to predict a feature map. Then we apply a boundary branch to predict a binary boundary map and a direction branch to predict a direction map and mask it with the binary boundary map. We apply boundary loss and direction loss on the predicted boundary map and direction map separately. In the testing stage, we first convert the direction map to offset map and then refine the segmentation results of any existing methods according to the offset map.

ploited the boundary information to improve the segmentation. For example, BNF [5] introduced a global energy model to consider the pairwise pixel affinities based on the boundary predictions. Gated-SCNN [52] exploited the duality between the segmentation predictions and the boundary predictions with a two-branch mechanism and a regularizer.

These methods [5,17,52,30] are highly dependent on the segmentation models and require careful re-training or fine-tuning. Different from them, SegFix does not perform either segmentation prediction or feature propagation and we instead refine the segmentation maps with an offset map directly. In other words, we only need to train a single unified SegFix model once w/o any further fine-tuning the different segmentation models (across multiple different datasets). We also empirically verify that our approach is complementary with the above methods, e.g., Gated-SCNN [52] and Boundary-Aware Feature Propagation [17].

Guided Up-sampling Network: The recent work [44,45] performed a segmentation guided offset scheme to address boundary errors caused by the bi-linear up-sampling. The main difference is that they do not apply any explicit supervision on their offset maps and require re-training for different models, while we apply explicit semantic-aware supervision on the offset maps and our offset maps can be applied to various approaches directly without any re-training. We also empirically verify the advantages of our approach.

Semantically Thinned Edge Alignment Learning (STEAL): The previous study STEAL [1] is the most similar work as it also predicts both boundary maps and direction maps (simultaneously) to refine the boundary segmentation results. To justify the main differences between STEAL and our SegFix, we summarize several key points as following: (i) STEAL predicts K independent boundary maps (associated with K categories) while SegFix only predicts a single boundary map w/o differentiating the different categories. (ii) STEAL first predicts the boundary map and then applies a fixed convolution on the boundary map to estimate the direction map while SegFix uses two parallel branches

to predict them independently. (iii) STEAL uses mean-squared-loss on the direction branch while SegFix uses cross-entropy loss (on the discrete directions). Besides, we empirically compare STEAL and our SegFix in the ablation study.

3 Approach

3.1 Framework

The overall pipeline of SegFix is illustrated in Figure 3. We first train a model to pick out boundary pixels (with the boundary maps) and estimate their corresponding interior pixels (with offsets derived from the direction maps) from only the image. We do not perform segmentation directly during training. We apply this model to generate offset maps from the images and use the offsets to get the corresponding pixels which should mostly be the more confident interior pixels, and thereby refine segmentation results from any segmentation model. We mainly describe SegFix scheme for semantic segmentation and we illustrate the details for instance segmentation in the Appendix.

Training stage. Given an input image $\mathbf{I}$ of shape $H \times W \times 3$, we first use a backbone network to extract a feature map $\mathbf{X}$, and then send $\mathbf{X}$ in parallel to (1) the *boundary branch* to predict a binary map $\mathbf{B}$, with 1 for the boundary pixels and 0 for the interior pixels, and (2) the *direction branch* to predict a direction map $\mathbf{D}$ with each element storing the direction pointing from the boundary pixel to the interior pixel. The direction map $\mathbf{D}$ is then masked by the binary map $\mathbf{B}$ to yield the input for the offset branch.

For model training, we use a binary cross-entropy loss as the boundary loss on $\mathbf{B}$ and a categorical cross-entropy loss as the direction loss on $\mathbf{D}$ separately.

Testing stage. Based on the predicted boundary map $\mathbf{B}$ and direction map $\mathbf{D}$, we apply the *offset branch* to generate a offset map $\Delta\mathbf{Q}$. A coarse label map $\mathbf{L}$ output by any semantic segmentation model will be refined as:

$$\widetilde{\mathbf{L}}_{\mathbf{p}_i} = \mathbf{L}_{\mathbf{p}_i + \Delta\mathbf{q}_i}, \tag{1}$$

where $\widetilde{\mathbf{L}}$ is refined label map, $\mathbf{p}_i$ represents the coordinate of the boundary pixel i, $\Delta\mathbf{q}_i$ is the generated offset vector pointing to an interior pixel, which is indeed an element of $\Delta\mathbf{Q}$. $\mathbf{p}_i + \Delta\mathbf{q}_i$ is the position of the identified interior pixel.

Considering that there might be some "fake" interior pixels [6] when the boundary is thick, we propose two different schemes as following: (i) re-scaling all the offsets by a factor, e.g., 2. (ii) iteratively applying the offsets (of the "fake" interior pixels) until finding an interior pixel. We choose (i) by default for simplicity as their performance is close.

During testing stage, we only need to generate the offset maps on test set *for once*, and could apply the same offset maps to refine the segmentation results from any existing segmentation model without requiring any prior information. In general, our approach is agnostic to any existing segmentation models.

[6] We use "fake" interior pixels to represent pixels (after offsets) that still lie on the boundary when the boundary is thick. Notably, we identify an pixel as interior pixel/boundary pixel if its value in the predicted boundary map $\mathbf{B}$ is 0/1.

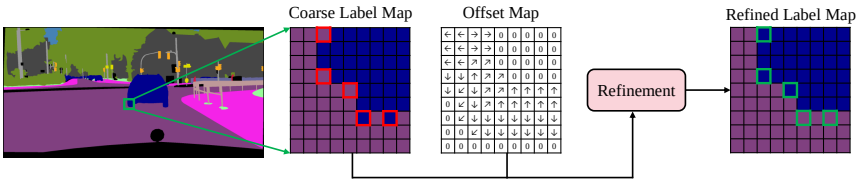


Fig. 4: **Illustrating the refinement mechanism of our approach:** we refine the coarse label map based on the offset map by replacing the labels of boundary pixels with the labels of (more) reliable interior pixels. We represent different offset vectors with different arrows. We mark the error positions in the coarse label map with □ and the corresponding corrected positions in the refined label map with □. For example, the top-most error pixel (class road) in the coarse label map is associated with a direction $\rightarrow$. We use the label (class car) of the updated position based on offset $(1, 0)$ as the refined label. Only one-step shifting based on the offset map already refines several boundary errors.

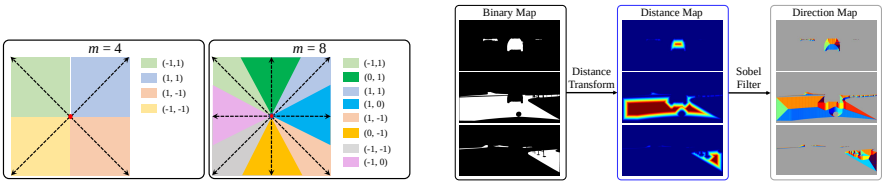
3.2 Network Architecture

Backbone. We adopt the recently proposed high resolution network (HRNet) [51] as backbone, due to its strengths at maintaining high resolution feature maps and our need to apply full-resolution boundary maps and direction maps to refine full-resolution coarse label maps. To further increase the resolution of the output feature map, we modify the original HRNet through adding an extra branch that maintains higher output resolution, i.e., $2\times$, called HRNet-$2\times$. We directly perform the boundary branch and the direction branch on the output feature map with the highest resolution. The resolution is $\frac{H}{s} \times \frac{W}{s} \times D$, where $s = 4$ for HRNet and $s = 2$ for HRNet-$2\times$. We empirically verify that our approach consistently improves the coarse segmentation results for all variations of our backbone choices in Section 4.2, e.g., HRNet-W18 and HRNet-W32.

Boundary branch/loss. We implement the boundary branch as 1×1 Conv $\rightarrow$ BN $\rightarrow$ ReLU with 256 output channels. We then apply a linear classifier (1×1 Conv) and up-sample the prediction to generate the final boundary map $\mathbf{B}$ of size $H \times W \times 1$. Each element of $\mathbf{B}$ records the probability of the pixel belonging to the boundary. We use binary cross-entropy loss as the boundary loss.

Direction branch/loss. Different from the previous approaches [1,3] that perform regression on continuous directions in $[0°, 360°)$ as the ground-truth, our approach directly predicts discrete directions by evenly dividing the entire direction range to m partitions (or categories) as our ground-truth ($m = 8$ by default). In fact, we empirically find that our discrete categorization scheme outperforms the regression scheme, e.g., mean squared loss in the angular domain [3], measured by the final segmentation performance improvements. We illustrate more details for the discrete direction map in Section 3.3.

We implement the direction branch as 1×1 Conv $\rightarrow$ BN $\rightarrow$ ReLU with 256 output channels. We further apply a linear classifier (1×1 Conv) and up-sample the classifier prediction to generate the final direction map $\mathbf{D}$ of size $H \times W \times m$. We mask the direction map $\mathbf{D}$ by multiplying by the (binarized) boundary map $\mathbf{B}$ to ensure that we only apply direction loss on the pixels identified as boundary



(a) Illustrating the directions → offsets. (b) Ground-truth generation procedure.

Fig. 5: (a) We divide the entire direction value range $[0°, 360°)$ to m partitions or categories (marked with different colors), For example, when $m = 4$, we have $[0°, 90°)$, $[90°, 180°)$, $[180°, 270°)$ and $[270°, 360°)$ correspond to 4 different categories separately. The above 4 direction categories correspond to offsets $(1, 1)$, $(-1, 1)$, $(-1, -1)$ and $(1, -1)$ respectively. The situation for $m = 8$ is similar. (b) Binary maps → Distance maps → Direction maps. The ground-truth binary maps are of category car, road and side-walk. We first apply distance transform on each binary map to compute the ground-truth distance maps. Then we use Sobel filter on the distance maps to compute the ground-truth direction maps. We choose different colors to represent different distance values or the direction values.

by the boundary branch. We use the standard category-wise cross-entropy loss to supervise the discrete directions in this branch.

Offset branch. The offset branch is used to convert the predicted direction map $\mathbf{D}$ to the offset map $\Delta\mathbf{Q}$ of size $H \times W \times 2$. We illustrate the mapping mechanism in Figure 5 (a). For example, the "upright" direction category (corresponds to the value within range $[0°, 90°)$) will be mapped to offset $(1, 1)$ when $m = 4$. Last, we generate the refined label map through shifting the coarse label map with the grid-sample scheme [28]. The process is shown in Figure 4.

3.3 Ground-truth generation and analysis

There may exist many different mechanisms to generate ground-truth for the boundary maps and the direction maps. In this work, we mainly exploit the conventional distance transform [32] to generate ground-truth for both semantic segmentation task and the instance segmentation task.

We start from the ground-truth segmentation label to generate the ground-truth distance map, followed by boundary map and direction map. Figure 5 (b) illustrates the overall procedure.

Distance map. For each pixel, our distance map records its minimum (Euclidean) distance to the pixels belonging to other object category. We illustrate how to compute the distance map as below.

First, we decompose the ground-truth label into K binary maps associated with different semantic categories, e.g., car, road, sidewalk. The k^{th} binary map records the pixels belonging to the k^{th} semantic category as 1 and 0 otherwise. Second, we perform distance transform [32] [7] on each binary map independently to compute the distance map. The element of k^{th} distance map encodes the distance from a pixel belonging to k^{th} category to the nearest pixel belonging to other categories. Such distance can be treated as the distance to the object

[7] We use `scipy.ndimage.morphology.distance_transform_edt` in implementation.

boundary. We compute a fused distance map through aggregating all the K distance maps.

Note that the values in our distance map are (always positive) different from the conventional signed distances that represent the interior/exterior pixels with positive/negative distances separately.

Boundary map. As the fused distance map represents the distances to the object boundary, we can construct the ground-truth boundary map through setting all the pixels with distance value smaller than a threshold γ as boundary[8]. We empirically choose small γ value, e.g., $\gamma = 5$, as we are mainly focused on the thin boundary refinement.

Direction map. We perform the Sobel filter (with kernel size 9×9) on the K distance maps independently to compute the corresponding K direction maps respectively. The Sobel filter based direction is in the range $[0^\circ, 360^\circ)$, and each direction points to the interior pixel (within the neighborhood) that is furthest away from the object boundary. We divide the entire direction range to m categories (or partitions) and then assign the direction of each pixel to the corresponding category. We illustrate two kinds of partitions in Figure 5 (a) and we choose the $m = 8$ partition by default. We apply the evenly divided direction map as our ground-truth for training. Besides, we also visualize some examples of direction map in Figure 5 (b).

Empirical Analysis. We apply the generated ground-truth on the segmentation results of three state-of-the-art methods including DeepLabv3 [12], HRNet [51] and Gated-SCNN [52] to investigate the potential of our approach. Specifically, we first project the ground-truth direction map to offset map and then refine the segmentation results on Cityscapes `val` based on our generated ground-truth offset map. Table 1 summarizes the related results. We can see that our approach significantly improves both the overall mIoU and the boundary F-score. For example, our approach ($m = 8$) improves the mIoU of Gated-SCNN by 3.1%. We may achieve higher performance through re-scaling the offsets for different pixels adaptively, which is not the focus of this work.

Discussion. The key condition for ensuring the effectiveness of our approach is that *segmentation predictions of the interior pixels are more reliable empirically.* Given accurate boundary maps and direction maps, we could always improve the segmentation performance in expectation. In other words, the segmentation performance ceiling of our approach is also determined by the interior pixels' prediction accuracy.

4 Experiments: Semantic Segmentation

4.1 Datasets & Implementation Details

Cityscapes [15] is a real-world dataset that consists of $2,975/500/1,525$ images with resolution 2048×1024 for training/validation/testing respectively. The dataset contains 19/8 semantic categories for semantic/instance segmentation task.

[8] We define the boundary pixels and interior pixels based on their distance values.

metric	method	w/o SegFix	w/ SegFix	
			$m = 4$	$m = 8$
mIoU	DeepLabv3 (Our impl.)	79.5	82.6 (+3.1)	82.4 (+2.9)
	HRNet-W48 (Our impl.)	81.1	84.1 (+3.0)	84.1 (+3.0)
	Gated-SCNN (Our impl.)	81.0	84.2 (+3.2)	84.1 (+3.1)
F-score	DeepLabv3 (Our impl.)	56.6	68.6 (+12.0)	68.4 (+11.8)
	HRNet-W48 (Our impl.)	62.4	73.8 (+11.4)	73.8 (+11.4)
	Gated-SCNN (Our impl.)	61.4	72.3 (+10.9)	72.3 (+10.9)

Table 1: **Improvements with ground-truth boundary offset on Cityscapes** `val`. We report both the segmentation performance mIoU and the boundary performance F-score (1px width).

ADE20K [64] is a very challenging benchmark consisting of around $20,000/2,000$ images for training/validation respectively. The dataset contains 150 fine-grained semantic categories.

GTA5 [49] is a synthetic dataset that consists of $12,402/6,347/6,155$ images with resolution 1914×1052 for training/validation/testing respectively. The dataset contains 19 semantic categories which are compatible with Cityscapes.

Implementation details. We perform the same training and testing settings on Cityscapes and GTA5 benchmarks as follow. We set the initial learning rate as 0.04, weight decay as 0.0005, crop size as 512×512 and batch size as 16, and train for 80K iterations. For the ADE20K benchmark, we set the initial learning as 0.02 and all the other settings are kept the same as on Cityscapes. We use "poly" learning rate policy with power $= 0.9$. For data augmentation, we all apply random flipping horizontally, random cropping and random brightness jittering within the range of $[-10, 10]$. Besides, we all apply syncBN [50] across multiple GPUs to stabilize the training. We simply set the loss weight as 1.0 for both the boundary loss and direction loss without tuning.

Notably, our approach does not require extra training or fine-tuning any semantic segmentation models. We only need to predict the boundary mask and the direction map for all the test images *in advance* and refine the segmentation results of any existing approaches accordingly.

Evaluation metrics. We use two different metrics including: *mask F-score* and top-1 *direction accuracy* to evaluate the performance of our approach during the training stage. Mask F-score is performed on the predicted binary boundary map and direction accuracy is performed on the predicted direction map. Especially, we only measure the direction accuracy within the regions identified as boundary by the boundary branch.

To verify the effectiveness of our approach for semantic segmentation, we follow the recent Gated-SCNN [52] and perform two quantitative measures including: *class-wise mIoU* to measure the overall segmentation performance on regions; *boundary F-score* to measure the boundary quality of predicted mask with a small slack in distance. In our experiments, we measure the boundary F-score using thresholds 0.0003, 0.0006 and 0.0009 corresponding to 1, 2 and 3 pixels respectively. We mainly report the performance with threshold as 0.0003 for most of our ablation experiments.

backbone	#param (M)	runtime (ms)	mask F-score	direction accuracy	mIoU△	F-score△
HRNet-W18	9.6	16	71.44	64.44	+0.8	+3.7
HRNet-W32	29.4	20	72.24	65.10	+0.9	+3.9
HRNet-2×	47.3	69	73.67	66.87	+1.0	+4.4

Table 2: **Influence of backbones.** The runtime is tested with an input image of resolution 2048×1024 on a single V100 GPU (PyTorch1.4 + TensorRT). SegFix reaches real-time speed with light-weight backbone, e.g., HRNet-W18 or HRNet-W32.

	boundary width				# directions		
	$\gamma = 3$	$\gamma = 5$	$\gamma = 10$	$\gamma = \infty$	$m = 4$	$m = 8$	$m = 16$
mIoU△	+0.94	+0.96	+0.95	+0.84	+0.97	+0.96	+0.96
F-score△	+4.1	+4.2	+4.1	+3.6	+4.1	+4.2	+4.2

Table 3: **Influence of the boundary width and direction number.** SegFix is robust to boundary width and direction number. We choose $\gamma = 5$ and $m = 8$ according to their F-scores.

4.2 Ablation Experiments

We conduct a group of ablations to analyze the influence of various factors within SegFix. We report the improvements over the segmentation baseline DeepLabv3 (mIoU/F-score is 79.5%/56.6%) if not specified.

Backbone. We study the performance of our SegFix based on three different backbones with increasing complexities, i.e., HRNet-W18, HRNet-W32 and HRNet-2×. We apply the same training/testing settings for all three backbones. According to the comparisons in Table 2, our SegFix consistently improves both the segmentation performance and the boundary quality with different backbone choices. We choose HRNet-2× in the following experiments if not specified as it performs best. Besides, we also report their running time in Table 2.

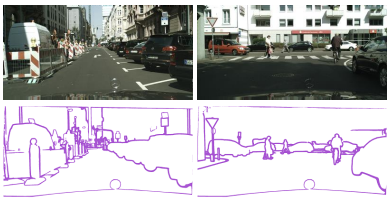


Fig. 6: **Qualitative results of our boundary branch prediction.** The 2 example images are selected from Cityscapes `val`. We can see that their predicted boundaries are of high quality.

Boundary branch. We verify that SegFix is robust to the choice of hyper-parameter γ within the boundary branch and illustrate some qualitative results.
□ **boundary width**: Table 3 shows the performance improvements based on boundary with different widths. We choose different γ values to control the boundary width, where smaller γ leads to thinner boundaries. We also report the performance with $\gamma = \infty$, which means all pixels is identified as boundary. We find their improvements are close and we choose $\gamma = 5$ by default.
□ **qualitative results**: Figure 6 shows the qualitative results with our boundary branch. We find that the predicted boundaries are of high quality. Besides, we also compute the F-scores between the boundary computed from the segmentation map of the existing approaches, e.g., Gated-SCNN and HRNet, and the

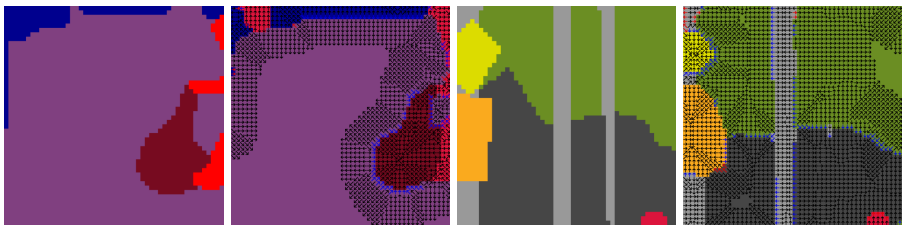


Fig. 7: **Qualitative results of our direction branch predictions.** The 1st and 3rd columns represent the ground-truth segmentation map. The 2nd and 4th columns illustrate the predicted directions with the segmentation map of HRNet as the background. We mark the directions that fix errors with blue arrow and directions that lead to extra errors with red arrow. Our predicted directions addresses boundary errors for various object categories such as bicycle, traffic light and traffic sign. (Better viewed zoom in)

predicted boundary from our boundary branch. The F-scores are around 70%, which (in some degree) means that their boundary maps are well aligned and ensures that more accurate direction predictions bring larger performance gains.

Direction branch. We analyze the influence of the direction number m and then present some qualitative results of our predicted directions.

□ **direction number**: We choose different direction numbers to perform different direction partitions and control the generated offset maps that are used to refine the coarse label map. We conduct the experiments with $m = 4$, $m = 8$ and $m = 16$. According to the reported results on the right 3 columns in Table 3, we find different direction numbers all lead to significant improvements and we choose $m = 8$ if not specified as our SegFix is less sensitive to the choice of m.

□ **qualitative results**: In Figure 7, we show some examples to illustrate that our predicted boundary directions improve the errors. Overall, the improved pixels (marked with blue arrow) are mainly distributed along the very thin boundary.

Comparison with GUM. We compare SegFix with the previous model-dependent guided up-sampling mechanism [44,45] based on DeepLabv3 as the baseline. We report the related results in Table 4. It can be seen that our approach significantly outperforms GUM measured by both mIoU and F-score. We achieve higher performance through combining GUM with our approach, which achieves 5.0% improvements on F-score compared to the baseline.

Comparison with DenseCRF. We compare our approach with the conventional well-verified DenseCRF [34] based on the DeepLabv3 as our baseline. We fine-tune the hyper-parameters of DenseCRF and set them empirically following [11]. According to Table 5, our approach not only outperforms DenseCRF but also is complementary with DenseCRF. The possible reasons for the limited mIoU improvements of DenseCRF might be that it brings more extra errors on the interior pixels.

	baseline	GUM (Our impl.)	SegFix	GUM+SegFix
mIoU	79.5	79.8 (+0.3)	80.5 (+1.0)	80.6 (+1.1)
F-score	56.6	57.7 (+1.1)	60.9 (+4.3)	61.6 (+5.0)

Table 4: **Comparison with GUM** [44]. SegFix not only outperforms GUM but also is complementary with GUM.

	baseline	DenseCRF	SegFix	DenseCRF+SegFix
mIoU	79.5	79.7 (+0.2)	80.5 (+1.0)	80.5 (+1.0)
F-score	56.6	60.9 (+4.3)	61.0 (+4.4)	64.1 (+7.5)

Table 5: **Comparison with DenseCRF** [34]. SegFix achieves comparable F-score improvements and much larger mIoU gains.

	Gated-SCNN	Gated-SCNN+SegFix
mIoU	81.0	81.5 (+0.5)
F-score	61.4	63.1 (+1.7)

Table 6: **Comparison with Gated-SCNN** [52] The result of Gated-SCNN is based on multi-scale testing.

	ADE20K		GTA5	
	baseline	+SegFix	baseline	+SegFix
mIoU	44.8	45.4(+0.6)	77.8	80.6(+2.8)
F-score	16.4	19.3(+2.9)	50.2	61.7(+11.5)

Table 7: **DeepLabv3 with SegFix on ADE20K and GTA5.** We all choose DeepLabv3 as the baseline.

Application to Gated-SCNN. Considering that Gated-SCNN [52] introduced multiple components to improve the performance, it is hard to compare our approach with Gated-SCNN fairly to a large extent. To verify the effectiveness of our approach to some extent, we first take the open-sourced Gated-SCNN (multi-scale testing) segmentation results on Cityscapes validation set as the coarse segmentation maps, then we apply the SegFix offset maps to refine the results. We report the results in Table 6 and SegFix improves the boundary F-score by 1.7%, suggesting that SegFix is complementary with the strong baseline that also focuses on improving the segmentation boundary quality. Besides, we also report the detailed category-wise improvements measured by both mIoU and boundary F-score in Table 8.

Comparison with STEAL. Due to the training code of STEAL [1] is not open-sourced, we simply apply the released checkpoints[9] to predict K semantic boundary maps and convert them to binary boundary map. We empirically find that the boundary quality of our SegFix (35.54%) is comparable with the carefully designed STEAL (35.86%) measured by F-score along the ground-truth boundary with 1-px width, suggesting that our method achieves nearly the state-of-the-art boundary detection performance. To verify whether SegFix can benefit from the more accurate boundary maps predicted by STEAL, we also train a SegFix model to only predict the direction map while using the (fixed) pre-computed boundary maps with STEAL. We find the result becomes slightly worse (80.5% $\rightarrow$ 80.32%) based on the coarse results with DeepLabv3.

[9] STEAL: https://github.com/nv-tlabs/STEAL

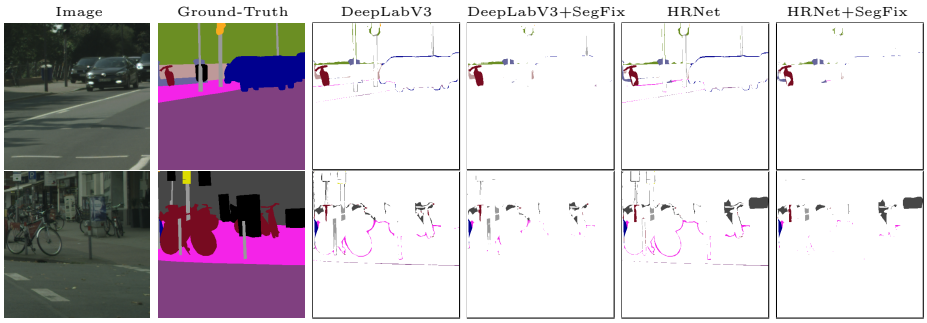


Fig. 8: **Qualitative comparison in terms of errors on Cityscapes** `val`. Our approach well addresses the existing boundary errors of various categories, e.g., car, bicycle and traffic sign, for both DeepLabv3 and HRNet.

4.3 Application to State-of-the-art

We generate the boundary maps and the direction maps in advance and apply them to the segmentation results of various state-of-the-art approaches without extra training or fine-tuning.

Cityscapes val: We first apply our approach on various state-of-the-art approaches (on Cityscapes `val`) including DeepLabv3, Gated-SCNN and HRNet. We report the category-wise performance improvements in Table 8. It can be seen that our approach significantly improves the segmentation quality along the boundaries of all the evaluated approaches. Figure 8 provides some qualitative examples of the improvements with our approach along the thin boundaries based on both DeepLabv3 and HRNet.

Cityscapes test: We further apply our approach on several recent state-of-the-art methods on Cityscapes `test` including PSANet [62], DANet [22], BFP [17], HRNet [51], Gated-SCNN [52], VPLR [65] and HRNet + OCR [58]. We directly apply the same model that are trained with only the $2,975$ training images without any other tricks, e.g., training with validation set or Mapillary Vistas [46], online hard example mining.

Notably, the state-of-the-art methods have applied various advanced techniques, e.g., multi-scale testing, multi-grid, performing boundary supervision or utilizing extra training data such as Mapillary Vistas or Cityscapes video, to improve their results. In Table 9, our model-agnostic boundary refinement scheme consistently improves all the evaluated approaches. For example, with our SegFix, "HRNet + OCR" achieves 84.5% on Cityscapes `test`. The improvements of our SegFix is in fact already significant considering the baseline is already very strong and the performance gap between top ranking methods is just around $0.1\% \sim 0.3\%$. We believe that lots of other advanced approaches might also benefit from our approach.

4.4 Experiments on ADE20K & GTA5

We evaluate our SegFix scheme on two other challenging semantic segmentation benchmarks including ADE20K and GTA5. We choose DeepLabv3 as our base-

width	method	road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrian	sky	person	rider	car	truck	bus	train	motorcycle	bicycle	mean
1px	DeepLabV3	70.7	44.4	50.0	45.9	42.3	48.3	45.8	46.5	49.5	45.4	60.5	43.0	55.9	56.9	76.6	84.5	92.3	70.5	45.9	56.6
	+ SegFix	**73.9**	**49.1**	**55.5**	**47.8**	**43.7**	**57.6**	**52.7**	**58.3**	**54.7**	**47.4**	**64.7**	**50.2**	**59.7**	**64.6**	**77.4**	**86.0**	**92.6**	**72.0**	**51.5**	**61.0**
	HRNet-W48	73.1	48.9	55.4	49.2	49.0	58.9	59.0	55.5	54.0	51.0	65.1	52.0	62.0	63.4	79.0	87.5	95.0	77.4	51.0	62.4
	+ SegFix	**74.8**	**51.9**	**58.2**	**50.9**	**49.7**	**63.6**	**64.0**	**61.6**	**57.1**	**52.5**	**66.8**	**56.8**	**64.4**	**67.5**	**79.7**	**88.7**	**95.2**	**77.7**	**55.0**	**65.1**
	Gated-SCNN	73.5	49.8	55.5	46.7	43.0	59.9	61.8	57.4	54.4	45.7	65.9	51.4	61.9	64.0	72.5	84.8	92.4	71.9	53.6	61.4
	+ SegFix	**74.2**	**51.3**	**57.7**	**47.2**	**45.3**	**64.0**	**63.8**	**61.2**	**56.7**	**46.9**	**66.6**	**55.6**	**64.0**	**66.9**	72.0	**85.0**	**92.6**	71.8	**55.9**	**63.1**
2px	DeepLabV3	79.1	57.5	62.2	49.3	45.5	64.1	54.5	61.3	62.6	49.8	72.2	54.8	62.4	71.6	78.0	86.5	92.7	72.3	54.7	64.8
	+ SegFix	**81.2**	**60.9**	**66.3**	**51.1**	**46.6**	**69.6**	**59.7**	**69.3**	**66.6**	**51.6**	**75.0**	**60.4**	**65.6**	**76.6**	**78.8**	**87.7**	**93.0**	**73.5**	**59.5**	**68.1**
	HRNet-W48	81.1	61.7	67.4	52.5	52.5	73.2	67.7	69.4	66.9	55.4	76.3	63.7	68.2	77.3	80.4	89.6	95.5	79.1	60.3	70.4
	+ SegFix	**82.1**	**63.7**	**69.1**	**54.0**	**52.8**	**75.2**	**71.1**	**72.5**	**69.1**	**56.7**	**77.2**	**66.9**	**70.4**	**79.5**	**80.9**	**90.3**	**95.6**	79.1	**63.3**	**72.1**
	Gated-SCNN	80.9	61.9	67.1	50.0	46.4	73.9	70.3	70.1	67.1	50.0	76.7	62.8	68.5	77.3	74.0	86.8	92.9	73.8	62.5	69.1
	+ SegFix	**81.5**	**63.0**	**68.6**	**50.5**	**48.5**	**75.9**	**71.1**	**72.1**	**68.6**	**51.2**	**77.1**	**65.9**	**70.3**	**78.9**	73.4	86.7	**93.0**	73.6	**64.6**	**70.2**
3px	DeepLabV3	84.1	65.8	70.7	52.0	47.9	72.5	60.8	70.2	72.2	53.2	79.9	62.9	67.3	79.8	79.0	87.8	93.0	73.7	61.6	70.2
	+ SegFix	**85.2**	**67.8**	**73.0**	**53.3**	**48.6**	**74.8**	**64.0**	**74.5**	**74.6**	**54.5**	**81.4**	**66.1**	**69.5**	**82.2**	**79.5**	**88.6**	**93.3**	**74.6**	**65.0**	**72.1**
	HRNet-W48	85.5	69.1	74.7	54.9	54.9	79.0	72.9	75.6	75.5	58.6	83.0	70.4	72.6	84.3	81.3	90.8	95.7	80.3	66.9	75.1
	+ SegFix	**86.0**	**70.3**	**75.4**	**55.8**	54.8	**79.5**	**74.9**	**77.0**	**76.8**	**59.5**	**83.3**	**72.0**	**74.0**	**84.9**	**81.6**	**91.2**	**95.8**	80.1	**68.6**	**75.9**
	Gated-SCNN	85.0	68.8	74.2	52.2	48.7	79.7	75.0	75.9	75.4	53.0	83.1	69.3	73.1	83.6	74.9	87.8	93.2	75.2	68.8	73.5
	+ SegFix	**85.3**	**69.6**	**74.9**	**52.5**	**50.6**	**80.3**	75.0	**76.7**	**76.3**	**54.0**	**83.3**	**71.1**	**74.2**	**84.2**	74.1	87.6	93.2	74.9	**70.0**	**74.1**

Table 8: **Boundary F-score with SegFix.** We illustrate the category-wise comparison with various baselines in terms of boundary F-score on Cityscapes `val`.

method	road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrian	sky	person	rider	car	truck	bus	train	motorcycle	bicycle	mean
PSANet	98.7	87.0	93.5	58.9	62.5	67.8	76.0	80.0	93.7	72.6	95.4	86.9	73.0	96.2	79.3	91.2	84.9	71.1	77.9	81.4
+ SegFix	98.7	**87.4**	**93.7**	**59.3**	**62.8**	**69.5**	**77.6**	**81.4**	**93.9**	**73.0**	**95.6**	**88.0**	**73.9**	**96.5**	**79.6**	**91.5**	**85.1**	**71.8**	**78.6**	**82.0**
DANet	98.6	86.1	93.5	56.1	63.3	69.7	77.3	81.3	93.9	72.9	95.7	87.3	72.9	96.2	76.8	89.4	86.5	72.2	78.2	81.5
+ SegFix	**98.7**	**86.6**	**93.7**	**56.5**	**63.5**	**71.4**	**78.7**	**82.4**	**94.1**	**73.2**	**95.9**	**88.2**	**73.7**	**96.5**	**77.0**	**89.7**	**86.8**	**72.8**	**78.8**	**82.0**
BFP	98.7	87.0	93.5	59.8	63.4	68.9	76.8	80.9	93.7	72.8	95.5	87.0	72.1	96.0	77.6	89.0	86.9	69.2	77.6	81.4
+ SegFix	98.7	**87.5**	**93.7**	**60.2**	**63.7**	**71.1**	**78.4**	**82.4**	**94.0**	**73.2**	**95.7**	**88.1**	**72.9**	**96.3**	**77.8**	**89.3**	**87.2**	**69.9**	**78.4**	**82.0**
HRNet	98.8	87.5	93.7	55.6	62.3	71.8	79.3	81.8	94.0	73.1	95.8	88.5	76.1	96.5	72.2	86.5	84.7	73.8	79.4	81.8
+ SegFix	98.8	**87.9**	**93.9**	**56.0**	**62.5**	**73.6**	**80.7**	**83.2**	**94.1**	**73.4**	**95.9**	**89.3**	**76.7**	**96.6**	**72.4**	**86.7**	**85.0**	**74.3**	**80.2**	**82.2**
VPLR	98.8	87.8	94.2	64.1	65.0	72.4	79.0	82.8	94.2	74.0	96.1	88.2	75.4	96.5	78.8	94.0	91.6	73.8	79.0	83.5
+ SegFix	98.8	**88.0**	**94.3**	**64.4**	**65.3**	**73.3**	**80.0**	**83.5**	**94.3**	**74.3**	**96.2**	**89.0**	**76.2**	**96.7**	**79.0**	**94.2**	**92.0**	**74.4**	**79.7**	**83.9**
HRNet + OCR	98.9	88.3	94.3	66.8	66.6	73.6	80.3	83.7	94.3	74.4	96.0	88.7	75.4	96.6	82.5	94.0	90.8	73.8	79.7	84.2
+ SegFix	98.9	88.3	**94.4**	**68.0**	**67.8**	**73.6**	**80.6**	**83.9**	**94.4**	**74.5**	**96.1**	**89.2**	**75.9**	**96.8**	**83.6**	**94.2**	**91.3**	**74.0**	**80.1**	**84.5**

Table 9: **Segmentation mIoU with SegFix:** Category-wise improvements of SegFix based on various state-of-the-art methods on Cityscapes `test`. Notably, "HRNet + OCR + SegFix" ranks the first place on the Cityscapes semantic segmentation leaderboard by the ECCV 2020 submission deadline.

method	person	rider	car	truck	bus	train	motorcycle	bicycle	mean (%)
Mask-RCNN	36.0	28.8	51.6	30.0	38.7	27.3	23.9	19.4	32.0
+ SegFix	**37.9**	**30.3**	**54.1**	**31.0**	**40.0**	**27.9**	**25.1**	**20.5**	**33.3 (+1.3)**
PointRend	36.6	29.7	53.7	29.9	40.4	33.3	23.6	19.6	33.3
+ SegFix	**38.7**	**31.1**	**56.2**	**31.1**	**41.6**	**34.1**	**24.6**	**20.7**	**34.8 (+1.5)**
PANet	41.5	33.6	58.2	31.8	45.3	28.7	28.2	24.1	36.4
+ SegFix	**43.3**	**34.9**	**60.4**	**32.9**	**47.0**	**30.1**	**29.1**	**24.7**	**37.8 (+1.4)**
PolyTransform	42.4	34.8	58.5	39.8	50.0	41.3	30.9	23.4	40.1
+ SegFix	**44.3**	**35.9**	**60.5**	**40.5**	**51.2**	**41.6**	**31.7**	**24.1**	**41.2 (+1.1)**

Table 10: **Results on Cityscapes Instance Segmentation task.** Our SegFix significantly improves the mask AP of Mask-RCNN [25], PointRend [33], PANet [39] and PolyTransform [37] on Cityscapes `test` (w/ COCO pre-training). Notably, "PolyTransform + SegFix" ranks the second place on the Cityscapes instance segmentation leaderboard by the ECCV 2020 submission deadline..

	DeepLabv3	HRNet-W18	DeepLabv3+SegFix	DeepLabv3+HRNet-W18
mIoU	79.5	79.4	80.3 (+0.8)	79.9 (+0.5)
F-score	56.6	57.0	60.3 (+3.7)	58.2 (+1.6)

Table 11: **Comparison with model ensemble.** "DeepLabv3+HRNet-W18" reports the results based on model ensemble and "DeepLabv3+SegFix" reports the results based on our SegFix. Our SegFix outperforms the model ensemble on both mIoU and F-score metrics. We report the improvements compared to the performance with DeepLabv3.

line on both datasets. As illustrated in Table 7, our approach also achieves significant performance improvements along the boundary on both benchmarks, e.g., the boundary F-score of DeepLabv3 gains 2.9%/11.5% on ADE20K `val`/GTA5 `test` separately.

4.5 Unified SegFix Model

We propose to train a single unified SegFix model on Cityscapes and ADE20K, and we report the improvements over DeepLabv3 as below: with a single unified SegFix model, the performance gains are 0.9%/3.8% on Cityscapes and 0.5%/2.7% on ADE20K measured by mIoU/F-score. We can see these improvements are comparable with the SegFix trained on each dataset independently. More experimental details are illustrated in the Appendix.

In general, we only need to train a single unified SegFix model to improve the boundary quality of various segmentation models across different datasets, thus SegFix is much more training friendly (and saves a lot of energy consumption) compared to the previous methods [5,52,17,41,40,30] that require re-training the existing segmentation models on each dataset independently.

4.6 Comparison with Model Ensemble

To investigate whether our SegFix mainly benefits from model ensemble, we conduct a group of experiments to compare our method with the standard model ensemble (that ensembles two segmentation models with the same compacity) under fair settings and report the results in Table 11. Specifically speaking, when processing a single image with resolution 1024×2048, the overall computation cost of DeepLabv3+SegFix/DeepLabv3+HRNet-W18 is 2054/2060 GFLOPs separately. We can see that SegFix outperforms the model ensemble, e.g., DeepLabv3+SegFix gains 1.9% (on F-score) over model ensemble method DeepLabv3+HRNet-W18, suggesting that our SegFix is capable to fix that boundary errors that the model ensemble fails to address. Besides, another advantage of our method lies at that we can use a single unified SegFix model across multiple datasets while the model ensemble requires training multiple different segmentation models on different datasets independently.

5 Experiments: Instance Segmentation

In Table 10, we illustrate the results of SegFix on Cityscapes instance segmentation task. We can find that the SegFix consistently improves the mean AP scores

over Mask-RCNN [25], PANet [39], PointRend [33] and PolyTransform [37]. For example, with SegFix scheme, PANet gains 1.4% points on the Cityscapes `test` set. We also apply our SegFix on the very recent PointRend and PolyTransform. Our SegFix consistently improves the performance of PointRend and PolyTransform by 1.5% and 1.1% separately, which further verifies the effectiveness of our method.

We use the public available checkpoints from Dectectron2[10] and PANet[11] to generate the predictions of Mask-RCNN, PointRend and PANet. Besides, we use the segmentation results of PolyTransform directly. More training/testing details of SegFix on Cityscapes instance segmentation task are illustrated in the Appendix. We believe that SegFix can be used to improve various other state-of-the-art instance segmentation methods directly w/o any prior requirements.

Notably, the improvements on the instance segmentation tasks ($+1.1\% \sim 1.5\%$) are more significant than the improvements on semantic segmentation task ($+0.3\% \sim 0.5\%$). We guess the main reason is that the instance segmentation evaluation (on Cityscapes) only considers 8 object categories without including the stuff categories. The performance of stuff categories is less sensitive to the boundary errors due to that their area is (typically) larger than the area of object categories. According to the category-wise results in Table 9, we can also find that the improvements on several object categories, e.g., person, rider, and truck, is more significant than the stuff categories, e.g., road, building.

6 Conclusion

In this paper, we have proposed a novel model-agnostic approach to refine the segmentation maps predicted by an unknown segmentation model. The insight is that the predictions of the interior pixels are more reliable. We propose to replace the predictions of the boundary pixels using the predictions of the corresponding interior pixels. The correspondence is learnt only from the input image. The main advantage of our method is that SegFix generalizes well on various strong segmentation models. Empirical results show that the effectiveness of our approach for both semantic segmentation and instance segmentation tasks. We hope our SegFix scheme can become a strong baseline for more accurate segmentation results along the boundary.

Acknowledgement: This work is partially supported by Natural Science Foundation of China under contract No. 61390511, and Frontier Science Key Research Project CAS No. QYZDJ-SSW-JSC009.

7 Appendix

First of all, we need to clarify that all of our semantic segmentation ablation experiments choose the DeepLabv3 as baseline if not specified. In Section 7.1, we

[10] Detectron2: https://github.com/facebookresearch/detectron2
[11] PANet: https://github.com/ShuLiu1993/PANet

illustrate the statistics of the proportions of boundary pixels over different categories as their scales vary so much. In Section 7.2, we report the category-wise mIoU improvements of our approach on Cityscapes `val`. In Section 7.3, we provide more details of our experiments with unified SegFix scheme. In Section 7.4, we present more details of our experiments on Cityscapes instance segmentation task. Last, in Section 7.5, we illustrate more qualitative results of our approach.

boundary width	road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle	mean
1	1.0	4.2	2.7	4.4	4.3	20.1	12.2	9.0	2.8	5.2	3.5	8.1	10.9	2.5	2.2	2.4	3.2	8.4	8.3	2.6
2	1.9	8.2	5.2	8.5	8.3	38.1	23.4	17.5	5.6	10.1	6.8	15.8	21.0	4.9	4.4	4.7	6.2	16.3	16.0	5.2
3	2.7	11.4	7.2	12.0	11.7	51.6	32.3	24.2	7.8	14.0	9.5	21.6	28.2	6.8	6.2	6.7	8.6	21.9	21.4	7.2
4	3.5	14.5	9.2	15.3	14.9	61.8	40.3	30.5	9.9	17.7	12.0	26.9	34.8	8.7	7.9	8.5	10.9	27.1	26.5	9.1
5	4.4	18.0	11.4	18.8	18.2	69.8	48.5	37.1	12.4	21.7	15.0	33.4	42.6	11.0	9.8	10.7	13.6	33.2	32.4	11.2

Table 12: The proportion of boundary pixels (with different widths) over different categories on Cityscapes `val` (%).

method	road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrian	sky	person	rider	car	truck	bus	train	motorcycle	bicycle	mean
DeepLabV3	98.4	86.5	93.1	63.9	62.6	66.1	72.2	80.0	92.8	66.3	95.0	83.3	65.5	95.3	74.5	89.0	80.0	67.4	78.4	79.5
+ SegFix	**98.5**	**87.1**	**93.5**	**64.6**	**63.1**	**69.0**	**74.9**	**82.4**	**93.2**	**66.7**	**95.3**	**84.9**	**66.9**	**95.8**	**75.0**	**89.6**	**80.7**	**68.4**	**79.7**	**80.5**
HRNet-W48	98.5	87.0	93.5	58.5	64.7	71.4	75.6	82.8	93.2	64.8	95.3	84.7	66.9	95.8	82.9	91.5	82.9	69.8	80.1	81.1
+ SegFix	98.5	**87.4**	**93.7**	**59.0**	**65.1**	**72.5**	**77.0**	**84.0**	**93.4**	**65.1**	**95.4**	**85.7**	**67.7**	**96.1**	**83.1**	**91.9**	**83.4**	**70.8**	**81.0**	**81.6**
Gated-SCNN	98.3	86.4	93.3	56.5	64.2	70.8	75.8	83.1	93.0	65.4	95.3	85.3	67.8	96.0	81.3	91.4	84.6	69.9	80.5	81.0
+ SegFix	**98.4**	**86.7**	**93.4**	**56.8**	**64.4**	**72.0**	**77.0**	**84.1**	**93.2**	**65.7**	**95.4**	**86.0**	**68.8**	**96.2**	**81.5**	**91.5**	**84.8**	**70.6**	**81.1**	**81.5**

Table 13: Category-wise mIoU improvements of SegFix based on various methods on Cityscapes `val`.

7.1 Statistics of Boundary Pixels

We collect some statistics of the proportion of the boundary pixels over different categories in Table 12. We can find that the boundary pixels occupy large proportions for three (small-scale) categories including *pole*, *traffic light* and *traffic sign*. In fact, the performance improvements (measured by mIoU) also mainly come from these three categories. For example, in Table 13, our SegFix improves the DeepLabv3's mIoUs of these three categories by 3.1%, 2.7% and 2.4% separately.

7.2 Category-wise mIoU Improvements

We perform the SegFix on the Cityscapes `val` segmentation results based on DeepLabv3 [11], Gated-SCNN [52] and HRNet [51]. We report the category-wise mIoU improvements in Table 13 and we can see that our approach significantly improves the performance on object categories including *pole*, *traffic light* and *traffic sign*. The key reason might be that the objects belonging to these categories tend to be of small scale, which benefit more from the accurate boundary.

7.3 Details of Unified SegFix Experiments

In our implementation, we use the same backbone HRNet-2× for SegFix and we illustrate the training policy as below: we set the batch size as 16 and construct each mini-batch by sampling 8 images from Cityscapes and 8 images from ADE20K. We choose the initial learning rate as 0.02 and all the other training settings are kept the same. the same learning rate policy, the crop size as 512 (for images from both datasets) and the same augmentation policy. As illustrated in the paper, the performance of unified SegFix is comparable with the performance of SegFix trained on each dataset separately. In general, the proposed unified SegFix is a general scheme that well addresses the boundary errors across multiple benchmarks.

7.4 Details of Experiments on Instance Segmentation

We generate the instance segmentation results of Mask-RCNN/PointRend based on the open-sourced Detectron2 [54], and we get the results of PANet [39] and PolyTransform [37] from the authors directly as our approach does not require training any segmentation models.

To predict suitable offset maps for instance segmentation, we start from the instance masks and re-compute the ground-truth distance maps, boundary maps and direction maps. Specifically, for the instance pixels, we first estimate a distance map based on each instance map and then merge all the instance based distance maps as the final distance map. We generate their direction maps and boundary maps following the same manner as the manner for semantic segmentation. We apply the predicted offset map on each predicted instance map separately during the testing stage. According to the experimental results on Cityscapes instance segmentation task, we can see that SegFix consistently improves the performance of various methods on Cityscapes `test`. We also believe the recent state-of-the-art methods might benefit from our SegFix.

7.5 More Qualitative Results

We illustrate more qualitative examples of the improvements (on semantic segmentation task) with our approach in Figure 9. We can see that our approach well addresses the errors along thin boundary. There still exist some errors located in the interior regions that our approach fail to address as we are mainly focused on the thin boundary refinement.

References

1. Acuna, D., Kar, A., Fidler, S.: Devil is in the edges: Learning semantic boundaries from noisy annotations. In: CVPR (2019)
2. Arbelaez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. PAMI (2010)

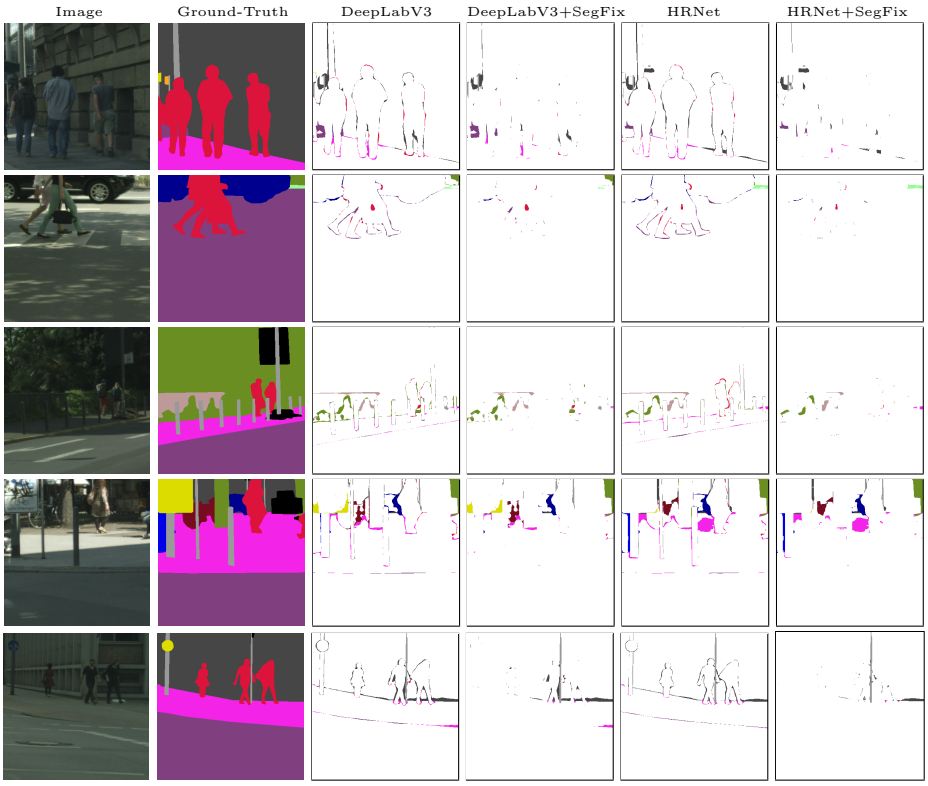


Fig. 9: Qualitative comparison in terms of errors on Cityscapes `val`. Our approach well addresses the existing boundary errors of various categories, e.g., person, pole and traffic sign, for both DeepLabv3 and HRNet.

3. Bai, M., Urtasun, R.: Deep watershed transform for instance segmentation. In: CVPR (2017)
4. Bertasius, G., Shi, J., Torresani, L.: High-for-low and low-for-high: Efficient boundary detection from deep object features and its applications to high-level vision. In: ICCV (2015)
5. Bertasius, G., Shi, J., Torresani, L.: Semantic segmentation with boundary neural fields. In: CVPR (2016)
6. Bischke, B., Helber, P., Folz, J., Borth, D., Dengel, A.: Multi-task learning for segmentation of building footprints with deep neural networks. In: ICIP (2019)
7. Caesar, H., Uijlings, J., Ferrari, V.: Coco-stuff: Thing and stuff classes in context. In: CVPR (2018)
8. Caselles, V., Kimmel, R., Sapiro, G.: Geodesic active contours. IJCV (1997)
9. Chen, L.C., Barron, J.T., Papandreou, G., Murphy, K., Yuille, A.L.: Semantic image segmentation with task-specific edge detection using cnns and a discriminatively trained domain transform. In: CVPR (2016)
10. Chen, L.C., Hermans, A., Papandreou, G., Schroff, F., Wang, P., Adam, H.: Masklab: Instance segmentation by refining object detection with semantic and direction features. In: CVPR (2018)
11. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. PAMI (2017)
12. Chen, L.C., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. arXiv:1706.05587 (2017)
13. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. In: ECCV (2018)
14. Chen, X., Williams, B.M., Vallabhaneni, S.R., Czanner, G., Williams, R., Zheng, Y.: Learning active contour models for medical image segmentation. In: CVPR (2019)
15. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: CVPR (2016)
16. Dangi, S., Yaniv, Z., Linte, C.: A distance map regularized cnn for cardiac cine mr image segmentation. arXiv:1901.01238 (2019)
17. Ding, H., Jiang, X., Liu, A.Q., Thalmann, N.M., Wang, G.: Boundary-aware feature propagation for scene segmentation. ICCV (2019)
18. Ding, H., Jiang, X., Shuai, B., Liu, A.Q., Wang, G.: Semantic correlation promoted shape-variant context for segmentation. In: CVPR (2019)
19. Ding, H., Jiang, X., Shuai, B., Qun Liu, A., Wang, G.: Context contrasted feature and gated multi-scale aggregation for scene segmentation. In: CVPR (2018)
20. Dollár, P., Zitnick, C.L.: Fast edge detection using structured forests. ArXiv (2014)
21. Fieraru, M., Khoreva, A., Pishchulin, L., Schiele, B.: Learning to refine human pose estimation. In: CVPRW (2018)
22. Fu, J., Liu, J., Tian, H., Li, Y., Bao, Y., Fang, Z., Lu, H.: Dual attention network for scene segmentation. In: CVPR (2019)
23. Gidaris, S., Komodakis, N.: Detect, replace, refine: Deep structured prediction for pixel wise labeling. In: CVPR (2017)
24. Hayder, Z., He, X., Salzmann, M.: Boundary-aware instance segmentation. In: CVPR (2017)
25. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: ICCV (2017)
26. Huang, L., Yuan, Y., Guo, J., Zhang, C., Chen, X., Wang, J.: Interlaced sparse self-attention for semantic segmentation. arXiv preprint arXiv:1907.12273 (2019)

27. Islam, M.A., Naha, S., Rochan, M., Bruce, N., Wang, Y.: Label refinement network for coarse-to-fine semantic segmentation. arXiv:1703.00551 (2017)
28. Jaderberg, M., Simonyan, K., Zisserman, A., et al.: Spatial transformer networks. In: NIPS (2015)
29. Kass, M., Witkin, A., Terzopoulos, D.: Snakes: Active contour models. IJCV (1988)
30. Ke, T.W., Hwang, J.J., Liu, Z., Yu, S.X.: Adaptive affinity fields for semantic segmentation. In: ECCV (2018)
31. Kim, Y., Kim, S., Kim, T., Kim, C.: Cnn-based semantic segmentation using level set loss. In: WACV (2019)
32. Kimmel, R., Kiryati, N., Bruckstein, A.M.: Sub-pixel distance maps and weighted distance transforms. JMIV (1996)
33. Kirillov, A., Wu, Y., He, K., Girshick, R.: Pointrend: Image segmentation as rendering. arXiv:1912.08193 (2019)
34. Krähenbühl, P., Koltun, V.: Efficient inference in fully connected crfs with gaussian edge potentials. In: NIPS (2011)
35. Kuo, W., Angelova, A., Malik, J., Lin, T.Y.: Shapemask: Learning to segment novel objects by refining shape priors. ICCV (2019)
36. Li, K., Hariharan, B., Malik, J.: Iterative instance segmentation. In: CVPR (2016)
37. Liang, J., Homayounfar, N., Ma, W.C., Xiong, Y., Hu, R., Urtasun, R.: Polytransform: Deep polygon transformer for instance segmentation. arXiv:1912.02801 (2019)
38. Lin, G., Milan, A., Shen, C., Reid, I.: Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In: CVPR (2017)
39. Liu, S., Qi, L., Qin, H., Shi, J., Jia, J.: Path aggregation network for instance segmentation. In: CVPR (2018)
40. Liu, S., De Mello, S., Gu, J., Zhong, G., Yang, M.H., Kautz, J.: Learning affinity via spatial propagation networks. In: NIPS (2017)
41. Liu, T., Ruan, T., Huang, Z., Wei, Y., Wei, S., Zhao, Y., Huang, T.: Devil in the details: Towards accurate single and multiple human parsing. arXiv:1809.05996 (2018)
42. Liu, Y., Cheng, M.M., Hu, X., Wang, K., Bai, X.: Richer convolutional features for edge detection. In: CVPR (2017)
43. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR (2015)
44. Mazzini, D.: Guided upsampling network for real-time semantic segmentation. arXiv preprint arXiv:1807.07466 (2018)
45. Mazzini, D., Schettini, R.: Spatial sampling network for fast scene understanding. In: CVPRW (2019)
46. Neuhold, G., Ollmann, T., Rota Bulo, S., Kontschieder, P.: The mapillary vistas dataset for semantic understanding of street scenes. In: ICCV (2017)
47. Osher, S., Sethian, J.A.: Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. Journal of Computational Physics (1988)
48. Papandreou, G., Zhu, T., Chen, L.C., Gidaris, S., Tompson, J., Murphy, K.: Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model. In: ECCV (2018)
49. Richter, S.R., Vineet, V., Roth, S., Koltun, V.: Playing for data: Ground truth from computer games. In: ECCV (2016)
50. Rota Bul, S., Porzi, L., Kontschieder, P.: In-place activated batchnorm for memory-optimized training of dnns. In: CVPR (2018)

51. Sun, K., Zhao, Y., Jiang, B., Cheng, T., Xiao, B., Liu, D., Mu, Y., Wang, X., Liu, W., Wang, J.: High-resolution representations for labeling pixels and regions. arXiv:1904.04514 (2019)
52. Takikawa, T., Acuna, D., Jampani, V., Fidler, S.: Gated-scnn: Gated shape cnns for semantic segmentation. ICCV (2019)
53. Wang, Z., Acuna, D., Ling, H., Kar, A., Fidler, S.: Object instance annotation with deep extreme level set evolution. In: CVPR (2019)
54. Wu, Y., Kirillov, A., Massa, F., Lo, W.Y., Girshick, R.: Detectron2. https://github.com/facebookresearch/detectron2 (2019)
55. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions. arXiv preprint arXiv:1511.07122 (2015)
56. Yu, Z., Feng, C., Liu, M.Y., Ramalingam, S.: Casenet: Deep category-aware semantic edge detection. In: CVPR (2017)
57. Yu, Z., Liu, W., Zou, Y., Feng, C., Ramalingam, S., Vijaya Kumar, B., Kautz, J.: Simultaneous edge alignment and learning. In: ECCV (2018)
58. Yuan, Y., Chen, X., Wang, J.: Object-contextual representations for semantic segmentation. arXiv preprint arXiv:1909.11065 (2019)
59. Yuan, Y., Wang, J.: Ocnet: Object context network for scene parsing. arXiv:1809.00916 (2018)
60. Zhang, H., Zhang, H., Wang, C., Xie, J.: Co-occurrent features in semantic segmentation. In: CVPR (2019)
61. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network (2017)
62. Zhao, H., Zhang, Y., Liu, S., Shi, J., Loy, C.C., Lin, D., Jia, J.: PSANet: Point-wise spatial attention network for scene parsing. In: ECCV (2018)
63. Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., Torr, P.H.S.: Conditional random fields as recurrent neural networks. In: ICCV (2015)
64. Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., Torralba, A.: Scene parsing through ade20k dataset. In: CVPR (2017)
65. Zhu, Y., Sapra, K., Reda, F.A., Shih, K.J., Newsam, S., Tao, A., Catanzaro, B.: Improving semantic segmentation via video propagation and label relaxation. In: CVPR (2019)