Author Rebuttal for NeurIPS 2019 Submission #964

We thank all the reviewers for their positive comments and valuable suggestions. This paper presents a linear-time *learnable tree filter* to capture long-range dependencies while preserving structural details for semantic segmentation. In the initial review, all reviewers agreed with the novelty and contributions of the *learnable tree filter*. We respond to all the comments as follows. In addition, we will carefully revise the manuscript to improve its readability before the final submission. As required by R5, an executable example (the same anonymous link as that in Supplementary Materials) has been provided. **All the source code will be released to the community soon.**

Response to Reviewer #1

Q1: *How to generate MST and initialize the weights? Details of muti-groups in Fig. 2?*
A1: Sorry for the confusion. Actually, the weights used for *MST construction* and *filtering* are different. The MST is constructed upon the low-level feature in encoder (defined as M_l in L141 and illustrated in Fig. 2). The weights for filtering do not need to be initialized, but are calculated based on high-level semantics (defined as $\boldsymbol{X}_l$ in L143 and illustrated in Fig. 2) in *every* step. *Multi-groups* in Fig. 2 represents calculating multiple groups of weights for the filter, which allows it to be sensitive to different components (please refer to L145-146 and Sec. 3.2 for more details).
Q2: *Can the proposed tree filter be used in encoder? How would the results be like?*
A2: Yes, it can be used in encoder. Actually, the experiments in Tab. 3 (when OS is 8) are conducted with encoder *only*. In addition, we inject the *Tree Filtering module* into C3-C5 of ResNet-50 and achieve a 71.7% mIoU, which is *inferior* to that in Tab. 3 (a 72.5% mIoU when applied to the decoder). It means higher-level semantic matters for the filter weights (decoder contains richer semantic cues, identical with the design in L40-43). This will be made clear.
Q3: *The evaluations on ADE20K?*
A3: Thanks for this suggestion. During rebuttal, we evaluate our methods using a ResNet-50 backbone on the ADE20K *val* set, and list the results in Tab. 7. This table will be added in the final version.

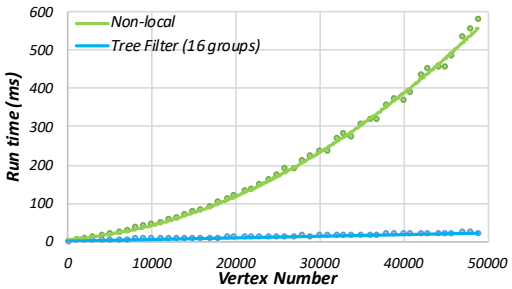

Figure 5: Runtime comparisons on Tesla V100.

Backbone	TF	MS	mIoU (%)	Pixel Acc (%)
ResNet-50	✗	✗	35.0	76.5
	✓	✗	40.0	79.3
	✓	✓	**41.1**	**80.2**

Table 7: Results on the ADE20K *val* set. **TF** denotes *multi-stage Tree Filtering modules* with decoder. **MS** indicates multi-scale testing strategy. All the experiments are conducted with a *vanilla* ResNet-50 backbone (*without* dilated convolutions or the ASPP module).

Response to Reviewer #4

Q1: *Additional improvements over the PSP and Non-local module?*
A1: Different from PSP [3] and Non-local [12] module, the proposed *Tree Filtering module* also preserves structural details when capturing long-range dependencies (refer to the qualitative analysis in Ablation Study and Supplementary Materials). To verify the effectiveness, we conduct experiments with additional PSP and Non-local module, and achieve 1.1% and 0.7% absolute gain (**75.4%** for PSP+TF and **74.9%** for NL+TF), respectively.
Q2: *Performance comparison.*
A2: Thanks for your comments. Actually, we adopt *vanilla* ResNet-101 *without* dilated convolutions or Dense blocks (adopted by PSANet [31] and DenseASPP [32]) as our backbone in Tab. 5 and Tab. 6. Of course, we will give more competitive results with stronger backbones as well as revise our claim in the final version.

Response to Reviewer #5

Q1: *Why the computational complexity of MST construction (Borůvka's algorithm) is linear?*
A1: Sorry for not having clarified this point clearly. Generally, the *Borůvka's algorithm* runs in $\mathcal{O}(E \log V)$. Nevertheless, as illustrated in Fig. 1, we build MST from a *4-connected* ***planar*** graph. When the input graph is planar, the computational complexity can be reduced to *linear* using *Contractive Borůvka's algorithm*. Of course, this can also be achieved by other algorithms (*e.g.,* [Karger *et.al.*, JACM, 1995]). We will clarify it in the final version.
Q2: *Provide the minimal working example and compare empirical runtime with other methods.*
A2: Following your suggestion, we have already provided the ***executable code*** (the same anonymous link with that in Supplementary Materials) as well as a benchmark for runtime comparison. We also illustrate the comparison against the non-local operation in Fig. 5. To clarify more details, we will release our source code to the community.
Q3: *How to select hyper-parameters?*
A3: For network training, we just follow traditional protocols (refer to Sec. 3.1) without bells-and-whistles. While for the proposed *Tree Filtering module*, we conduct ablation studies (especially for the *equipped stage* and *group number* in Sec. 3.2) and choose the best-performed combination of hyper-parameters.