NEURAL INFORMATION PROCESSING SYSTEMS

Label-efficient Segmentation via Affinity Propagation

Wentong Li[1*], Yuqian Yuan[1*], Song Wang[1], Wenyu Liu[1], Dongqi Tang[2], Jian Liu[2], Jianke Zhu[1†], Lei Zhang[3]

[1]Zhejiang University [2]Ant Group [3]The Hong Kong Polytechnical University

Code: https://github.com/CircleRadon/APro

Project Page: https://LiWentomng.github.io/apro/

01/Background

□ Label-efficient segmentation with Sparse Annotations

- Compared with *fully supervision with mask annotations*, there are several sparse forms:

Bounding Box

Scribble

Single Point

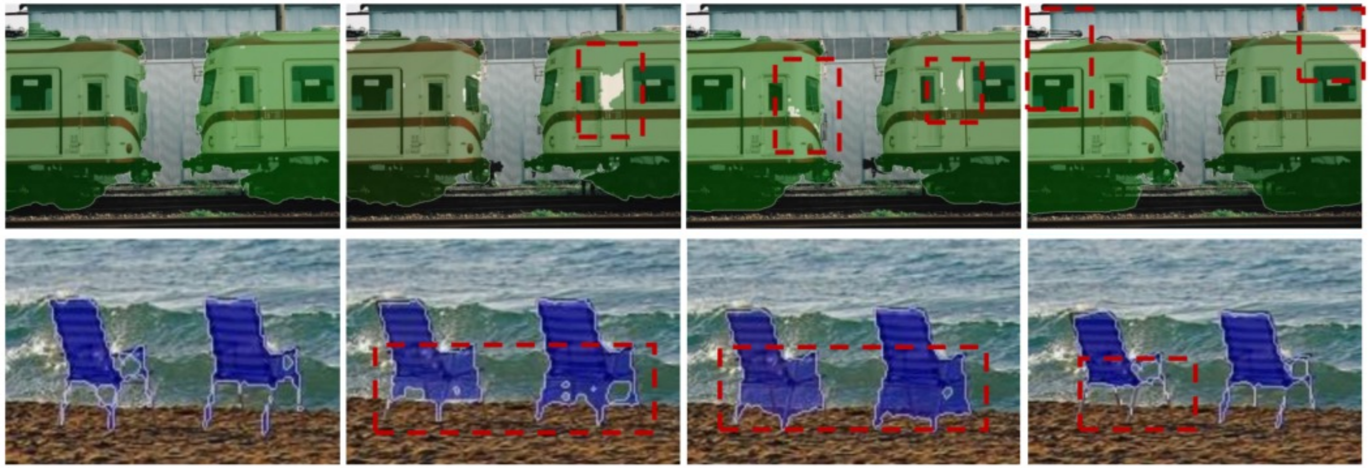
Pretrained CLIP

Mask Prediction

01/Background

Limitations of Previous Methods

- Previous methods adopt the *local appearance kernel* (LAB color space or RGB color space)
  - Cannot **capture global context cues** and **long-range affinity dependencies**.
  - Fail to take the intrinsic **topology of objects** into account, lacks capability of **detail preservation**.

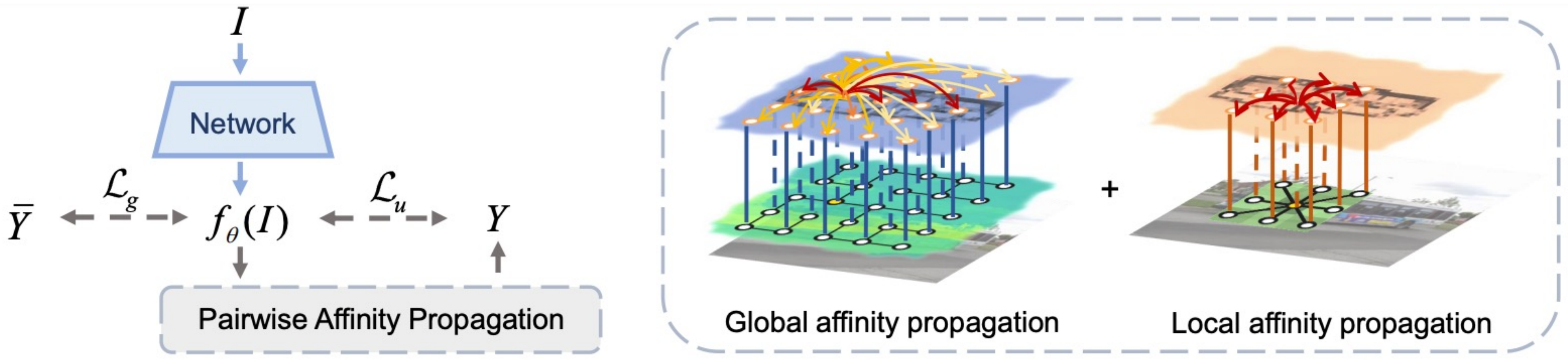


APro(Ours) CRF Loss[1] TreeEnergy Loss[2] Pairwise Loss[3]

[1] Shiyi Lan et al, Vision transformers are good mask auto-labelers. *In CVPR, 2023*.
[2] Zhiyuan Liang et al, Tree energy loss: Towards sparsely annotated semantic segmentation. *In CVPR, 2022*.
[3] Zhi Tian et al, Boxinst: High-performance instance segmentation with box annotations. *In CVPR, 2021*.

02/Our Method

APro



(a) (b)

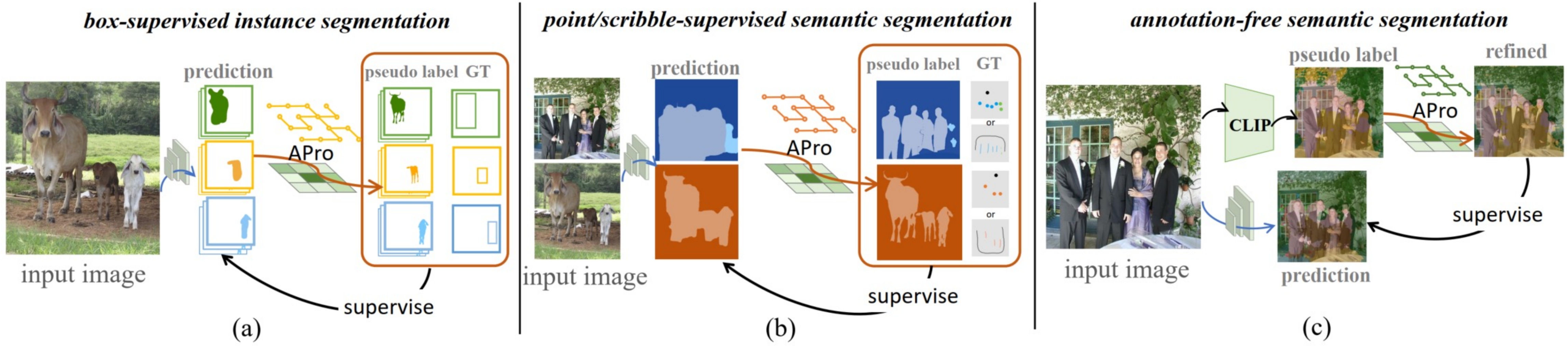
- We define the generation of pseudo label Y as an **affinity propagation process (APro)**:

$$y_i = \frac{1}{z_i} \sum_{j \in \tau} \phi(x_j)\psi(x_i, x_j) \quad (1)$$

- The proposed approach consists of **global affinity propagation (GP)** and **local affinity propagation (LP)** to generate accurate pseudo labels.

02/Our Method

APro: A general plug-in component

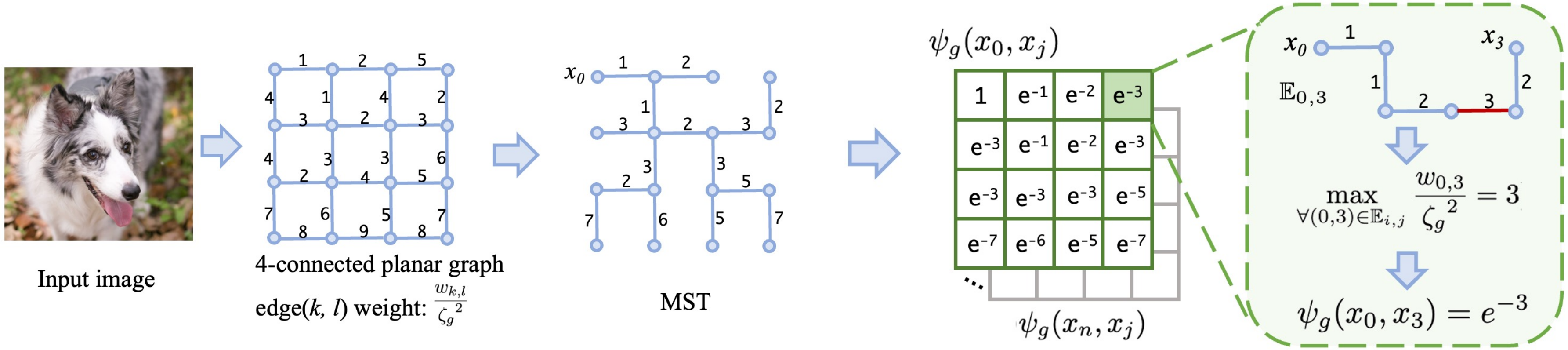


- Box-supervised instance segmentation
- Point/scribble-supervised semantic segmentation
- Annotation-free semantic segmentation

02/Our Method

Global Affinity Propagation

- tree-based sparse graph



- global pairwise potential ψ_g:

$$\psi_g(x_i, x_j) = \underset{\forall j \in \mathcal{V}}{\mathcal{T}(I_i, I_j)} = \exp(-\max_{\forall(k,l)\in\mathbb{E}_{i,j}} \frac{w_{k,l}}{\zeta_g{}^2}) \qquad (1)$$

$$y_i^g = \frac{1}{z_i^g} \sum_{j\in\mathcal{V}} \phi(x_j)\psi_g(x_i, x_j), \;\; z_i^g = \sum_{j\in\mathcal{V}} \psi_g(x_i, x_j) \qquad (2)$$

02/Our Method

Efficient Implementation

- ***Lazy Propagation*** scheme

$$\mathcal{Z}(\delta)_{k^*} = \mathcal{Z}(\delta)_{k^*} + \begin{cases} \exp(-w_{k,l}/{\zeta_g}^2)S(\delta)_l & \mathcal{U}_k.\text{rank} > \mathcal{U}_l.\text{rank}, \\ \exp(-w_{k,l}/{\zeta_g}^2)S(\delta)_l - \mathcal{Z}(\delta)_{l^*} & \text{otherwise}, \end{cases} \quad (1)$$

- Global affinity propagation term

$$LProp(\delta)_i = \delta_i + \sum_{r \in Asc_{\mathcal{G}_T}(i) \cup \{i\}} \mathcal{Z}(\delta)_r, \quad (2)$$

- Time complexity $\mathcal{O}(N \log N)$.

Effic. Imple.	Ave. Runtime
✗	4.3×10^3
✓	0.8

Algorithm 1: Algorithm for GP process

Input: Tree $\mathcal{G}_T \in \mathbb{N}^{e\times 2}$; Pairwise distance $\boldsymbol{w} \in \mathbb{R}^N$; Dense predictions $\phi(x) \in \mathbb{R}^N$; Vertex num N; Edge num $e = N - 1$; Set of vertices $\mathcal{V}$.
Output: $y^g \in \mathbb{R}^N$.
$\Lambda \leftarrow \mathbf{1} \in \mathbb{R}^N$
$F \leftarrow \{0, 1, 2, ..., N-1\}$ ▷ Initialize each vertex as a connected block
Sort $\{\mathcal{G}_T, \boldsymbol{w}\}$ in ascending order of $\boldsymbol{w}$. ▷ Quick Sort
for $(k,l) \in \mathcal{G}_T, w_i \in \boldsymbol{w}$ **do**
  $a \leftarrow \text{find}(k)$, $b \leftarrow \text{find}(l)$ ▷ Find the root node with Path Compression
  Update$\{\mathcal{Z}(\phi)_a, \mathcal{Z}(\Lambda)_a, \mathcal{Z}(\phi)_b, \mathcal{Z}(\Lambda)_b\}$ ▷ Add lazy tag
  if $S_a < S_b$ **then**
    swap(a, b) ▷ Merge by Rank
  $F_b \leftarrow a$ ▷ Merge two connected blocks
for $v \in \mathcal{V}$ **do**
  $p \leftarrow \text{find}(v)$
  for $\delta \in \{\phi, \Lambda\}$ **do**
    if $p = v$ **then**
      $LProp(\delta)_v = \mathcal{Z}(\delta)_v + \delta_v$
    else
      $LProp(\delta)_v = \mathcal{Z}(\delta)_p + \mathcal{Z}(\delta)_v + \delta_v$
  $y^g_v = \frac{LProp(\phi)_v}{LProp(\Lambda)_v}$ ▷ Normalization
return y^g

02/Our Method

Local Affinity Propagation

- Use gaussian kernel
- The local pairwise term ψ_s

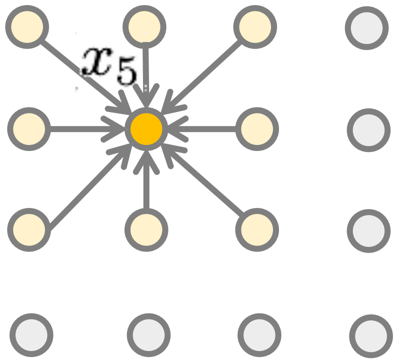
$$\psi_s(x_i, x_j) = \underset{j \in \mathcal{N}(i)}{\mathcal{K}(I_i, I_j)} = \exp\left(\frac{-|I_i - I_j|^2}{\zeta_s^2}\right) \tag{1}$$

- The pseudo label y^s can be obtained via:

$$y_i^s = \frac{1}{z_i^s} \sum_{j \in \mathcal{N}(i)} \phi(x_j)\psi_s(x_i, x_j), \quad z_i^s = \sum_{j \in \mathcal{N}(i)} \psi_s(x_i, x_j), \tag{2}$$

- 5x faster than MeanField-based method[1] .

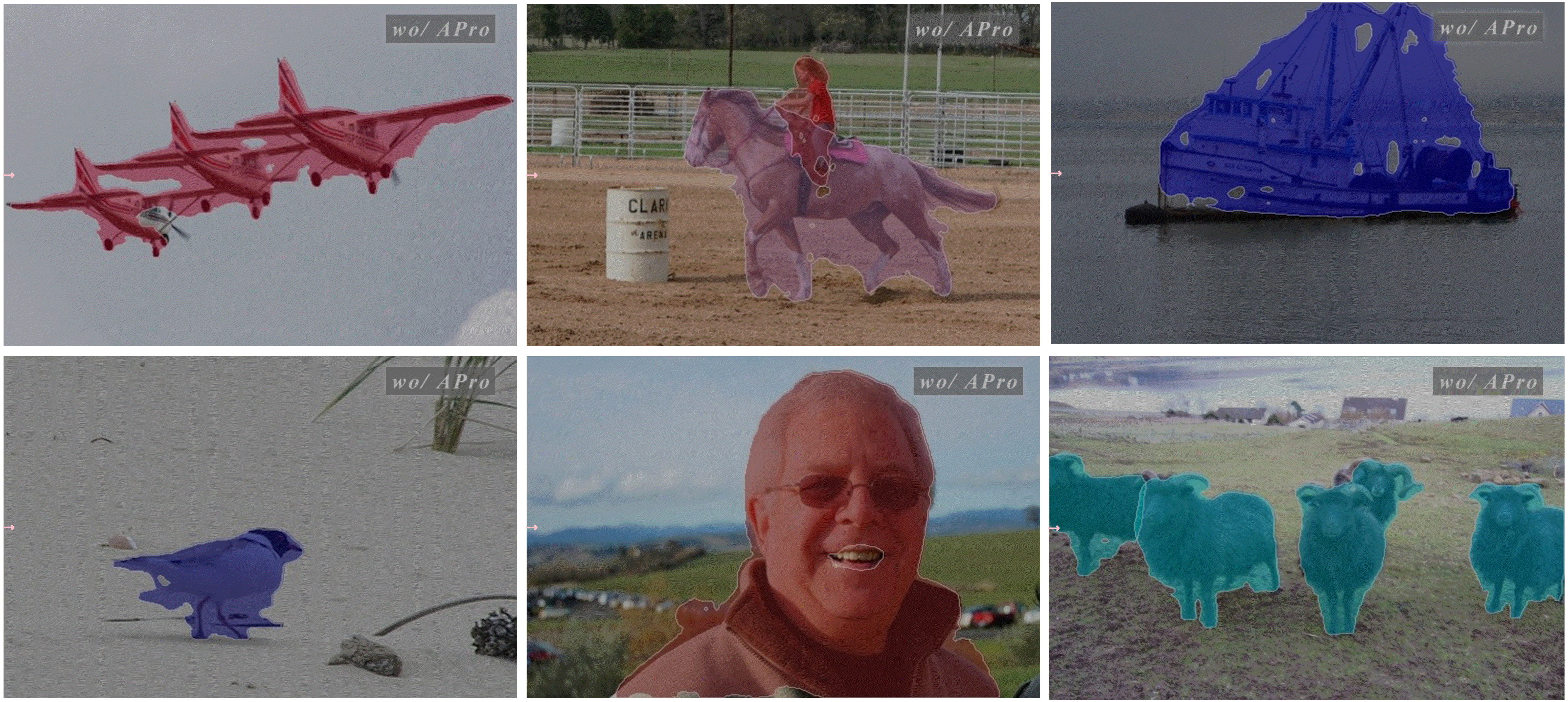
Gaussian Kernel



[1] Shiyi Lan et al, Vision transformers are good mask auto-labelers. *In CVPR, 2023*.

03/Experiments

Weakly-supervised Instance Segmentation



03/Experiments

□ Weakly-supervised Instance Segmentation

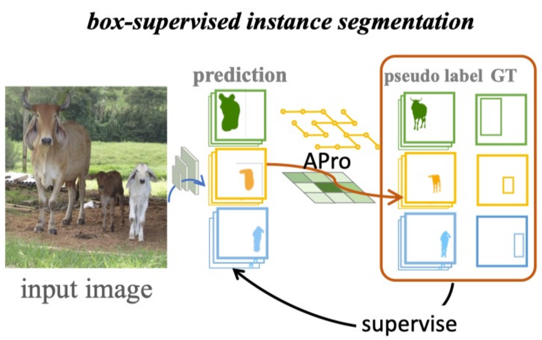


Table 1: Quantitative results (§4.1) on Pascal VOC [43] and COCO `val` [49] with mask AP(%).

Method	Backbone	#Epoch	Pascal VOC AP	Pascal VOC AP_{50}	Pascal VOC AP_{75}	COCO AP	COCO AP_{50}	COCO AP_{75}
BBTP [NeurIPS19] [16]	ResNet-101	12	23.1	54.1	17.1	21.1	45.5	17.2
BoxInst [CVPR21] [8]	ResNet-50	36	34.3	58.6	34.6	31.8	54.4	32.5
DiscoBox [ICCV21] [17]	ResNet-50	36	-	59.8	35.5	31.4	52.6	32.2
BoxLevelset [ECCV22] [9]	ResNet-50	36	36.3	64.2	35.9	31.4	53.7	31.8
SOLOv2 Framework								
Pairwise Loss [CVPR21] [8]	ResNet-50	12	35.7	64.3	35.1	31.0	52.8	31.5
TreeEnergy Loss [CVPR22] [7]	ResNet-50	12	35.0	64.4	34.7	30.9	52.9	31.3
CRF Loss [CVPR23] [10]	ResNet-50	12	35.0	64.7	34.9	30.9	53.1	31.4
APro(Ours)	ResNet-50	12	**37.1**	**65.1**	**37.0**	**32.0**	**53.4**	**32.9**
Pairwise Loss [CVPR21] [8]	ResNet-50	36	36.5	63.4	38.1	32.4	54.5	33.4
TreeEnergy Loss [CVPR22] [7]	ResNet-50	36	36.1	63.5	36.1	31.4	54.0	31.2
CRF Loss [CVPR23] [10]	ResNet-50	36	35.9	64.0	35.7	32.5	54.9	33.2
APro(Ours)	ResNet-50	36	38.4	65.4	39.8	32.9	55.2	33.6
APro(Ours)	ResNet-101	36	**40.5**	**67.9**	**42.6**	**34.3**	**57.0**	**35.3**
Mask2Former Framework								
Pairwise Loss [CVPR21] [8]	ResNet-50	12	35.2	62.9	33.9	33.8	57.1	34.0
TreeEnergy Loss [CVPR22] [7]	ResNet-50	12	36.0	65.0	34.3	33.5	56.7	33.7
CRF Loss [CVPR23] [10]	ResNet-50	12	35.7	64.3	35.2	33.5	57.5	33.8
APro(Ours)	ResNet-50	12	37.0	65.1	37.0	34.4	57.7	35.3
APro(Ours)	ResNet-50	50	42.3	70.6	44.5	36.1	62.0	36.7
APro(Ours)	ResNet-101	50	43.6	72.0	45.7	38.0	63.6	38.7
APro(Ours)	Swin-L	50	**49.6**	**77.6**	**53.1**	**41.0**	**68.3**	**41.9**

03/Experiments

Weakly-supervised Semantic Segmentation

point/scribble-supervised semantic segmentation

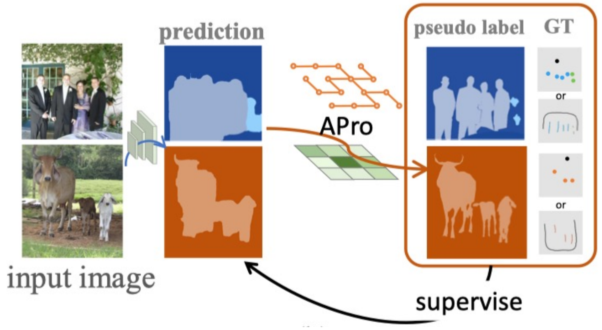


Table 2: Quantitative results (§4.2) on Pascal VOC2012 [51] val with mean IoU(%).

Method	Backbone	Supervision	CRF Post.	mIoU
†KernelCut Loss [ECCV18] [6]	DeepLabV2	Point	✓	57.0
*TEL [CVPR22] [7]	LTF		✗	66.8
APro(Ours)	LTF		✗	**67.7**
†NormCut Loss [CVPR18] [59]	DeepLabV2	Scribble	✓	74.5
†DenseCRF Loss [ECCV18] [6]	DeepLabV2		✓	75.0
†KernelCut Loss [ECCV18] [6]	DeepLabV2		✓	75.0
†GridCRF Loss [ICCV19] [36]	DeepLabV2		✗	72.8
PSI [ICCV21] [36]	DeepLabV3		✗	74.9
*TEL [CVPR22] [7]	LTF		✗	76.2
APro(Ours)	LTF		✗	**76.6**

†:adopting multi-stage training, ∗:our re-implementation.

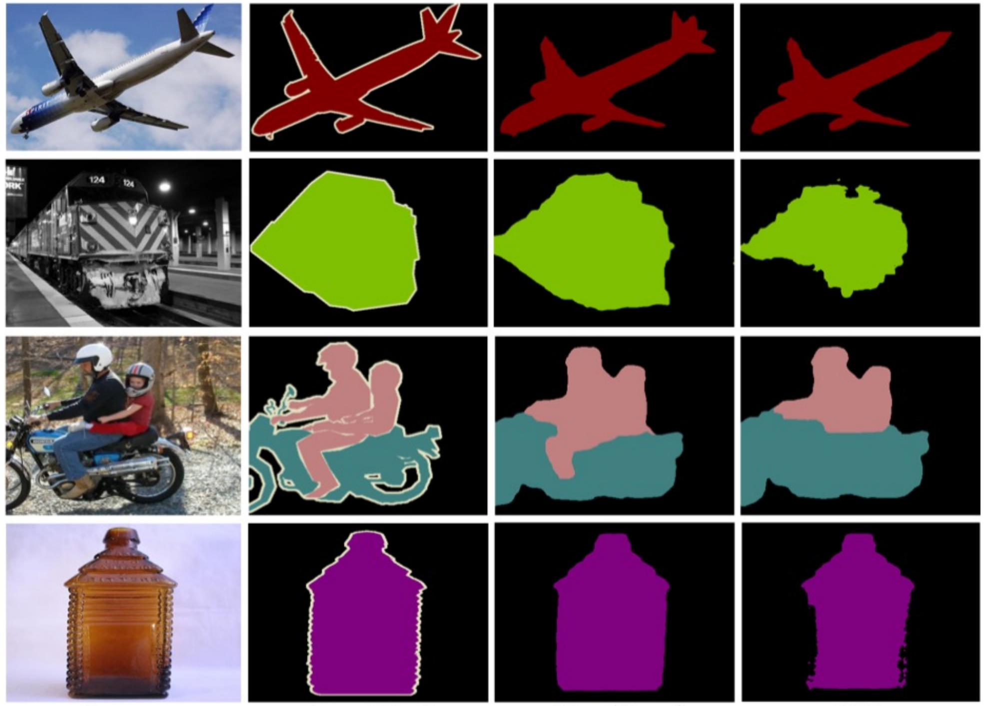


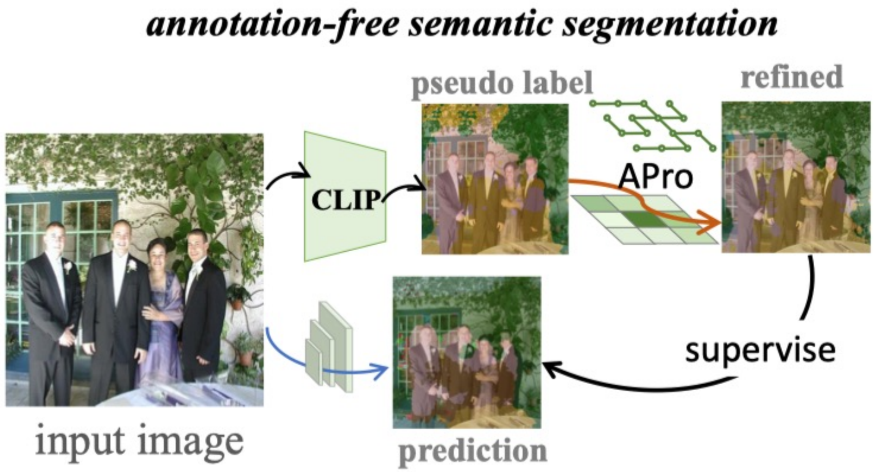
Image | Ground Truth | APro(Ours) | TEL

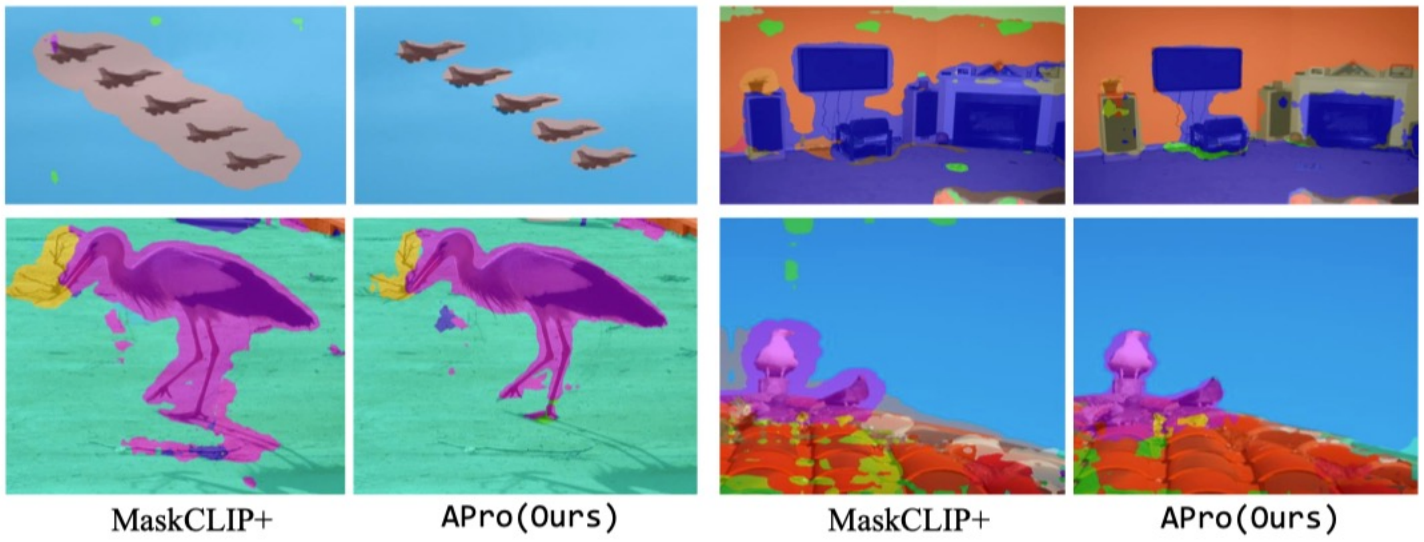
03/Experiments

CLIP-guided Semantic Segmentation

Table 3: Quantitative results (§4.3) on Pascal VOC2012 [51] val, Pascal Context [60] val, and COCO-Stuff [61] val with mean IoU (%).

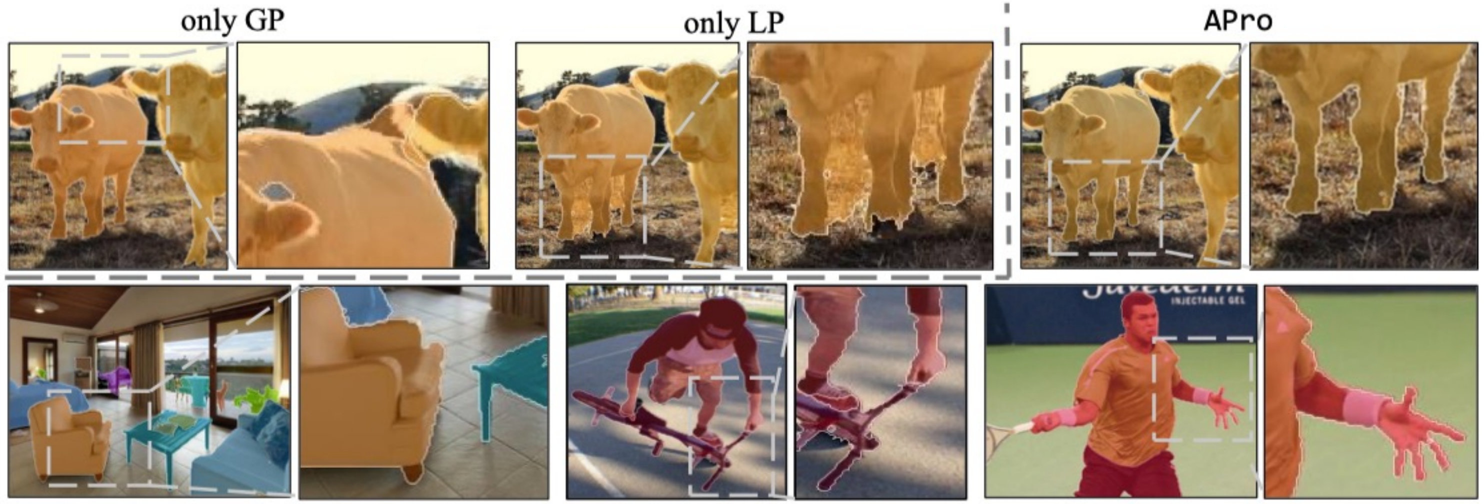
Method	CLIP Model	VOC2012	Context	COCO.
MaskCLIP+ [ECCV22] [40]	ResNet-50	58.0	23.9	13.6
APro(Ours)		**61.6** ↑3.6	**25.4** ↑1.5	**14.6** ↑1.0
MaskCLIP+ [ECCV22] [40]	ResNet-50×16	67.5	25.2	17.3
APro(Ours)		**70.4** ↑2.9	**26.5** ↑1.3	**18.2** ↑0.9
MaskCLIP+ [ECCV22] [40]	ViT-B/16	73.6	31.1	18.0
APro(Ours)		**75.1** ↑1.5	**32.6** ↑1.5	**19.5** ↑1.5

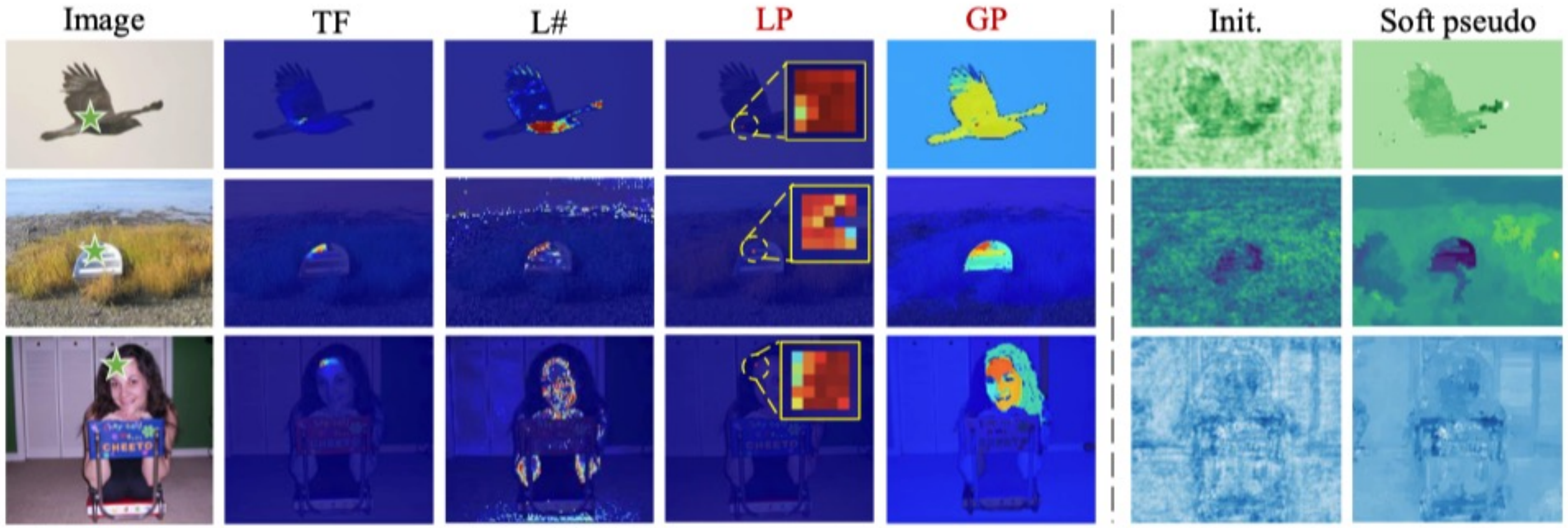




03/Experiments

Visual comparisons





03/Experiments

Diagnostic Experiments

Table 4: Effects of unary and pairwise terms.

Unary	Global Pairwise	Local Pairwise	AP	AP_{50}	AP_{75}
✓			25.9	57.0	20.4
✓	✓		36.3	63.9	37.0
✓		✓	36.0	64.3	35.6
✓	✓	✓	**38.4**	**65.4**	**39.8**

Table 5: Comparisons with tree-based methods.

Method	AP	AP_{50}	AP_{75}
TreeFilter [19]	36.1	63.5	36.1
TreeFilter [19] + Local Pairwise	36.8	64.4	36.5
Global + Local Pairwise (Ours)	**38.4**	**65.4**	**39.8**

Table 6: Comparisons on local pairwise affinity modeling.

LP(Ours)		MeanField[10]	
Iteration	AP	Iteration	AP
10	35.8	20	35.2
20	**36.0**	30	35.5
30	35.7	50	35.5
50	35.6	100	**35.9**

Table 7: Generation of soft pseudo labels.

Method	AP	AP_{50}	AP_{75}
GP-LP-C	36.8	63.7	37.8
LP-GP-C	37.7	65.1	39.1
GP-LP-P	**38.4**	**65.4**	**39.8**

04/Takeaways

- We proposed a novel universal component for weakly-supervised segmentation by formulating it as an **affinity propagation process**.
- A global and a local pairwise affinity term were introduced.
- Experiments on three typical label-efficient segmentation tasks proved the effectiveness of `APro`.
    - box-supervised instance segmentation
    - point/scribblesupervised semantic segmentation
    - CLIP-guided annotation-free semantic segmentation

- Code available: https://github.com/CircleRadon/APro
- Project homepage: https://liwentomng.github.io/apro