STEdge: Self-Training Edge Detection With Multilayer Teaching and Regularization

Yunfan Ye, Renjiao Yi, Zhiping Cai, *Member, IEEE*, and Kai Xu, *Senior Member, IEEE*

***Abstract*— Learning-based edge detection has hereunto been strongly supervised with pixel-wise annotations which are tedious to obtain manually. We study the problem of *self-training edge detection*, leveraging the untapped wealth of large-scale unlabeled image datasets. We design a self-supervised framework with multilayer regularization and self-teaching. In particular, we impose a consistency regularization which enforces the outputs from each of the multiple layers to be consistent for the input image and its perturbed counterpart. We adopt L0-smoothing as the "perturbation" to encourage edge prediction lying on salient boundaries following the cluster assumption in self-supervised learning. Meanwhile, the network is trained with multilayer supervision by pseudo labels which are initialized with Canny edges and then iteratively refined by the network as the training proceeds. The regularization and self-teaching together attain a good balance of precision and recall, leading to a significant performance boost over supervised methods, with lightweight refinement on the target dataset. Through extensive experiments, our method demonstrates strong cross-dataset generality and can improve the original performance of edge detectors after self-training and fine-tuning.**

***Index Terms*— Consistency regularization, edge detection, pseudo labels, self-training.**

I. Introduction

EDGE detection is a fundamental low-level task in computer vision, which aims to extract object boundaries and visually salient edges from natural images. Various high-level tasks have greatly benefited from edge detection, such as object detection and segmentation [1], [2], [3], [4], [5], [6], video interpolation [7], image inpainting [8], and denoising [9].

Traditional methods accentuate edges based on local features such as gradients [4], [10], [11]. Recently, deep learning approaches have achieved great success due to their ability of capturing more global context and hence producing more meaningful edges. Examples include HED [12], RCF [13], BDCN [14], DexiNed [15], PiDiNet [16], and EDTER [17].

Manuscript received 22 November 2022; revised 9 May 2023; accepted 3 July 2023. Date of publication 19 July 2023; date of current version 30 October 2024. This work was supported in part by the National Key Research and Development Program of China under Grant 2018AAA0102200, in part by NSFC under Grant 62132021 and Grant 62002375, in part by the Natural Science Foundation of Hunan Province of China under Grant 2022RC3061, Grant 2022RC1104, and Grant 2021JJ40696, and in part by NUDT Research under Grant ZK22-52. *(Yunfan Ye and Renjiao Yi are co-first authors.) (Corresponding authors: Zhiping Cai; Kai Xu.)*

The authors are with the Department of Computer Science, National University of Defense Technology, Changsha 410073, China (e-mail: zpcai@nudt.edu.cn; kevin.kai.xu@gmail.com).

Digital Object Identifier 10.1109/TNNLS.2023.3292905

A downside of supervised learning-based edge detection is the requirement of large amount of pixel-level annotations, which is extremely tedious to obtain manually. Meanwhile, manual labels are biased from person to person, the Multicue [18] dataset were labeled by six annotators whose annotations are often inconsistent and the ground truths were obtained by taking the average. To this end, we study the problem *self-supervised edge detection*, leveraging the untapped wealth of large-scale unlabeled image dataset. In fact, we found that self-training is especially suited for edge detection when reasonably designed and trained. Highly accurate and generalizable models can be learned with a large collection of unlabeled data.

We design a network that is self-trained with multilayer regularization and self-teaching. This multilayer self-training is inspired by the supervised HED method [12] which shows that edge detection networks are best learned with deep supervision on multiple layers. For the purpose of self-training, we impose *consistency regularization* [19], [20] which enforces the output from each of the multiple layers to be consistent for the input image and its perturbed counterpart. In particular, we adopt L0-smoothing [21] as the "perturbation" to encourage edge prediction lies in salient boundaries. This conforms with the cluster assumption in generic self-supervised learning [22] as L0-smoothing [21] is inherently a color-based pixel clustering. This way, our network learns to discriminate salient edges.

Meanwhile, our network is self-trained with multilayer teaching (supervision) by pseudo labels. The pseudo labels are initialized with Canny edges and iteratively refined by the network as training proceeds. In each iteration, the edge map output by the network is first binarized and pixel-wise multiplied with the low-threshold (over-detected) Canny edge map before being used as pseudo labels for the next round. This is essentially an entropy minimization which helps improve pixel classification with low-density separation [23].

The multilayer regularization and self-teaching together achieve a good balance of precision and recall of edge detection, making our method realizes good cross-dataset generalization. After self-trained on *COCO* validation dataset without using labels, it attains 1.2% improvement for optimal dataset scale (ODS) and 1.8% for optimal image scale (OIS) when tested on the unseen *BIPED* dataset, compared to supervised methods trained on *BSDS* dataset using all labels. Also, on *BIPED* dataset, based on our self-training method, two backbones of DexiNed and PiDiNet finetuned on 50% of the training set already outperforms all the state-of-the-art methods finetuned on 100% of the training set. With the same

training set, our method improves the original backbones in all cases.

In a nutshell, our contributions are mainly as follows.

1) We propose the first framework which enables self-training edge detection from single images.
2) We introduce multilayer consistency regularization and self-teaching in the context of self-trained edge detection.
3) We conduct extensive evaluations of the self-trained network and demonstrate its strong cross-dataset generality and the potential to promote edge detectors.

II. Related Work

A. Edge Detection

Existing edge detection methods can be categorized into three groups: traditional edge detector, learning-based methods, and deep learning-based ones.

Traditional edge detectors focus on utilizing image gradients to generate edges such as Sobel [24] and Canny [10]. Although they suffer from noisy pixels and do not consider semantic understanding, they are still widely used in applications such as image segmentation [4], [5] and image inpainting [8].

Learning based methods usually integrate various low-level features and train detectors to generate object-level contours, based on priors such as gradient descent [4] and decision tree [11]. Although these methods achieve better performance than traditional edge detectors, they have many limitations in challenging scenarios.

Amounts of deep learning-based methods have been proposed with the success of convolutional neural network (CNN). In early stage, there are patch-based approaches like DeepEdge [25] and DeepContour [26] which take pre-divided patches as input of CNNs to decide edge pixels. HED [12] is a pioneering work of end-to-end edge detection, with a network architecture based on VGG16 [27] and parameters adopted from pretrained models on ImageNet dataset [28]. Based on HED, RCF [13] combines richer features from each CNN layer and BDCN [14] proposed a bidirectional cascade structure to train the network with layer-specific supervisions. Efforts have also been made to design lightweight architectures for efficient edge detection including [15], [16], [29], where DexiNed [15] introduces Xception [30] to edge detection network and PiDiNet integrates the traditional edge detection operators into CNN models. In this article, unlike previous works, we neither focus on the network architecture design and efficiency improvements. We first introduce self-training into edge detection and explore a framework to utilize unlabeled dataset.

B. Self-Training

Self-training is a semi-supervised or unsupervised learning strategy that iteratively train the network with constantly updated pseudo labels for unlabeled training data. It is widely studied on image classification [31], [32], [33], and recently applied to high-level vision tasks such as semi-supervised segmentation [34], [35], [36], [37], [38]. However, in their cases, the pretrained models used in self-training are usually trained supervised on labeled data. Recently, some zero-shot methods based on transferable and adversarial networks [39], [40], [41], [42] also show the potential for pretraining models.

The most similar framework ULE [43] for self-training edge detection is based on optical flow estimation, by iteratively detecting motion edges as pseudo labels from videos and re-training the edge detector. However, learning from consecutive frames need extra cost of storage and calculation, and the focus of only motion edges limits the performance. In this work, we aim to enable self-training edge detection from single images.

To adapt with self-training on noisy pseudo labels, various kinds of perturbations and consistency methods are studied. Image perturbation methods [19], [20] augment the input images randomly and constraint the predictions of augmented images to be consistent with the original one. Feature perturbation method [44] uses multiple decoders and enforces the consistency between decoder outputs. Network perturbation method [45] applies two networks of the same structure with different initialization and impose the consistency between the predictions of perturbed networks. Moreover, Chen et al. [46] imposes the consistency using predictions of one network to supervise the other one. In this work, with the inspiration of previous methods, we also impose consistency regularization in our self-training framework. Specifically, L0-smoothing [21] is adopted to help the network learn to discriminate salient edges based on the cluster assumption, and a post-process is delicately designed for entropy minimization which helps improve pixel classification with low-density separation [23].

III. Method

In Section III-A, we provide problem formulations and overview of the pipeline including a self-training scheme consisting of multilayer teaching and consistencies. In Section III-B, we introduce multilayer teaching by noisy pseudo labels, to enable training on unlabeled images. In Section III-C, we introduce a multilayer consistency constraint to regularize the multilayer teaching, where we enforce the outputs from each of the multiple layers to be consistent for the input image and its smoothed counterpart. In Section III-D, we introduce the iterative self-training process including post-process for entropy minimization. In Section III-E, we adopt an uncertainty-aware strategy to filter pseudo labels for further retraining to avoid overfitting noisy edge pixels and thus improve the performances.

A. Method Overview

We aim to propose a self-supervised training scheme to utilize unlabeled images for edge detection.

To initialize the network as our phase-one model, we use pseudo labels generated by Canny [10] as supervision. We observe that even some edge pixels are missing in pseudo labels, the phase-one model can still predict themselves correctly, rather than overfitting to noisy pseudo labels, as most pixels are labeled correctly. It motivates the idea of using the phase-one model as initialization, and gradually improves pseudo labels in the self-training phase. Actually, there are

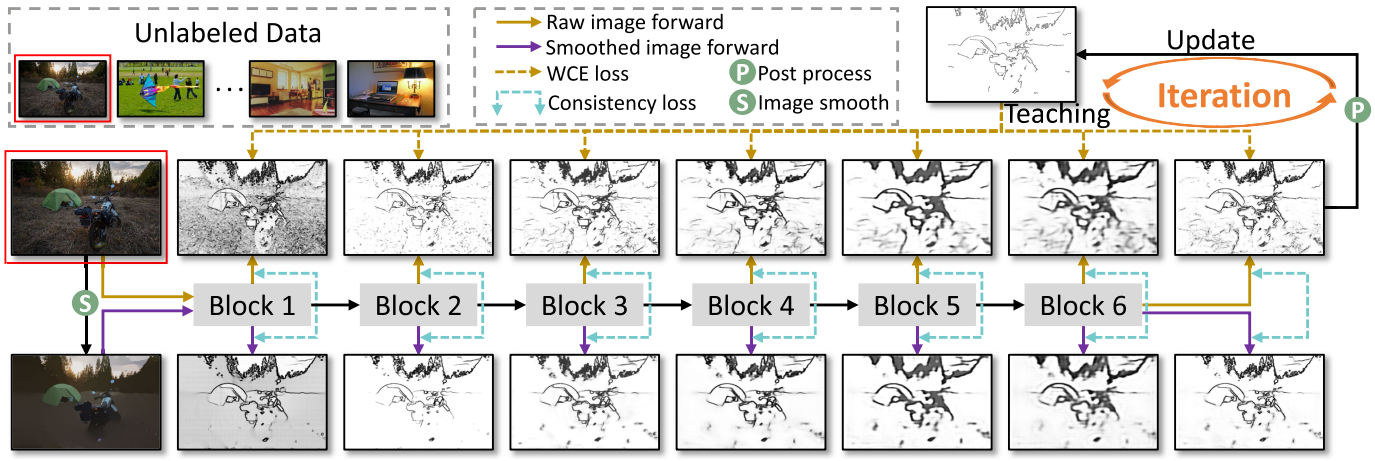

Fig. 1. Our self-training framework. The input is an unlabeled image and its smoothed counterpart. Consistency regularization and teaching happen in multiple layers of their output edge maps. Pseudo labels are iteratively generated and send into the network again for self-training.

several merits of adopting pseudo label learning for edge detection. First, the pixel-level annotations for edge detection are expensive and tedious to get by brute force. As a result, there are few edge detection datasets available. Also, scale of current datasets with annotations are quite small, leading overfitting in supervised training. Naturally, it motivates our work on utilizing large-scale unlabeled image datasets by self-training scheme. Moreover, despite that pseudo labels inevitably contain noises, most of them are correct enough. By introducing multilayer consistency regularization, pseudo labels can be gradually stable and less noisy to serve as extra training data to promote the performance.

As shown in Fig. 1, in the self-training phase, the pseudo labels and network predictions are improved jointly and iteratively. The pseudo labels teach the network at multiple layers with the proposed multilayer consistency as regularization, to suppress the complicated textures and prevent the network from generating redundant edges. During the self-training, the input image X and its perturbed one X' are sent into the same network $f(x)$. Multilayer consistency losses are defined between predictions of X and X'. In this article, we apply Gaussian blur followed by L0-smoothing [21] to X to get X'. The process can be logically demonstrated as below

$$\begin{array}{ccccc} X & \rightarrow & f(x) & \rightarrow & \hat{P} \rightarrow Y \\ \downarrow & \nearrow & & \searrow & \updownarrow \\ X' & & & & \hat{P}' \end{array} \tag{1}$$

where $\hat{P}$ is a set of predicted edge maps generated from multiple layers of the network, $\hat{P} = [\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n]$, where $\hat{p}_i$ has the same size as input image, and n is the number of outputs from each upsampling block. $\hat{P}$ and $\hat{P}'$ are predicted edge maps of multiple layers from X and X' respectively. Y denotes the pseudo labels post-processed from $\hat{P}$ for the next round training based on entropy minimization.

B. Multilayer Teaching

Deeply supervised schemes are studied and proven to be effective on edge detection [12]. The receptive field varies from different network layers. Low-level layers with small receptive fields are likely to detect fine details, while high-level layers tend to decide semantic boundaries. Providing teaching of edge maps at multiple layers is essential and effective (See Section IV-C for more details). Previously, edge detection networks perform better with deep supervisions at multiple layers [12], followed by several recent methods [13], [14], [15]. Since the distribution of edge/non-edge pixels is heavily biased in a natural image, we adopt the robust weighted cross entropy loss denoted as $\mathcal{L}_{\text{wce}}$ [13] at multiple layers to drive the training. Loss at pixel x_i of an upsampling block n is calculated by

$$l^n_{\text{wce}}(x_i, W) = \begin{cases} \alpha \cdot \log(1 - P(x_i, W)), & \text{if } y_i = 0 \\ 0, & \text{if } 0 < y_i < \eta \\ \beta \cdot \log P(x_i, W), & \text{otherwise} \end{cases} \tag{2}$$

in which

$$\begin{aligned} \alpha &= \lambda \cdot \frac{|Y^+|}{|Y^+| + |Y^-|} \\ \beta &= \frac{|Y^-|}{|Y+| + |Y-|} \end{aligned} \tag{3}$$

where W denotes the collection of all network parameters, Y^+ and Y^- denote the number of edge pixels and none-edge pixels in the ground truth, respectively. λ is a hyper-parameter to balance positive and negative samples. η is a threshold to filter out the less confident edge pixels in ground truths, to avoid confusing the network. Thus, loss after an upsampling block n for input image X of size $w \times h$ can be represented as

$$l^n_{\text{wce}}(X, W) = \sum_{i=1}^{w \times h} l^n_{\text{wce}}(x_i, W). \tag{4}$$

We define different weight δ^n for each level, and the final $\mathcal{L}_{\text{wce}}$ is calculated as

$$\mathcal{L}_{\text{wce}} = \sum_{n=1}^{N} \delta^n \times l^n_{\text{wce}}(X, W). \tag{5}$$

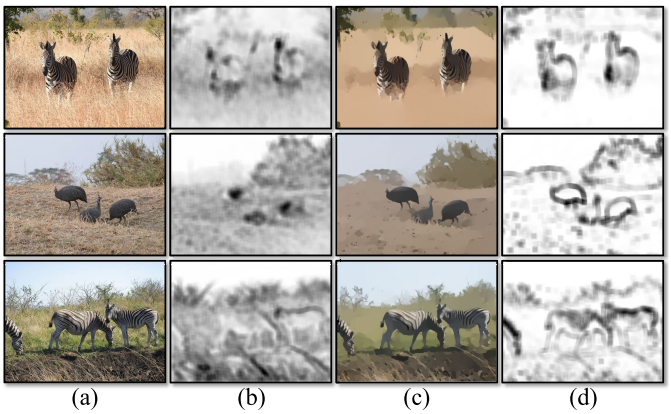

Fig. 2. **Cluster assumption in edge detection.** (a) Examples from COCO dataset. (b) *For raw images*. The average Euclidean distance between each patch of size 20 × 20 centered at a given spatial location extracted from the input images, and its 8 neighboring patches. (c) Results from applying Gaussian blur followed by L0-smoothing to raw images. (d) *For smoothed images*. The average Euclidean distance map. Darker regions indicate higher distances.

C. Multilayer Regularization

Consistency regularization is widely studied to make the decision boundary lies in low-density areas. We enforce the consistency at multiple layers between the input image and its smoothed counterpart. Here we apply Gaussian blur followed by L0-smoothing [21] to the input image, imposing consistency to help the network converge to predict more visually salient edges. As illustrated in Fig. 2, edge pixels are easier to be classified in smoothed images which suppress complicated texture pixels.

The multilayer consistency works as a regularizer to prevent the network from generating increasingly redundant and noisy edges as in pseudo labels. Although image noises and redundant edges are expected to be filtered out after L0-smoothing as an edge-preserving smoothing approach, it unavoidably causes over-sharpening in challenging circumstances [21]. Therefore, edge maps predicted from smoothed images still contain noises and it has been proven in [47] that though categorical cross entropy loss converges rapidly, it is sensitive to noises. To avoid overfitting to incorrect labels during self-training, we apply L2 norm, a symmetrical loss function, which is more noise-tolerant on classification problems based on the theory of risk minimization [47], [48], where edge detection is exactly a binary classification problem at pixel level. As L2 loss is theoretically and practically effective in our self-training method for its noise tolerance. The pixel-wise squared difference is computed between the edge map of each block predicted from X and the corresponding edge map from X'. The multilayer consistency loss for block n is formulated as

$$l^n_{\text{mlc}}(X, W) = \sum_{i=1}^{w\times h}\left[P(x_i, W) - P\left(x'_i, W\right)\right]^2 \tag{6}$$

same with the $\mathcal{L}_{\text{wce}}$, the complete $\mathcal{L}_{\text{mlc}}$ with layer weights δ is calculated as

$$\mathcal{L}_{\text{mlc}} = \sum_{n=1}^{N} \delta^n \times l^n_{\text{mlc}}(X, W). \tag{7}$$

Combining multilayer teaching and regularization, with a trade-off weight μ, the final loss is calculated as

$$\mathcal{L} = \mathcal{L}_{\text{wce}} + \mu\mathcal{L}_{\text{mlc}}. \tag{8}$$

Algorithm 1 Self-Training Edge Detection (**STEdge**)

Input:
The unlabeled images X; The number of epochs trained in each round E; The termination parameter $T\%$.

Output:
The well-trained edge detector M with parameters θ;
The images X with reasonable edge pseudo labels Y.

1: Initialize the network weights supervised trained by Eq. 5. Initial pseudo labels are generated by performing Canny of high threshold on blurred images:
$Y = Canny(Blur(X), thres\_high)$,
$\theta_0 = Train(M(\theta), X, Y, \mathcal{L}_{\text{wce}})$;
2: Prepare smoothed images for consistency regularization:
$X' = L0\_smoothing(Blur(X))$;
3: **for** round k in $\{1, \ldots, K\}$ **do**
4: Get the edge maps predicted by current weights:
$\hat{P} = [\hat{p_1}, \hat{p_2}, \ldots, \hat{p_n}] = M(\theta_{k-1}, X)$;
5: Post-process to get new pseudo labels by performing element-wise multiplication on over-detected Canny and our binarized edge map:
$A_k = Adaptive\_binarization(\hat{p_n})$,
$C_k = Canny(Blur(X), thres\_low)$,
$Y_k = Connectivity\_filter(A_k \odot C_k)$;
6: Train on current pseudo labels for E epochs by Eq. 5 and Eq. 7:
$\theta_k = Train(M(\theta_{k-1}), X, X', Y, \mathcal{L}_{\text{wce}}, \mathcal{L}_{\text{mlc}})$;
7: Decide if the self-training process is stable and needed to be terminated, the number of edge pixels in the pseudo labels produced in round k is termed N^k_{edge}:
if $\frac{N^i_{edge} - N^{i-1}_{edge}}{N^i_{edge}} < T\%$ **then**
break;
end if.
8: **end for**.

D. Iterative Self-Training

Our approach includes two phases: the initialization phase and the self-training phase. Phase-one training aims to get an initial model to warm up self-training, which is needless to be very powerful. In phase-one, we apply Canny of high thresholds to the unlabeled dataset D^u to generate pseudo labels, then train the network using weighted cross entropy loss.

Phase two is the iterative self-training, optimizing the network and pseudo labels simultaneously as training proceeds. Based on entropy minimization, the predicted edge maps are post-processed to be the updated labels for the next round. In Algorithm 1, $\hat{p_n}$ is the network prediction of the last layer, with values between 0 and 1 at each pixel. In post-processing, $\hat{p_n}$ is multiplied with over-detected Canny edges (with low thresholds) in a pixel-wise level, and then filtered by connected

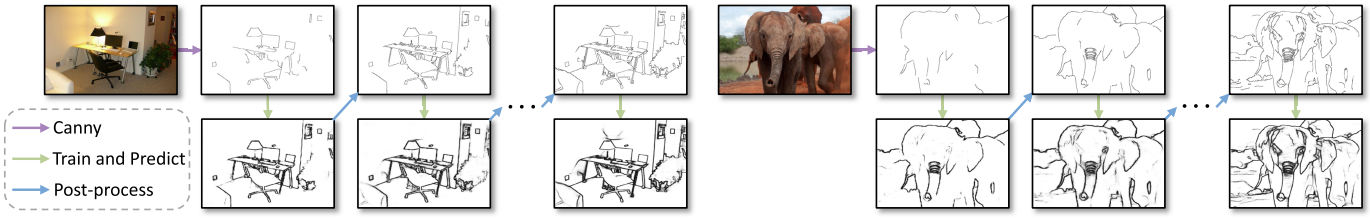

Fig. 3. Evolving process of two examples during the iterative training. For each example, the top left is the input image, the first row illustrates the updating pseudo labels, and the second row shows the network predictions as iteration proceeds.

areas with low connectivity to clear noise edges. The updating of pseudo labels is critical from two aspects. On the one hand, if we treat the predicted edge maps as new pseudo labels, the loss and gradients will be all zeros. On the other hand, when the predicted probability map is transferred into binary edge map, the entropy of each pixel is minimized, enforcing low-density seperations [23] by predicted edges.

During training, we repeat the following process.

1) Predict then post-process $\hat{p_n}$ to generate new pseudo labels Y;
2) Training the network for E epochs using updated pseudo labels.

We refer to an iteration of (1) and (2) as one *round*. The network is self-trained iteratively for several rounds until convergence. There are several factors enabling the pseudo labels to become gradually stable during self-training. First, in our post-processing, we apply Canny with preset thresholds to generate updated pseudo labels, which means the edge pixels have an upper bound (Canny edges on original input). Also, though Canny edges contain a lot of noisy pixels other than salient edges, our multilayer regularization is introduced to encourage higher importance on salient edges and suppress noisy ones. With the premise of limited upper bound, as the consistency loss converges, the pseudo labels become gradually better and stable during the self-training phase.

The self-training process is terminated when the pseudo edge maps change slightly. Specifically, we finish the self-training when the ratio of increased edge pixel number to the total edge pixel number is smaller than a termination value $T\%$ (We set $T = 2$ in all experiments). We summarize the whole self-training process as in Algorithm 1. The qualitative evolving process is shown in Fig. 3.

E. Uncertainty-Aware Re-Training

To make full use of the pseudo labels from each self-training round, and avoid overfitting noisy ones, we adopt an uncertainty-aware strategy to filter pseudo labels for further retraining. Specifically, only the edge pixels that appears in pseudo labels of every self-training round that can be treated as true edge pixels (set to 1), while other edge pixels are uncertain ones [set to a value which is smaller than η in (2)] that will be ignored during the loss calculation when retraining. Examples are presented in Fig. 4. The filtered pseudo labels are also generated without any human annotations, which can better play as extra and free datasets to be trained by any learning-based edge detectors from the scratch.

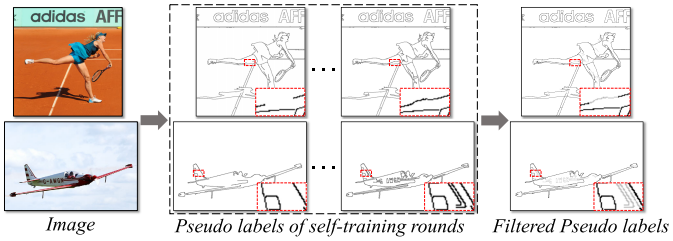

Fig. 4. Two examples from *COCO val2017* dataset of the uncertainty-aware pseudo label filtering process. Zoom-in is recommended for better observation. Edge pixels of low responses are uncertain pixels to be ignored when retraining with filtered pseudo labels.

IV. Experiments

A. Datasets

Datasets adopted in our experiments include *COCO* [49], *BSDS* [4], *Multicue* [18], and *BIPED* [15]. *COCO* is a widely used computer vision dataset containing common objects for natural scenes, in this article, we adopt *COCO val2017* of 5000 images as unlabeled dataset for the self-training. *BSDS* is designed for image segmentation and boundary detection, consisting of 200, 100, and 200 images for training, validation, and testing, respectively. Each image is labeled by multiple annotators and the final ground truth is calculated by taking their average. As previous works, we augment the training and validation sets with flipping (2×), scaling (3×), and rotation (16×) during the training. It also plays as the training set of several state-of-the-art networks to compare the generalization ability. *Multicue* is a commonly-used benchmark dataset for edge detection, containing 100 images from challenging natural scenes. Each scene contains a sequence of images from the left and right views, captured by a stereo camera. Only the last frames of every left-view sequences are labeled by annotators. For these 100 images, each of them is annotated by several people as well. As previous works, we randomly split them into training and evaluation sets, consisting of 80 and 20 images, respectively. *BIPED* dataset contains 250 annotated images of outdoor scenes, splitting into a training set of 200 images and a testing set of 50 images. All images are carefully annotated at single-pixel width. The resolution of *Multicue* and *BIPED* are both 1280 × 720. Thus, we augment each image by flipping (2×), cropping (3×), and rotation (16×), leading to a training set that is 96 times larger than the original dataset. We conduct ablation study and the evaluation of cross-dataset generality on *BIPED* dataset. The performances after finetuning on *BSDS*, *Multicue*, and *BIPED* are also compared with the state-of-the-art edge detectors.

TABLE I
ABLATION STUDIES OF SEVERAL CRITICAL MODULES ON *BIPED* DATASET

Method	ODS	OIS	AP
Canny (20, 40)	.266	\	\
Canny (100, 200)	.680	\	\
Canny (200, 300)	.597	\	\
Phase-one Model	.705	.724	.684
Consistency (only last-layer teaching)	.739	.757	.697
Multi-layer teaching	.754	.769	.738
Multi-layer teaching + Consistency	.760	.783	**.798**
Multi-layer teaching + Consistency + Re-training	**.788**	**.803**	.754

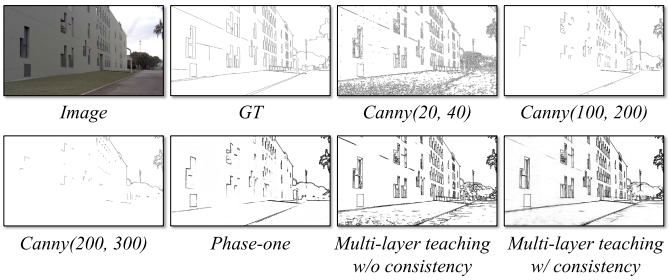

Fig. 5. Edge maps from Canny detectors with different thresholds and three different settings of our STEdge on the unseen *BIPED* dataset. STEdge are trained without any human annotations.

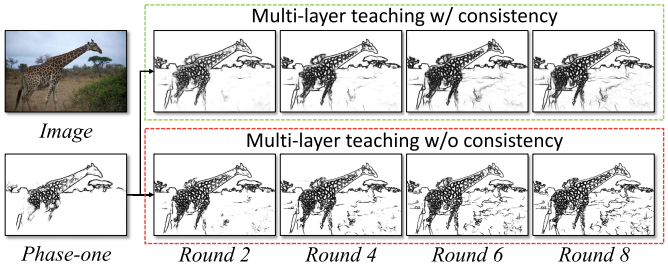

Fig. 6. Example from *COCO val2017* of different rounds self-trained on *COCO val2017* with and without consistency. Initializing from the same phase-one model, the edge maps trained with consistency learning are qualitatively much more reasonable.

B. Implementation Details and Evaluation Metrics

The proposed self-training framework is implemented in PyTorch [50], and can be applied to any edge detection network with light refinements. The filtered pseudo labels can also be trained with any edge detectors from the scratch. Following previous works, the parameter λ to balance positive and negative samples and the threshold η in $\mathcal{L}_{\text{wce}}$ is set to 1.1 and 0.3 in all cases, respectively. In the final loss function $\mathcal{L} = \mathcal{L}_{\text{wce}} + \mu\mathcal{L}_{\text{mlc}}$, the selection of the trade-off weight μ do not bring much difference and we just set it to 1, and the termination value $T\%$ in Algorithm 1 is set to 2% (generally 6–8 iterations). No data augmentation strategies are adopted when self-training with pseudo labels. All Experiments are conducted on an NVIDIA GeForce RTX 3080 Ti GPU with 12 GB memory.

In this article, two current lightweight edge detectors DexiNed [15] and PiDiNet [16] are adopted to show the effectiveness of our STEdge. Both networks are trained with the Adam optimizer [51]. For the backbone of DexiNed, the learning rate is 0.0001 with the batch size of 8. For the backbone of PiDiNet, we train the network with the batch size of 24, decaying in a multistep way at an initial learning rate of 0.005.

For evaluations, like previous works, standard nonmaximum suppression (NMS) will be applied to thin detected edges before evaluation. We adopt three commonly used evaluation metrics for edge detection, the F-measure of ODS, the F-measure of OIS, and average precision (AP). ODS and OIS are two strategies to transform the output probability map into a binary edge map. ODS employs a fixed threshold for all images in the dataset while OIS chooses an optimal threshold for each image. The F-measure is defined as $F = (2 \cdot P \cdot R)/(P + R)$, where P denotes precision and R denotes recall. AP represents the area under the precision–recall curve. However, in some cases when the precision–recall curve is shorter and does not cover the whole range, it gives a lower AP even if the predictions are better and satisfactory. In such cases, AP may be less reliable for evaluations compared with ODS and OIS. For ODS and OIS, the maximum allowed distances between corresponding pixels from predicted edges and ground truths are set to 0.0075 for all experiments.

C. Ablation Study

We evaluate the effectiveness of each part of our STEdge pipeline by conducting several ablations.

In terms of the DexiNed backbone, Table I presents the quantitative comparisons of Canny detectors with different thresholds, the phase-one model, only last-layer teaching, multilayer teaching with and without consistency regularization and the final model retrained with filtered uncertainty-aware pseudo labels. Qualitative comparisons are in Fig. 5.

As expected, it is difficult to set proper thresholds for Canny detectors to fit all images, which leads to low F-scores. However, from the noisy Canny pseudo labels, the networks successfully learn general features for edge detection. The phase-one model training from Canny labels already outperforms Canny detectors. Also, we can observe that the performance drops significantly if only the last layer is supervised, which reveals the necessity of multilayer teaching. Meanwhile, there will be much more redundant edge pixels without regularization by consistency loss. As shown in Fig. 6, the consistency loss works as a regularizer to stop the network from labeling weak noisy edges that violate the consistencies.

Through multilayer teaching and consistency, the networks continue to get performance boosting. Additionally, we also demonstrate that if we introduce uncertainty to further filter pseudo labels and retraining with them, the network performance will continue to improve. Note that the whole self-training process employ only *unlabeled* data.

Fig. 3 illustrates some examples from the *COCO-val2017* dataset. The initial pseudo label is sparse and noisy from Canny detector with high thresholds. As iterative training proceeds, the predicted edge maps and pseudo labels are evolving together, gradually getting better.

We also evaluate the influences of different Canny pseudo labels used in the phase-one model, quantitative results and the

TABLE II

QUANTITATIVE EVALUATION OF SELF-TRAINING WITH UNLABELED *COCO-val2017* DATASET AND INITIALIZING WITH CANNY OF VARIOUS THRESHOLDS

Canny Threshold	ODS	OIS	AP
20-40	.724	.746	.678
50-100	.763	.784	.796
100-200	.759	.782	.777
150-300	.763	.784	.798
200-300	.760	.782	.798
300-400	.713	.731	667

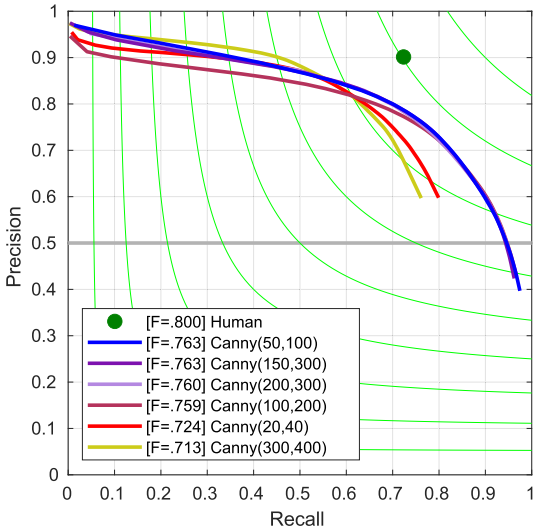

Fig. 7. Final performance self-trained from phase-one models with different Canny thresholds. Except for some extreme cases, curves of moderate initialization have similar performances.

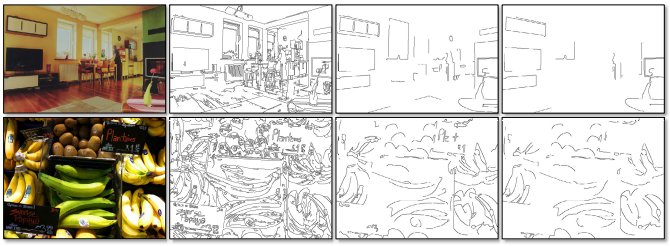

Image *Canny(20, 40)* *Canny(200, 300)* *Canny(300, 400)*

Fig. 8. Examples of Canny detectors with various thresholds on *COCO–val2017* dataset.

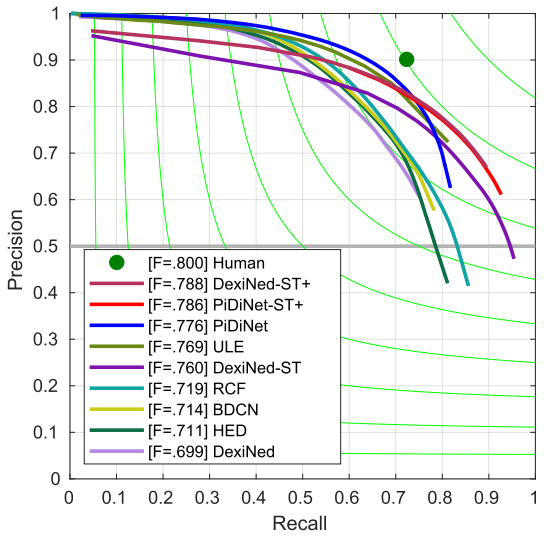

Fig. 9. Precision–recall curves of state-of-the-art methods trained on *BSDS* and evaluated on *BIPED*.

TABLE III

COMPARISONS BETWEEN STATE-OF-THE-ART METHODS EVALUATED ON *BIPED*. "-ST" MEANS ADOPTING OUR SELF-TRAINING STRATEGY BASED ON AN EDGE DETECTION BACKBONE, AND "-ST+" MEANS APPLYING FURTHER UNCERTAINTY-AWARE RETRAINING ON *COCO-val2017* (5 K IMAGES) AS DESCRIBED IN SECTION III-E. ULE [43] IS SELF-TRAINED ON (VSB) [53] AND YTB [54] DATASETS (∼50 K FRAMES). * **DENOTES TRAINING WITH PSEUDO LABELS WITHOUT HUMAN ANNOTATIONS, AND SAME FOR OTHER TABLES**

Method	Pretraining	Training	ODS	OIS	AP
HED	ImageNet	BSDS	.711	.725	.714
RCF	ImageNet	BSDS	.719	.732	.749
BDCN	ImageNet	BSDS	.714	.725	.687
DexiNed	\	BSDS	.699	.716	.664
PiDiNet	\	BSDS	.776	.785	.740
DexiNed-ST	\	BSDS*	.719	.734	.732
DexiNed-ST	\	COCO-val2017*	.760	.783	**.798**
DexiNed-ST+	\	COCO-val2017*	.788	.803	.754
DexiNed-ST+	BSDS	COCO-val2017*	.768	.795	.768
PiDiNet-ST+	\	COCO-val2017*	.786	.804	.259
PiDiNet-ST+	BSDS	COCO-val2017*	**.790**	**.809**	.387
ULE [43]	\	VSB*+YTB*	.769	.782	.742

precision–recall curves are shown in Table II and Fig. 7. When the number of edges on initial edge maps is moderate, initializing from different Canny pseudo labels does not make a big difference to the final model. However, the final performance drops if the edge pixels on initial edge maps are too sparse (300–400) or too dense (20–40). The sensitivity to the quality of initial pseudo labels is one of the limitations of our method and could be tackled by introducing extra constraints. Some visual examples are provided in Fig. 8. Since the qualitative performance shows setting thresholds as (200, 300) is slightly better, we use this setting as initialization throughout this article.

D. Comparisons of Generalization

In this section, we study the generalization ability of the proposed approach. We compare our network with several state-of-the-art networks including HED [12], RCF [13], BDCN [14], DexiNed [15], and PiDiNet [16]. For generalization, we train all these backbones on *BSDS*, and evaluate them on *BIPED*. All other methods are trained supervisedly with labeled edge maps while our network is self-trained without using any human annotations. Note that our networks do not initialized from any pretrained models while others are pretrained with ImageNet [28].

Moreover, we also compare the performance with the most similar self-training edge detection framework ULE (Unsupervised Learning of Edges) [43]. ULE starts with Sobel edges [24] and adopts EpicFlow [52] for optical flow estimation. The motion edges are detected on colored flow maps and then applied to train the new edge detector, which will further detect edges for the next iteration. Since optical flow estimation needs consecutive frames, ULE combines videos from two different datasets: the Video Segmentation Benchmark (VSB) [53] and the YouTube Object dataset (YTB) [54], leading to ∼50 K frames after frame filtering.

The Precision-Recall curves are presented in Fig. 9. By observing the quantitative results in Table III and

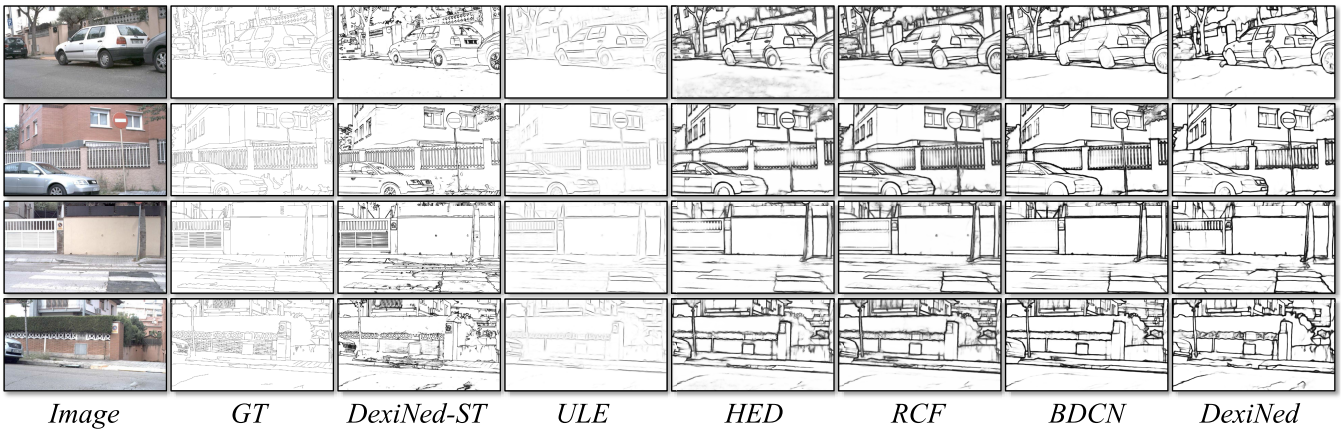

Fig. 10. Qualitative comparisons on *BIPED* dataset. ULE [43] is self-trained on VSB and YTB dataset (∼50 K frames) and "DexiNed-ST" means self-trained on *COCO val2017* dataset (5 K images) without any human annotations using the DexiNed backbone. Other networks are trained on labeled *BSDS* dataset.

TABLE IV

COMPARISON WITH THE STATE-OF-THE-ART WORKS ON *Multicue* DATASET. 50% MEANS FINETUNING USING HALF OF THE TRAINING SET

Method	Pretraining	Training	ODS	OIS	AP
Human	\	\	.750	\	\
Multicue	\	Multicue	.830	\	\
HED	ImageNet	Multicue	.851	.864	.890
RCF	ImageNet	Multicue	.857	.862	\
BDCN	ImageNet	Multicue	.891	.898	.935
DexiNed	\	Multicue	.872	.881	.871
DexiNed-ST+	COCO-val2017*	Multicue 50%	.889	.896	.943
DexiNed-ST+	COCO-val2017*	Multicue	**.895**	**.900**	**.948**
PiDiNet	\	Multicue	.855	.860	\
PiDiNet-ST+	COCO-val2017*	Multicue 50%	.878	.885	.903
PiDiNet-ST+	COCO-val2017*	Multicue	**.881**	**.887**	.901

TABLE V

COMPARISON WITH THE STATE-OF-THE-ART WORKS ON *BIPED* DATASET. 50% MEANS FINETUNING USING HALF OF THE TRAINING SET, AND SAME FOR OTHER TABLES

Method	Pretraining	Training	ODS	OIS	AP
HED	ImageNet	BIPED	.829	.847	.869
RCF	ImageNet	BIPED	.843	.859	.882
BDCN	ImageNet	BIPED	.839	.854	.887
DexiNed	\	BIPED	.857	.861	.805
DexiNed-ST+	COCO-val2017*	BIPED 50%	.859	.873	.737
DexiNed-ST+	COCO-val2017*	BIPED	**.867**	**.878**	.732
PiDiNet	\	BIPED	.868	.876	.912
PiDiNet-ST+	COCO-val2017*	BIPED 50%	.876	.881	.913
PiDiNet-ST+	COCO-val2017*	BIPED	**.878**	**.883**	**.923**

qualitative results in Fig. 10, several interesting conclusions can be drawn.

1) Edge detectors with our self-training strategy outperform other methods significantly with more precise details, including those supervised on *BSDS* and self-trained on larger datasets: In Tables IV and V, previous works are capable of performing well when training and evaluating on the same datasets. However, from Table III we can see that their generalization ability is limited when evaluating on unseen datasets, showing that the generalization ability of edge detection networks remains an open problem, which is important in practical applications. ULE can detect most motion edges well; however, there also exist other types of semantic edges in natural images, which limits its performance.
2) Training on the pseudo labels generated by our self-training method can bring free performance boost to edge detectors: Compared with the original version, the DexiNed self-trained on unlabeled *BSDS* dataset can already achieve comparable performance on unseen *BIPED* dataset. The performance further boosts significantly for both the backbones of DexiNed and PiDiNet when self-trained on the *COCO val* dataset, whether the network is pretrained (on *BSDS*) or not, revealing the potential of exploring more unlabeled datasets.

E. Comparisons With the State-of-the-Arts

1) Performance on BSDS: We compare our model with traditional detectors including SCG [55], PMI [56], and OEF [57], and deep-learning-based detectors including DeepContour [26], HED [12], RCF [13], BDCN [14], DexiNed [15], and PiDiNet [16]. The best results of all the methods are taken from their publications. Some qualitative results are illustrated in Fig. 11. With the aid of STEdge, edge detectors can predict more detailed edges accurately. Quantitative results are reported in Table VI. As lightweight edge detectors compared with VGG-architecture-based networks [12], [13], [14], DexiNed and PiDiNet with our self-training method still achieve comparable performance with much faster inference time. Also, both of them can benefits from pre-training on *COCO val* dataset for free performance boost, for DexiNed, it attains 1.4% improvement for ODS and 1.4% for OIS, and 0.7% for ODS and 0.9% for OIS in terms of PiDiNet.

2) Performance on Multicue: For a fair comparison, we follow the experiments of previous works [13], [14], finetuning our self-trained model on the split 80% training set and test on the remaining 20%. We average the scores of three independent trials as the final results. Some Qualitative examples are presented in Fig. 12. After self-training on *COCO val2017*, even finetuned on 50% of the training set, the edge detectors can achieve almost the same performances compared with those finetuned on the whole training set, showing the potential of our self-training method for few-

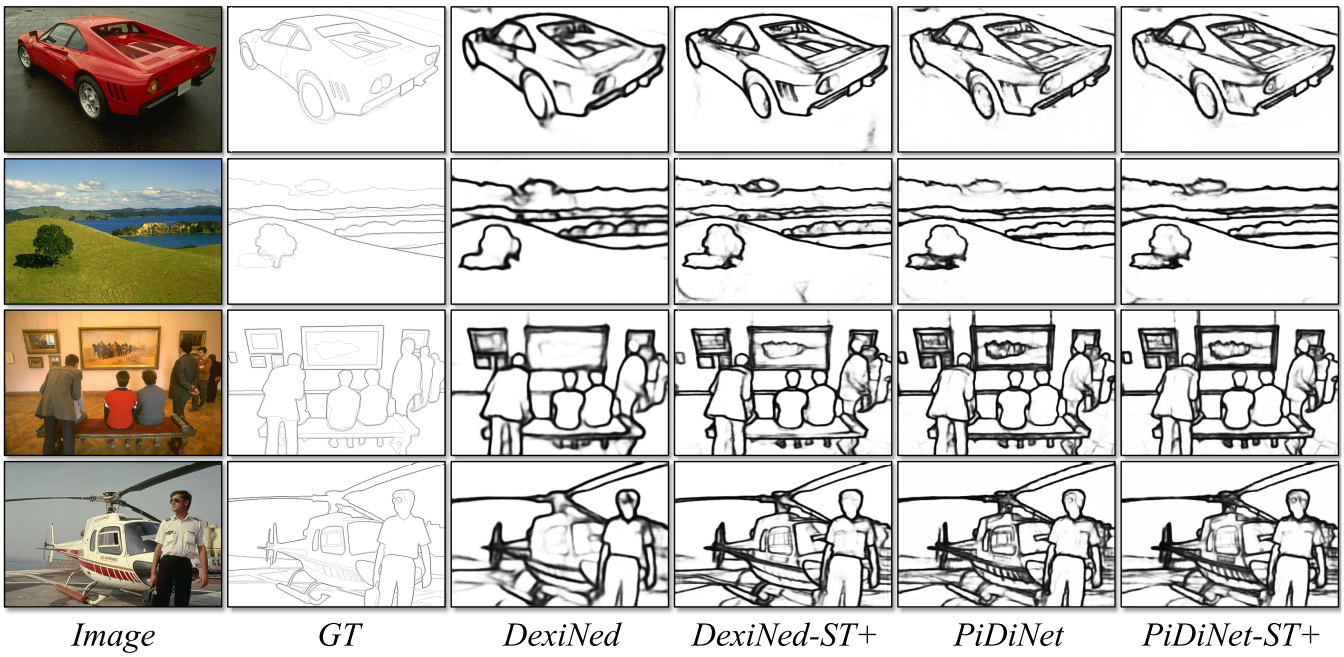

Fig. 11. Qualitative detection examples from *BSDS* dataset before NMS of two backbones combined with our STEdge."-ST+" means applying our whole self-training pipeline with edge detection detectors, and same for other figures. With the aid of STEdge, more detailed edges are predicted precisely compared with the original ones.

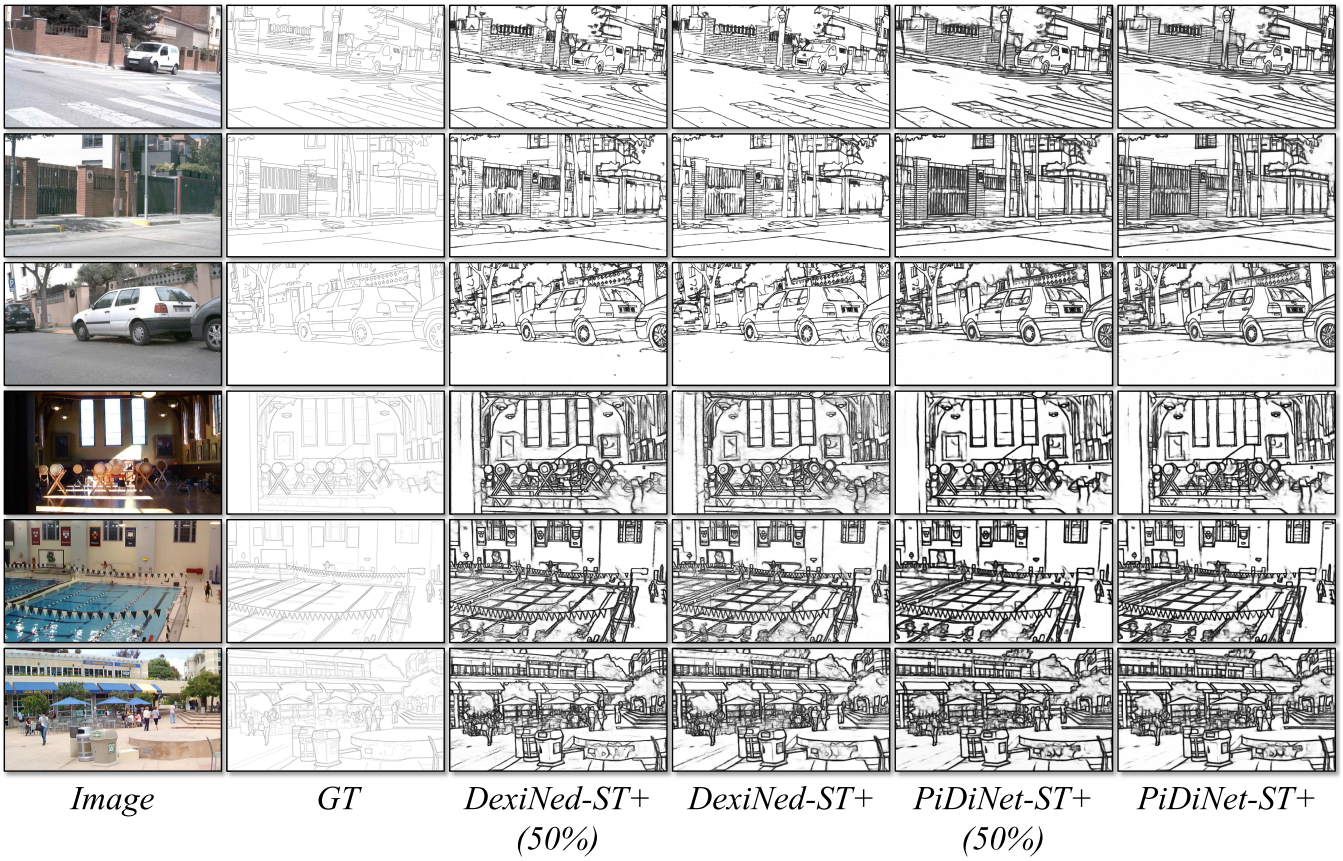

Fig. 12. Qualitative detection results before NMS of two backbones combined with our STEdge. The first three rows are examples from *BIPED* dataset and the last three rows come from *Multicue* dataset. "(50%)" means training using half of the training set. With the aid of our STEdge, edge detectors finetuned only on half of the training set already achieve the state-of-the-art performances.

shot edge detection. The comparisons to recent state-of-the-art methods are reported in Table IV, where our STEdge with DexiNed backbone achieves the best performance. Moreover, we can observe that, for both DexiNed and PiDiNet with STEdge, even finetuned on 50% of the training set, their performance already outperform most of previous methods and their original performance without self-training on *COCO val*. Especially, if trained with the same 100% training set,

TABLE VI
QUANTITATIVE RESULTS OF SEVERAL RECENT STATE-OF-THE-ART WORKS ON *BSDS* DATASET

Method	Pretraining	Training	ODS	OIS	AP
SCG	\	BSDS	.739	.758	.773
PMI	\	BSDS	.741	.769	.799
OEF	\	BSDS	.746	.770	.820
DeepContour	\	BSDS	.757	.776	.800
HED	ImageNet	BSDS	.788	.808	.840
RCF	ImageNet	BSDS	.798	.815	\
BDCN	ImageNet	BSDS	**.806**	**.826**	**.847**
DexiNed	\	BSDS	.760	.779	.690
DexiNed-ST+	COCO-val2017*	BSDS	**.774**	**.793**	.770
PiDiNet	\	BSDS	.789	.803	\
PiDiNet-ST+	COCO-val2017*	BSDS	**.796**	**.812**	**.796**

with the aid of our self-training strategy, DexiNed attains 2.3% improvement for ODS and 1.9% for OIS, and PiDiNet attains 2.6% improvement for ODS and 2.7% for OIS.

3) Performance on BIPED: BIPED is a recent dataset with well-annotated edge maps. We also finetune the self-trained models on 50% and 100% of the training split, which correspond to 100 and 200 images, respectively. As shown in Table V, DexiNed and PiDiNet with STEdge method also already achieve the best performance. With more labeled images (100%) for training, the performance gets substantially higher. By self-training on *COCO val* dataset freely, DexiNed attains 0.7% improvement for ODS and 1.6% for OIS, PiDiNet attains 1% improvement for ODS and 0.7% for OIS.

The experiments demonstrate the superiority of the proposed self-training scheme and multilayer consistency regularization. By exploring unlabeled datasets, the generalization ability and performance are both improved significantly.

V. CONCLUSION AND LIMITATIONS

We propose a simple but effective self-training framework for edge detection named STEdge, to leverage unlabeled image datasets. To the best of our knowledge, it is among the first self-training pipeline proposed for edge detection. The framework consists of multilayer teaching by noisy pseudo labels and consistency regularization to suppress complicated textures. During iterative training, network predictions and the noisy pseudo labels are evolving simultaneously. Experimental results show the superiority of the proposed STEdge on generalization ability and edge detection performance on several benchmark datasets. In the future, we plan to explore the full pipeline of utilizing unlabeled web images for self-training edge detection and further improve the cross-dataset generality.

Limitations: The performance drops when the initial edge maps are too sparse or too dense, the initialization strategy with designed constraints could be one of the future direction. Although applying STEdge to edge detectors can achieve better performances, the utilization of unlabeled data is still unsatisfactory. Some pseudo labels still contain noisy pixels surrounding image areas of complicated textures, degenerating the self-training performance. A well-designed pseudo label sampling strategy during re-training is also an interesting direction to explore.

REFERENCES

[1] J. Li, Z. Wang, Z. Pan, Q. Liu, and D. Guo, "Looking at boundary: Siamese densely cooperative fusion for salient object detection," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Oct. 6, 2021, doi: 10.1109/TNNLS.2021.3113657.

[2] G. Zhu, J. Li, and Y. Guo, "Supplement and suppression: Both boundary and nonboundary are helpful for salient object detection," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Nov. 24, 2021, doi: 10.1109/TNNLS.2021.3127959.

[3] M. Li, D. Chen, S. Liu, and F. Liu, "Semisupervised boundary detection for aluminum grains combined with transfer learning and region growing," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Dec. 20, 2021, doi: 10.1109/TNNLS.2021.3133760.

[4] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 898–916, May 2011.

[5] C. Rother, V. Kolmogorov, and A. Blake, "'GrabCut'—Interactive foreground extraction using iterated graph cuts," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 309–314, 2004.

[6] C. Li, W. Xia, Y. Yan, B. Luo, and J. Tang, "Segmenting objects in day and night: Edge-conditioned CNN for thermal image semantic segmentation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 7, pp. 3069–3082, Jul. 2021.

[7] B. Zhao and X. Li, "Edge-aware network for flow-based video frame interpolation," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Jun. 8, 2022, doi: 10.1109/TNNLS.2022.3178281.

[8] K. Nazeri, E. Ng, T. Joseph, F. Z. Qureshi, and M. Ebrahimi, "EdgeConnect: Generative image inpainting with adversarial edge learning," 2019, *arXiv:1901.00212*.

[9] F. Fang, J. Li, Y. Yuan, T. Zeng, and G. Zhang, "Multilevel edge features guided network for image denoising," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 9, pp. 3956–3970, Sep. 2021.

[10] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 6, pp. 679–698, Nov. 1986.

[11] P. Dollár and C. L. Zitnick, "Fast edge detection using structured forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 8, pp. 1558–1570, Aug. 2015.

[12] S. Xie and Z. Tu, "Holistically-nested edge detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1395–1403.

[13] Y. Liu, M. Cheng, X. Hu, K. Wang, and X. Bai, "Richer convolutional features for edge detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5872–5881.

[14] J. He, S. Zhang, M. Yang, Y. Shan, and T. Huang, "Bi-directional cascade network for perceptual edge detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3823–3832.

[15] X. Soria, E. Riba, and A. Sappa, "Dense extreme inception network: Towards a robust CNN model for edge detection," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2020, pp. 1912–1921.

[16] Z. Su et al., "Pixel difference networks for efficient edge detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 5097–5107.

[17] M. Pu, Y. Huang, Y. Liu, Q. Guan, and H. Ling, "EDTER: Edge detection with transformer," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 1392–1402.

[18] D. A. Mély, J. Kim, M. McGill, Y. Guo, and T. Serre, "A systematic comparison between visual cues for boundary detection," *Vis. Res.*, vol. 120, pp. 93–107, Mar. 2016.

[19] G. French, T. Aila, S. Laine, M. Mackiewicz, and G. Finlayson, "Semi-supervised semantic segmentation needs strong, high-dimensional perturbations," 2019, *arXiv:1906.01916*.

[20] J. Kim, J. Jang, H. Park, and S. Jeong, "Structured consistency loss for semi-supervised semantic segmentation," 2020, *arXiv:2001.04647*.

[21] L. Xu, C. Lu, Y. Xu, and J. Jia, "Image smoothing via L_0 gradient minimization," in *Proc. SIGGRAPH Asia Conf.*, Dec. 2011, pp. 1–12.

[22] O. Chapelle and A. Zien, "Semi-supervised classification by low density separation," in *Proc. Int. Workshop Artif. Intell. Statist.*, 2005, pp. 57–64.

[23] Y. Grandvalet and Y. Bengio, "Semi-supervised learning by entropy minimization," in *Proc. CAP*, vol. 367, 2005, pp. 281–296.

[24] J. Kittler, "On the accuracy of the Sobel edge detector," *Image Vis. Comput.*, vol. 1, no. 1, pp. 37–42, Feb. 1983.

[25] G. Bertasius, J. Shi, and L. Torresani, "DeepEdge: A multi-scale bifurcated deep network for top-down contour detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 4380–4389.

[26] W. Shen, X. Wang, Y. Wang, X. Bai, and Z. Zhang, "DeepContour: A deep convolutional feature learned by positive-sharing loss for contour detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3982–3991.

[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[28] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[29] J. K. Wibisono and H.-M. Hang, "FINED: Fast inference network for edge detection," 2020, *arXiv:2012.08392*.

[30] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1800–1807.

[31] H. Scudder, "Probability of error of some adaptive pattern-recognition machines," *IEEE Trans. Inf. Theory*, vol. IT-11, no. 3, pp. 363–371, Jul. 1965.

[32] S. Fralick, "Learning to recognize patterns without a teacher," *IEEE Trans. Inf. Theory*, vol. IT-13, no. 1, pp. 57–64, Jan. 1967.

[33] D.-H. Lee et al., "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Proc. Workshop Challenges Represent. Learn. ICML*, vol. 3, no. 2, 2013, p. 896.

[34] L.-C. Chen et al., "Naive-student: Leveraging semi-supervised learning in video sequences for urban scene segmentation," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2020, pp. 695–714.

[35] B. Zoph et al., "Rethinking pre-training and self-training," 2020, *arXiv:2006.06882*.

[36] Y. Zhu et al., "Improving semantic segmentation via self-training," 2020, *arXiv:2004.14960*.

[37] W.-C. Hung, Y.-H. Tsai, Y.-T. Liou, Y.-Y. Lin, and M.-H. Yang, "Adversarial learning for semi-supervised semantic segmentation," 2018, *arXiv:1802.07934*.

[38] M. S. Ibrahim, A. Vahdat, M. Ranjbar, and W. G. Macready, "Semi-supervised semantic image segmentation with self-correcting networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2020, pp. 12715–12725.

[39] X. Chang, P. Ren, P. Xu, Z. Li, X. Chen, and A. Hauptmann, "A comprehensive survey of scene graphs: Generation and application," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 1, pp. 1–26, Jan. 2021.

[40] L. Zhang et al., "TN-ZSTAD: Transferable network for zero-shot temporal activity detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 3, pp. 3848–3861, Mar. 2022.

[41] M. Li, P. Huang, X. Chang, J. Hu, Y. Yang, and A. Hauptmann, "Video pivoting unsupervised multi-modal machine translation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 3, pp. 3918–3932, Mar. 2022.

[42] C. Yan et al., "ZeroNAS: Differentiable generative adversarial networks search for zero-shot learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 12, pp. 9733–9740, Dec. 2022.

[43] Y. Li, M. Paluri, J. M. Rehg, and P. Dollár, "Unsupervised learning of edges," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1619–1627.

[44] Y. Ouali, C. Hudelot, and M. Tami, "Semi-supervised semantic segmentation with cross-consistency training," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 12671–12681.

[45] Z. Ke, D. Wang, Q. Yan, J. Ren, and R. Lau, "Dual student: Breaking the limits of the teacher in semi-supervised learning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6727–6735.

[46] X. Chen, Y. Yuan, G. Zeng, and J. Wang, "Semi-supervised semantic segmentation with cross pseudo supervision," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 2613–2622.

[47] A. Ghosh, H. Kumar, and P. Sastry, "Robust loss functions under label noise for deep neural networks," in *Proc. AAAI Conf. Artif. Intell.*, vol. 31, no. 1, 2017, pp. 1–7.

[48] A. Ghosh, N. Manwani, and P. S. Sastry, "Making risk minimization tolerant to label noise," *Neurocomputing*, vol. 160, pp. 93–107, Jul. 2015.

[49] T.-Y. Lin et al., "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2014, pp. 740–755.

[50] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 8026–8037.

[51] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[52] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid, "EpicFlow: Edge-preserving interpolation of correspondences for optical flow," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1164–1172.

[53] F. Galasso, N. S. Nagaraja, T. J. Cárdenas, T. Brox, and B. Schiele, "A unified video segmentation benchmark: Annotation, metrics and analysis," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 3527–3534.

[54] A. Prest, C. Leistner, J. Civera, C. Schmid, and V. Ferrari, "Learning object class detectors from weakly annotated video," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3282–3289.

[55] X. Ren and L. Bo, "Discriminatively trained sparse code gradients for contour detection," in *Proc. 25th Int. Conf. Neural Inf. Process. Syst.*, vol. 1, 2012, pp. 584–592.

[56] P. Isola, D. Zoran, D. Krishnan, and E. H. Adelson, "Crisp boundary detection using pointwise mutual information," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2014, pp. 799–814.

[57] S. Hallman and C. C. Fowlkes, "Oriented edge forests for boundary detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1732–1740.

Yunfan Ye received the bachelor's degree from Xiamen University, Xiamen, China, in 2017, and the master's degree from the Stevens Institute of Technology, Hoboken, NJ, USA, in 2019. He is currently pursuing the Ph.D. degree with the National University of Defense Technology, Changsha, China.

His research interests include edge detection, computer vision, graphics, and their applications.

Renjiao Yi received the bachelor's degree in computer science from the National University of Defense Technology, Changsha, China, in 2013, and the Ph.D. degree from Simon Fraser University, Burnaby, BC, Canada, in 2019.

She is currently an Associate Professor with the National University of Defense Technology. Her research interests include 3-D vision, inverse rendering, and their augmented reality applications.

Zhiping Cai (Member, IEEE) received the bachelor's, master's, and Ph.D. degrees (Hons.) in computer science and technology from the National University of Defense Technology (NUDT), Changsha, China, in July 1996, April 2002, and December 2005, respectively.

He is currently a Full Professor with the Networking Engineering Department, College of Computer, NUDT. His current research interests include big data, network security, and network virtualization.

Kai Xu (Senior Member, IEEE) received the Ph.D. degree in computer science from the National University of Defense Technology (NUDT), Changsha, China, in 2011.

From 2008 to 2010, he worked as a Visiting Ph.D. degree with the GrUVi Laboratory, Simon Fraser University, Burnaby, BC, Canada. He is currently a Professor with the School of Computer Science, NDUT. He is also an Adjunct Professor with Simon Fraser University. His current research interests include data-driven shape analysis and modeling, and 3-D vision and robot perception and navigation.