Joint Topology-preserving and Feature-refinement Network for Curvilinear Structure Segmentation

Mingfei Cheng[1*] Kaili Zhao[1*] Xuhong Guo[2] Yajing Xu[1] Jun Guo[1]
[1]School of Artificial Intelligence, Beijing University of Posts and Telecommunications.
[2]DOCOMO Beijing Communications Laboratories Co., Ltd.

Abstract

Curvilinear structure segmentation (CSS) is under semantic segmentation, whose applications include crack detection, aerial road extraction, and biomedical image segmentation. In general, geometric topology and pixel-wise features are two critical aspects of CSS. However, most semantic segmentation methods only focus on enhancing feature representations while existing CSS techniques emphasize preserving topology alone. In this paper, we present a Joint Topology-preserving and Feature-refinement Network (JTFN) that jointly models global topology and refined features based on an iterative feedback learning strategy. Specifically, we explore the structure of objects to help preserve corresponding topologies of predicted masks, thus design a reciprocative two-stream module for CSS and boundary detection. In addition, we introduce such topology-aware predictions as feedback guidance that refines attentive features by supplementing and enhancing saliencies. To the best of our knowledge, this is the first work that jointly addresses topology preserving and feature refinement for CSS. We evaluate JTFN on four datasets of diverse applications: Crack500, CrackTree200, Roads, and DRIVE. Results show that JTFN performs best in comparison with alternative methods. Code is available.[1]

1. Introduction

Curvilinear Structure Segmentation (CSS) [19, 37] is to segment binary masks of curvilinear objects such as concrete cracks, aerial roadmaps, blood vessel, and neuron boundary, *etc*. An accurate CSS in computer vision can help automatically detect concrete cracks captured by unmanned aerial vehicle [53], extract road networks from aerial images [4,38], assist doctors to recognize lesions from medical images [48]. While significant progress has been made in CSS and semantic segmentation communities, at least two critical problems remain: topology preserving and feature refinement. Like human's labeling process, topology preserving seeks to preserve overall geometric connectivity; feature-refinement aims to render details.

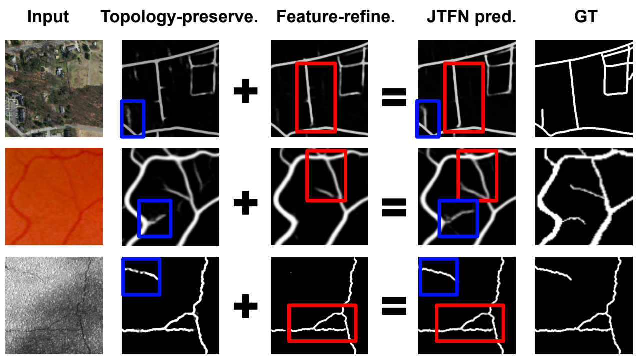


Figure 1. Effects of Joint Topology-preserving and Feature-refinement Network (JTFN) on road network extraction, blood vessels segmentation, and crack detection. JTFN can preserve global connectivity and render details simultaneously. Red and blue rectangles mark effect regions of topology-preserving and feature refinement separately.

Typically, general CSS methods [14, 26, 39] tend to emphasize topology preserving. For instance, [14] explicitly defines connected components and holes as local topology, but only optimizes limited pixels for topology, thus leads to the method is sensitive to complex backgrounds. Another work [26] implicitly assumes features produced by pre-trained VGG have natural topology, but such strong assumptions cause limited improvements. Hence, a robust, loose assumption, and global topology-preserving CSS method is demanding.

In contrast, general semantic segmentation works mostly focus on feature representations, which integrate additional contextual or spatial information [7, 8, 15, 42, 44, 59]. For context-aware methods, dilated convolutions or pyramid designs are applied to enhance receptive fields [7, 8, 44], and non-local modules are used to raise self-attentions [15, 42, 59]. For spatial-aware methods, multi-scale or skip-connected architectures are introduced to aggregate different features to diversify representations [20, 21]. Though these methods improve representations relatively, refining features for details is still problematic for CSS. Correspond-

*These authors contributed equally.
[1]https://github.com/zkl20061823

ingly, [9, 26, 33, 40, 41] directly bring predicted masks as additional input channels and update the predictions iteratively. To sum up, to refine features, it is inspiring to study how to utilize both context- and spatial-aware predicted masks as feedback guidance.

To address the aforementioned problems, we propose a novel framework, Joint Topology-preserving and Feature-refinement Network (JTFN), to tackle two problems with one stone. JTFN aims to jointly model global topology and refined features based on an iterative feedback learning strategy. Specifically, JTFN explores object boundary as global-topology regularizations of predicted masks, and designs a *Feature Interactive Module* (FIM) to reciprocate features between CSS and boundary detection. The learned topology-aware predictions naturally facilitate both attentive context- and spatial-aware features. To utilize such attentive masks as feedback guidances during feature refinement, JTFN proposes a *Gated Attentive Unit* (GAU) to supplement and enhance saliencies. Furthermore, JTFN embeds FIM and GAU in a feedback loop from feature learning to prediction updating, thus not only refines features but also rectifies final predictions accordingly.

Fig. 1 illustrates examples of JTFN's effects on diverse CSS applications, and obtains the results from **Base-FIM**, **Base-GAU**, and JTFN respectively (will be illustrated in Sec. 4). From the results, we can see that JTFN can preserve global topology and refined features simultaneously. Comprehensive experiments on Crack500 [55], CrackTree200 [60], Roads [25], and DRIVE [24] validate the effectiveness of JTFN in comparison with ablations and alternatives.

2. Related Work

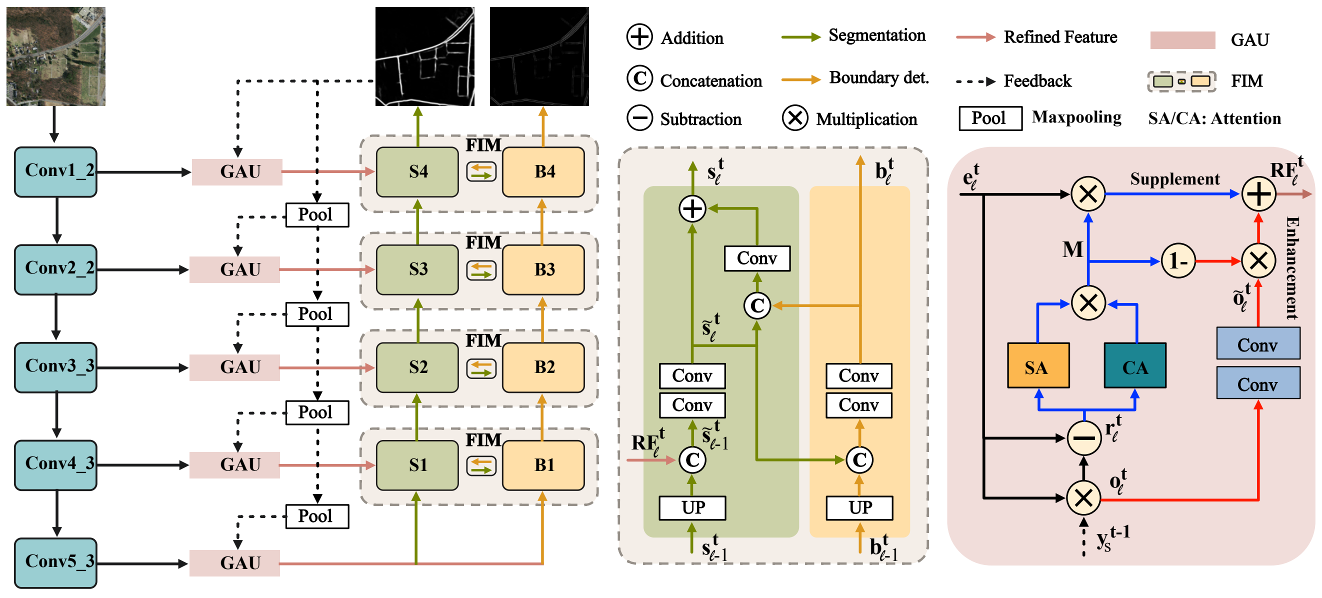
Semantic Segmentation: Given an input image, the goal of semantic segmentation (SS) is to assign pixel-level labels as 0 or 1, thus SS belongs to dense prediction. Fully Convolutional Network (FCN) [23] is a pioneer work that helps reduce computation cost and preserve spatial information. Broad SS methods follow the FCN stream and achieve inspiring results. Here, we categorize the methods into 2 types: context-aware and spatial-aware.

Context-aware methods encourage to perceive of contextual information from larger fields (*e.g.*, dilated convolution), neighboring pixels (*e.g.*, non-local modules), or hierarchical layers (*e.g.* pyramid). [7, 8, 44, 54] apply dilated convolution to enlarge receptive fields with limited extra parameters. Feature pyramid-based methods [22, 57, 58] aggregate features of different scale by utilizing parallel spatial pooling. Besides, [15, 42, 59] exploit non-local module to model global context by building long-range dependencies among neighboring pixels. However, the above methods lose some spatial details due to down-sampling operations used. To mitigate the loss, spatial-aware methods mostly are built on encoder-decoder architectures [2] which recover spatial information layer by layer in decoder module. For CSS, details matter, and more precise spatial information is necessary. Thus skip connections are proposed to directly concatenate spatial features in encoder to the ones in decoder [31, 56]. In addition, [20, 21] introduce richer spatial features in skip connections by aggregating multiple scales features. Instead, we introduce feature refinement in skip connection module, and utilize both context and spatial aware predictions as attentive guidance to select saliencies.

Refined Segmentation: Refined segmentation aims to refine predictions when initial "coarse predictions" are obtained. We broadly group refined segmentation methods into three categories: offline postprocessing-based, cascade refinement-based, and feedback loop-based.

Offline postprocessing-based methods [7, 13] are sensitive to various scenes since thresholds are manually set. Cascade refinement-based methods design additional modules that sequentially update predictions [17,30,39,51]. For instance, [30, 51] propose a residual refinement module to learn the difference between "coarse predictions" and the ground truth. Wang *et al.* [39] add another light encoder-decoder network to generate finer patches that can replace the corresponding patches in "coarse predictions". However, these methods require extra modules, consequently, bring more parameters. To address this problem, feedback loop-based method (*e.g.* DRU [41]) builds a feedback scheme by iteratively bringing "coarse predictions" as another channel of input image [9,26,33,40]. Besides, instead of embedding "coarse predictions" to inputs, CPD [49] utilizes the predictions to guide deep layer features in encoders. But CPD still adds extra parameters due to separate cascaded decoders are exploited to reconstruct refined features. By contrast, first, JTFN not only refines predictions but also refines features; secondly, JTFN refines features from shallow to deep layers; third, JTFN treats "coarse predictions" as context- and spatial-aware guidance.

Curvilinear Structure Segmentation: The two critical aspects of Curvilinear Structure Segmentation (CSS) are topology preserving and feature refinement [14, 26, 33, 38, 39, 53]. But existing general CSS methods center on topology preserving. TopoNet [14] learns local topology by optimizing limited critical points, thus is sensitive to various scenes. [26] assumes that pre-trained VGG [34] can produce natural topology of objects, but such assumption causes limited improvements. Accordingly, the sub-tasks of CSS, such as crack detection [53], road extraction [4,38], and biomedical image segmentation [31,33,39], also only consider one aspect alone. For instance, [4] involves extra annotated data to keep connectivities of road network. FPHBN [53] diversifies receptive field by aggregating pyramid features for crack detection. Instead, our JTFN jointly addresses topology preserving and feature refinement.



(a) JTFN architecture (b) FIM (c) GAU

Figure 2. Joint Topology-preserving and Feature-refinement Network (JTFN). (a) Architecture of JTFN. Our model consists of three phases: five encoding layers, reciprocative decoders (FIM) from semantic segmentation (**S**) to boundary detection (**B**), and skip connections embedded with feature refinement (GAU). (b) Feature Interactive Module (FIM). (c) Gated Attentive Unit (GAU).

3. Joint Topology-preserving and Feature-refinement Network (JTFN)

In this section, we first discuss the JTFN architecture. Then we illustrate topology preserving and feature refinement separately. Finally, we explain the iterative training and inference process for JTFN.

3.1. JTFN Architecture

Fig. 2(a) illustrates the JTFN architecture. The architecture is built on UNet [31]. For encoders, we truncate the first five layers of VGG16 [34] and remove the last three fully-connected layers and the last pooling layer of VGG-16. Thus we obtained five hierarchical encoding features that are in charge of gaining knowledge. We name these five encoding layers as same as VGG-16, like Conv1_2 in Fig. 2(a). Four decoding layers are used for reconstructing spatial and semantic features, thus comprehensive priors are important for extracting meaningful info. Here, we introduce object connectivity as topology priors and design a reciprocative module (*i.e.*, FIM) to exchange features from semantic segmentation (**S**) to boundary detection (**B**). **S** and **B** are two homogeneous streams and each **S** or **B** contains two sets of "conv + bn + relu" and one upsampling. Additionally, to mitigate the spatial loss of encoders due to stacks of pooling operations, we add skip connections between each corresponding encoding and decoding layers for feature refinement. In this step, we embed a Gated Attentive Unit (GAU) in each skip connection, to introduce context- and spatial-aware predictions as feedback guidance that selects saliencies for better refinement. Consequently, this iterative feedback learning strategy not only refines features but also helps update predictions simultaneously.

3.2. Feature Interactive Module (FIM)

[16, 43, 52] support that topology and boundary connectivity are related, and demonstrate that topology connectivity can help identify boundary. Thus we utilize object boundary to represents the topology of curvilinear structures. Technically, segmentation masks are highly associated with corresponding object boundaries. In addition, boundary map annotations can be automatically obtained by performing Canny operator [6] on existing segmentation labels. Hence, we exploit features from boundary detection as supervision of topology preserving for segmentation. Inspired by this, we build a feature interactive module (FIM) to model feature interchange between segmentation and boundary detection. It is worth noting that we only utilize FIM and boundary detection in decoders because decoding layers emphasize semantic features (such as topology) more compared to encoding layers [1]. Fig. 2(b) shows the brief architecture of FIM. The left blue part corresponds to the learned features for semantic segmentation (**S**) and the yellow one is for boundary detection (**B**). Here we denote lower-case ${s_\ell}^t$ and ${b_\ell}^t$ as the learned features in **S** and **B** separately, where ℓ means the layer number and t is for # learning iterations. Thus FIM is designed as follows:

$$\mathbf{s}_\ell^t, \mathbf{b}_\ell^t = F_\ell(\mathbf{s}_{\ell-1}^t, \mathbf{b}_{\ell-1}^t, \mathrm{RF}_\ell^t; \theta_F^t), \tag{1}$$

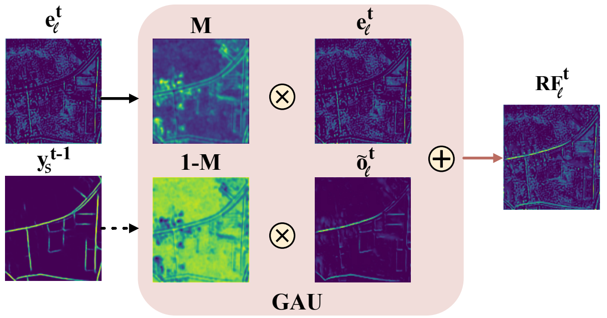


Figure 3. Feature refinement in GAU. Under feedback $\mathbf{y}_s^{t-1}$'s guidance, the learned encoding features $\mathbf{e}_\ell^t$ is refined as RF_ℓ^t.

where RF_ℓ^t is a refined feature from GAU, and θ_F represents the parameters of FIM. The $\mathbf{S}$ branch learns $\mathbf{s}_\ell^t$ through three parts: concatenating refined features RF_ℓ^t and the previous S's feature $\mathbf{s}_{\ell-1}^t$, updating features by two convs (denoted as $\tilde{\mathbf{s}}_{\ell-1}^t$), and learning attentive features with a residual attention which incorporates updated $\mathbf{B}$ features $\mathbf{b}_\ell{}^t$. As an auxiliary stream, the $\mathbf{B}$ branch simply concatenates the updated features $\tilde{\mathbf{s}}_\ell^t$ and the previous $\mathbf{B}$'s feature $\mathbf{b}_{\ell-1}^t$. Here $\tilde{\mathbf{s}}_\ell^t$ is obtained by taking two convs after $\tilde{\mathbf{s}}_{\ell-1}^t$. With the FIM designs, features are reciprocated from segmentation to boundary detection, thus global topology is preserved.

3.3. Gated Attentive Unit (GAU)

The predicted masks naturally show context- and spatial-aware attentions. Instead of directly including the predictions to inputs [9, 26, 33, 40], JTFN back-feeds predictions as feature-refinement guidance that helps enhance feature representations in skip connections. In addition, this feedback loop from predictions to feature refinement encourages refining features and iteratively updating predictions at the same time. Specifically, we present a Gated Attentive Unit (GAU) that both supplements residual and enhances existing saliencies, thus attentive features are obtained.

Fig. 2(c) illustrates the GAU architecture. The unit consists of two functions: *supplement* and *enhancement*. The inputs of GAU are encoding features $\mathbf{e}_\ell^t$ and previous predictions $\mathbf{y}_s^{t-1}$. With GAU, $\mathbf{e}_\ell^t$ will be enhanced as refined features RF_ℓ^t under the $\mathbf{y}_s^{t-1}$'s guidance, and then be skip-connected to decoders for better reconstruction. We formulate GAU as below:

$$\begin{aligned}\mathrm{RF}_\ell^t &= \mathrm{sp}_\ell^t + \mathrm{eh}_\ell^t\\ &= \mathbf{M} \times \mathbf{e}_\ell^t + (1-\mathbf{M}) \times \tilde{\mathbf{o}}_\ell^t,\end{aligned} \tag{2}$$

where sp_ℓ^t means *supplement* and eh_ℓ^t represents *enhancement*; $\mathbf{M}$ is denoted as the attentive-residual mask and $\times$ is for element-wise multiplication.

Specifically, *supplement* (the blue lines) aims to make up for residual attentive features that exist in the learned predicted masks but not or weakly exist in $\mathbf{e}_\ell^t$. We define *supplement* as follows:

$$\begin{aligned}\mathrm{sp}_\ell^t &= \mathbf{M} \times \mathbf{e}_\ell^t\\ \mathbf{M} &= \mathrm{sigmoid}[SA(\mathbf{r}_\ell^t) \times CA(\mathbf{r}_\ell^t)]\\ \mathbf{r}_\ell^t &= \mathbf{e}_\ell^t - \mathbf{o}_\ell^t\\ \mathbf{o}_\ell^t &= \mathbf{e}_\ell^t \times \mathbf{y}_s^{t-1}.\end{aligned} \tag{3}$$

Refer to Fig. 2(c), from left bottom to top, the detailed process is as follows: (1) obtain overlapping attentive features $\mathbf{o}_\ell^t$ by multiplying $\mathbf{e}_\ell^t$ and $\mathbf{y}_s^{t-1}$, *i.e.*, existing attentive features; (2) get residual attentive features $\mathbf{r}_\ell^t$ by finding difference between current learned features $\mathbf{e}_\ell^t$ and the overlapping attentive features $\mathbf{o}_l^t$; (3) generate spatial- and channel-wise masks $\mathbf{M}$ based on residual attentive features $\mathbf{r}_\ell^t$; (4) generate supplement features sp_ℓ^t by masking input features $\mathbf{e}_\ell^t$ with $\mathbf{M}$. Here, the spatial-wise attention SA and channel-wise attention CA share the same structures as [47] used for better feature representations.

By contrast, *enhancement* (the red lines) seeks to enhance existing attentive features that exist in both predicted masks and current $\mathbf{e}_\ell^t$. We formulate its process as below:

$$\mathrm{eh}_\ell^t = (1-\mathbf{M}) \times \tilde{\mathbf{o}}_\ell^t(\mathbf{y}_s^{t-1}). \tag{4}$$

Here $\tilde{\mathbf{o}}_\ell^t(\mathbf{y}_s^{t-1})$ is obtained by two convs after overlapping attentive features $\mathbf{o}_\ell^t$. That's to say, *enhancement* strengthens existing attentive features. Fig. 3 shows feature learning of GAU. The features are obtained from the layer ($\ell = 2, t = 3$) that is trained for road network extraction. From the results, we can see residual attentive features (*i.e.* vertical lines) are maintained or supplemented, and existing attentive features (*i.e.* two horizontal lines) are enhanced.

3.4. Iterative Training & Inference

Training: In training, JTFN adopts an iterative feedback learning strategy, which traverse between accumulating a total loss from $\mathbf{T}$ iterations of forward passes, and then performing backpropagation. Specifically, a batch of CSS images $\mathbf{I} \in \mathbf{R}^{C\times H\times W}$ go through T-iterations updates. Specifically, for iteration t, JTFN first learns an encoder module. Then JTFN models two reciprocative decoders for semantic segmentation and boundary detection (denoted as $\mathbf{S}$_decoder and $\mathbf{B}$_decoder separately), thus generates segmentation and boundary predictions $\mathbf{y}_s^t$ and $\mathbf{y}_b^t$:

$$\mathbf{y}_s^t, \mathbf{y}_b^t = \mathbf{F}_{\text{JTFN}}(\mathbf{I}, \mathbf{y}_s^{t-1}; \theta), \tag{5}$$

where θ is the overall parameters of JTFN. At last, JTFN brings a segmentation prediction $\mathbf{y}_s^t$ as a feedback input of GAU for feature refinement. In addition, we utilize binary cross-entropy loss (BCE) to minimize the divergence be-

Algorithm 1 Iterative Training

Input: Input image $\mathbf{I} \in \mathbf{R}^{C\times H\times W}$, segmentation annotation $\mathbf{G}_s \in \mathbf{R}^{1\times H\times W}$, boundary annotation $\mathbf{G}_b \in \mathbf{R}^{1\times H\times W}$, initialized prediction $\mathbf{y}_s^0 = 1$, and hyper-parameters α and λ.
Output: Predicted segmentation and boundary masks: y_s^t and y_b^t.

```
1: for $itr = 0, \ldots, \#epoch$ do
2:     //Feed-forward:
3:     for $t = 1, \ldots, \mathbf{T}$ do
4:         $\mathbf{e}_\ell^t = \text{encoder}(\mathbf{I})$
5:         $\mathbf{y}_s^t = \mathbf{S}\_\text{decoder}(\mathbf{s}_{\ell-1}^t, \text{RF}_\ell^t(\mathbf{e}_\ell^t, \mathbf{y}_s^{t-1}), \mathbf{b}_\ell^t)$
6:         $\mathbf{y}_b^t = \mathbf{B}\_\text{decoder}(\mathbf{b}_{\ell-1}^t, \tilde{\mathbf{s}}_\ell^t)$
7:         $L_t = L_{bce}(\mathbf{y}_s^t, \mathbf{G}_s) + \lambda L_{bce}(\mathbf{y}_b^t, \mathbf{G}_b)$
8:     end for
9:     Minimize $L = \sum_t \alpha_t L_t$
10:    Backward and Update JTFN.
11: end for
```

tween predictions and ground-truth $(\mathbf{G}_s, \mathbf{G}_b)$:

$$\begin{aligned} L_t &= L_{bce}(\mathbf{y}_s^t, \mathbf{G}_s) + \lambda_b L_{bce}(\mathbf{y}_b^t, \mathbf{G}_b) \\ &= -\sum_i^{H\times W} \{[\mathbf{G}_s^i \log(\mathbf{y}_s^{i,t}) + (1-\mathbf{G}_s^i)\log(1-\mathbf{y}_s^{i,t})] \\ &\quad + [\mathbf{G}_b^i \log(\mathbf{y}_b^{i,t}) + (1-\mathbf{G}_b^i)\log(1-\mathbf{y}_b^{i,t})]\}, \end{aligned} \tag{6}$$

We set the balancing coefficient λ_b as 1 in experiments. After T-rounds forward computation, total loss can be obtained as:

$$L = \sum_{t=1}^{T} \alpha_t L_t, \tag{7}$$

where α_t is the loss weight and here we set weights equally in each iteration. See Algo. 1 for the detailed feedforward computation and backward update.

Inference: In prediction, we obtain predictions only from **S**, the semantic segmentation head. To refine predictions, we iteratively perform T feedforward computations and get the final predicted masks. Here the threshold for binarizing predictions is set as 0.5.

4. Experiments

4.1. Datasets

To validate the effectiveness of CSS on various applications, we perform JTFN on below four datasets: Crack500 [55] and CrackTree200 [60] for crack detection, DRIVE dataset [24] for blood vessel segmentation, Massachusetts Roads dataset [25] for aerial road network extraction.

Cracks: Crack500 [55] and CrackTree200 [60] both contain concrete/pavement crack images with various shapes and cluttered backgrounds. Crack500 is the existing largest public dataset that consists of 1896 training images and 1124 test images. The widths and shapes of cracks in Crack500 are varying from a large range that makes crack detection challenging. CrackTree200 has 206 pavement images that are only annotated with one-pixel width masks. Hence, CrackTree200 is feasible for topology-aware evaluation. Following [26]'s settings, we dilate the ground truth masks by 4-pixels in evaluation. Since public dataset partitions are not provided in [60], in experiments, we use 164 images for training and the rest for testing.

DRIVE: DRIVE dataset [24] is designed for blood vessel segmentation of medical images. The dataset only contains 40 retina images. The limited number of images can evaluate our model's performance on the small dataset. Following [39], we set 20 images for training and 20 for testing.

Roads: Massachusetts Roads dataset [25] is the largest available dataset for road network extraction whose images are taken from airborne craft. The dataset contains different roads, like small paths, highways, *etc*., which provide different scenarios for evaluation. The dataset has 1108 training and 49 test images.

Table 1. Ablations of **Boundary-only (BO), FIM, and GAU**.

Architecture	+BO	+FIM	+GAU	F1	Quality
Base				78.86	77.56
Base-C				80.44	79.60
Base-BO	✓			80.27	78.79
Base-FIM	✓	✓		82.07	80.75
Base-GAU			✓	81.84	81.55
JTFN	✓	✓	✓	84.19	82.96

4.2. Implementation Details

Evaluation metrics. To evaluate the topology and pixel-wise accuracy in CSS, we choose two types of metrics correspondingly. For pixel-wise evaluation, we choose *F1 score*, *Precision*, and *Recall* that have widely been used in existing semantic segmentation methods [14, 26, 32]. *Precision* and *Recall* are computed by comparing predicted and ground-truth masks in pixel-level and F1 coordinates the agreement between *Precision* and *Recall*: $F_1 = 2\frac{Precision \times Recall}{Precision + Recall}$. Compared to pixel-wise metrics are sensitive to small changes, topology-based is relatively robust. For evaluating topology, following [46], we adopt *Correctness*, *Completeness* and *Quality*, which measure the similarity between predicted skeletons and ground truth within a threshold. In our experiments, the threshold is set to 2.

Training details. Our proposed network is built with PyTorch [29] and trained on a single NVIDIA RTX 3090. For data augmentation, we use horizontal flipping, random cropping, and random rotation with 90°, 180° and 270°. Our model is optimized by Adam [18] with an initial learning rate of 10^{-3} and a weight decay of 5×10^{-4}. During training, all training samples are cropped to 256×256, and our network is trained with a mini-batch size 2. In addition, we produce GT boundary by fusing boundaries from Canny

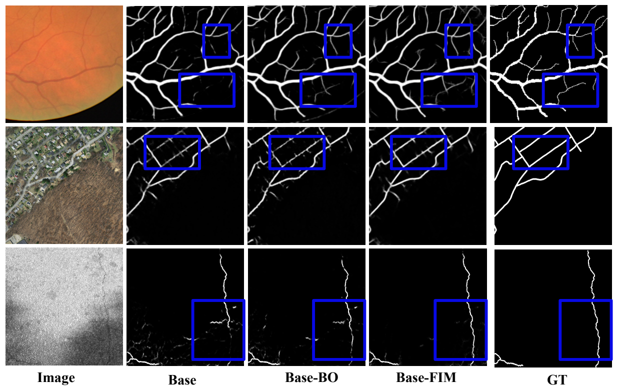


Figure 4. Examples of ablations w/ boundary or FIM. Three ablation models are performed: **Base**, **Base-BO**, and **Base-FIM**. The blue rectangle represents effect regions of topology-preserving.

operator [6] and a spatial gradient deriving method.

4.3. Ablation Study

To analyze JTFN, we performed extensive ablations with comprehensive metrics: pixel-wise based F1 and topology-based Quality. We show quantitative results on CrackTree200 [60] and qualitative results on DRIVE [24], Roads [25], and CrackTree200. Here, our baseline model (denoted as **Base**) is obtained from JTFN excluding boundary detection, FIM, and GAU, *i.e.* a UNet-like structure.

Boundary-only (BO): To demonstrate the objects' boundary is effective for preserving the topology of curvilinear structures, we only include a parallel decoder of boundary detection to original **Base**, and denote the model as **Base-BO**. Observing results in Table 1, we see **Base-BO** obtains about 1.41 points higher than **Base** in F1 and achieves 1.23 points higher in Quality. We can infer that including boundary to base model can help improve accuracy. The second and third columns of Fig. 4 show the segmentation examples of **Base** and **Base-BO** models individually. When zooming in the regions, we can see **Base-BO** can learn new connectivities in various curvilinear structures, like the missing structure in blood vessels. Thus we believe including boundary to base segmentation model can help preserve topology of curvilinear structures.

FIM: To fully take advantage of boundary information when reconstructing semantic segmentation, we propose Feature Interactive Module (FIM) to interchange features between boundary detection and semantic segmentation (*i.e.* CSS). To analyze the effectiveness of FIM, here we include FIM to each pair of (segmentation, boundary detection) decoders, and denote this model as **Base-FIM**. As the results shown in Table 1, compared to **Base-BO**, **Base-FIM** performs 1.8 points higher in F1 and 1.95 points higher in Quality. In addition, Comparing third and fourth columns of Fig. 4, **Base-FIM** removes noisy connectivity of roads and cracks, and obtains more complete topologies. These results prove that our proposed FIM consistently performs

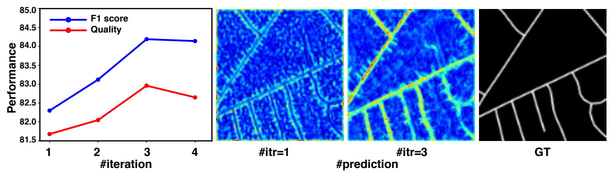


Figure 5. Iterative leaning of JTFN. JTFN achieves best results at #itr 3. Topology and details are gradually complemented.

best by gradually including boundary information and interactive modules for topology enhancement.

GAU: To mitigate the spatial loss and enhance semantic information from encoders to decoders, we design a GAU embedded in skip connections, which module helps learn attentive features under the guidance of predictions. To validate the performance of GAU, we include GAU to **Base** without adding **Boundary** or **FIM**. Observing results of **Base** and **Base-GAU** in Table 1, we can see **Base-GAU** improve the results by a large margin with 2.98 points in F1 and 3.99 points in Quality. In addition, we compare a new baseline with concatenation operation **C**, instead of GAU module. We can see **Base-GAU** outperforms **Base-C** by 1.4 and 1.95 points in F1 and Quality. The results show that GAU is effective in CSS segmentation.

Including both FIM and GAU: We have validated the effectiveness of FIM and GAU alone. Here, we will demonstrate the combination of these two modules. Instead of incrementally adding them together, our proposed JTFN combines these two modules in an iterative learning strategy. From the results in Table 1, JTFN obtains 5.33 points gains in F1 (84.19 *vs.* 78.86) and 5.4 points gains in Quality (82.96 *vs.* 77.56), compared to **Base** model. Additionally, compared with the separate modules, JTFN achieves around 2 points increase for both F1 and Quality compared to **Base-FIM**, and performs (2.35, 1.41) points better of (F1, Quality) than **Base-GAU**. We can see JTFN consistently performs best among extensive ablations. We can infer that topology-preserving (FIM) and feature refinement (GAU) mutually benefit during joint learning.

Number of iterations: Since JTFN goes through FIM and GAU in an iterative learning strategy. Here we study how many iterations are enough to obtain a speed-accuracy tradeoff. We train our JTFN up to 4 iterations when the accuracy starts to decrease. Fig. 5 illustrates experimental results of JTFN on CrackTree200 by comprehensive topology and pixel-wise metrics (*i.e.* F1 and Quality). For #itr 1, only with FIM, JTFN is guided by an initialized mask and achieves satisfactory results: (82.30, 81.68) of (F1, Quality). With the updates of feedback masks, for #itr 3, JTFN significantly surpasses #itr 1 by a large margin (84.19 *vs.* 82.30) of F1 and (82.96, *vs.* 81.68) of Quality. JTFN starts to decrease performance at #itr 4, thus, we select #itr as 3 in below experiments. It is worth noting that the results from #itr 2 to 3, even 4 are substantially better than #itr 1 with

Table 2. Comparisons with alternative CSS methods across segmentation-based and topology-based metrics. Results of alternatives were obtained by reproducing codes provided by authors. Red numbers indicate best results, blue numbers are second best.

Method	CrackTree200			Crack500			DRIVE			Roads		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
UNet [31]	79.16	78.95	78.42	62.22	68.85	61.83	82.74	80.59	81.41	62.55	52.63	55.70
VGG-UNet [26]	83.49	80.43	81.84	58.18	60.26	51.79	81.17	82.05	81.25	64.79	57.47	59.65
TopoNet [14]	81.85	77.80	79.03	66.81	62.68	60.06	82.94	80.29	81.36	62.25	55.83	57.36
DRU [41]	84.80	77.46	80.49	61.94	71.43	62.82	84.36	80.82	82.30	60.62	56.96	57.42
JTFN (ours)	85.87	82.58	84.19	68.81	69.06	65.76	82.71	83.40	82.81	65.14	59.04	60.65
Method	**Correct.**	**Complete.**	**Quality.**	**Correct.**	**Complete.**	**Quality**	**Correct.**	**Complete.**	**Quality**	**Correct.**	**Complete.**	**Quality**
UNet [31]	82.05	85.93	76.15	31.66	34.98	19.56	55.60	46.29	33.82	62.89	58.06	48.56
VGG-UNet [26]	86.95	85.36	80.08	25.45	32.75	15.67	53.04	33.36	31.25	67.23	61.34	53.98
TopoNet [14]	85.50	83.92	77.36	30.02	37.26	19.83	55.67	46.95	34.22	62.46	61.47	50.77
DRU [41]	87.81	84.02	79.47	30.51	36.02	19.78	56.03	48.86	35.36	62.04	60.42	49.83
JTFN (ours)	88.30	87.42	82.96	36.72	39.26	24.12	57.09	49.28	36.02	68.65	63.37	56.05

Table 3. Comparisons with SOA alternatives on CrackTree200.

Method	Correct.	Complete.	Quality
CrackTree [60]	79.0	92.0	73.92
Reg-AC [35]	10.7	92.83	10.61
VGG-UNet [26]	85.50	83.92	77.36
TopoNet [14]	86.95	85.36	80.08
JTFN (ours)	88.3	87.42	82.96

Table 4. Comparisons with SOA alternatives on Roads.

Method	Correct.	Complete.	Quality
Reg-AC [35]	25.37	34.78	17.19
RoadTracer [3]	43.50	51.30	30.80
MSP-Tracer [45]	48.80	55.20	34.30
JTFN (ours)	68.65	63.37	56.05

the same computation cost. This indicates that an iterative learning strategy is beneficial for optimizing predictions. From the predictions in Fig. 5, we can see the confidence becomes higher with iterative learning increasing.

4.4. Comparisons

To validate the effectiveness of our JTFN, we compare with alternatives in general CSS methods and each branch task separately, including crack detection, road network extraction, and blood vessel segmentation.

Comparative methods: For general CSS methods' comparisons, we select topology-based methods and the closest semantic-based method (*i.e.* DRU [41]) to JTFN. Correspondingly, VGG-UNet [26] and TopoNet [14] both are alternative topology-preserving methods. Closest to JTFN, the state-of-the-art DRU [41] also adopts iterative learning strategy and exploits feature refinement module, *i.e.*, **refined segmentation** as Sec.2 discussed. However, mimicked RNN's idea, DRU simply embeds predictions to input channel and only refines features in the last layer of encoder without any attentive guidance. In addition, since UNet [31] is a commonly used and effective structure in CSS, here, we include UNet as a baseline. In experiments, thanks to the codes provided by authors, we reproduce the results for fair comparisons. To further show the effectiveness of JTFN, we compare JTFN to state-of-the-art methods in each sub-tasks. These methods vary from hand-crafted features-based [5, 36, 50] to recent CNN based [24, 39]. All numbers of alternatives are obtained from original papers.

Comparisons with alternative CSS methods: Table 2 shows results of JTFN and alternatives on four datasets across six metrics, which evaluate both pixel-wise accuracy (F1, Precision, Recall) and topology preserving (Correctness, Completeness, and Quality). Compared to alternatives, JTFN achieves 22 best scores, 1 second best, and 1 third best score over all 24 scores. The highest increment among pixel-wise based metrics is around 5 points (*e.g.* from 60.62 to 65.14 for Precision of Roads); and the largest improvement on topology based metrics is about 6 points (*i.e.* from 30.51 to 36.72 for Correctness of Crack500). Thus we infer that JTFN obtains consistent improvements on both topology-preserving and pixel-wise refinement. By contrast, the alternative methods fail to improve consistently, for example, DRU mostly performs better than baseline UNet but sometimes even worse, such as Precision of Roads (*i.e.* 60.62 *vs.* 62.55), and Correctness of Crack500 (30.51 vs 31.66) *etc.* Compared to the closest method DRU, JTFN only performs less effectively on Recall of Crack500 (69.06 *vs.* 71.43) and Precision of DRIVE (82.71 *vs.* 84.36). To further understand JTFN, Fig. 6 shows segmentation examples of JTFN and the alternatives. From the 1st row to 4th row correspond to results for road extraction, one-pixel width crack detection, width-changing crack detection, and blood vessel segmentation. We can see that JTFN delineates curvilinear structure (*e.g.* roads or cracks) better compared with alternatives. Besides, JTFN can capture more details that appear near boundaries like blood vessels.

Comparisons with SOA alternatives: Tables 3-4 show comparisons between JTFN and SOA alternatives in each branch tasks. For crack detection, not many existing works focus on the field, and in most scenarios, topology-preserving works are main-stream, like Reg-AC [35] which is only for centerline detection. Besides, for road network

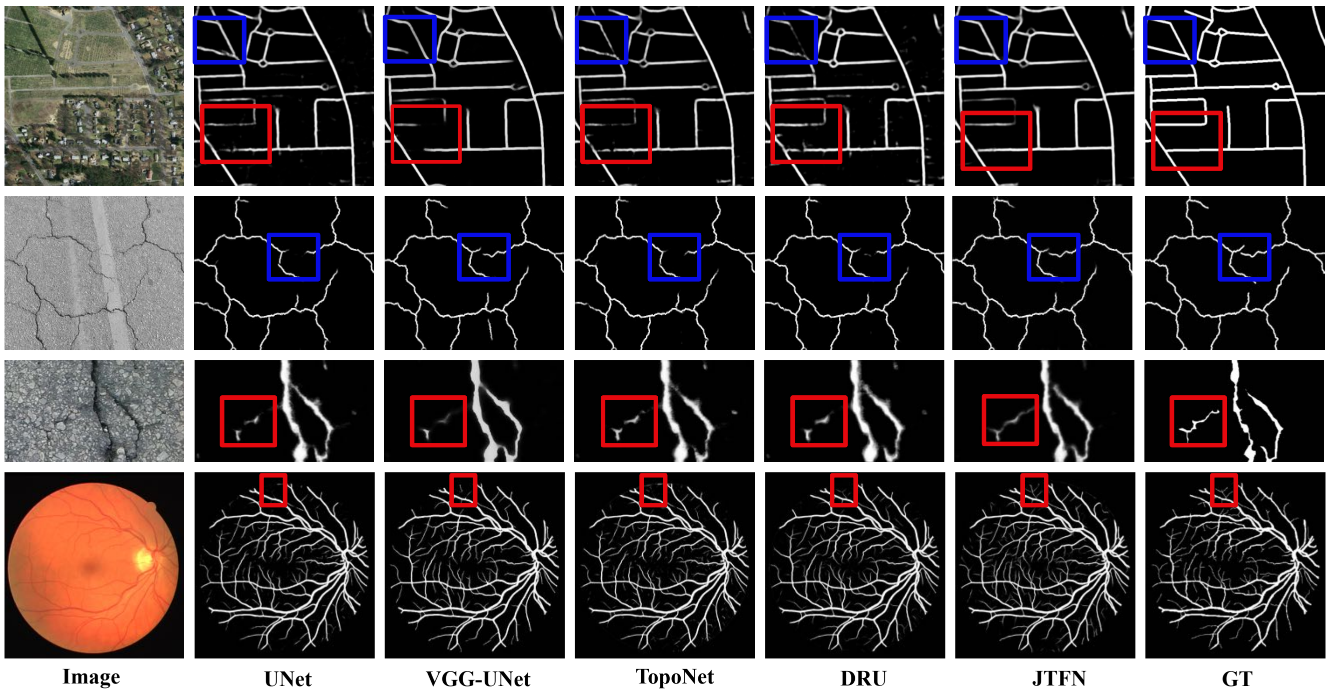


Figure 6. Examples of segmentation for JTFN and alternatives. We can conclude that topology-preserving (TP) encodes boundary connectivity to improve broken edges, and feature-refinement (FR) iteratively refines details such as curvilinear structure or thickness of edges. Blue rectangle represents the area we can see effects of TP and red rectangle is for FR. Zoom in for better visualization.

extraction, connectivity is critical, thus topology-preserving is also very important. Thus, according to the characteristic of these two tasks, topology-based metrics are selected for evaluation. From the results of CrackTree200 in Table 3, JTFN performs best over Correctness and Quality and less effective on completeness of cracks. Besides, JTFN achieves best across all the topology-based metrics on Roads dataset shown in Table 4. Overall, we see JTFN performs effectively in topology-aware applications. Not like Reg-AC which only has effects on topology, JTFN also can render details of segmentation. For blood vessel segmentation, details and topology are both critical aspects, thus pixel-wise accuracy and topology-aware metrics are applied on DRIVE dataset. Table 5 shows JTFN performs best and second best overall the metrics. From the above discussion, we conclude that JTFN achieved state-of-the-art results.

Table 5. Comparisons with SOA alternatives on DRIVE.

Method	F1	Correct.	Complete.	Quality
Wavelets [36]	76.18	49.18	21.34	17.82
SE [10]	65.84	37.94	16.03	12.59
CE-Net [12]	71.09	33.69	25.96	17.14
HED [50]	79.59	43.83	41.57	27.03
KBoost [5]	80.03	46.89	42.01	28.40
N^4Fields [11]	80.52	56.50	36.46	28.43
CRFs [28]	78.12	49.39	40.31	28.56
CS^2 [27]	81.63	56.62	46.26	34.12
DRIU [24]	82.21	47.34	47.25	31.37
PolicyNet [39]	83.53	57.68	46.39	34.32
JTFN (ours)	82.81	57.09	49.28	36.02

5. Conclusion

We have proposed Joint Topology-preserving and Feature-refinement Network (JTFN) for curvilinear structure segmentation. JTFN contains two critical modules: Feature Interactive Module (FIM) and Gated Attentive Unit (GAU), then combines these two modules via an iterative learning strategy. FIM seeks to address topology preserving by introducing boundary detection (**B**) as a geometric constraint of semantic segmentation (**S**), and reciprocates features between two tasks. Using such topology-aware predictions as feedback guidance, GAU refines attentive features by supplementing and enhancing saliencies. We provide extensive ablation studies to justify FIM and GAU. Comparisons with general CSS methods validate the effectiveness of whole JTFN on topology-preserving and segmentation accuracy. For each sub-tasks of CSS (*e.g.* blood vessel segmentation), JTFN consistently outperforms state-of-the-art alternatives. Potential extensions of JTFN include adaptively weighing loss for each iteration, thus refining learning process and refine predictions efficiently.

Acknowledgement. Research reported in this paper was supported by the Natural Science Foundation of China under grants 62076036 and 62076031.

References

[1] Md Amirul Islam, Mrigank Rochan, Neil DB Bruce, and Yang Wang. Gated feedback refinement network for dense image labeling. In *CVPR*, 2017.

[2] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *TPAMI*, 39, 2017.

[3] Favyen Bastani, Songtao He, Sofiane Abbar, Mohammad Alizadeh, Hari Balakrishnan, Sanjay Chawla, Sam Madden, and David DeWitt. Roadtracer: Automatic extraction of road networks from aerial images. In *CVPR*, 2018.

[4] Anil Batra, Suriya Singh, Guan Pang, Saikat Basu, CV Jawahar, and Manohar Paluri. Improved road connectivity by joint learning of orientation and segmentation. In *CVPR*, 2019.

[5] Carlos Becker, Roberto Rigamonti, Vincent Lepetit, and Pascal Fua. Supervised feature learning for curvilinear structure segmentation. In *MICCAI*, 2013.

[6] John Canny. A computational approach to edge detection. *TPAMI*, 1986.

[7] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 40, 2017.

[8] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018.

[9] Ho Kei Cheng, Jihoon Chung, Yu-Wing Tai, and Chi-Keung Tang. Cascadepsp: Toward class-agnostic and very high-resolution segmentation via global and local refinement. In *CVPR*, 2020.

[10] Piotr Dollár and C Lawrence Zitnick. Structured forests for fast edge detection. In *ICCV*, 2013.

[11] Yaroslav Ganin and Victor Lempitsky. N4-fields: Neural network nearest neighbor fields for image transforms. In *ACCV*, 2014.

[12] Zaiwang Gu, Jun Cheng, Huazhu Fu, Kang Zhou, Huaying Hao, Yitian Zhao, Tianyang Zhang, Shenghua Gao, and Jiang Liu. Ce-net: Context encoder network for 2d medical image segmentation. *IEEE transactions on medical imaging*, pages 2281–2292, 2019.

[13] Qibin Hou, Ming-Ming Cheng, Xiaowei Hu, Ali Borji, Zhuowen Tu, and Philip HS Torr. Deeply supervised salient object detection with short connections. In *CVPR*, 2017.

[14] Xiaoling Hu, Fuxin Li, Dimitris Samaras, and Chao Chen. Topology-preserving deep image segmentation. In *NeurIPS*, 2019.

[15] Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei, and Wenyu Liu. Ccnet: Criss-cross attention for semantic segmentation. In *ICCV*, 2019.

[16] Viren Jain, Benjamin Bollmann, Mark Richardson, Daniel R Berger, Moritz N Helmstaedter, Kevin L Briggman, Winfried Denk, Jared B Bowden, John M Mendenhall, Wickliffe C Abraham, et al. Boundary learning by optimization with topological constraints. In *CVPR*, 2010.

[17] Won-Dong Jang and Chang-Su Kim. Interactive image segmentation via backpropagating refinement scheme. In *CVPR*, 2019.

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2014.

[19] Max WK Law and Albert CS Chung. Three dimensional curvilinear structure detection using optimally oriented flux. In *ECCV*, 2008.

[20] Di Lin, Yuanfeng Ji, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. Multi-scale context intertwining for semantic segmentation. In *ECCV*, 2018.

[21] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *CVPR*, 2017.

[22] Wei Liu, Andrew Rabinovich, and Alexander C Berg. Parsenet: Looking wider to see better. 2016.

[23] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[24] Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Pablo Arbeláez, and Luc Van Gool. Deep retinal image understanding. In *MICCAI*, 2016.

[25] Volodymyr Mnih. *Machine learning for aerial image labeling*. University of Toronto (Canada), 2013.

[26] Agata Mosinska, Pablo Marquez-Neila, Mateusz Koziński, and Pascal Fua. Beyond the pixel-wise loss for topology-aware delineation. In *CVPR*, 2018.

[27] Lei Mou, Yitian Zhao, Li Chen, Jun Cheng, Zaiwang Gu, Huaying Hao, Hong Qi, Yalin Zheng, Alejandro Frangi, and Jiang Liu. Cs-net: channel and spatial attention network for curvilinear structure segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2019.

[28] José Ignacio Orlando and Matthew Blaschko. Learning fully-connected crfs for blood vessel segmentation in retinal images. In *MICCAI*, 2014.

[29] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[30] Xuebin Qin, Zichen Zhang, Chenyang Huang, Chao Gao, Masood Dehghan, and Martin Jagersand. Basnet: Boundary-aware salient object detection. In *CVPR*, 2019.

[31] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[32] Mojtaba Seyedhosseini, Mehdi Sajjadi, and Tolga Tasdizen. Image segmentation with cascaded hierarchical models and logistic disjunctive normal networks. In *ICCV*, 2013.

[33] Wei Shen, Bin Wang, Yuan Jiang, Yan Wang, and Alan Yuille. Multi-stage multi-recursive-input fully convolutional networks for neuronal boundary detection. In *ICCV*, 2017.

[34] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. 2015.

[35] Amos Sironi, Engin Türetken, Vincent Lepetit, and Pascal Fua. Multiscale centerline detection. *TPAMI*, 38, 2015.

[36] João VB Soares, Jorge JG Leandro, Roberto M Cesar, Herbert F Jelinek, and Michael J Cree. Retinal vessel segmentation using the 2-d gabor wavelet and supervised classification. *TMI*, 25, 2006.

[37] Engin Turetken, Carlos Becker, Przemyslaw Glowacki, Fethallah Benmansour, and Pascal Fua. Detecting irregular curvilinear structures in gray scale and color imagery using multi-directional oriented flux. In *ICCV*, 2013.

[38] Carles Ventura, Jordi Pont-Tuset, Sergi Caelles, Kevis-Kokitsi Maninis, and Luc Van Gool. Iterative deep learning for road topology extraction. *BMVC*, 2018.

[39] Feigege Wang, Yue Gu, Wenxi Liu, Yuanlong Yu, Shengfeng He, and Jia Pan. Context-aware spatio-recurrent curvilinear structure segmentation. In *CVPR*, 2019.

[40] Linzhao Wang, Lijun Wang, Huchuan Lu, Pingping Zhang, and Xiang Ruan. Salient object detection with recurrent fully convolutional networks. *TPAMI*, 41, 2018.

[41] Wei Wang, Kaicheng Yu, Joachim Hugonot, Pascal Fua, and Mathieu Salzmann. Recurrent u-net for resource-constrained segmentation. In *ICCV*, 2019.

[42] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018.

[43] Yue Wang, Jie Gao, and Joseph SB Mitchell. Boundary recognition in sensor networks by topological methods. In *International conference on Mobile computing and networking*.

[44] Yunchao Wei, Huaxin Xiao, Honghui Shi, Zequn Jie, Jiashi Feng, and Thomas S Huang. Revisiting dilated convolution: A simple approach for weakly-and semi-supervised semantic segmentation. In *CVPR*, 2018.

[45] Yao Wei, Kai Zhang, and Shunping Ji. Simultaneous road surface and centerline extraction from large-scale remote sensing images using cnn-based segmentation and tracing. *TGRS*, 58, 2020.

[46] Christian Wiedemann, Christian Heipke, Helmut Mayer, and Olivier Jamet. Empirical evaluation of automatically extracted road axes. *Empirical evaluation techniques in computer vision*, 12.

[47] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *ECCV*, 2018.

[48] Dijia Wu, David Liu, Zoltan Puskas, Chao Lu, Andreas Wimmer, Christian Tietjen, Grzegorz Soza, and S Kevin Zhou. A learning based deformable template matching method for automatic rib centerline extraction and labeling in ct images. In *CVPR*, 2012.

[49] Zhe Wu, Li Su, and Qingming Huang. Cascaded partial decoder for fast and accurate salient object detection. In *CVPR*, 2019.

[50] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *ICCV*, 2015.

[51] Ning Xu, Brian Price, Scott Cohen, and Thomas Huang. Deep image matting. In *CVPR*, 2017.

[52] Zhenhua Xu, Yuxiang Sun, and Ming Liu. Topo-boundary: A benchmark dataset on topological road-boundary detection using aerial images for autonomous driving. *arXiv preprint arXiv:2103.17119*, 2021.

[53] Fan Yang, Lei Zhang, Sijia Yu, Danil Prokhorov, Xue Mei, and Haibin Ling. Feature pyramid and hierarchical boosting network for pavement crack detection. *T-ITS*, 21, 2019.

[54] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. 2016.

[55] Lei Zhang, Fan Yang, Yimin Daniel Zhang, and Ying Julie Zhu. Road crack detection using deep convolutional neural network. In *ICIP*, 2016.

[56] Xiaoning Zhang, Tiantian Wang, Jinqing Qi, Huchuan Lu, and Gang Wang. Progressive attention guided recurrent network for salient object detection. In *CVPR*, 2018.

[57] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017.

[58] Mingmin Zhen, Jinglu Wang, Lei Zhou, Shiwei Li, Tianwei Shen, Jiaxiang Shang, Tian Fang, and Long Quan. Joint semantic segmentation and boundary detection using iterative pyramid contexts. In *CVPR*, 2020.

[59] Zhen Zhu, Mengde Xu, Song Bai, Tengteng Huang, and Xiang Bai. Asymmetric non-local neural networks for semantic segmentation. In *ICCV*, 2019.

[60] Qin Zou, Yu Cao, Qingquan Li, Qingzhou Mao, and Song Wang. Cracktree: Automatic crack detection from pavement images. *PRL*, 33, 2012.