2021 ICCV OCTOBER 11-17 VIRTUAL

LabOR: Labeling Only if Required for Domain Adaptive Semantic Segmentation


Inkyu Shin


Dong-Jin Kim


Jae Won Cho


Sanghyun Woo


Kwanyong Park


In So Kweon

Korea Advanced Institute of Science and Technology(KAIST)
Robotics and Computer Vision Lab(RCV Lab)

KAIST

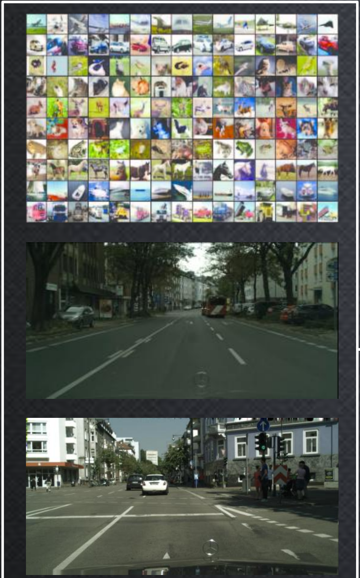
RCV ROBOTICS AND COMPUTER VISION LAB

Bottleneck in Deep learning

Deep learning framework consists of three things:

1. Data Acquisition
2. Data Labeling
3. Model Training

<Sensor>



<Labeling Company>

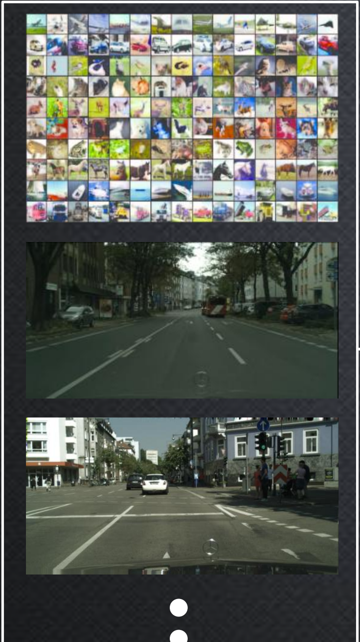
Loading

Bottleneck in Deep learning

Deep learning framework consists of three things:

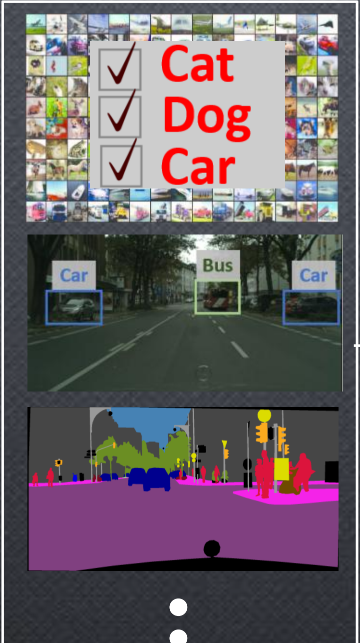
1. Data Acquisition

<Sensor>



2. Data Labeling

<Labeling Company>



3. Model Training

<Deep learning model>

Training

Prediction

Bottleneck in Deep learning

Deep learning framework consists of three things:

1. Data Acquisition

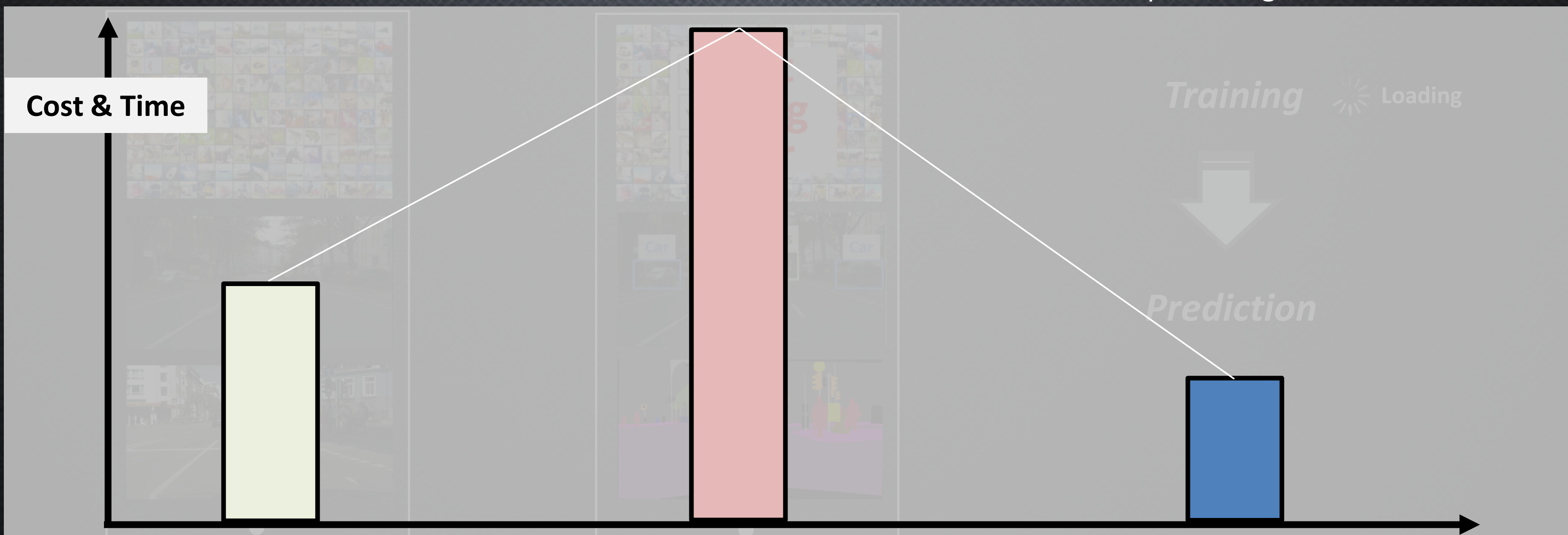
<Sensor>

2. Data Labeling

<Labeling Company>

3. Model Training

<Deep learning model>



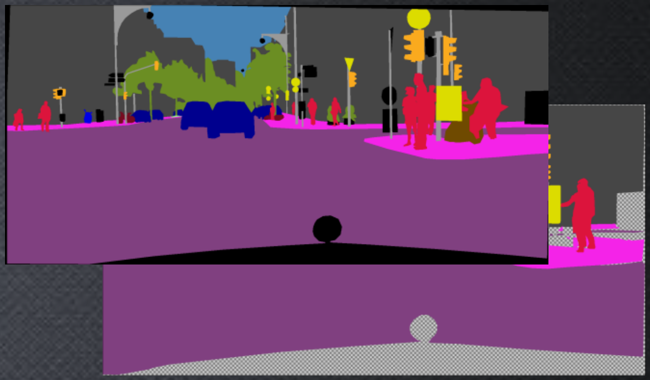
Data labeling is the main bottleneck in deep learning framework.

Labeling for Semantic Segmentation

Specifically, we need a pixel-level labeling for semantic segmentation.

Images

Human annotators



Ground truth

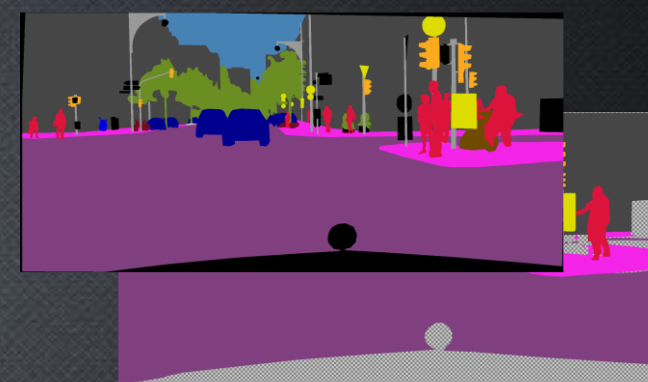
Labeling for Semantic Segmentation

There are two major issues on real-world labeling for semantic segmentation

#1. labor-cost(Time and Expense)

Images

Human annotators



Ground truth

Cost about ***9 minutes and 30$*** per one label[1]

[1] Zlateski et al., On the importance of Label Quality for Semantic Segmentation. CVPR 2018

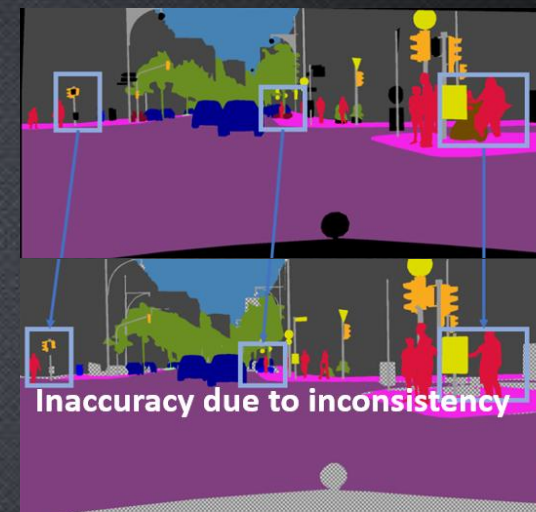
Labeling for Semantic Segmentation

There are two major issues on real-world labeling for semantic segmentation

#2. labeling inaccuracy

Images
(t, t+1 frames)

Human annotators



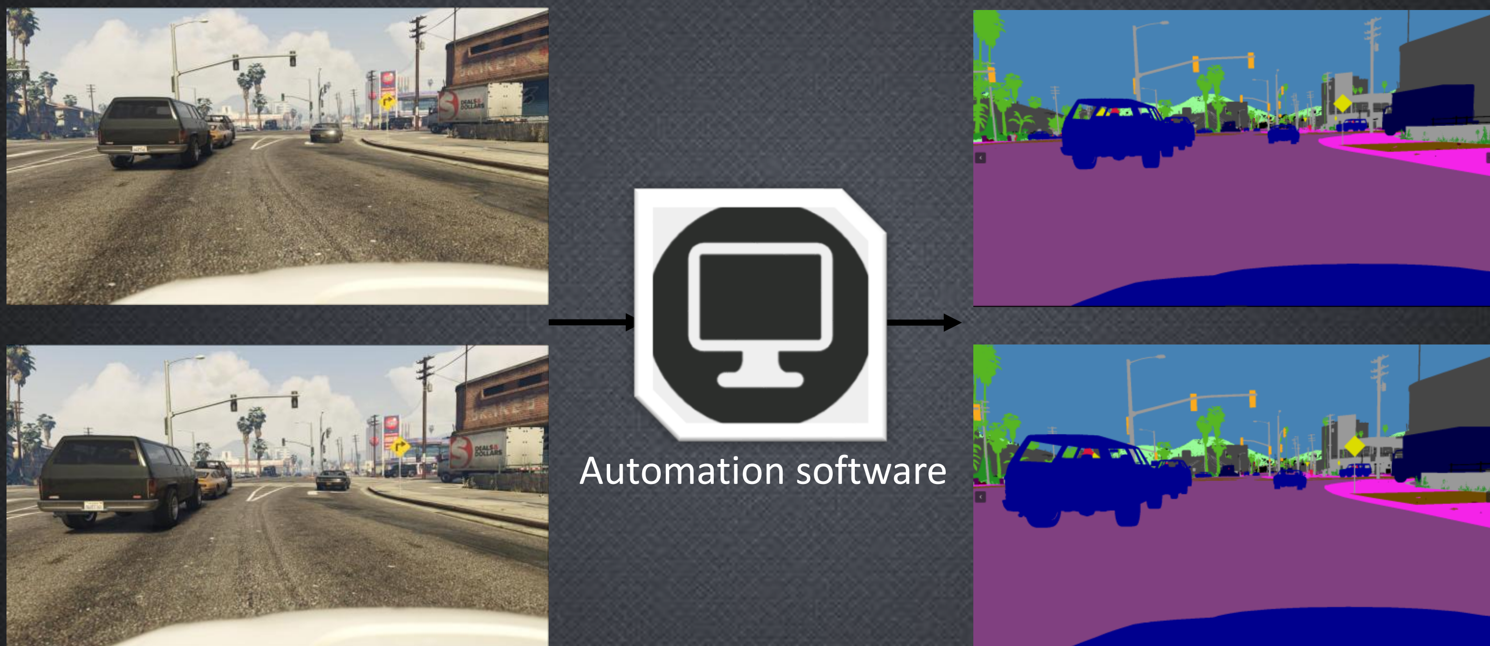
Ground truth
(t, t+1 frames)

Human error due to intensive labor [1]

[1] Machine Learning Data Label., www.basic.ai/data/label

Labeling for Semantic Segmentation

To solve these issues, data from simulator[1,2,3] are used with very small amount of labeling cost and high accuracy

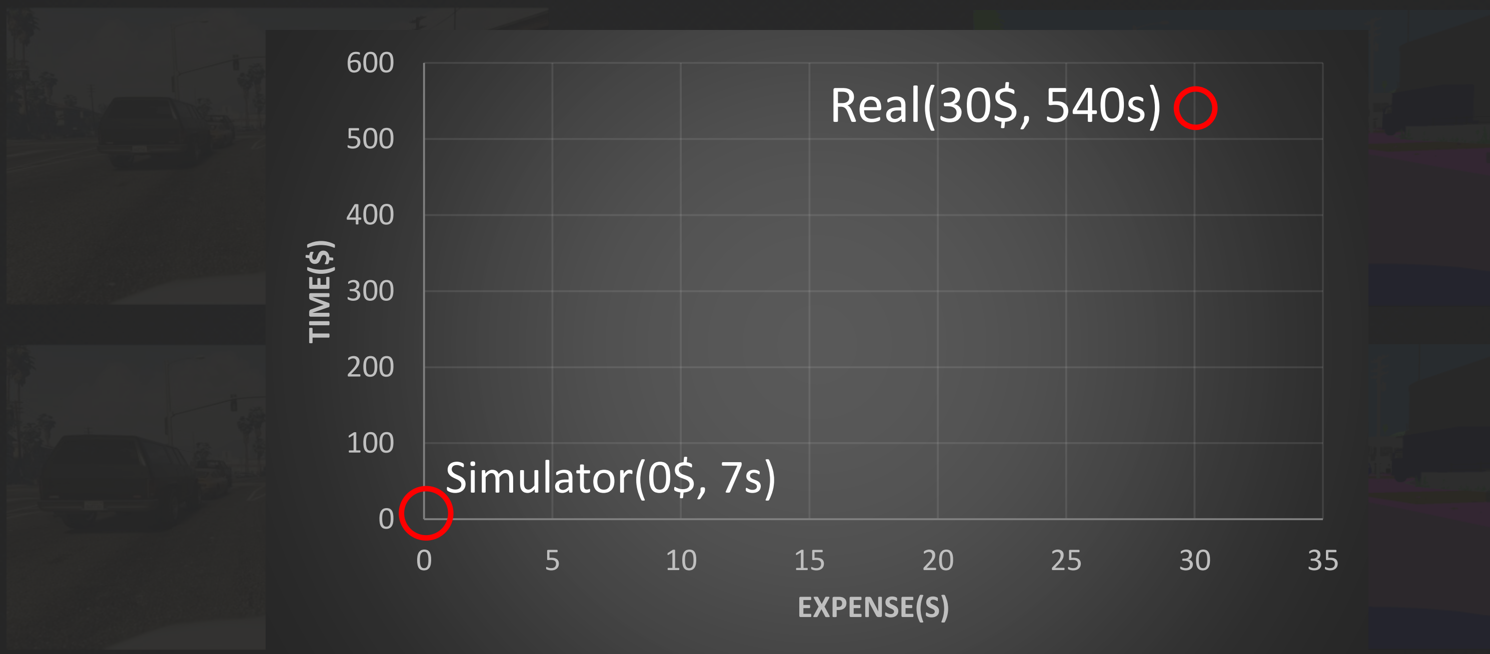


Cost about ***7 seconds*** *and* ***almost 0$*** per one label [1]

[1] SR Richter et al., Playing for Benchmarks. ICCV 2017.
[2] SR Richter et al., Playing for Data: Ground Truth from Computer Games. ECCV 2016.
[3] G Ros et al., The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes. CVPR 2016

Labeling for Semantic Segmentation

To solve these issues, data from simulator[1,2,3] are used with very small amount of labeling cost and high accuracy.



The effectiveness of using simulator data

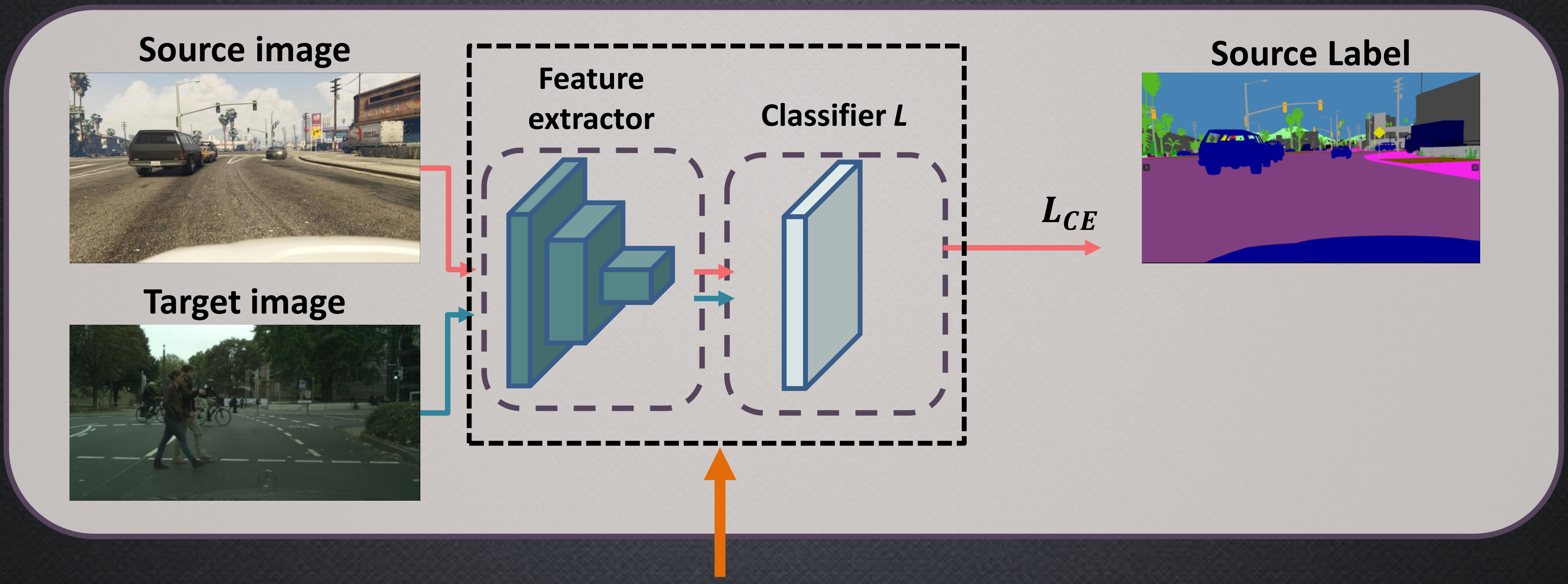
Cost about 7 seconds and almost 0$ per one label [1]

[1] SR Richter et al., Playing for Benchmarks. ICCV 2017.
[2] SR Richter et al., Playing for Data: Ground Truth from Computer Games. ECCV 2016.
[3] G Ros et al., The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes. CVPR 2016

2021 ICCV OCTOBER 11-17 VIRTUAL

Unsupervised Domain Adaptation

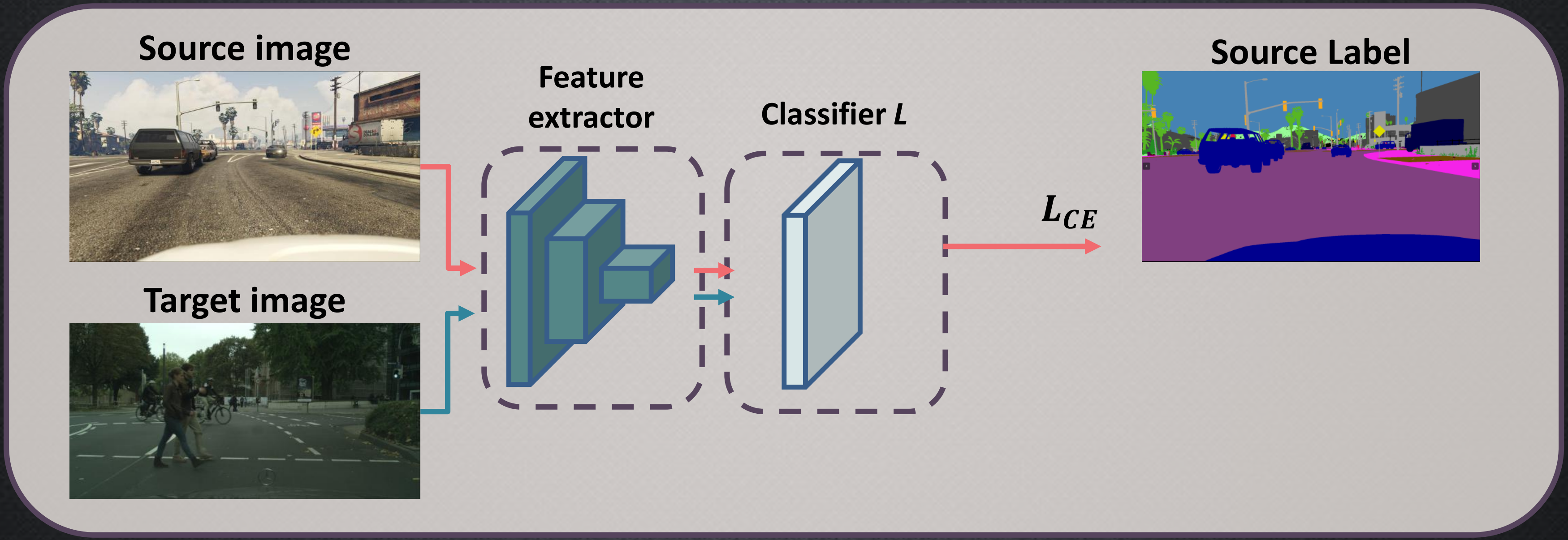
UDA aims to transfer the knowledge learned from labeled simulator data.
- Make segmentation model to be **domain adaptive**.



Make it to be domain adaptive

Previous Domain Adaptation

Previous Domain Adaptation (UDA) consists of **three paradigms**:

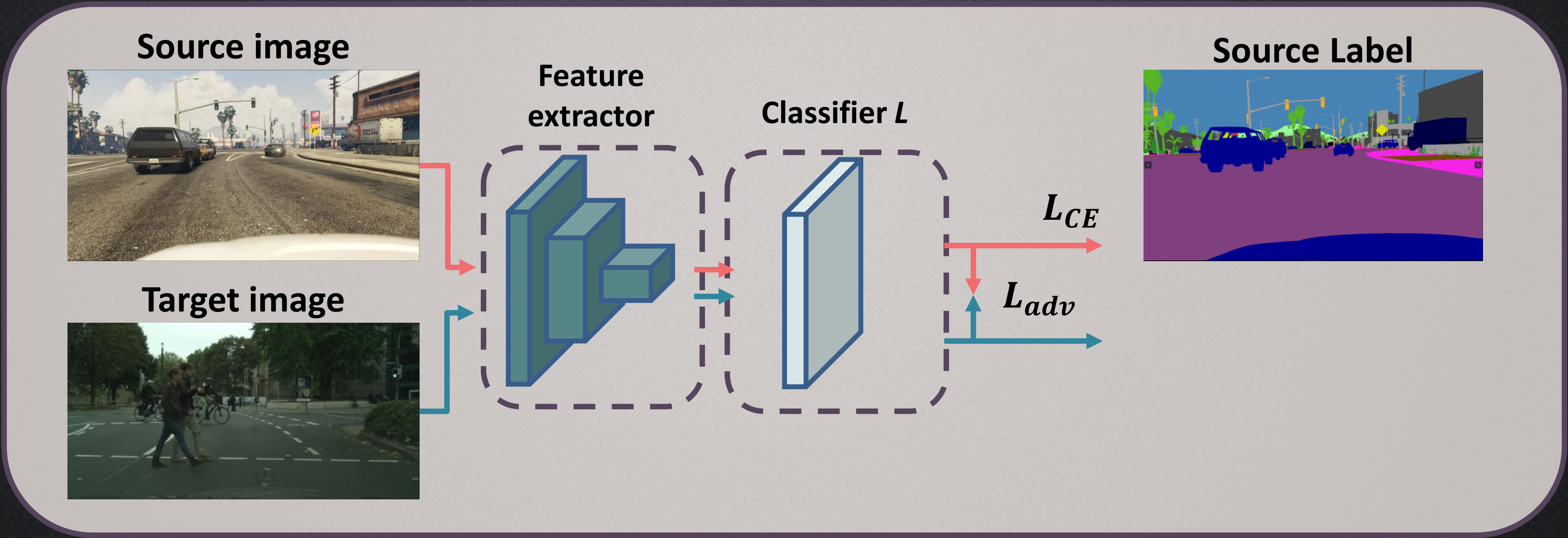


Previous Domain Adaptation

Previous Domain Adaptation (UDA) consists of **three paradigms**:

Adversarial Learning based DA (L_{adv})

- AdaptSegNet[1], Advent[2], CLAN[3], PatchAdapt[4], MaxSquare[5]



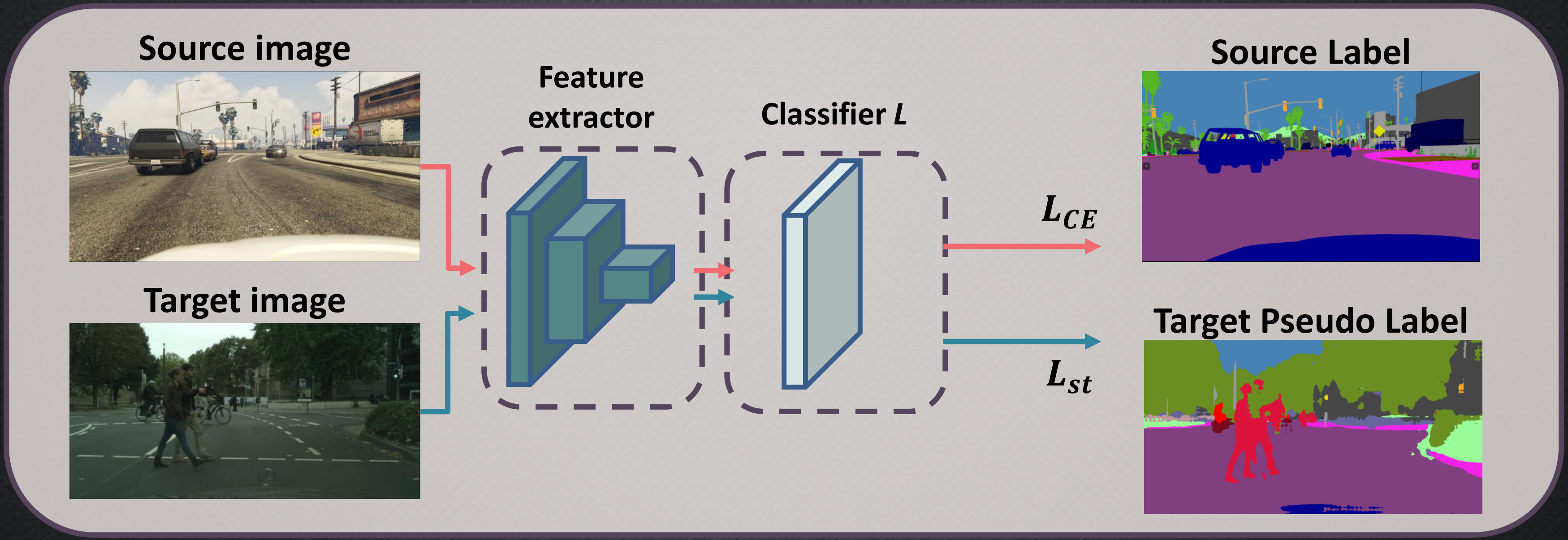
[1] Learning to Adapt Structured Output Space for Semantic Segmentation, Tsai et al., CVPR 2018
[2] ADVENT: Adversarial Entropy Minimization for Domain Adaptation in Semantic Segmentation, Vu et al., CVPR 2019
[3] Taking A Closer Look at Domain Shift: Category-level Adversaries for Semantics Consistent Domain Adaptation, Luo et al., CVPR 2019
[4] Domain Adaptation for Structured Output via Discriminative Patch Representations, Tsai et al., ICCV 2019
[5] Domain Adaptation for Semantic Segmentation with Maximum Squares Loss, Chen et al., ICCV 2019

Previous Domain Adaptation

Previous Domain Adaptation (UDA) consists of **three paradigms**:

Self-training based DA (L_{st})

- CBST[1], CRST[2]



[1] Unsupervised Domain Adaptation for Semantic Segmentation via Class-Balanced Self-Training, Zou et al., ECCV 2018

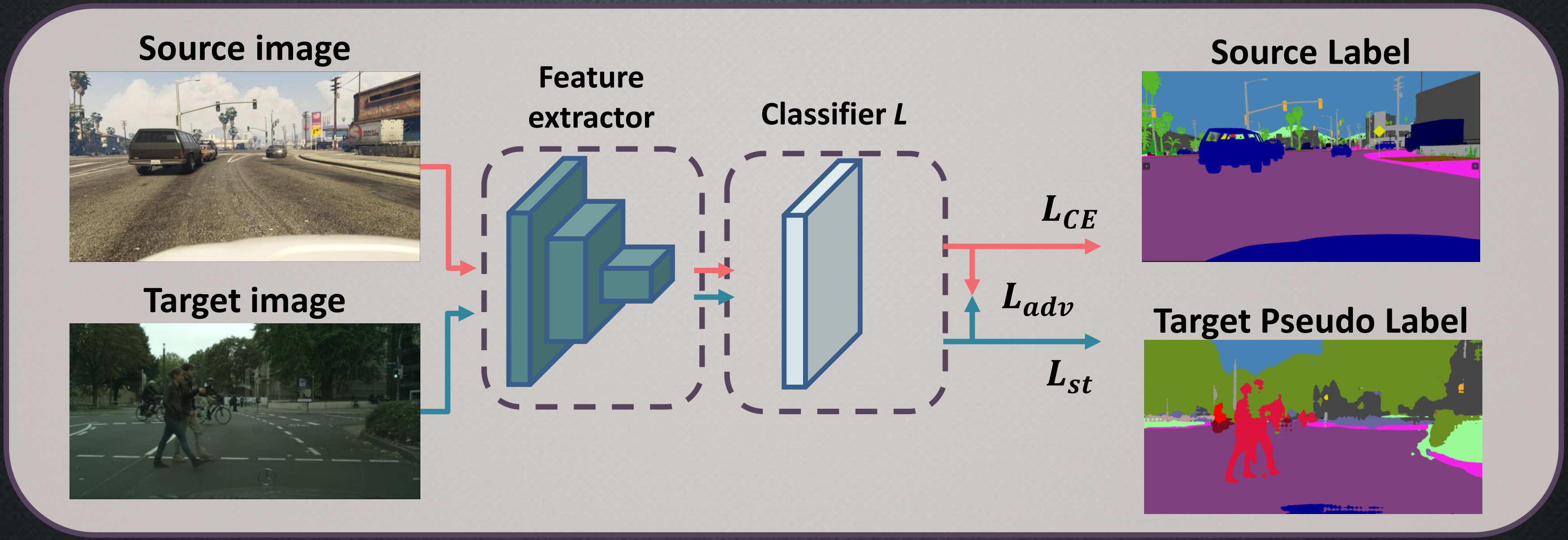
[2] Confidence Regularized Self-Training, Zou et al., ICCV 2019

Previous Domain Adaptation

Previous Domain Adaptation (UDA) consists of **three paradigms**:

Adversarial + Self-training based DA ($L_{adv} + L_{st}$)

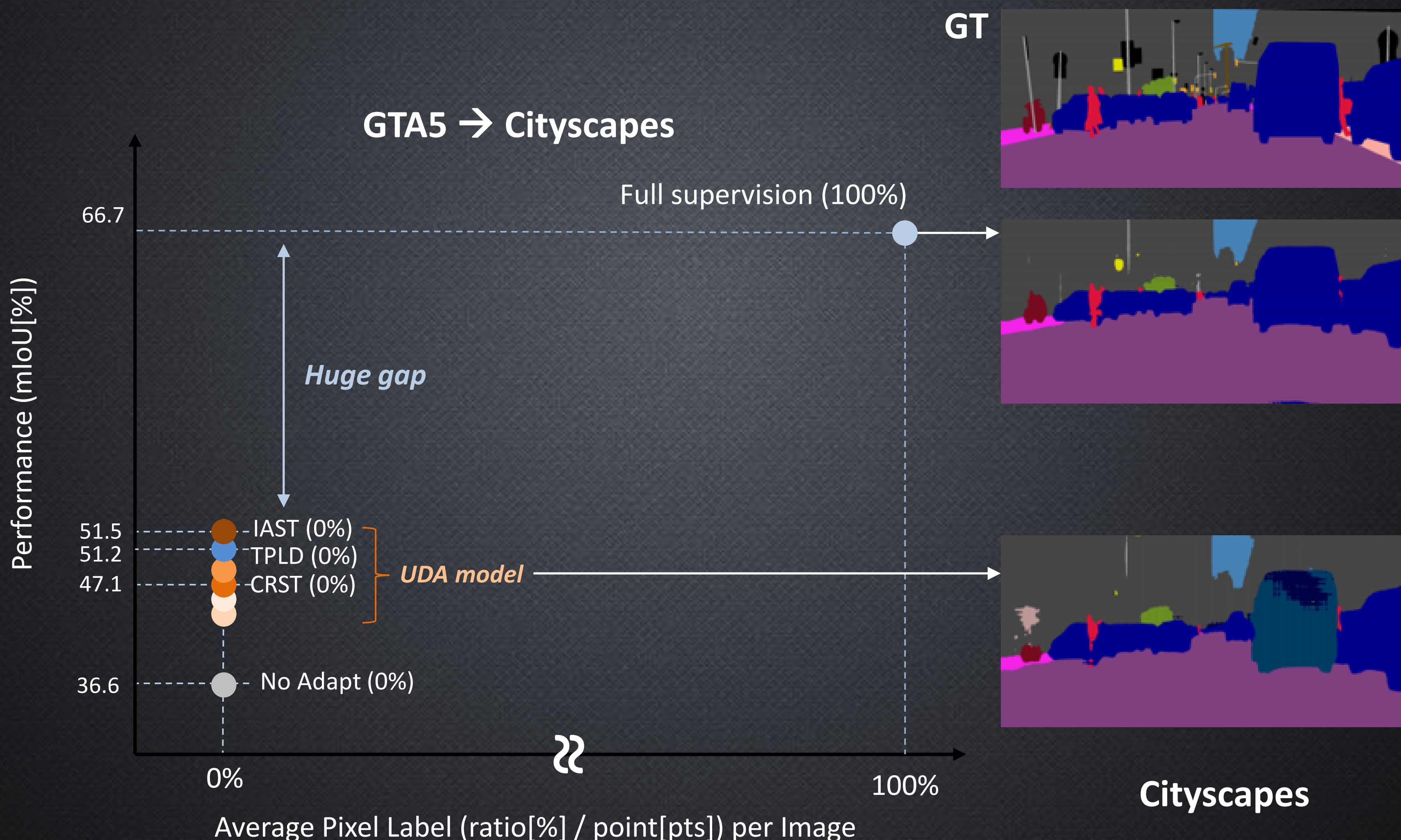
- BDL[1], IntraDA[2], SIM[3], TPLD[4], IAST[5]



[1] Bidirectional Learning for Domain Adaptation of Semantic Segmentation, Li et al., CVPR 2019
[2] Unsupervised Intra-domain Adaptation for Semantic Segmentation through Self-Supervision, Pan et al., CVPR 2020
[3] Differential Treatment for Stuff and Things: A Simple Unsupervised Domain Adaptation Method for Semantic Segmentation, Wang et al., CVPR 2020
[4] Two-phase Pseudo Label Densification for Self-training based Domain Adaptation, Shin et al., ECCV 2020
[5] Instance Adaptive Self-Training for Unsupervised Domain Adaptation, Mei et al., ECCV 2020

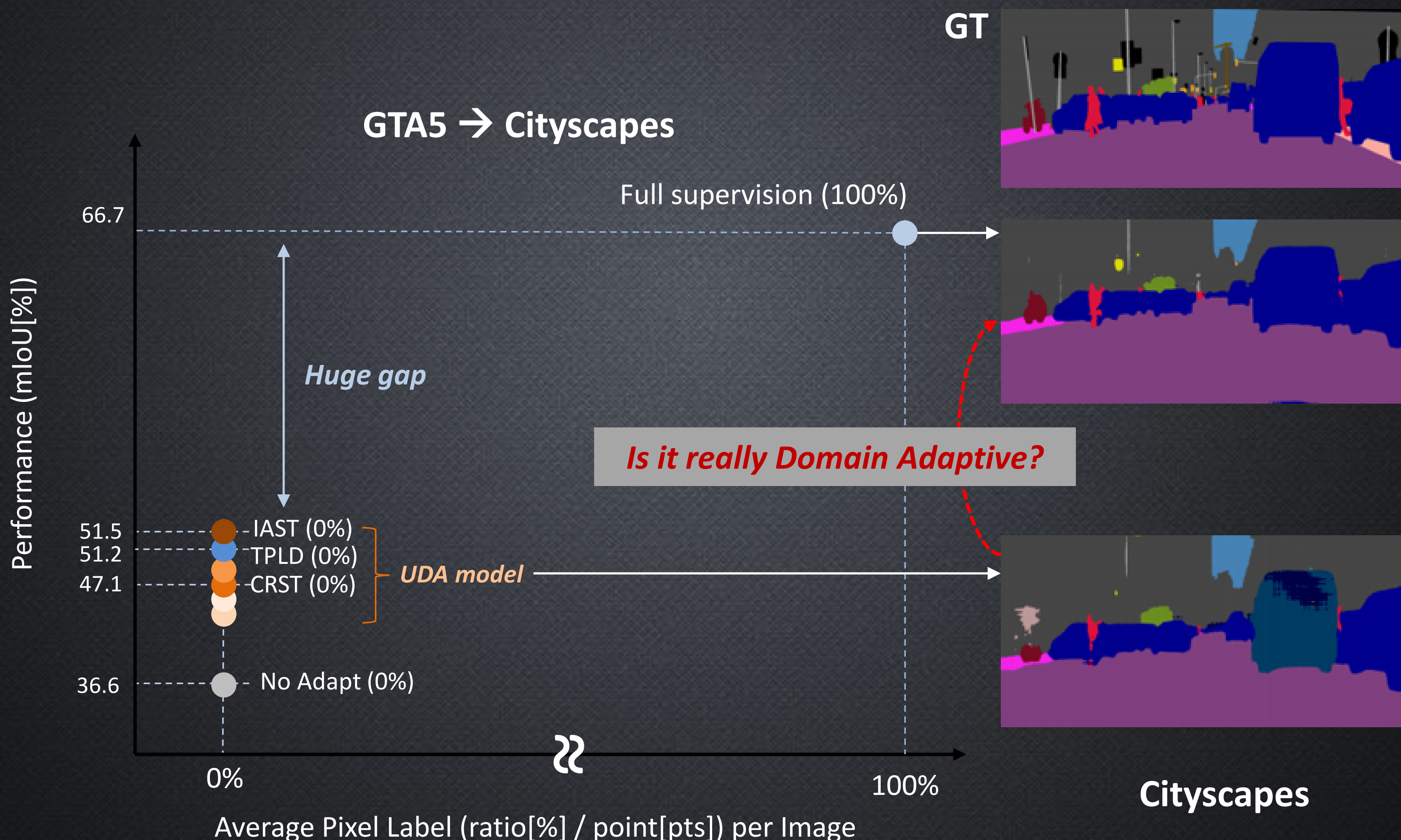
Previous Domain Adaptation

The performance limitations are clear as it still lags far behind the fully supervised model.



Previous Domain Adaptation

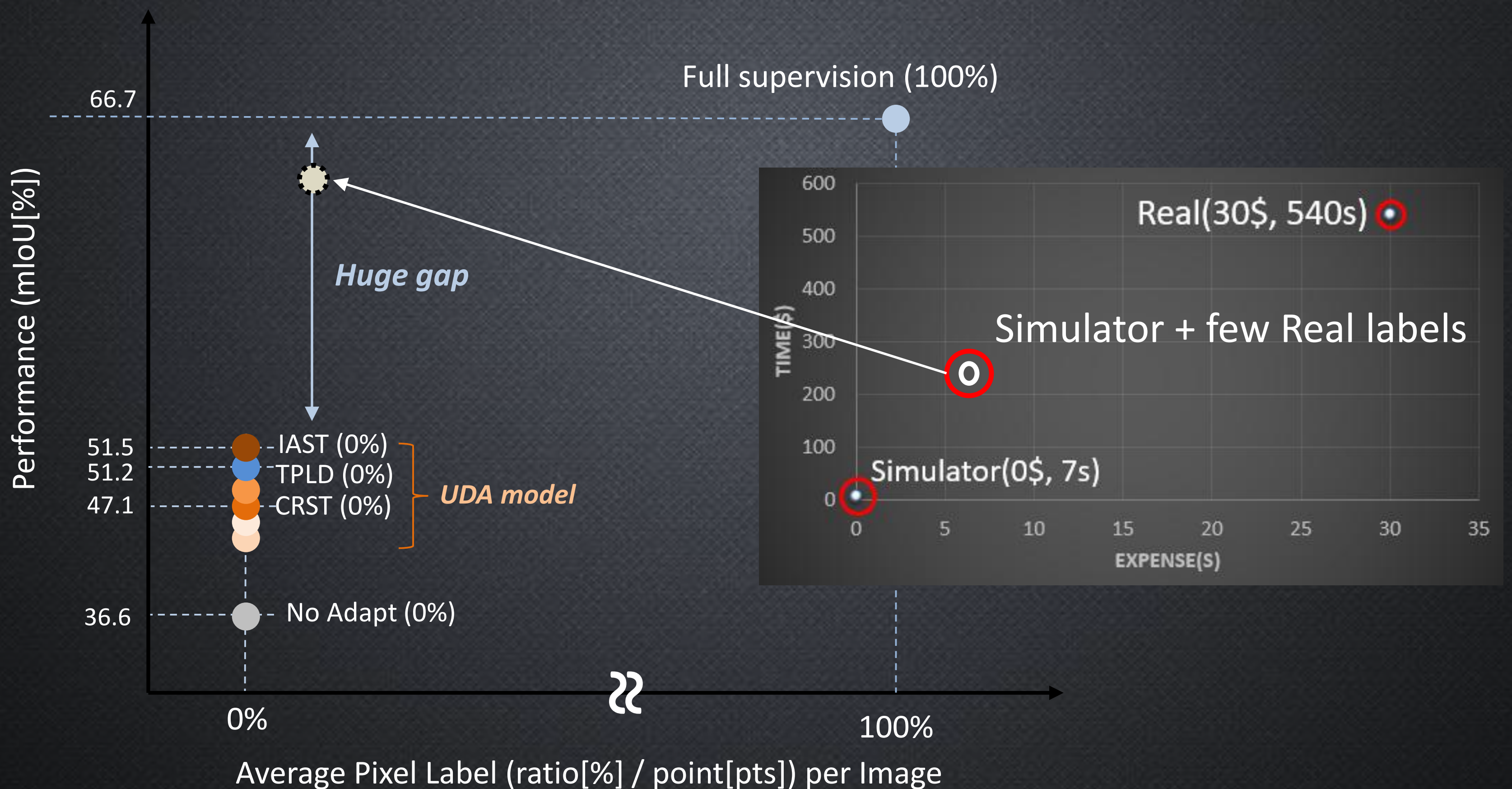
The performance limitations are clear as it still lags far behind the fully supervised model.



Domain Adaptation with real-data few labels

Assuming we are not able to obtain real-data labels is not practical.

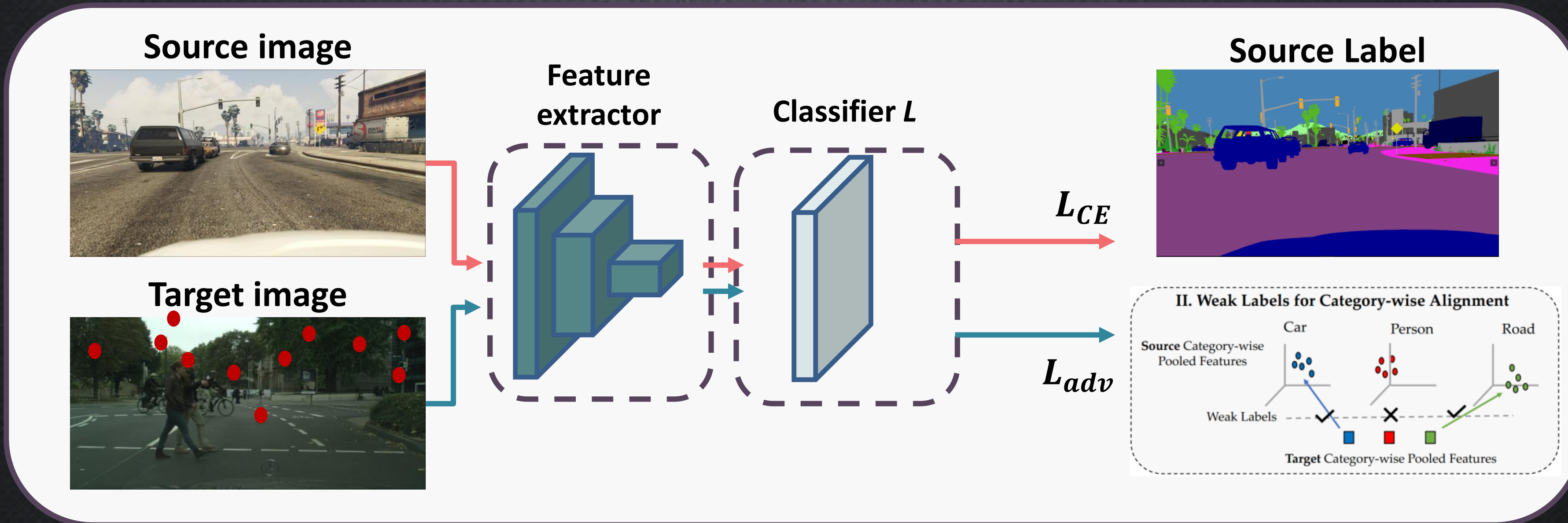
→ We need to find a way to boost the performance even with very few real data labels



Related work – Domain Adaptation with Weak Label

Weak Domain Adaptation (WDA)[1] uses point labels.

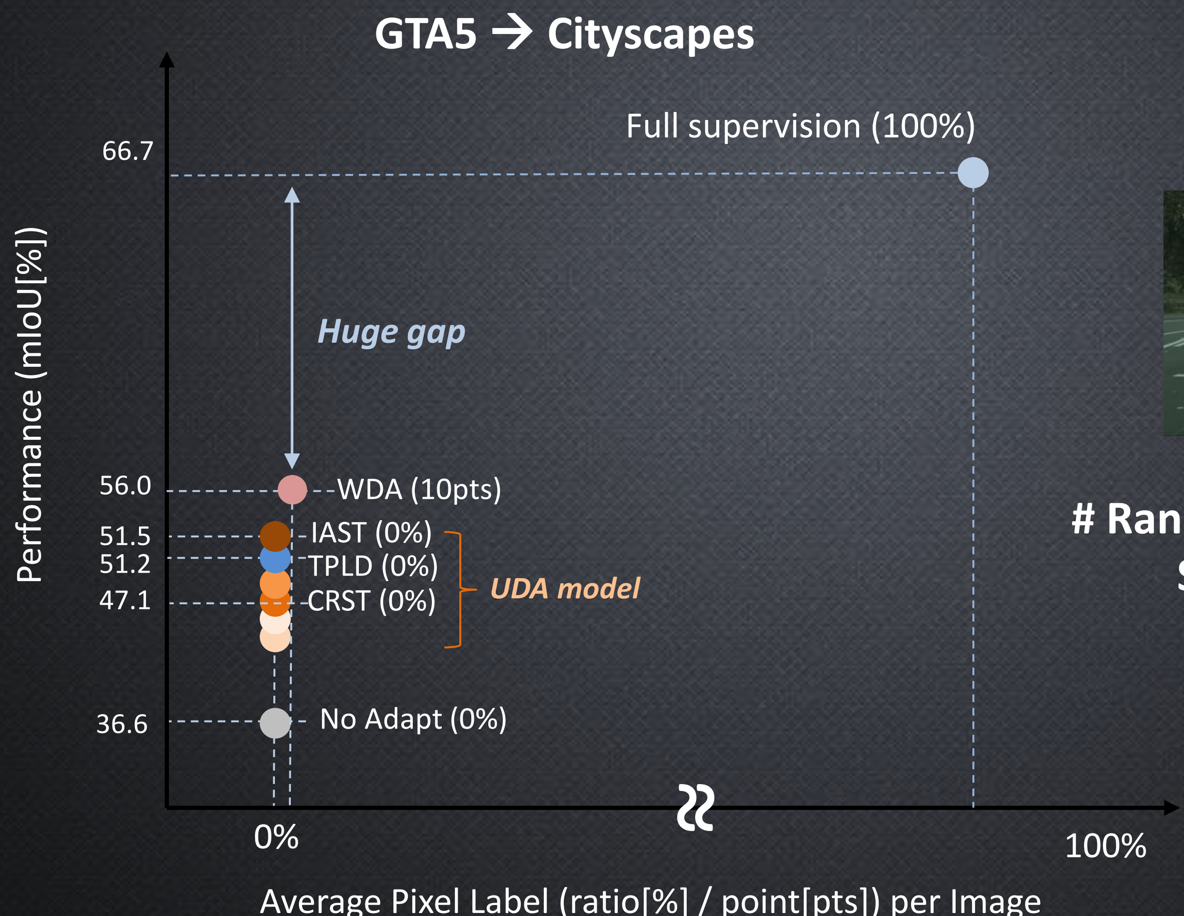
Point labels from each category (Human picks randomly)

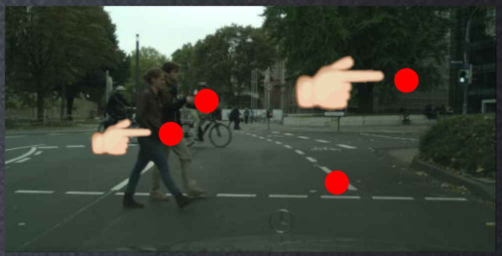


[1] Domain Adaptive Semantic Segmentation Using Weak Labels, Paul et al., ECCV 2020

Related work – Domain Adaptation with Weak Label

- Still lags far behind the fully-supervised model performance.
- It could be sensitive to human's randomness.





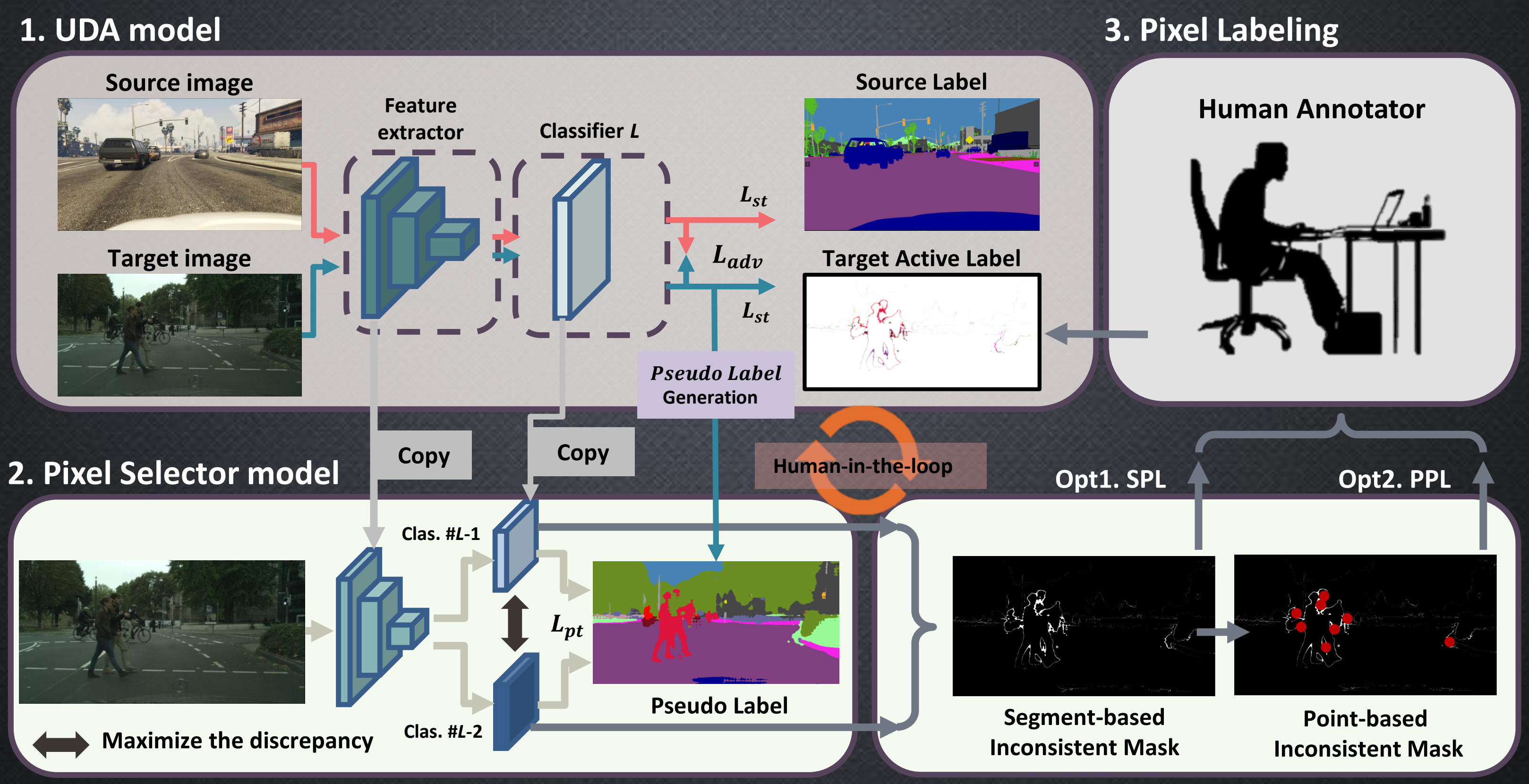
 # Random Pick in each category → Sensitive to randomness

Our method - LabOR

We propose Active Domain Adaptation, LabOR (Labeling Only if Require).

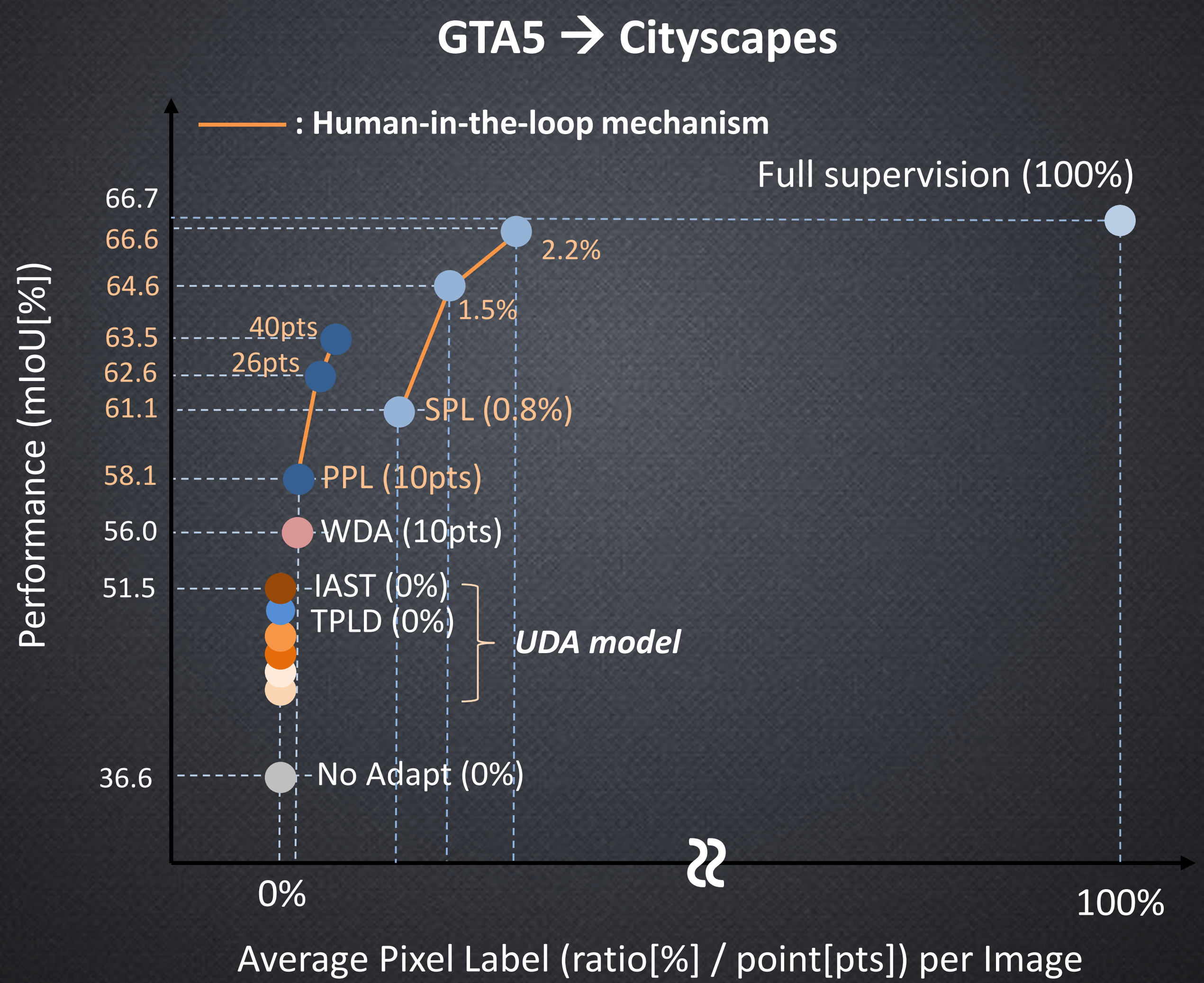
- model (Pixel Selector) lets us know where we need to label

- SPL and PPL method depending on how to labeling



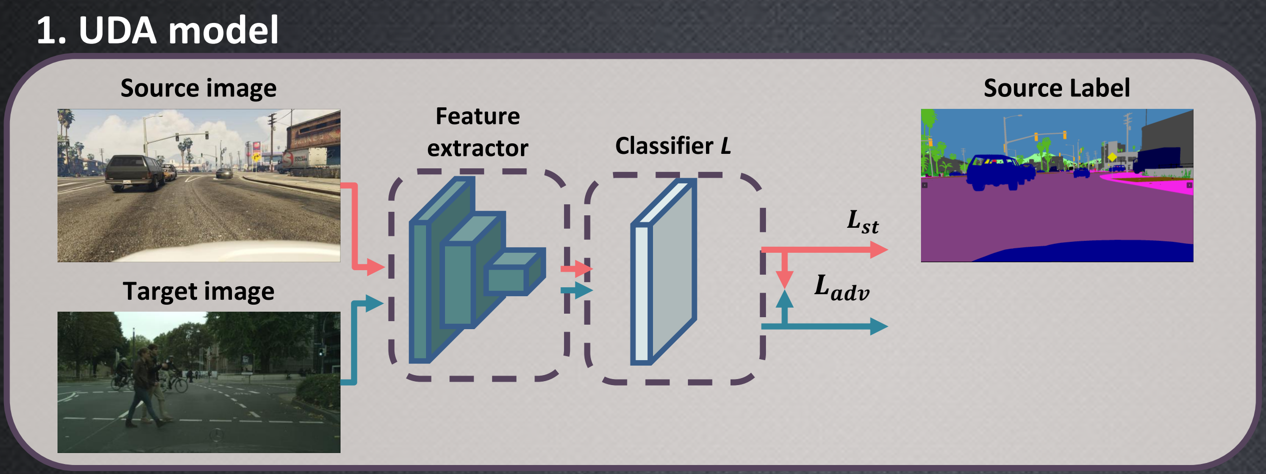
Our method - LabOR

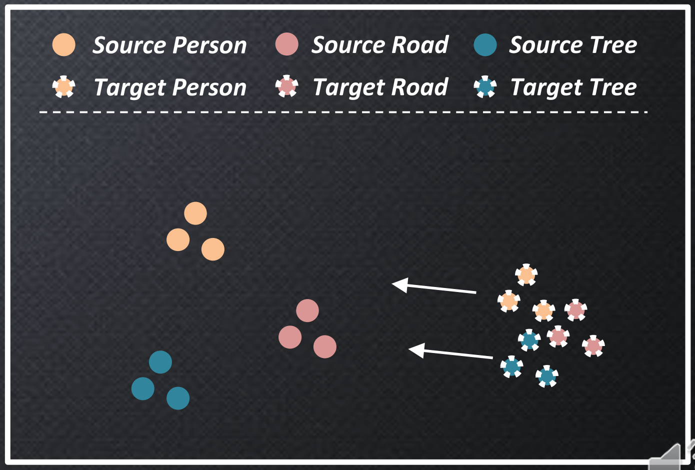
Our method achieves very near to supervised performance.



Our method - LabOR

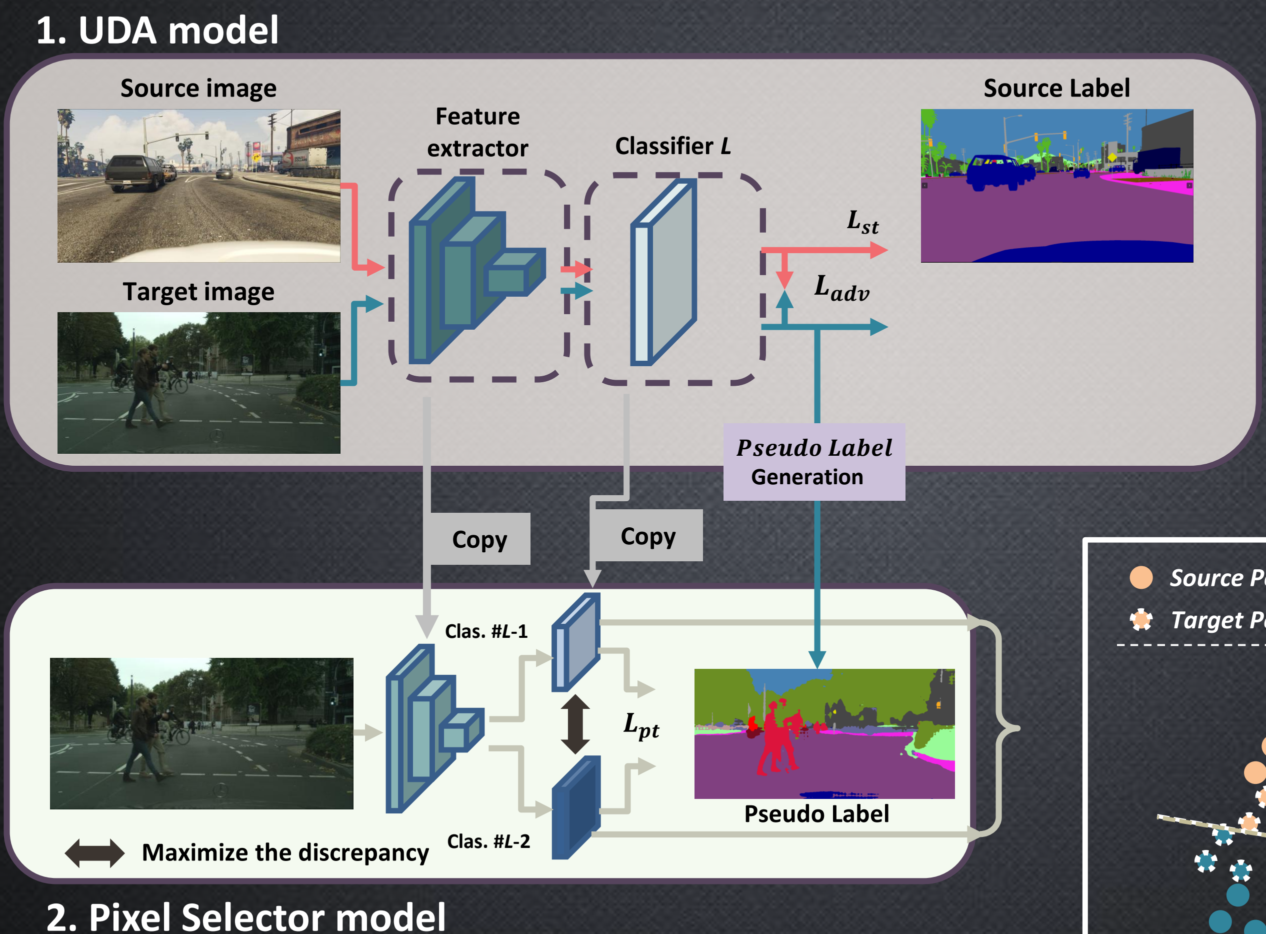
1. UDA model provides pseudo label for pixel selector

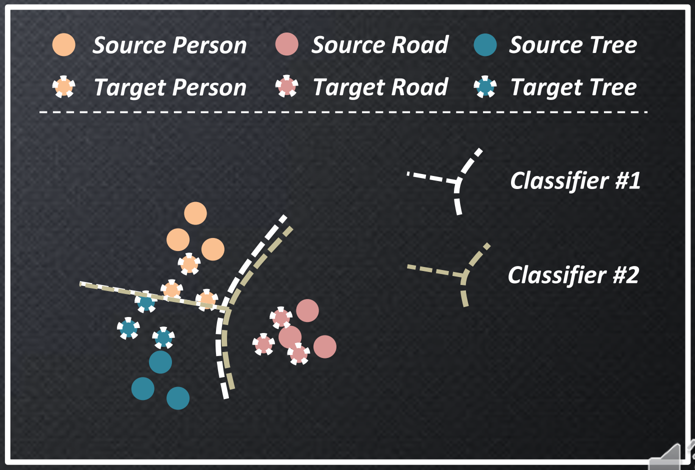




Our method - LabOR

1. UDA model provides pseudo label for pixel selector

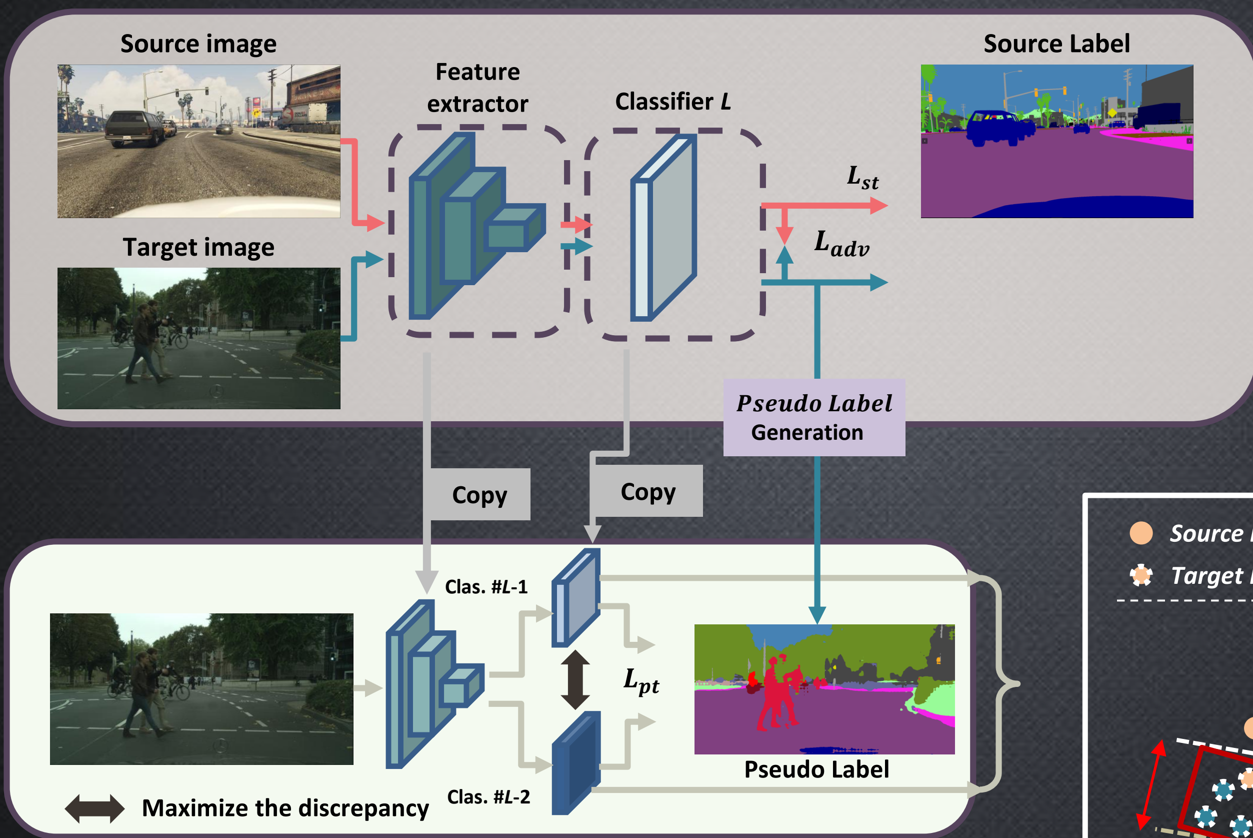




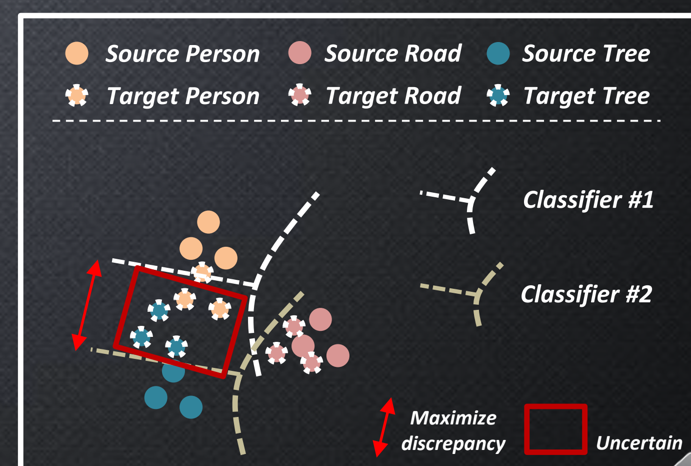
Our method - LabOR

2. Pixel selector model is trained to bring out inconsistent mask via maximizing classifiers' discrepancy

1. UDA model

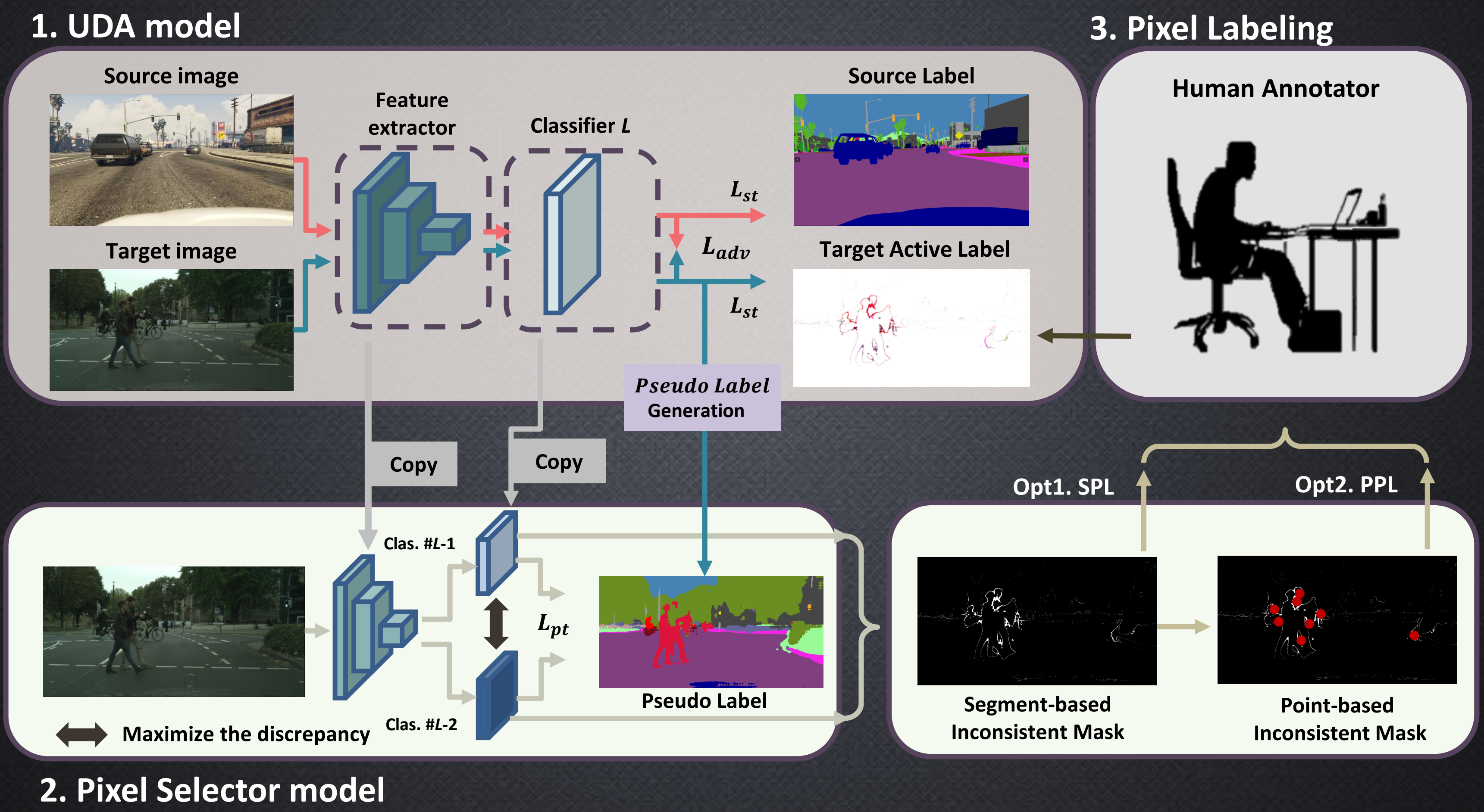


2. Pixel Selector model



Our method - LabOR

3. Based on inconsistent mask (SPL or PPL), conduct pixel labeling and training on target data.



PPL (Point based Pixel-Labeling)

Pseudo Label

×

SPL region

PPL (Point based Pixel-Labeling)

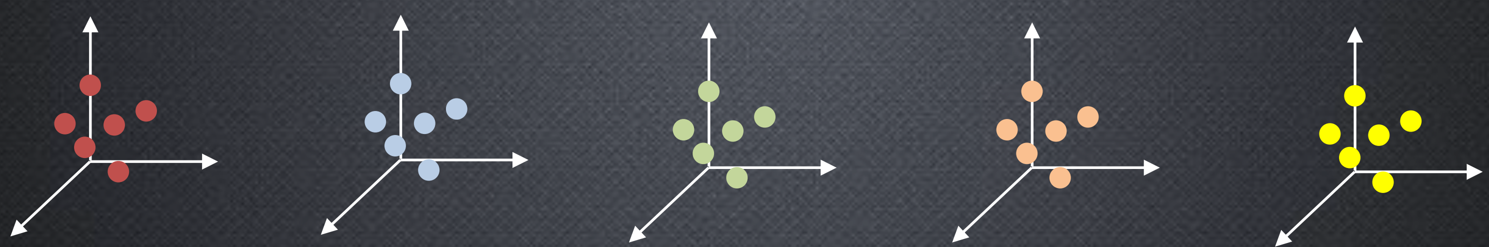
Pseudo Label

×

SPL region

Uncertain pixels D^k **for class *k* as follows**

D^1 **(ex., person)** D^2 D^3 D^4 D^5



• • •

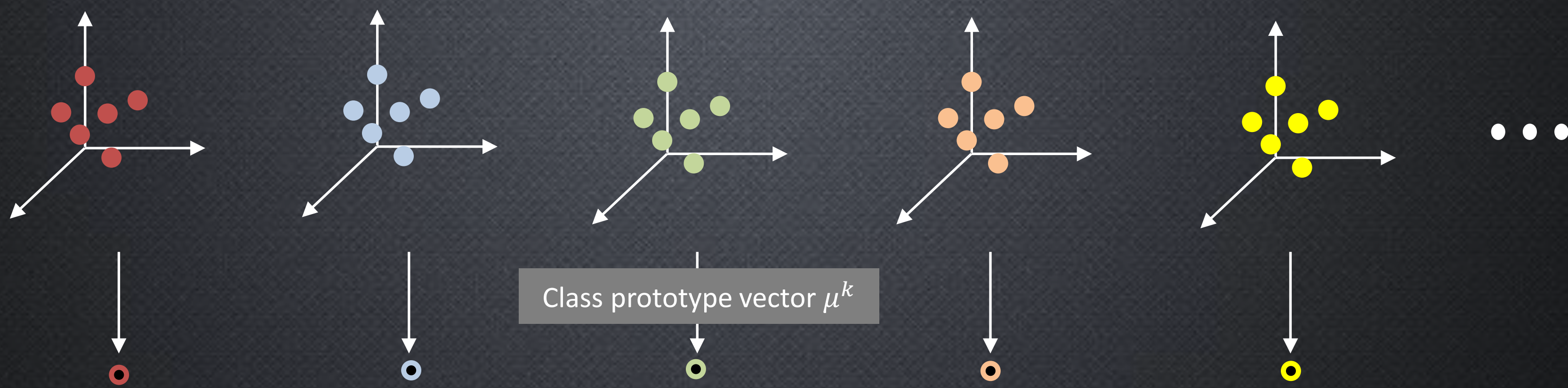
PPL (Point based Pixel-Labeling)

Pseudo Label

SPL region

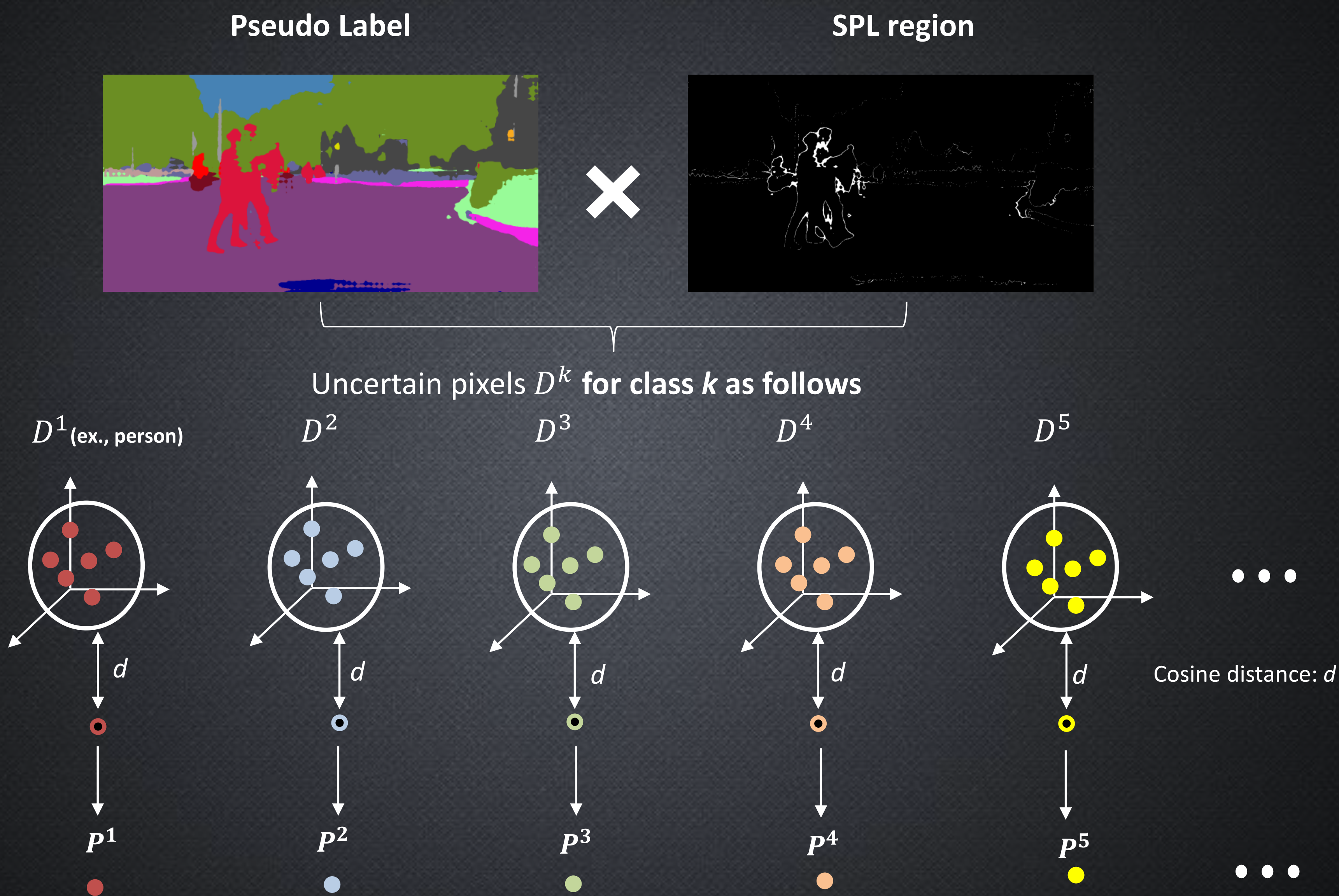
Uncertain pixels D^k **for class *k* as follows**

D^1 **(ex., person)** D^2 D^3 D^4 D^5



$$\mu^k = \frac{1}{|\mathcal{D}^k|} \sum_{(i,j)\in\mathcal{D}^k} \hat{\mathbf{Y}}_{i,j} \in \mathbb{R}^K$$

PPL (Point based Pixel-Labeling)

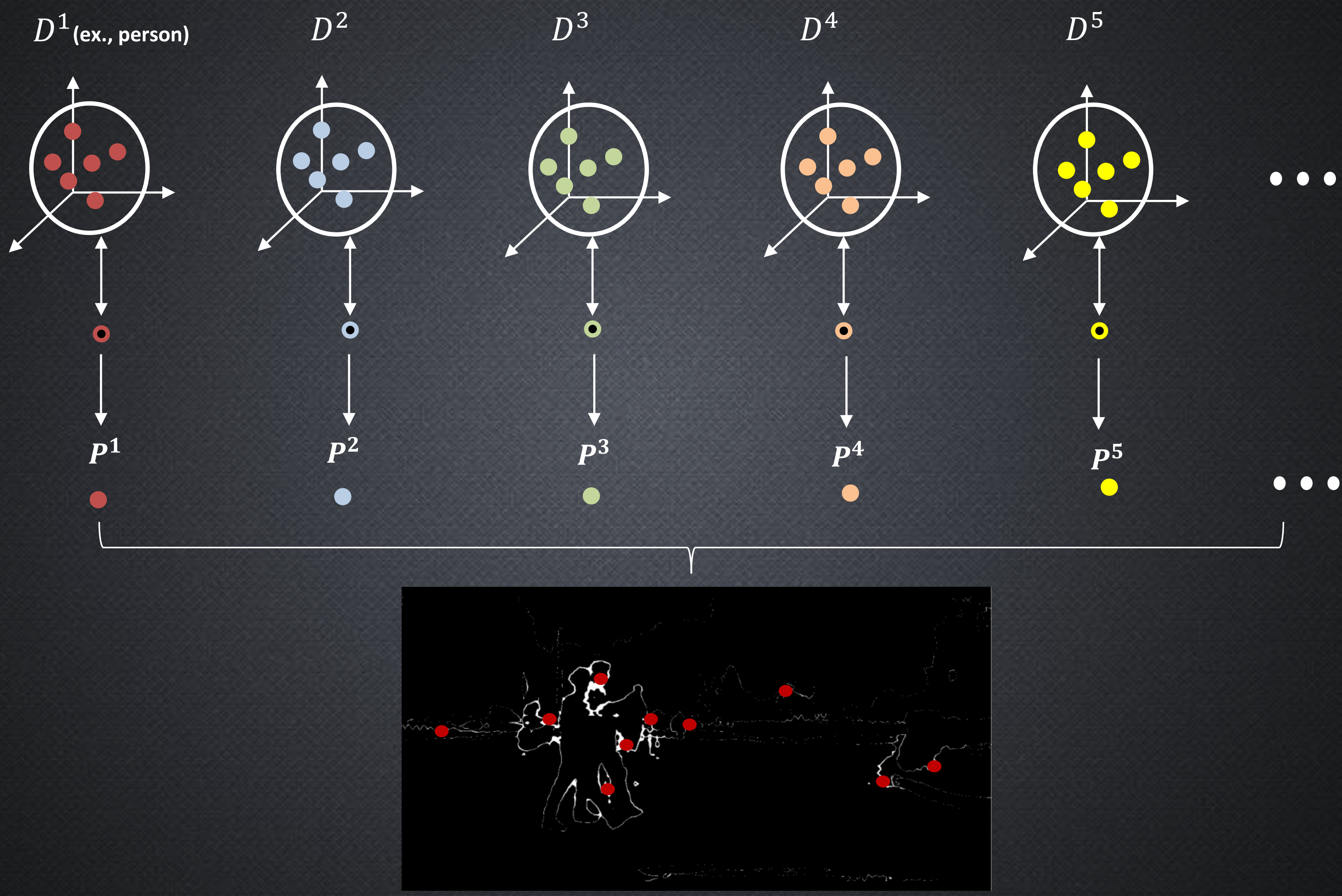


$$P(\mathbf{x}) = \left\{ \underset{(i,j)\in\mathcal{D}^k}{\arg\min}\, d\left(\mu^k, \hat{\mathbf{Y}}_{i,j}\right) \right\}_{k=1}^{K}$$

Selected points P

PPL (Point based Pixel-Labeling)

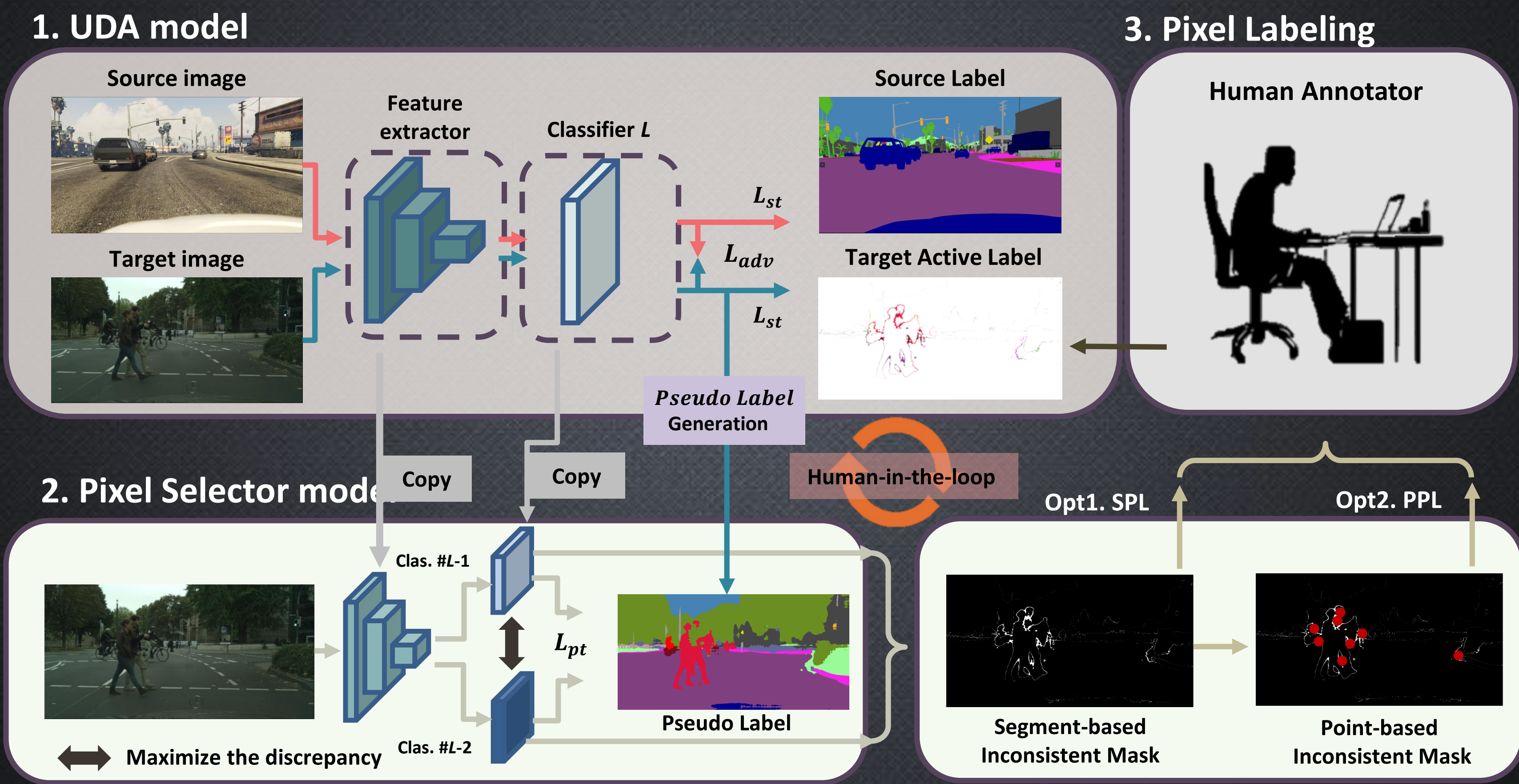
Uncertain pixels D^k **for class *k* as follows**



PPL

Our method - LabOR

This process is conducted iteratively as human-in-the-loop manner.



Our method - LabOR

Our Algorithm

Algorithm 1: Pixel Selector Model

Input: Source data $\mathcal{S}$, Target data $\mathcal{T}$, Initialized model $f_\theta \circ g_\phi(\cdot)$

Output: The model with adapted weights on target dataset $f_\theta \circ g_\phi(\cdot)$

begin

Pre-train the model on the source dataset.
And, initial adapt with adversarial learning.
$\theta, \phi \leftarrow \arg\min_{\theta,\phi} \mathcal{L}_s(\theta,\phi) + \mathcal{L}_{adv}(\theta,\phi)$ (Eq. (1))

for *3 Stages* **do**

Define auxiliary layers and copy weights
$\theta_1' \leftarrow \theta, \quad \theta_2' \leftarrow \theta, \quad \phi' \leftarrow \phi$
Apply self-training (Eq. (3))
$\phi', \theta_1', \theta_2' \leftarrow \arg\min\limits_{\phi', \theta_1', \theta_2'} \mathcal{L}_{\text{self}}(\phi', \theta_1', \theta_2')$
Maximize classifier discrepancy (Eq. (4))
$\phi', \theta_1', \theta_2' \leftarrow \arg\min\limits_{\phi'} \max\limits_{\theta_1', \theta_2'} \mathcal{L}_{\text{dis}}(\phi', \theta_1', \theta_2')$

for $\mathbf{x}_t \in \mathcal{T}$ **do**

Generate $M(\mathbf{x}_t; \phi', \theta_1', \theta_2')$ with Eq. (2).
if *SPL* **then**
Annotate inconsistency mask
$\tilde{\mathbf{Y}}_t(\mathbf{x}_t) \leftarrow M(\mathbf{x}_t) \odot \mathbf{Y}_t$
else if *PPL* **then**
Select representative points
$P(\mathbf{x}_t) = \text{SelectPt}(M, p(\mathbf{Y}|\mathbf{x}_t; \theta, \phi))$
Annotate the points
$\tilde{\mathbf{Y}}_t(\mathbf{x}_t) \leftarrow P(\mathbf{x}_t) \odot \mathbf{Y}_t$

Train the model with the pseudo labels
$\theta, \phi \leftarrow \arg\min_{\theta,\phi} \mathcal{L}_t(\theta,\phi)$ (Eq. (5))

Pretrain on source & Adversarial learning

$$\mathcal{L}_s(\theta,\phi) = \mathbb{E}_{(\mathbf{x}_s,\mathbf{Y}_s)\in\mathcal{S}}\big[\text{CE}(\mathbf{Y}_s, p(\mathbf{Y}|\mathbf{x}_s;\theta,\phi))\big]. \quad (1)$$

$$\mathcal{L}_{adv}(\theta,\phi) = \mathbb{E}_{\mathbf{x}_s\in\mathcal{S},\mathbf{x}_t\in\mathcal{T}}\big[\text{Adv}(p(\mathbf{x}_s;\theta,\phi), p(\mathbf{x}_t;\theta,\phi))\big].$$

Our method - LabOR

Our Algorithm

Algorithm 1: Pixel Selector Model

Input: Source data $\mathcal{S}$, Target data $\mathcal{T}$, Initialized model $f_\theta \circ g_\phi(\cdot)$

Output: The model with adapted weights on target dataset $f_\theta \circ g_\phi(\cdot)$

```
begin
    Pre-train the model on the source dataset.
    And, initial adapt with adversarial learning.
    θ, φ ← arg min_{θ,φ} L_s(θ, φ) + L_adv(θ, φ) (Eq. (1))
    for 3 Stages do
        Define auxiliary layers and copy weights
        θ'_1 ← θ,  θ'_2 ← θ,  φ' ← φ
        Apply self-training (Eq. (3))
        φ', θ'_1, θ'_2 ← arg min_{φ',θ'_1,θ'_2} L_self(φ', θ'_1, θ'_2)
        Maximize classifier discrepancy (Eq. (4))
        φ', θ'_1, θ'_2 ← arg min_{φ'} max_{θ'_1,θ'_2} L_dis(φ', θ'_1, θ'_2)
        for x_t ∈ T do
            Generate M(x_t; φ', θ'_1, θ'_2) with Eq. (2).
            if SPL then
                Annotate inconsistency mask
                Ỹ_t(x_t) ← M(x_t) ⊙ Y_t
            else if PPL then
                Select representative points
                P(x_t) = SelectPt(M, p(Y|x_t; θ, φ))
                Annotate the points
                Ỹ_t(x_t) ← P(x_t) ⊙ Y_t
        Train the model with the pseudo labels
        θ, φ ← arg min_{θ,φ} L_t(θ, φ) (Eq. (5))
```

Pretrain on source & Adversarial learning

$$\mathcal{L}_s(\theta,\phi) = \mathbb{E}_{(\mathbf{x}_s,\mathbf{Y}_s)\in\mathcal{S}}\big[\mathrm{CE}(\mathbf{Y}_s, p(\mathbf{Y}|\mathbf{x}_s;\theta,\phi))\big]. \tag{1}$$

$$\mathcal{L}_{adv}(\theta,\phi) = \mathbb{E}_{\mathbf{x}_s\in\mathcal{S},\mathbf{x}_t\in\mathcal{T}}\big[\mathrm{Adv}(p(\mathbf{x}_s;\theta,\phi), p(\mathbf{x}_t;\theta,\phi))\big].$$

Maximize Classifiers' Discrepancy

$$\begin{aligned}\mathcal{L}_{\text{self}}&(\phi', \theta'_1, \theta'_2) \\ &= \mathbb{E}_{\mathbf{x}_t\in\mathcal{T}}\big[\mathrm{CE}(\arg\max_K \hat{\mathbf{Y}}_t, p(\mathbf{Y}|\mathbf{x}_t;\theta'_1,\phi')) \\ &\qquad + \mathrm{CE}(\arg\max_K \hat{\mathbf{Y}}_t, p(\mathbf{Y}|\mathbf{x}_t;\theta'_2,\phi'))\big].\end{aligned} \tag{3}$$

$$\begin{aligned}&\min_{\phi'}\max_{\theta'_1,\theta'_2}\mathcal{L}_{\text{dis}}(\phi', \theta'_1, \theta'_2) \\ &= \min_{\phi'}\max_{\theta'_1,\theta'_2}\mathbb{E}_{\mathbf{x}_t\in\mathcal{T}}\Big[||f_{\theta'_1}\circ g_{\phi'}(\mathbf{x}_t) - f_{\theta'_2}\circ g_{\phi'}(\mathbf{x}_t)||_1\Big].\end{aligned} \tag{4}$$

Our method - LabOR

Our Algorithm

Algorithm 1: Pixel Selector Model

Input: Source data $\mathcal{S}$, Target data $\mathcal{T}$, Initialized model $f_\theta \circ g_\phi(\cdot)$

Output: The model with adapted weights on target dataset $f_\theta \circ g_\phi(\cdot)$

begin

Pre-train the model on the source dataset.

And, initial adapt with adversarial learning.

$\theta, \phi \leftarrow \arg\min_{\theta,\phi} \mathcal{L}_s(\theta, \phi) + \mathcal{L}_{adv}(\theta, \phi)$ (Eq. (1))

for *3 Stages* **do**

Define auxiliary layers and copy weights

$\theta_1^{'} \leftarrow \theta, \quad \theta_2^{'} \leftarrow \theta, \quad \phi^{'} \leftarrow \phi$

Apply self-training (Eq. (3))

$\phi^{'}, \theta_1^{'}, \theta_2^{'} \leftarrow \arg\min_{\phi^{'}, \theta_1^{'}, \theta_2^{'}} \mathcal{L}_{\text{self}}(\phi^{'}, \theta_1^{'}, \theta_2^{'})$

Maximize classifier discrepancy (Eq. (4))

$\phi^{'}, \theta_1^{'}, \theta_2^{'} \leftarrow \arg\min_{\phi^{'}} \max_{\theta_1^{'}, \theta_2^{'}} \mathcal{L}_{\text{dis}}(\phi^{'}, \theta_1^{'}, \theta_2^{'})$

for $\mathbf{x}_t \in \mathcal{T}$ **do**

Generate $M(\mathbf{x}_t; \phi^{'}, \theta_1^{'}, \theta_2^{'})$ with Eq. (2).

if *SPL* **then**

Annotate inconsistency mask

$\tilde{\mathbf{Y}}_t(\mathbf{x}_t) \leftarrow M(\mathbf{x}_t) \odot \mathbf{Y}_t$

else if *PPL* **then**

Select representative points

$P(\mathbf{x}_t) = \text{SelectPt}(M, p(\mathbf{Y}|\mathbf{x}_t; \theta, \phi))$

Annotate the points

$\tilde{\mathbf{Y}}_t(\mathbf{x}_t) \leftarrow P(\mathbf{x}_t) \odot \mathbf{Y}_t$

Train the model with the pseudo labels

$\theta, \phi \leftarrow \arg\min_{\theta,\phi} \mathcal{L}_t(\theta, \phi)$ (Eq. (5))

Pretrain on source & Adversarial learning

$$\mathcal{L}_s(\theta, \phi) = \mathbb{E}_{(\mathbf{x}_s, \mathbf{Y}_s) \in \mathcal{S}} \left[\text{CE}(\mathbf{Y}_s, p(\mathbf{Y}|\mathbf{x}_s; \theta, \phi)) \right]. \qquad (1)$$

$$\mathcal{L}_{adv}(\theta, \phi) = \mathbb{E}_{\mathbf{x}_s \in \mathcal{S}, \mathbf{x}_t \in \mathcal{T}} \left[\text{Adv}(p(\mathbf{x}_s; \theta, \phi), p(\mathbf{x}_t; \theta, \phi)) \right].$$

Maximize Classifiers' Discrepancy

$$\begin{aligned} &\mathcal{L}_{\text{self}}(\phi^{'}, \theta_1^{'}, \theta_2^{'}) \\ &= \mathbb{E}_{\mathbf{x}_t \in \mathcal{T}} \Big[\text{CE}(\arg\max_K \hat{\mathbf{Y}}_t, p(\mathbf{Y}|\mathbf{x}_t; \theta_1^{'}, \phi^{'})) \\ &\qquad + \text{CE}(\arg\max_K \hat{\mathbf{Y}}_t, p(\mathbf{Y}|\mathbf{x}_t; \theta_2^{'}, \phi^{'})) \Big]. \end{aligned} \qquad (3)$$

$$\begin{aligned} &\min_{\phi^{'}} \max_{\theta_1^{'}, \theta_2^{'}} \mathcal{L}_{\text{dis}}(\phi^{'}, \theta_1^{'}, \theta_2^{'}) \\ &= \min_{\phi^{'}} \max_{\theta_1^{'}, \theta_2^{'}} \mathbb{E}_{\mathbf{x}_t \in \mathcal{T}} \left[|| f_{\theta_1^{'}} \circ g_{\phi^{'}}(\mathbf{x}_t) - f_{\theta_2^{'}} \circ g_{\phi^{'}}(\mathbf{x}_t) ||_1 \right]. \end{aligned} \qquad (4)$$

Pixel Labeling (SPL or PPL)

- SPL on all inconsistent mask
- PPL in SelectPt [1] from inconsistent mask

Our method - LabOR

Our Algorithm

Algorithm 1: Pixel Selector Model

Input: Source data $\mathcal{S}$, Target data $\mathcal{T}$, Initialized model $f_\theta \circ g_\phi(\cdot)$

Output: The model with adapted weights on target dataset $f_\theta \circ g_\phi(\cdot)$

```
begin
    Pre-train the model on the source dataset.
    And, initial adapt with adversarial learning.
    θ, φ ← arg min_{θ,φ} L_s(θ, φ) + L_adv(θ, φ) (Eq. (1))
    for 3 Stages do
        Define auxiliary layers and copy weights
        θ'_1 ← θ,  θ'_2 ← θ,  φ' ← φ
        Apply self-training (Eq. (3))
        φ', θ'_1, θ'_2 ← arg min_{φ',θ'_1,θ'_2} L_self(φ', θ'_1, θ'_2)
        Maximize classifier discrepancy (Eq. (4))
        φ', θ'_1, θ'_2 ← arg min_{φ'} max_{θ'_1,θ'_2} L_dis(φ', θ'_1, θ'_2)
        for x_t ∈ T do
            Generate M(x_t; φ', θ'_1, θ'_2) with Eq. (2).
            if SPL then
                Annotate inconsistency mask
                Ỹ_t(x_t) ← M(x_t) ⊙ Y_t
            else if PPL then
                Select representative points
                P(x_t) = SelectPt(M, p(Y|x_t; θ, φ))
                Annotate the points
                Ỹ_t(x_t) ← P(x_t) ⊙ Y_t
        Train the model with the pseudo labels
        θ, φ ← arg min_{θ,φ} L_t(θ, φ) (Eq. (5))
```

Pretrain on source & Adversarial learning

$$\mathcal{L}_s(\theta, \phi) = \mathbb{E}_{(\mathbf{x}_s, \mathbf{Y}_s)\in\mathcal{S}}\big[\mathrm{CE}(\mathbf{Y}_s, p(\mathbf{Y}|\mathbf{x}_s; \theta, \phi))\big]. \tag{1}$$

$$\mathcal{L}_{adv}(\theta, \phi) = \mathbb{E}_{\mathbf{x}_s\in\mathcal{S}, \mathbf{x}_t\in\mathcal{T}}\big[\mathrm{Adv}(p(\mathbf{x}_s; \theta, \phi), p(\mathbf{x}_t; \theta, \phi))\big].$$

Maximize Classifiers' Discrepancy

$$\begin{aligned}&\mathcal{L}_{\text{self}}(\phi', \theta'_1, \theta'_2)\\ &= \mathbb{E}_{\mathbf{x}_t\in\mathcal{T}}\Big[\mathrm{CE}(\arg\max_K \hat{\mathbf{Y}}_t, p(\mathbf{Y}|\mathbf{x}_t; \theta'_1, \phi'))\\ &\qquad + \mathrm{CE}(\arg\max_K \hat{\mathbf{Y}}_t, p(\mathbf{Y}|\mathbf{x}_t; \theta'_2, \phi'))\Big].\end{aligned} \tag{3}$$

$$\begin{aligned}&\min_{\phi'}\max_{\theta'_1,\theta'_2}\mathcal{L}_{\text{dis}}(\phi', \theta'_1, \theta'_2)\\ &= \min_{\phi'}\max_{\theta'_1,\theta'_2}\mathbb{E}_{\mathbf{x}_t\in\mathcal{T}}\Big[||f_{\theta'_1}\circ g_{\phi'}(\mathbf{x}_t) - f_{\theta'_2}\circ g_{\phi'}(\mathbf{x}_t)||_1\Big].\end{aligned} \tag{4}$$

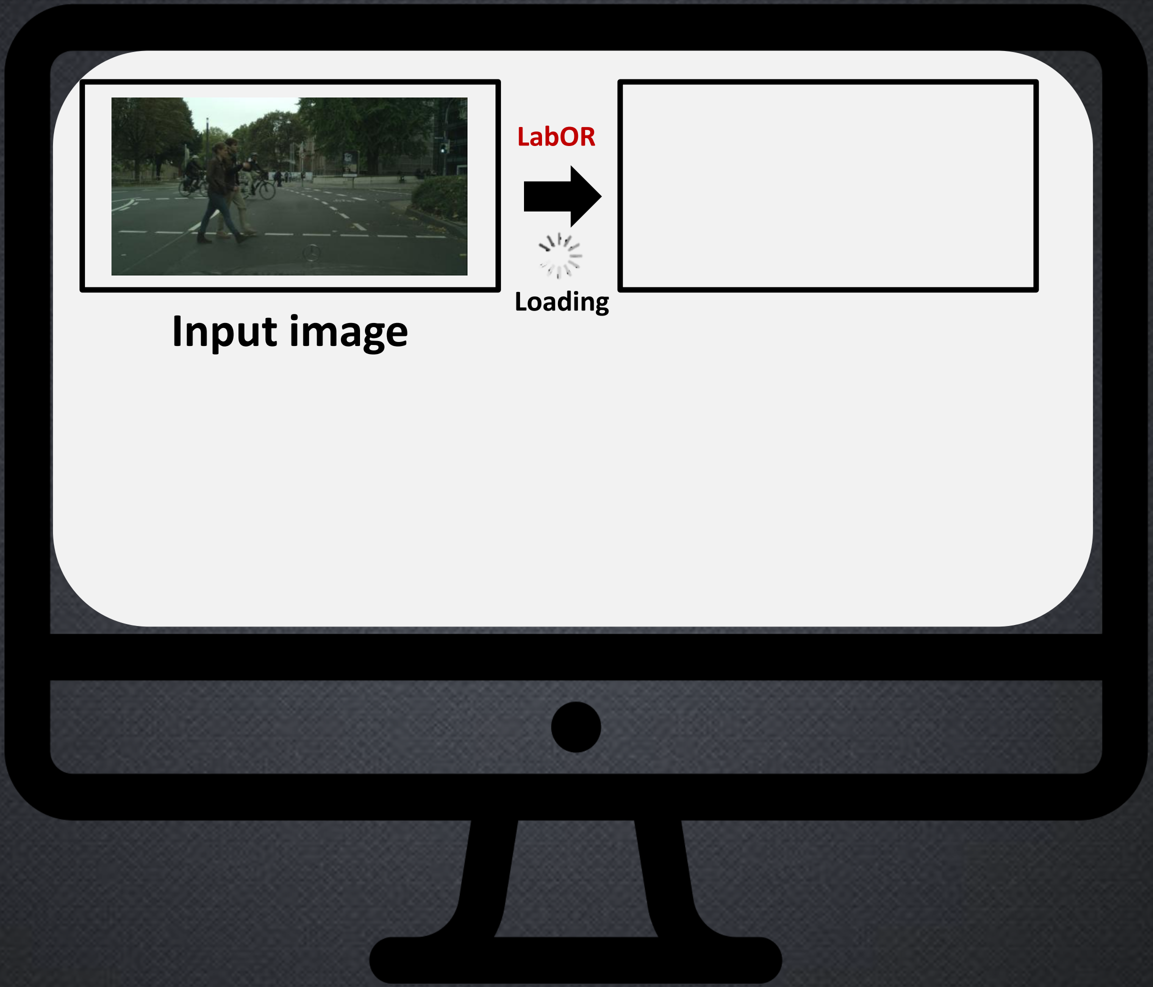
Pixel Labeling (SPL or PPL)

- SPL on all inconsistent mask(M)
- PPL in SelectPt [1] from inconsistent mask

Training with selected target label

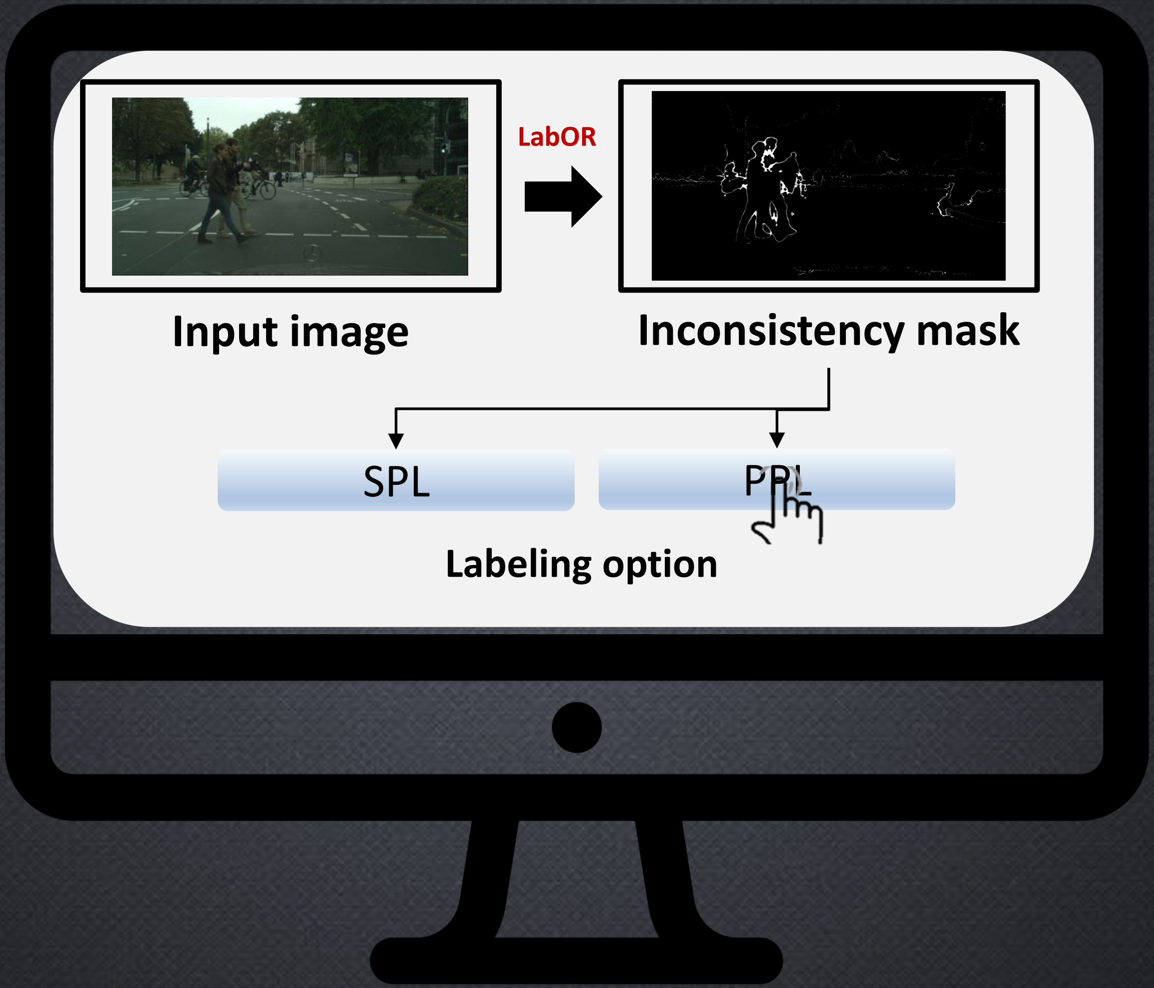
$$\mathcal{L}_t(\theta, \phi) = \mathbb{E}_{\mathbf{x}_t\in\mathcal{T}}\big[\mathrm{CE}(\mathbf{Y}_t(\mathbf{x}_t), p(\mathbf{Y}|\mathbf{x}_t; \theta, \phi))\big] \tag{5}$$

Labeling Procedure



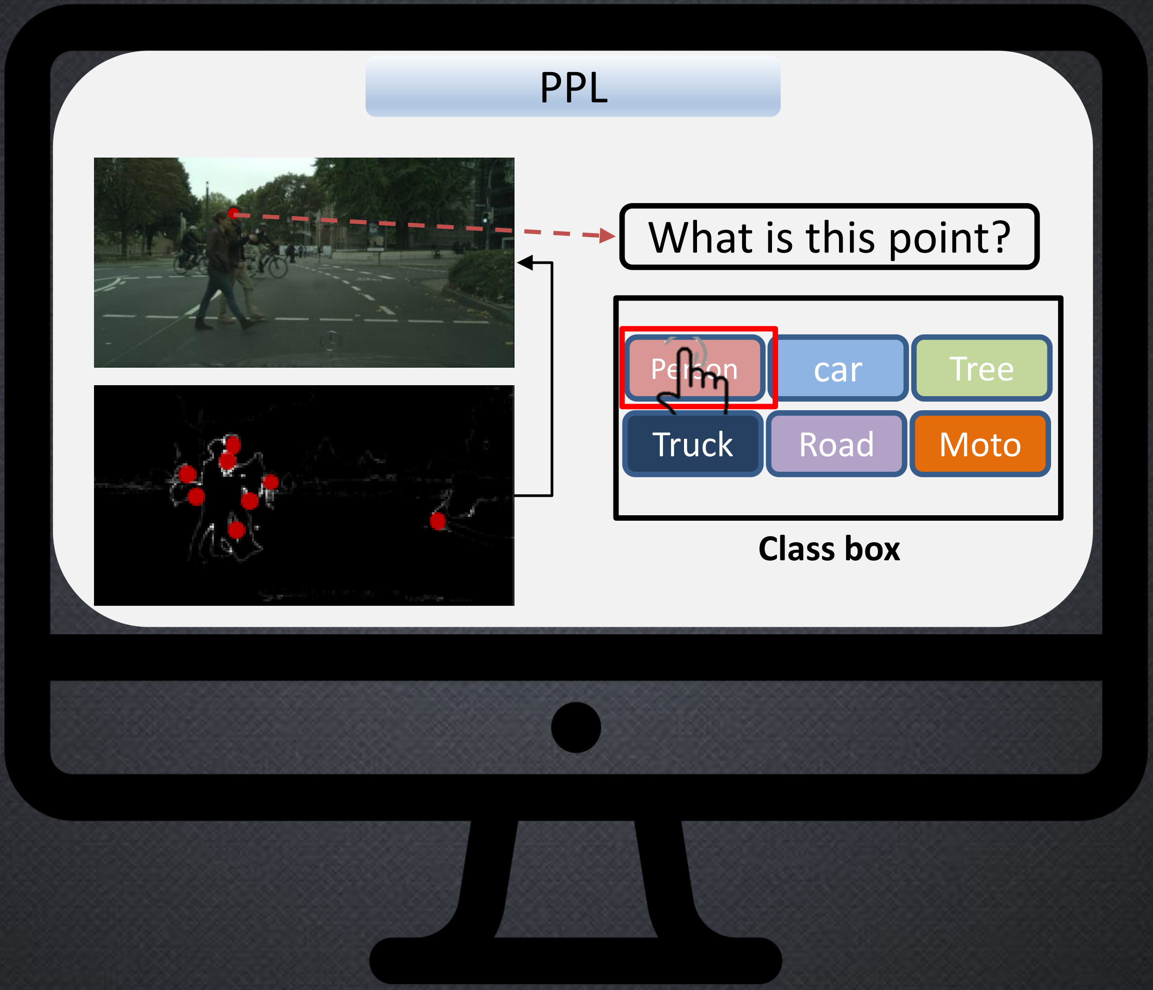
<LabOR-embedded labeling tool>

Labeling Procedure



<LabOR-embedded labeling tool>

Labeling Procedure



<LabOR-embedded labeling tool>

ICCV 2021 OCTOBER 11-17 VIRTUAL

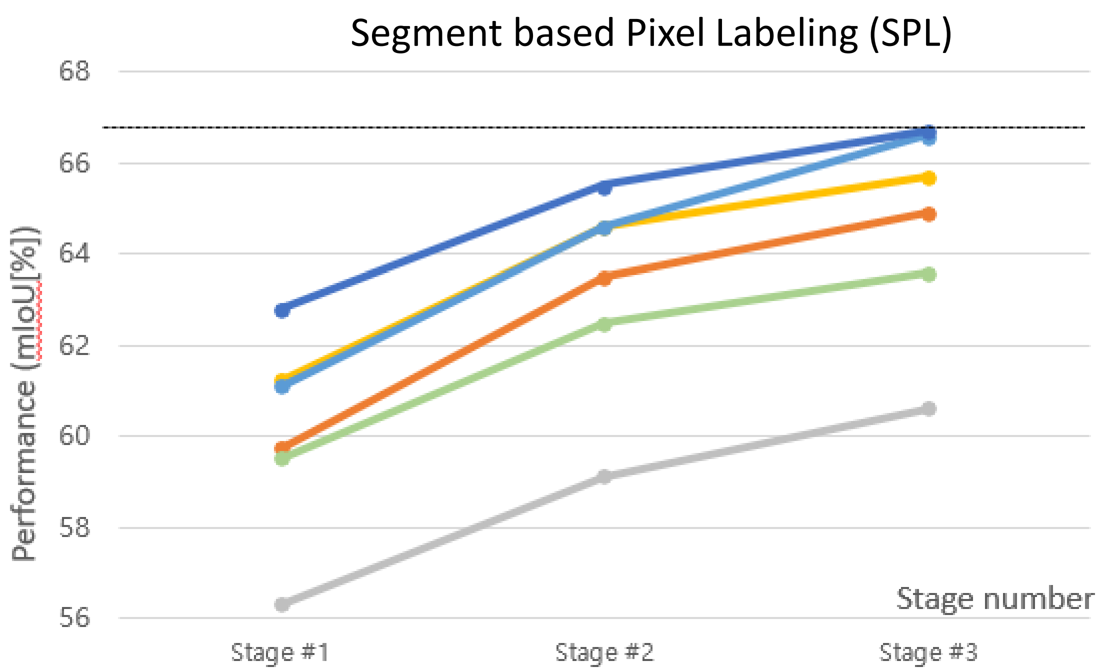
Our result - LabOR

Our method achieves very near to fully supervised model performance.

GTA5 → Cityscapes																				
Method	Road	SW	Build	Wall	Fence	Pole	TL	TS	Veg.	Terrain	Sky	PR	Rider	Car	Truck	Bus	Train	Motor	Bike	mIoU
No Adapt	75.8	16.8	77.2	12.5	21.0	25.5	30.1	20.1	81.3	24.6	70.3	53.8	26.4	49.9	17.2	25.9	6.5	25.3	36.0	36.6
AdaptSegNet [43]	86.5	36.0	79.9	23.4	23.3	35.2	14.8	14.8	83.4	33.3	75.6	58.5	27.6	73.7	32.5	35.4	3.9	30.1	28.1	42.4
ADVENT [44]	89.9	36.5	81.2	29.2	25.2	28.5	32.3	22.4	83.9	34.0	77.1	57.4	27.9	83.7	29.4	39.1	1.5	28.4	23.3	43.8
SIMDA [46]	90.6	44.7	84.8	34.3	28.7	31.6	35.0	37.6	84.7	43.3	85.3	57.0	31.5	83.8	42.6	48.5	1.9	30.4	39.0	49.2
LTIR [18]	92.9	55.0	85.3	34.2	31.1	34.9	40.7	34.0	85.2	40.1	87.1	61.0	31.1	82.5	32.3	42.9	0.3	36.4	46.1	50.2
PCEDA [47]	91.0	49.1	85.6	37.2	29.7	33.7	38.1	39.2	85.4	35.4	85.1	61.1	32.8	84.1	45.6	46.9	0.0	34.2	44.5	50.5
FDA [48]	92.5	53.3	82.4	26.5	27.6	36.4	40.6	38.9	82.3	39.8	78.0	62.6	34.4	84.9	34.1	53.1	16.9	27.7	46.4	50.5
CBST [50]	91.8	53.5	80.5	32.7	21.0	34.0	28.9	20.4	83.9	34.2	80.9	53.1	24.0	82.7	30.3	35.9	16.0	25.9	42.8	45.9
CRST(MRKLD) [52]	91.0	55.4	80.0	33.7	21.4	37.3	32.9	24.5	85.0	34.1	80.8	57.7	24.6	84.1	27.8	30.1	26.9	26.0	42.3	47.1
TPLD [40]	94.2	60.5	82.8	36.6	16.6	39.3	29.0	25.5	85.6	44.9	84.4	60.6	27.4	84.1	37.0	47.0	31.2	36.1	50.3	51.2
IAST [26]	93.8	57.8	85.1	39.5	26.7	26.2	43.1	34.7	84.9	32.9	88.0	62.6	29.0	87.3	39.2	49.6	23.2	34.7	39.6	51.5
WDA [31] (Point)	94.0	62.7	86.3	36.5	32.8	38.4	44.9	51.0	86.1	43.4	87.7	66.4	36.5	87.9	44.1	58.8	23.2	35.6	55.9	56.4
Ours (PPL: Point)	96.1	71.8	88.8	47.0	46.5	42.2	53.1	60.6	89.4	55.1	91.4	70.8	44.7	90.6	56.7	47.9	39.1	47.3	62.7	**63.5**
Ours (SPL: Segment)	**96.6**	**77.0**	**89.6**	**47.8**	**50.7**	**48.0**	**56.6**	**63.5**	**89.5**	**57.8**	**91.6**	**72.0**	**47.3**	**91.7**	**62.1**	**61.9**	**48.9**	**47.9**	**65.3**	**66.6**
Supervised	96.9	77.1	89.8	45.6	49.9	47.4	55.8	64.1	90.0	58.2	92.8	71.9	46.9	91.4	60.3	65.8	54.3	44.6	64.7	66.7

Our result - LabOR

Self baseline for SPL



Label (pts)	RAND	SCONF	ENT	TE	SPL	Sup
Stage #1	1	1	1	7.5	0.7	100
Stage #2	2	1.8	1.8	9.2	1.5	100
Stage #3	3	2.2	2.0	10.1	2.2	100

RAND: random segment label ratio

SCONF: sorting from softmax max value

ENT: sorting from entropy value per pixel

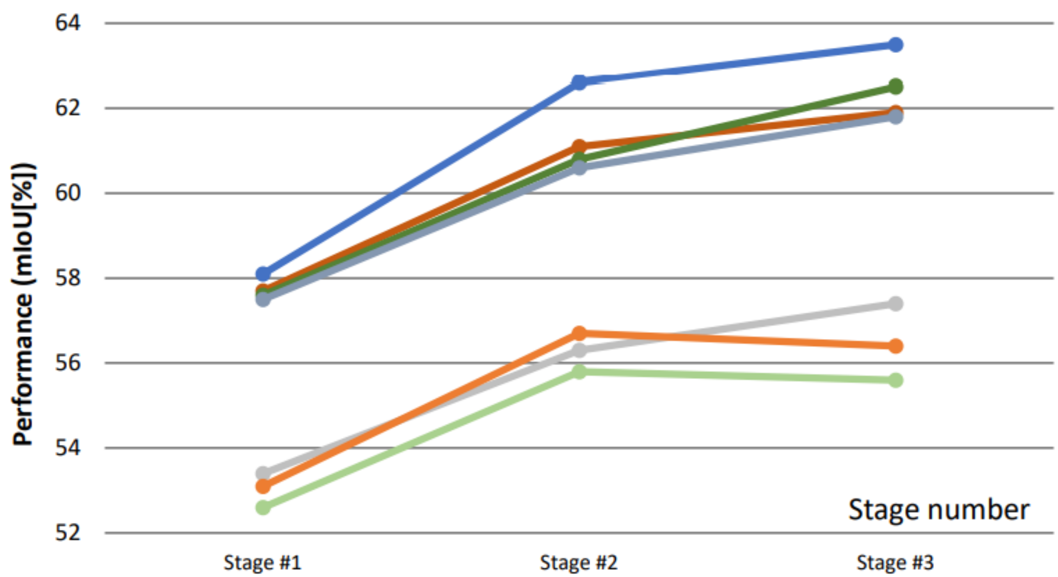
TE: inconsistent mask from temporal ensemble

SPL (Ours)

Our result - LabOR

Self baseline for PPL

Point based Pixel Labeling (PPL)



Labels (pts)	RAND	SCONF	ENT	PPL(SCONF)	PPL(ENT)	PPL(Sim_worst)	PPL(Sim_best)
Stage #1	15	15	15	15	15	12.7	12.7
Stage #2	30	29.9	29.8	30	30	26.2	26.5
Stage #3	45	43.6	41.7	45	45	39.3	40.1

RAND

SCONF

ENT

PPL (SCONF)

PPL (ENT)

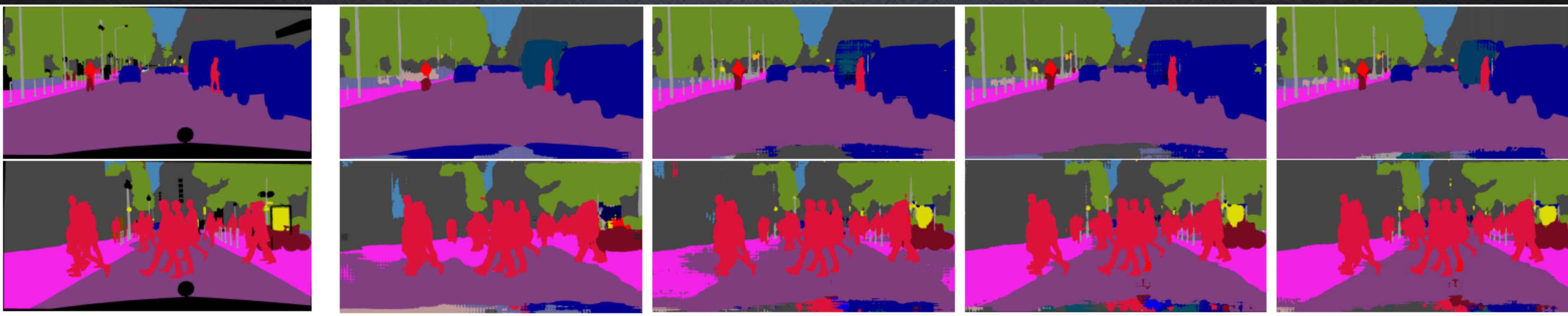
PPL (Sim_worst)

PPL (Sim_best[ours])

Our result - LabOR

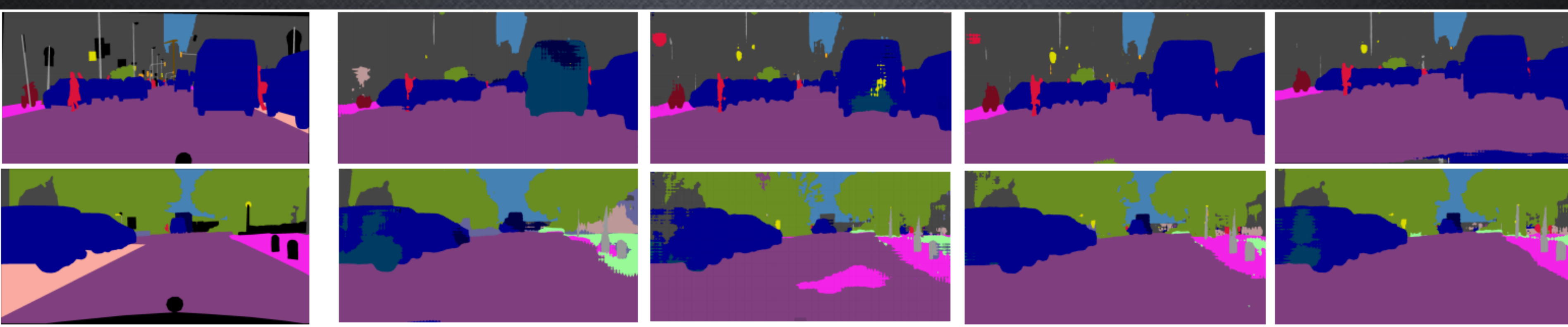
Qualitative result

SPL



GT | (a) IAST(0%) | (b) SCONF(~2.2%) | (c) SPL(~2.2%) | (d) Sup(100%)

PPL



GT | (a) IAST(0pts) | (b) SCONF(45pts) | (c) PPL(40pts) | (d) Sup(100%)

Our analysis - LabOR

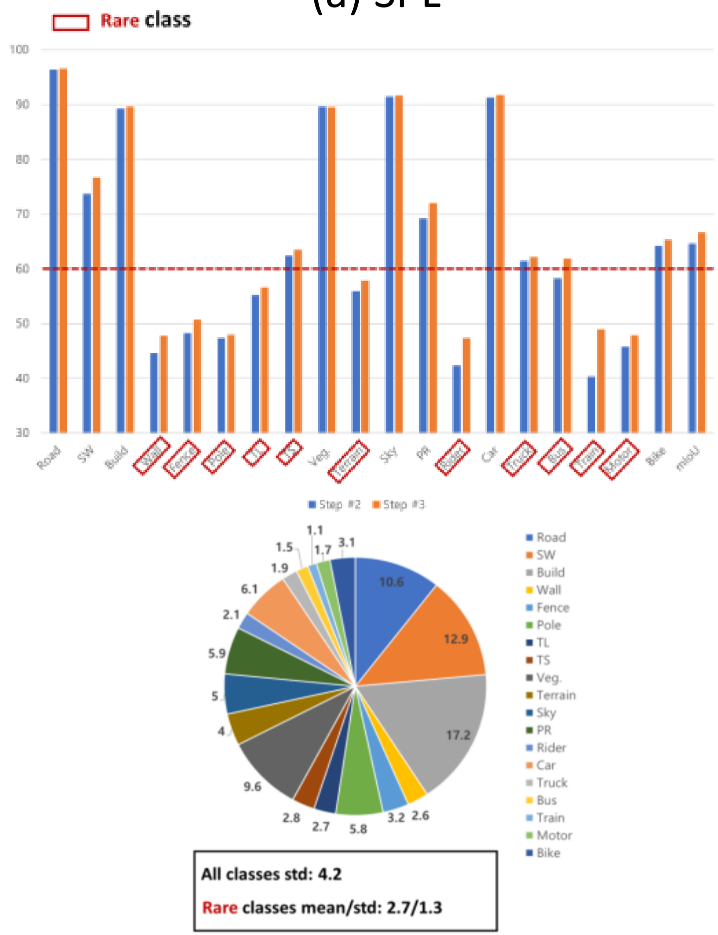
The visualization of the generated regions to label
- Ours SPL can avoid 'lump' of labels.

GT Ent SPL PPL

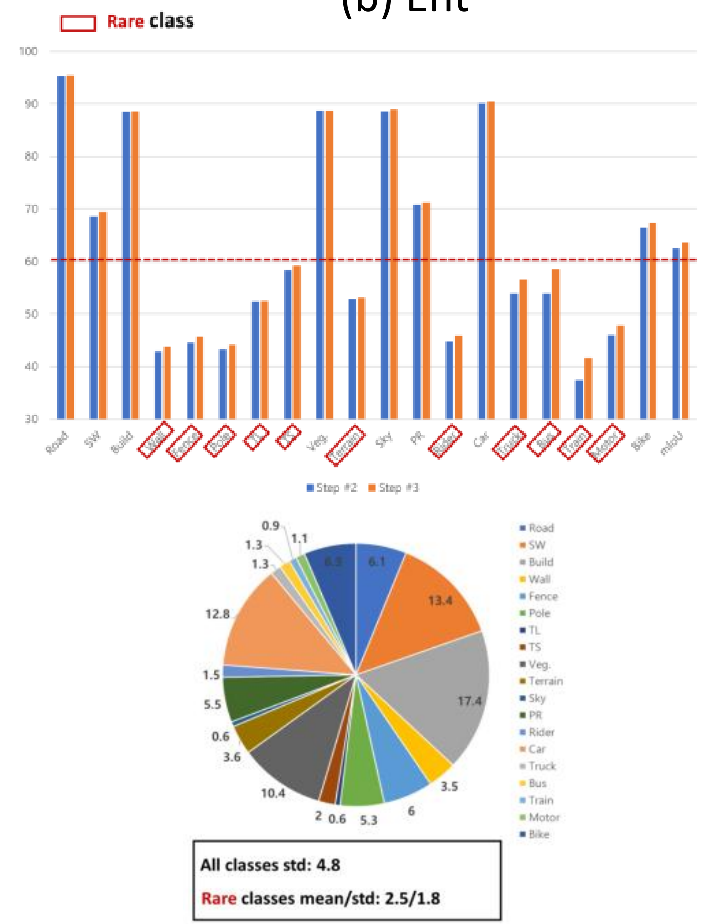
Our analysis - LabOR

The diversity of pixel classes selected for Ours and Entropy
- Our SPL has a much more even distribution in many of the classes, while Entropy has many classes that are rarely selected

(a) SPL



(b) Ent



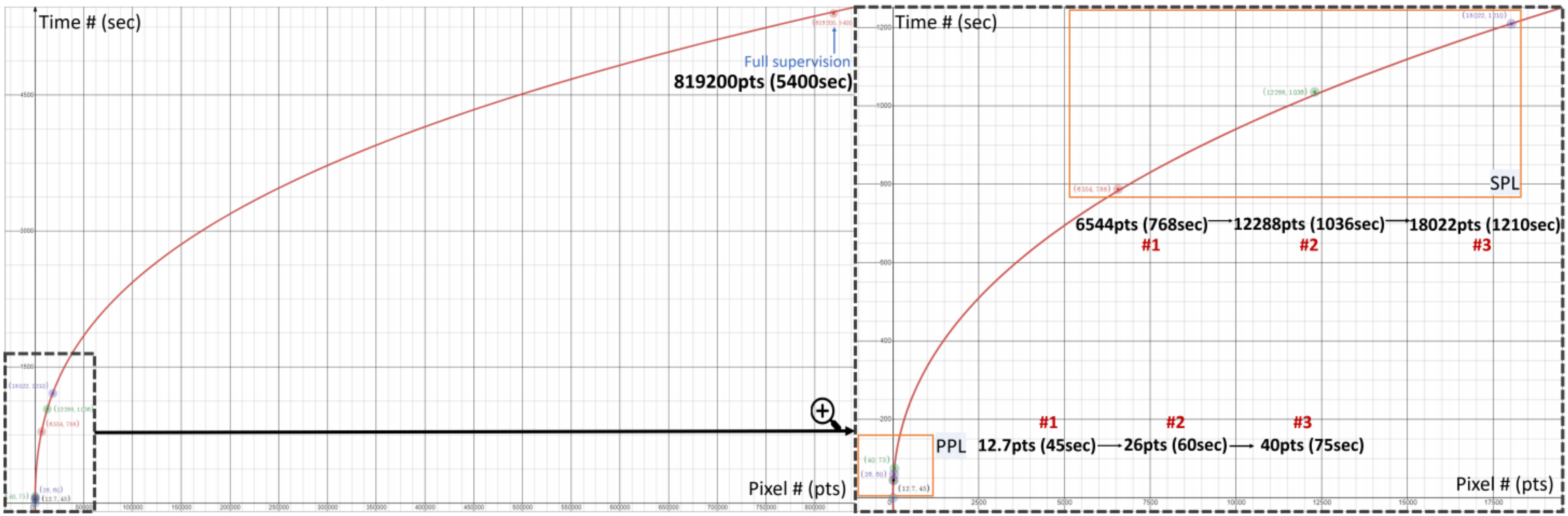
2021 ICCV OCTOBER 11-17 VIRTUAL

Our analysis - LabOR

Evaluation on Labeling Effort

$$x = 3.7 \times 10^{-6} y^3 + 7.9 \times 10^{-3} y^2 - 8.0 \times 10^{-2} y.$$

$(0,0), (12.7, 45), (40, 75), (819200, 5400)$



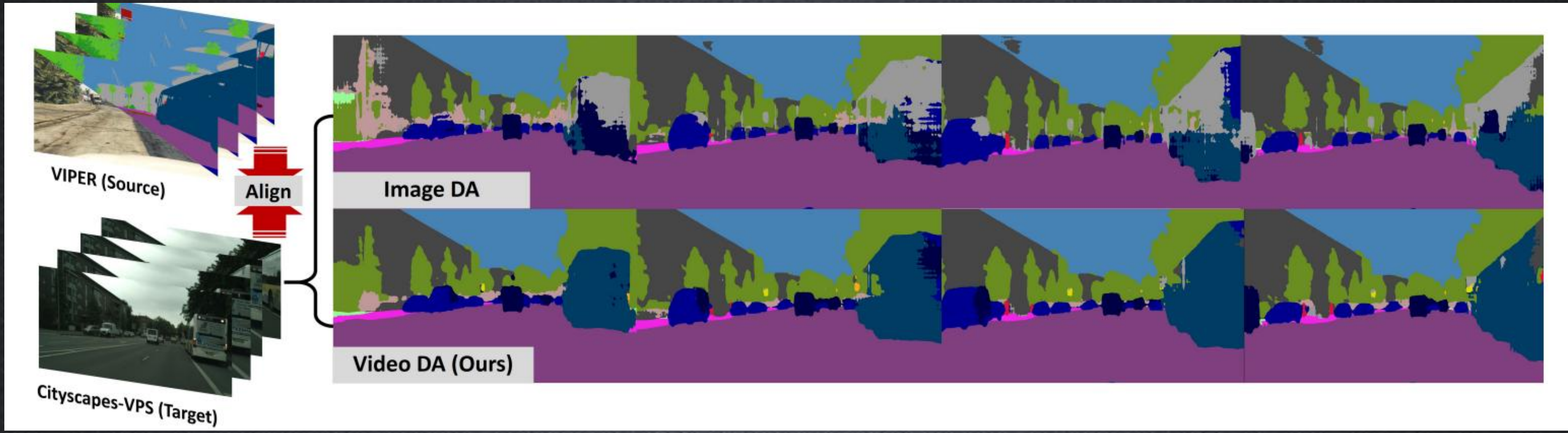
(a) Labeling cost graph (b) Zoomed-in Labeling cost graph

Labeling Type	Seconds (s)	Minutes (mins)	Hours (hr)	mIoU (%)
PPL	223,115	3,719	62	63.5
SPL	3,599,750	59,996	1,000	66.6
Full Image Label	16,065,000	267,750	4,463	66.7

https://www.nec-labs.com/~mas/WeakSegDA/

Ongoing & Future work

1. Unsupervised Domain Adaptation for Video Semantic Segmentation *(Ongoing)*



https://arxiv.org/pdf/2107.11052.pdf

2. LabOR for Video Domain Adaptation *(Future)*

Thank you

Contact: dlsrbgg33@kaist.ac.kr

Homepage: dlsrbgg33.github.io