Supplementary for Generalize then Adapt: Source-Free Domain Adaptive Semantic Segmentation

In this supplementary, we provide detailed theoretical insights, extensive implementation details with additional qualitative and quantitative performance analysis. Towards reproducible research, we have released our code and trained network weights at our project page: https://sites.google.com/view/sfdaseg.
This supplementary is organized as follows:

- Section 1: Notations (Table 1)
- Section 2: Extended theoretical insights
  - Discussion on Result 1 (Sec. 2.1)
  - Augmentation selection criteria (Sec. 2.2)
- Section 3: Implementation details
  - Experimental settings (Sec. 3.1, Table 7)
  - Vendor-side training (Sec. 3.2, Algo. 1)
  - Client-side training (Sec. 3.3, Algo. 2)
  - Optimization details (Sec. 3.4)
- Section 4: Analysis
  - Optimal choice of AGs (Sec. 4.1, Fig. 1B, Table 2)
  - Empirical evaluation of Result 1 (Sec. 4.2, Table 3)
  - Impact of cPAE (Sec. 4.3, Fig. 1C)
  - Time complexity analysis (Sec. 4.4, Table 4)
  - Qualitative analysis (Sec. 4.5, Fig. 2, 3, 4, 5)
  - Quantitative analysis (Sec. 4.6, Table 5, 6)

1. Notations

We summarize the notations used in the paper in Table 1. The notations are listed under 5 groups viz. Distributions, Datasets, Networks, Samples/Outputs and Theoretical.

Table 1. **Notation Table**

	Symbol	Description
Distribution	p_s	Marginal source input distribution
	p_t	Marginal target input distribution
	p_y	Marginal output distribution
	p_{s_i}	Marginal i^{th} aug. source input distr.
Datasets	$\mathcal{D}_s$	Labeled source dataset
	$\mathcal{D}_{s_i}$	i^{th}-AG source dataset
	$\mathcal{D}_{s_g}$	*Global* source dataset
	$\mathcal{D}_{s_i}^{-}$	i^{th}-*leave-one-out* source dataset
	$\mathcal{D}_t$	Unlabeled target dataset
	$\hat{\mathcal{D}}_t$	Pseudo-labeled target dataset
Networks	F	Shared CNN backbone
	H_g	Global output head
	H_i	i^{th} non-global output head
	F_g	Backbone F and first block of H_g
	Q	Conditional Prior-enforcing AE
Samples / Outputs	(x_s, y_s)	Labeled source sample
	$\mathcal{T}_i(\cdot)$	i^{th} augmentation
	(x_{s_i}, y_s)	i^{th} augmentation sample
	x_t	Unlabeled target sample
	$(x_t, \hat{y}_t)$	Pseudo-labeled target sample
	h_g	Output of $H_g \circ F$
	h_i	Output of $H_i \circ F$
Theoretical	$\epsilon_t(h)$	Expected target error of hypothesis h
	α	Source convex combination weights
	$\mathcal{H}^\alpha$	Hypothesis subspace for particular α
	t_j	j^{th} target domain

2. Extended theoretical insights

2.1. Discussion on Result 1

To analyze possible solutions to the paradigm defined in Definition 1 of the paper, we introduce three configurations: **1)** ERM (empirical risk minimization) *i.e.* weighting each multi-source domain equally, **2)** domain-experts++ (*DE++*) *i.e.* an ERM subspace together with K subspaces formed from one specific domain each, **3)** leave-one-out++ (*LO++*) *i.e.* an ERM subspace together with K subspaces formed by all domains except one each. We restate the result here:

$$\begin{aligned} \epsilon_t(h \in \mathcal{A}^{\text{LO++}}) &\leq \epsilon_t(h \in \mathcal{H}^{\text{ERM}}) \\ \epsilon_t(h \in \mathcal{A}^{\text{DE++}}) &\leq \epsilon_t(h \in \mathcal{H}^{\text{ERM}}) \end{aligned} \tag{1}$$

Proof. We give the proof by contradiction. Let the optimal hypothesis from *LO++* have a higher target error than ERM *i.e.* $\epsilon_t(h \in \mathcal{A}^{\text{LO++}}) > \epsilon_t(h \in \mathcal{H}^{\text{ERM}})$. However, this implies that it cannot be the optimal hypothesis because an ERM hypothesis within *LO++* has a lower target error. This is a contradiction which proves the result. The same can be

shown for the *DE++* case. In other words, the ERM subspace in *DE++* or *LO++* can provide the optimal hypothesis in the worst case and the equality in Eq. 1 will hold.

2.2. Augmentation selection criteria

From Eq. 3 of the paper, we know that the augmentation $\mathcal{T}_i$ modifies the non-causal domain-related factor f_s by a weight γ_i and introduces a new augmentation-related factor f_i without altering the causal class factor f_y. For a general augmentation, we cannot restrict γ_i *i.e.* $\gamma_i \in \mathbb{R}$. However, since we need to restrict the number of domains K, augmentations need to be filtered out. The important criteria for this filtering is the ability of the augmentation to simulate new domains while suppressing the original domain. Assuming that each augmentation is capable of introducing new domain-specific features, it is crucial for it to reduce the influence of the original domain. Thus, in Definition 2 in the paper, the selection criteria is $|\gamma_i| < 1$.

Due to the hypothetical nature of the decomposition of a sample into class-specific and domain-specific factors, it is not feasible to use the criteria on γ_i directly. However, models usually rely on both class-specific and domain-specific factors for prediction [22]. Thus, the performance of a standard single-source-trained model on augmented samples can be a good measure of the residual original domain dependency. An augmentation causing low performance for the pretrained model is likely to be suppressing the original domain factors. In other words, the model is unable to latch onto the original domain factors for prediction. This gives the surrogate condition in Eq. 4 of the paper. However, the candidate augmentations must be manually filtered to not select those that perturb the class-relevant factors since it is not ensured through this criteria.

3. Implementation Details

In this section, we describe the network architectures, datasets, the training process used for vendor-side and client-side training and other implementation details.

3.1. Experimental settings

a) Network architectures. Following [15, 33], we employ 2 widely-used network architectures for the DA setting on semantic segmentation, DeepLabv2 [2] with ResNet101 [7] backbone and FCN8s [18] with VGG16 [27] backbone. For DeepLabv2-ResNet101 and FCN8s-VGG16, we define the shared backbone F upto *Layer3* block and *Conv4* block respectively. The remaining networks are taken as output head for both. Thus, at the inference stage, our model has exactly the same amount of parameters as used in the prior-arts. We use a fully convolutional architecture for the conditional Prior-enforcing Auto-Encoder (cPAE) with asymmetric encoder and decoder as shown in Table 7.

b) Datasets. We extensively evaluate the proposed approach on two popular synthetic-to-real benchmarks *i.e.*, GTA5→Cityscapes and SYNTHIA→Cityscapes. For GTA5 [23], we resize the image to 1280×720 before randomly cropping to 1024×512. Whereas, for SYNTHIA [24], we resize to 1280×760 and random crop to 1024×512 following [33]. For Cityscapes [5], we resize the image to 1024×512. For GTA5 we use 24500 images for training and 466 images for validation. Whereas, for SYNTHIA we use 9000 images for training and 400 images for validation. For client-side evaluation, we use Cityscapes training dataset for training and the standard validation set for testing [33]. Following previous works [20, 29, 38], we use multi-scale testing to report the final performance.

c) Augmentations. We provide extra details about the AGs to enhance the reproducibility of our experiments.

Aug-A: We used images from a style transfer dataset [9] and release them with the code. For random noise, we used uniform sampling from 0 to 255 for every pixel location.

Aug-B: We used this augmentation from the code release of [10] as provided. No controllable parameter available.

Aug-C: We set the strength of stylization (α) to 0.3 which balances stylization and content preservation.

Aug-D: For snow and frost augmentation, we uniformly sample the severity between 1 and 3 (max. severity 5 possible in [11]) to balance stylization and content preservation.

Aug-E: No controllable parameter in cartoon AG [11].

3.2. Vendor-side training

The vendor-side training involves multi-head SoMAN training followed by cPAE training, as described in Algo. 1.

In SoMAN training, at each iteration, an image sampled from the source dataset D_s is augmented using a random AG. Since we train each head in a *leave-one-out* manner, the global head and $K-1$ non-global heads are trained at each iteration (L2-L14). We update the parameters of each head using a separate optimizer (L13). The momentum parameters in the optimizers adaptively scale the gradients thereby avoiding loss-scaling hyperparameters.

In cPAE training, we use the trained non-global heads from SoMAN to generate noisy seg-maps for denoising auto-encoder training. We augment source samples with a randomly chosen i^{th} AG (L17-19). Next, these are passed through the corresponding i^{th} non-global head which was trained to be sensitive to that AG, thereby yielding noisy seg-maps (L20). The cPAE predictions for these noisy seg-maps are used to compute the cross-entropy loss with the ground truth seg-maps. This loss is minimized using a SGD optimizer to update the cPAE parameters (L21-L23).

3.3. Client-side training

The client-side training requires optimal head identification, pseudo-label extraction and self-training, as in Algo. 2.

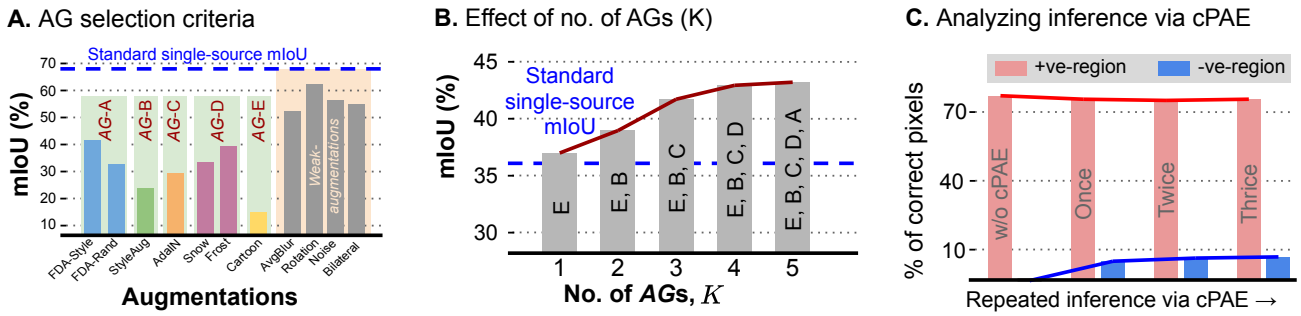

Figure 1. **A.** Selecting AGs from a set of candidate augmentations via inference through a standard single-source trained model (see Sec. 4.1). **B.** Performance of vendor-side trained models varying K on Cityscapes. Performance saturates as K reaches 5 (see Sec. 4.1). **C.** Impact of cPAE on correctly (+ve) and incorrectly (-ve) predicted regions on Cityscapes for a given model (see Sec. 4.3).

In optimal head identification, for a given target, the head with the lowest $\epsilon_t(h)$ has to be chosen as per Result 1. Since the computation of $\epsilon_t(h)$ is intractable, we choose a proxy *i.e*. average self-entropy on the target training set (L2). Intuitively, the head closest to the target domain will be selected as it would be the most confident (lowest self-entropy).

In pseudo-label extraction, we first process the entire target training dataset (L3-L11) and store the spatial class predictions (L8) and the prediction probabilities (L9). To avoid noisy predictions, we determine class-wise thresholds (L12-L16) which are set at 33% of the most confident predictions per class. Finally, we apply the class-wise thresholds (L17-23) and assign an unlabeled '*unknown*' class to the pixels which do not satisfy the threshold. These unlabeled pixels are not considered in the loss computation during training.

In self-training, we use the pseudo-labeled target dataset to train in a supervised manner (L24-30). Specifically, we train a block under the shared backbone F (L29) using the optimal head predictions and pseudo-labels. For DeepLabv2-ResNet101, this block is *Layer3* while for FCN8s-VGG16, it is *Conv3+Conv4* blocks. We find this to perform better than training the entire backbone F. We perform 3 rounds of offline pseudo-label extraction and self-training, following [33]. Further, we find that the performance does not increase with more than 3 rounds.

3.4. Optimization details

We implement our framework on PyTorch [21]. Following [15], for DeepLabv2-ResNet101, we use SGD optimizer with momentum 0.9, initial learning rate 2.5e-4 with a polynomial learning rate decay with power 0.9, and weight decay 5e-4. For FCN8s-VGG16, we use Adam [14] optimizer with initial learning rate 1e-5 with step decay of 0.1 at every 20k steps. We train for 50k iterations each in vendor-side and in each round of self-training. With mixed precision, we use batch size 2 on a GTX1080Ti GPU.

4. Analysis

In this section, we analyze the choice of AGs, the empirical evaluation of Result 1 and the impact of cPAE. We show extended quantitative and qualitative evaluations of our approach and training time comparisons with prior arts.

4.1. Optimal choice of AGs

The choice of number of AGs and the AGs themselves is critical to the success of vendor-side training. We describe the AG candidates and those used by prior arts in Table 2. Firstly, it is important to choose diverse AGs to facilitate the learning of both domain-invariant and domain-specific features by the proposed SoMAN. Secondly, a higher number of AGs (K) and correspondingly non-global output heads incur additional computational cost in the vendor-side training. Thus, it is crucial to determine a low enough K that yields a significant performance improvement.

Towards the first, Fig. 1A shows the selection of diverse AGs from a set of candidates using the performance of a standard single-source trained model. This is according to Definition 2 in the paper using mIoU (task metric) with a threshold of 25%. We observe that weaker augmentations that do not produce enough domain-shift get filtered out.

Towards the second, we analyze the effect of number of AGs on the performance in Fig. 1B. Particularly, we evaluate vendor-side trained models with varying number of AGs used during training. Since there are multiple ways to choose from the set of candidate AGs, we choose the most diverse ones first to ensure the best possible performance for a lower K. In other words, the AG giving the most deterioration in mIoU for a standard single-source trained model is chosen first. Fig. 1A shows that the order is E, B, C, D, A. Using this order to choose AGs for $K = \{1, 2, \ldots, 5\}$, we observe that performance saturates as K reaches 5. Thus, we infer that the computational burden of adding more AGs would not result in a substantial performance improvement.

Algorithm 1 Pseudo-code for vendor-side training

Input: source dataset $\mathcal{D}_s$, standard single-source trained model F_s, H_s

▷ *Note that CE denotes class-weighted cross-entropy.*

Step 1: *SoMAN training*

Initialize $F, H_g, \{H_i\}_{i=1}^K$ using F_s, H_s
for $iter < MaxIter$ **do**:
  $x_s, y_s \leftarrow$ batch sampled from $\mathcal{D}_s$
  $i_1 \leftarrow \text{rand}(1, K)$
  $\tilde{x}_s \leftarrow \mathcal{T}_{i_1}(x_s)$
  $h_g \leftarrow H_g(F(\tilde{x}_s))$
  Compute $\mathcal{L}_g \leftarrow \text{CE}(h_g, y_s)$
  for i in range(K) except $\{i_1\}$ **do**:
    $h_i \leftarrow H_i(F(\tilde{x}_s))$
    Compute $\mathcal{L}_i \leftarrow \text{CE}(h_i, y_s)$
  end for
  update $\theta_{H_g}, \{\theta_{H_i}\}_{i=1, i\neq i_1}^K$ by minimizing $\mathcal{L}_g, \{\mathcal{L}_i\}_{i=1, i\neq i_1}^K$ using separate optimizers
end for

Step 2: *cPAE training*

Randomly initialize cPAE Q; freeze $F, H_g, \{H_i\}_{i=1}^K$ from Step 1 training.
for $iter < MaxIter$ **do**:
  $x_s, y_s \leftarrow$ batch sampled from $\mathcal{D}_s$
  $i \leftarrow \text{rand}(1, K)$ ▷ Randomly choose an AG
  $x_{s_i} \leftarrow \mathcal{T}_i(x_s)$ ▷ Augment x_s
  $h_i \leftarrow H_i(F(x_{s_i}))$
  $\hat{h}_i \leftarrow Q(h_i, F_g(x_{s_i}))$
  Compute $\mathcal{L}_q \leftarrow \text{CE}(\hat{h}_i, y_s)$
  update θ_Q by minimizing $\mathcal{L}_q$ using SGD optimizer
end for

4.2. Empirical evaluation of Result 1

To empirically evaluate Result 1 of the paper, we measure the performance of each SoMAN head for a variety of target scenarios as shown in Table 3. We observe that different heads of the SoMAN give the best performance in different target scenarios. Further, in most scenarios, at least one of the *leave-one-out* heads performs better than ERM. This is in line with Result 1 *i.e.* one of the leave-one-out heads gives a lower or equal target risk than ERM.

For Foggy-Cityscapes (0.02) [25], we observe that ERM is optimal which can be considered as a worst-case scenario (large domain-shift). On the other hand, Foggy-Cityscapes (0.01) and (0.005) both have LO-E as the optimal head since they represent similar domain-shifts. Further, for NTHU-Cross-City [4], different heads are optimal for different cities since each city presents a different domain-shift.

Algorithm 2 Pseudo-code for client-side training

Input: Trained SoMAN $(F, H_g, \{H_i\}_{i=1}^K)$ and cPAE (Q) from vendor, unlabeled target dataset $\mathcal{D}_t$

▷ *Let $[\cdot]$ denote the indexing operation, $\cdot||\cdot$ denote the append operation, $|\cdot|$ denote the cardinality, and C be the number of classes.*

Step 1: *Optimal head identification*

$i' \leftarrow \arg\min_{i \in \{g,[K]\}} \sum_{x \in \mathcal{D}_t} \{-\langle h_i, \log h_i \rangle\}$ where $h_i = H_i(F(x)) \, \forall \, i$ ▷ Lowest average self-entropy

Step 2: *Pseudo-label extraction*

$Y_p \leftarrow \{\}$ ▷ Empty ordered list
$W \leftarrow \{\}$ ▷ Empty ordered list
$X_p \leftarrow \{\}$ ▷ Empty ordered list
for x in $\mathcal{D}_t$ **do**:
  $\hat{h} = Q(H_{i'}(F(x)), F_g(x))$
  $y_p \leftarrow \arg\max_{c \in C} \hat{h}[c]$ ▷ Class predictions
  $w \leftarrow \max_{c \in C} \hat{h}[c]$ ▷ Predicted class probabilities
  $W, Y_p, X_p \leftarrow W \,||\, w, Y_p \,||\, y_p, X_p \,||\, x$
end for
$t \leftarrow \{\}$ ▷ List of class-wise thresholds
for c in range(C) **do**:
  Store all prediction probabilities of class c in p_x
  $p_x \leftarrow W[Y_p == c]$
  $p_x \leftarrow \text{sort}(p_x)$
  Set threshold at top 33% most confident predictions
  $t \leftarrow t \,||\, p_x[0.66|p_x|]$
end for
$\hat{\mathcal{D}}_t \leftarrow \{\}$ ▷ Empty ordered list
for y_p, w, x_p in Y_p, W, X_p **do**:
  for c in range(C) **do**:
    Assign class-id, $C+1$ representing '*unknown*', to pixels with probability $<$ class threshold $t[c]$
    $y_p[(w < t[c]) \& (y_p == c)] \leftarrow C + 1$
  end for
  $\hat{\mathcal{D}}_t \leftarrow \hat{\mathcal{D}}_t \,||\, (x_p, y_p)$
end for

Step 3: *Source-free self-training adaptation*

Obtain $F, H_{i'}$ from vendor (or last self-training round)
for $iter < MaxIter$ **do**:
  $x_t, y_t \leftarrow$ batch sampled from $\hat{\mathcal{D}}_t$
  $\hat{h} \leftarrow H_{i'}(F(x_t))$
  Compute $\mathcal{L}_t \leftarrow \text{CE}(\hat{h}, y_t)$
  update trainable parameters of θ_F by minimizing $\mathcal{L}_t$ using SGD optimizer
end for

Table 2. AG candidates and augmentations used by prior arts. TF indicates target-free *i.e.* whether augmentation requires target data.

Method	TF	Description
FDA [33]	×	Target images as reference for stylization in Fourier domain.
LTIR [13]	×	Uses style-swap [3] with target images as reference.
BDL [15]	×	Image-to-image translation for source→target conversion.
LDR [30]	×	Image-to-image translation for target→source conversion.
Ours-A	✓	FDA [33] with random and style images as reference.
Ours-B	✓	Stylization by randomly sampled style embedding [10].
Ours-C	✓	AdaIN [9] layers for stylization using reference images.
Ours-D	✓	Varying levels of weather augmentations [11]: frost and snow.
Ours-E	✓	Converts image into texture-less cartoon-like image [11].
Ours-W1	✓	Blurring using a 5×5 average filter.
Ours-W2	✓	Rotating image by an angle $\in [-15, 15]$ degrees.
Ours-W3	✓	Adding random noise to the image.
Ours-W4	✓	Edge-preserved smoothing using bilateral filtering.

Table 3. Empirical evaluation of Result 1 for vendor-side SoMAN heads with mIoU for various target scenarios. LO indicates leave-one-out head while ERM is the global head. 0.005, 0.01, and 0.02 indicate the levels of fog in the dataset. We observe that different heads are optimal for different target domains.

Head	Cityscapes	Foggy-Cityscapes			NTHU-Cross-City			
		0.005	0.01	0.02	Rio	Rome	Taipei	Tokyo
ERM	43.1	43.6	42.4	**38.3**	47.0	48.7	43.4	44.5
LO-A	42.4	43.0	41.6	36.7	45.4	**48.9**	43.2	45.4
LO-B	42.2	42.2	40.7	36.1	**49.0**	47.7	42.1	46.5
LO-C	43.1	43.0	41.7	37.8	48.1	48.6	43.8	**46.7**
LO-D	**43.5**	43.4	41.7	37.0	45.6	47.9	**43.9**	45.3
LO-E	43.2	**43.9**	**42.6**	37.9	45.5	47.0	43.6	45.9

4.3. Impact of cPAE

We train the proposed cPAE as a denoising autoencoder to encourage spatial regularities in the segmentation predictions. In Fig. 1C, we analyze the effect of inference via cPAE. Particularly, we hypothesize that the cPAE should not distort regions that were correctly predicted by the segmentation network. This is desirable to retain the inductive bias in the absence of target labels. For a given segmentation network, we determine these regions, denoted as +ve-regions, and the regions where the segmentation network failed, denoted as -ve-regions. Next, we use the cPAE on the segmentation predictions and evaluate the performance in the two previously determined regions. We also evaluate repeated inference via cPAE *i.e.* passing the output of the cPAE through the cPAE again. We observe that the performance in the +ve-region is almost the same while it improves in the -ve-region. Further, we observe that repeated inference via cPAE does not give any significant improvement. Thus, we resort to inferring via cPAE only once.

4.4. Time complexity analysis

Our proposed client-side adaptation trains the shared backbone F partially and does not require any additional networks like adversarial discriminators during the training.

Table 4. Training time comparison of adaptation step with prior arts. AN indicates whether additional networks are involved in the adaptation training.

Method	AN	Training time (per iter.) (seconds)
PCEDA [32]	✓	0.94
FDA [33]	×	0.90
Ours	×	**0.31**

Further, we offer a simple adaptation pipeline without requiring access to source data. These factors lead to a lower training time (Table 4) for our client-side adaptation compared to prior arts while maintaining *state-of-the-art* adaptation performance. This makes it suitable for practical and even online adaptation scenarios.

We perform the analysis (Table 4) by measuring the average time taken for forward pass, backward pass and the optimizer step (network weights update) for each method. For FDA and PCEDA, the time per iteration is higher since they train the entire model while using FFT-based augmentation and an image-to-image translation network respectively. For a fair comparison, we use batch size 1 (without automatic mixed precision) for all methods and evaluate on a machine with Intel Xeon E3-1200 CPU, 32GB RAM and a single 11GB NVIDIA GTX1080Ti GPU using Python 3, PyTorch 1.6 and CUDA 10.2.

4.5. Qualitative analysis

We provide extended qualitative results of our proposed approach on GTA5→Cityscapes [23, 5] in Fig. 5. Further, we show examples of the paired samples used for the training of cPAE in Fig. 2 and examples of the devised AGs applied to the GTA5 dataset in Fig. 3. We also show qualitative results of online adaptation to NTHU-Cross-City [4] and Foggy-Cityscapes [25] in Fig. 4.

We also observe some failure cases in Fig. 5 (indicated by white circles) where merged-region problems occur for smaller-sized classes in the scene. More explicit ways of inculcating shape priors may improve the performance further. We plan to explore this direction in our future work.

4.6. Quantitative analysis

We provide extended quantitative results on the GTA5→Cityscapes and SYNTHIA→Cityscapes [24] benchmarks for semantic segmentation in Tables 5, 6. We obtain *state-of-the-art* performance across all settings even against non-source-free approaches.

References

[1] Mathilde Bateson, Hoel Kervadec, Jose Dolz, Herve Lombaert, and Ismail Ben Ayed. Source-relaxed domain adaptation for image segmentation. In *MICCAI*, 2020. 7

[2] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image

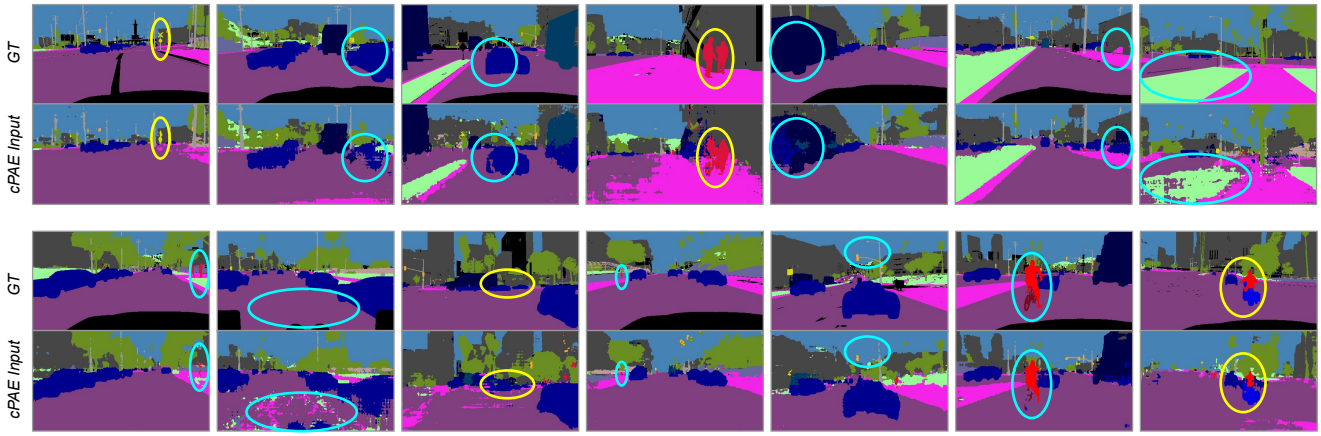

Figure 2. Paired samples for cPAE training. cPAE is trained as a denoising autoencoder to encourage structural regularity in segmentation predictions and alleviate merged-region (yellow circle) and splitted-region (blue circle) problems. *Best viewed in color.*

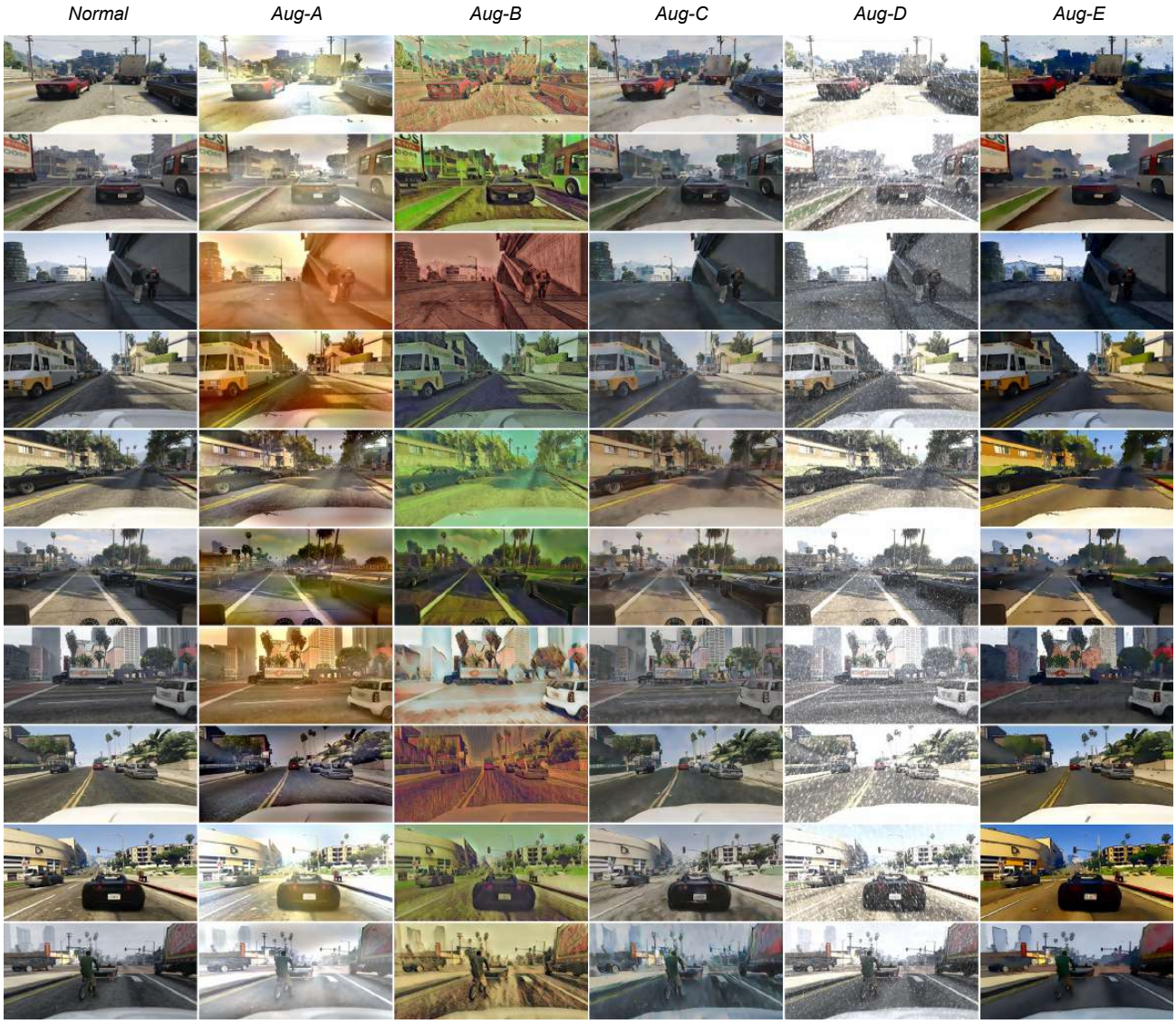

Figure 3. Examples of AGs applied to GTA5 dataset images. Notice the diversity in the augmentations. *Best viewed in color.*

Table 5. **Quantitative evaluation on GTA5→Cityscapes**. Performance on different segmentation architectures: A (DeepLabv2 ResNet-101), B (FCN8s VGG-16). SF indicates whether the method supports *source-free* adaptation. *Ours (V)* indicates use of our vendor-side AGs with prior art and * indicates reproduced by us using released code. We observe better or competitive performance on minority classes like motorcycle compared to non-source-free prior arts.

Method	Arch.	SF	road	sidewalk	building	wall	fence	pole	t-light	t-sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle	mIoU
PLCA [12]	A	×	84.0	30.4	82.4	35.3	24.8	32.2	36.8	24.5	85.5	37.2	78.6	66.9	32.8	85.5	40.4	48.0	8.8	29.8	41.8	47.7
CrCDA [8]	A	×	92.4	55.3	82.3	31.2	29.1	32.5	33.2	35.6	83.5	34.8	84.2	58.9	32.2	84.7	40.6	46.1	2.1	31.1	32.7	48.6
PIT [19]	A	×	87.5	43.4	78.8	31.2	30.2	36.3	39.9	**42.0**	79.2	37.1	79.3	65.4	37.5	83.2	46.0	45.6	25.7	23.5	49.9	50.6
TPLD [26]	A	×	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
RPT [35]	A	×	89.7	44.8	86.4	**44.2**	30.6	**41.4**	**51.7**	33.0	87.8	39.4	86.3	**65.6**	24.5	**89.0**	36.2	46.8	17.6	39.1	**58.3**	53.2
FADA [29]	A	×	91.0	50.6	86.0	43.4	29.8	36.8	43.4	25.0	86.8	38.3	87.4	64.0	38.0	85.2	31.6	46.1	6.5	25.4	37.1	50.1
IAST [20]	A	×	94.1	58.8	85.4	39.7	29.2	25.1	43.1	34.2	84.8	34.6	88.7	62.7	30.3	87.6	42.3	50.3	24.7	35.2	40.2	52.2
Ours (V) + FADA*	A	×	91.2	51.0	**86.6**	43.6	30.3	37.1	43.7	25.2	**87.9**	40.2	88.2	64.7	**38.4**	85.5	32.0	46.8	6.6	25.9	37.5	50.6
Ours (V) + IAST*	A	×	**94.8**	59.4	86.2	40.5	29.5	25.5	43.8	34.7	85.9	34.9	89.5	63.4	30.8	88.3	42.6	50.7	25.3	35.7	40.9	52.8
URMA [28]	A	✓	92.3	55.2	81.6	30.8	18.8	37.1	17.7	12.1	84.2	35.9	83.8	57.7	24.1	81.7	27.5	44.3	6.9	24.1	40.4	45.1
SRDA* [1]	A	✓	90.5	47.1	82.8	32.8	28.0	29.9	35.9	34.8	83.3	39.7	76.1	57.3	23.6	79.5	30.7	40.2	0.0	26.6	30.9	45.8
Ours (w/o cPAE)	A	✓	90.9	48.6	85.5	35.3	31.7	36.9	34.7	34.8	86.2	47.8	88.5	61.7	32.6	85.9	46.9	50.4	0.0	38.9	52.4	51.6
Ours (w/ cPAE)	A	✓	91.7	53.4	86.1	37.6	**32.1**	37.4	38.2	35.6	86.7	**48.5**	**89.9**	62.6	34.3	87.2	**51.0**	50.8	4.2	**42.7**	53.9	**53.4**
BDL [15]	B	×	89.2	40.9	81.2	29.1	19.2	14.2	29.0	19.6	83.7	35.9	80.7	54.7	23.3	82.7	25.8	28.0	2.3	25.7	19.9	41.3
LTIR [13]	B	×	92.5	54.5	**83.9**	**34.5**	**25.5**	**31.0**	30.4	18.0	84.1	39.6	83.9	53.6	19.3	81.7	21.1	13.6	**17.7**	12.3	6.5	42.3
LDR [30]	B	×	90.1	41.2	82.2	30.3	21.3	18.3	33.5	23.0	84.1	37.5	81.4	54.2	24.3	83.0	**27.6**	32.0	8.1	29.7	26.9	43.6
FADA [29]	B	×	92.3	51.1	83.7	33.1	29.1	28.5	28.0	21.0	82.6	32.6	85.3	55.2	28.8	83.5	24.4	37.4	0.0	21.1	15.2	43.8
PCEDA [32]	B	×	90.2	44.7	82.0	28.4	28.4	24.4	33.7	**35.6**	83.7	**40.5**	75.1	54.4	28.2	80.3	23.8	39.4	0.0	22.8	30.8	44.6
SFDA [17]	B	✓	81.8	35.4	82.3	21.6	20.2	25.3	17.8	4.7	80.7	24.6	80.4	50.5	9.2	78.4	26.3	19.8	11.1	6.7	4.3	35.8
Ours (w/o cPAE)	B	✓	90.1	44.2	81.7	31.6	19.2	27.5	29.6	26.4	81.3	34.7	82.6	52.5	24.9	83.2	25.3	**41.9**	8.6	15.7	32.2	43.4
Ours (w/ cPAE)	B	✓	**92.9**	**56.9**	82.5	20.4	6.0	**30.8**	**34.7**	33.2	**84.6**	17.0	**88.9**	**62.3**	**30.7**	**85.1**	15.3	40.6	10.2	**30.1**	**50.4**	**45.9**

Table 6. **Quantitative evaluation on SYNTHIA→Cityscapes**. Performance on different segmentation architectures: A (DeepLabv2 ResNet-101), B (FCN8s VGG-16). mIoU and mIoU* are averaged over 16 and 13 categories respectively. SF indicates whether the method supports *source-free* adaptation.

Method	Arch.	SF	road	sidewalk	building	wall*	fence*	pole*	t-light	t-sign	vegetation	sky	person	rider	car	bus	motorcycle	bicycle	mIoU	mIoU*
CAG [34]	A	×	84.8	41.7	**85.5**	-	-	-	13.7	23.0	**86.5**	78.1	**66.3**	28.1	81.8	21.8	22.9	49.0	-	52.6
APODA [31]	A	×	86.4	41.3	79.3	-	-	-	22.6	17.3	80.3	81.6	56.9	21.0	84.1	49.1	24.6	45.7	-	53.1
PyCDA [16]	A	×	75.5	30.9	83.3	**20.8**	0.7	32.7	27.3	**33.5**	84.7	85.0	64.1	25.4	85.0	45.2	21.2	32.0	46.7	53.3
TPLD [26]	A	×	80.9	44.3	82.2	19.9	0.3	40.6	20.5	30.1	77.2	80.9	60.6	25.5	84.8	41.1	24.7	43.7	47.3	53.5
USAMR [37]	A	×	83.1	38.2	81.7	9.3	1.0	35.1	30.3	19.9	82.0	80.1	62.8	21.1	84.4	37.8	24.5	53.3	46.5	53.8
RPL [36]	A	×	87.6	41.9	83.1	14.7	1.7	36.2	31.3	19.9	81.6	80.6	63.0	21.8	86.2	40.7	23.6	53.1	47.9	54.9
IAST [20]	A	×	81.9	41.5	83.3	17.7	**4.6**	32.3	30.9	28.8	83.4	85.0	65.5	30.8	86.5	38.2	33.1	52.7	49.8	57.0
RPT [35]	A	×	89.1	47.3	84.6	14.5	0.4	**39.4**	**39.9**	30.3	86.1	86.3	60.8	25.7	**88.7**	49.0	28.4	**57.5**	51.7	59.5
URMA [28]	A	✓	59.3	24.6	77.0	14.0	1.8	31.5	18.3	32.0	83.1	80.4	46.3	17.8	76.7	17.0	18.5	34.6	39.6	45.0
Ours (w/o cPAE)	A	✓	89.0	44.6	80.1	7.8	0.7	34.4	22.0	22.9	82.0	86.5	65.4	33.2	84.8	45.8	38.4	31.7	48.1	55.5
Ours (w/ cPAE)	A	✓	**90.5**	**50.0**	81.6	13.3	2.8	34.7	25.7	33.1	83.8	**89.2**	66.0	**34.9**	85.3	**53.4**	**46.1**	46.6	**52.0**	**60.1**
PyCDA [16]	B	×	80.6	26.6	74.5	2.0	0.1	18.1	13.7	14.2	80.8	71.0	48.0	19.0	72.3	22.5	12.1	18.1	35.9	42.6
SD [6]	B	×	87.1	36.5	79.7	-	-	-	13.5	7.8	81.2	76.7	50.1	12.7	**78.0**	**35.0**	4.6	1.6	-	43.4
FADA [29]	B	×	80.4	35.9	**80.9**	2.5	0.3	**30.4**	7.9	22.3	81.8	83.6	48.9	16.8	77.7	31.1	13.5	17.9	39.5	46.0
BDL [15]	B	×	72.0	30.3	74.5	0.1	0.3	24.6	10.2	25.2	80.5	80.0	54.7	23.2	72.7	24.0	7.5	44.9	39.0	46.1
PCEDA [32]	B	×	79.7	35.2	78.7	1.4	0.6	23.1	10.0	**28.9**	79.6	81.2	51.2	**25.1**	72.2	24.1	**16.7**	**50.4**	41.1	48.7
Ours (w/o cPAE)	B	✓	88.5	45.4	79.8	2.8	2.2	27.4	18.4	25.4	82.4	83.6	55.9	12.1	72.8	25.6	3.5	12.9	40.0	46.7
Ours (w/ cPAE)	B	✓	**89.9**	**48.8**	**80.9**	**2.9**	**2.5**	28.1	**19.5**	26.2	**83.7**	**84.9**	**57.4**	17.8	75.6	28.9	4.3	17.2	**41.3**	**48.9**

Table 7. **Network architecture of cPAE. Conv*** denotes standard convolutional layer followed by a batch-normalization with Parametric-ReLU non-linearity, **Dconv** denotes standard convolutional layer with Dilation, **Tanh** denotes hyperbolic tangent non-linearity, $\bigoplus$ denotes element-wise tensor addition, **Conv**** denotes standard convolutional layer followed by a batch-normalization with ReLU non-linearity, **Tconv*** denotes transpose convolutional layer followed by a batch-normalization with Parametric-ReLU non-linearity, $||$ denotes channel-wise concatenation, F_g consists of `SoMAN` backbone F and first block of global head H_g and input x is an RGB image ($512 \times 1024 \times 3$).

	Layer	Input	Type	Filter | Stride | Dilation	Output Size
Encoder	C_1	$\hat{y}$	Conv*	$7 \times 7, 64 \mid 1 \mid$ -	$512 \times 1024 \times 64$
	C_2	C_1	Conv*	$3 \times 3, 128 \mid 2 \mid$ -	$256 \times 512 \times 128$
	C_3	C_2	Conv*	$7 \times 7, 128 \mid 1 \mid$ -	$256 \times 512 \times 128$
	C_4	C_3	Conv*	$3 \times 3, 256 \mid 2 \mid$ -	$128 \times 256 \times 256$
	C_5	C_4	Conv*	$7 \times 7, 256 \mid 1 \mid$ -	$128 \times 256 \times 256$
	C_6	C_5	Conv*	$3 \times 3, 512 \mid 2 \mid$ -	$64 \times 128 \times 512$
	C_7	$C_6, F_g(x)$	$\|\|$	-	$64 \times 128 \times 2560$
	C_8	C_7	Dconv	$3 \times 3, 512 \mid 1 \mid 2$	$64 \times 128 \times 512$
	C_9	C_7	Dconv	$3 \times 3, 512 \mid 1 \mid 4$	$64 \times 128 \times 512$
	C_{10}	C_7	Dconv	$3 \times 3, 512 \mid 1 \mid 8$	$64 \times 128 \times 512$
	C_{11}	C_7	Dconv	$3 \times 3, 512 \mid 1 \mid 16$	$64 \times 128 \times 512$
	C_{12}	C_8, C_9, C_{10}, C_{11}	$\bigoplus$	-	$64 \times 128 \times 512$
	C_{13}	C_{12}	Conv+Tanh	$1 \times 1, 512 \mid 1 \mid$ -	$64 \times 128 \times 512$
Decoder	C_{14}	C_{13}	Conv**	$3 \times 3, 512 \mid 1 \mid$ -	$64 \times 128 \times 512$
	C_{15}	C_{14}	Conv*	$3 \times 3, 512 \mid 1 \mid$ -	$64 \times 128 \times 512$
	C_{16}	C_{15}	Conv*	$7 \times 7, 256 \mid 1 \mid$ -	$64 \times 128 \times 256$
	C_{17}	C_{16}	Tconv*	$3 \times 3, 256 \mid 2 \mid$ -	$128 \times 256 \times 256$
	C_{18}	C_{17}	Conv*	$7 \times 7, 128 \mid 1 \mid$ -	$128 \times 256 \times 128$
	C_{19}	C_{18}	Tconv*	$3 \times 3, 64 \mid 2 \mid$ -	$256 \times 512 \times 64$
	C_{20}	C_{19}	Conv	$7 \times 7, 19 \mid 1 \mid$ -	$256 \times 512 \times 19$
	Upsampling	C_{20}	Interpolation	-	$512 \times 1024 \times 19$

segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017. 2

[3] Tian Qi Chen and Mark Schmidt. Fast patch-based style transfer of arbitrary style, 2016. 5

[4] Yi-Hsin Chen, Wei-Yu Chen, Yu-Ting Chen, Bo-Cheng Tsai, Yu-Chiang Frank Wang, and Min Sun. No more discrimination: Cross city adaptation of road scene segmenters. In *ICCV*, 2017. 4, 5

[5] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 2, 5

[6] Liang Du, Jingang Tan, Hongye Yang, Jianfeng Feng, Xiangyang Xue, Qibao Zheng, Xiaoqing Ye, and Xiaolin Zhang. Ssf-dan: Separated semantic feature based domain adaptation network for semantic segmentation. In *ICCV*, 2019. 7

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2

[8] Jiaxing Huang, Shijian Lu, Dayan Guan, and Xiaobing Zhang. Contextual-relation consistent domain adaptation for semantic segmentation. In *ECCV*, 2020. 7

[9] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017. 2, 5

[10] Philip T Jackson, Amir Atapour-Abarghouei, Stephen Bonner, Toby P Breckon, and Boguslaw Obara. Style augmentation: Data augmentation via style randomization. In *CVPR Workshops*, 2019. 2, 5

[11] Alexander B. Jung, Kentaro Wada, Jon Crall, Satoshi Tanaka, Jake Graving, Christoph Reinders, Sarthak Yadav, Joy Banerjee, Gábor Vecsei, Adam Kraft, Zheng Rui, Jirka Borovec, Christian Vallentin, Semen Zhydenko, Kilian Pfeiffer, Ben Cook, Ismael Fernández, François-Michel De Rainville, Chi-Hung Weng, Abner Ayala-Acevedo, Raphael Meudec, Matias Laporte, et al. imgaug. `https://github.com/aleju/imgaug`, 2020. Online; accessed 01-Feb-2020. 2, 5

[12] Guoliang Kang, Yunchao Wei, Yi Yang, Yueting Zhuang, and Alexander Hauptmann. Pixel-level cycle association: A new perspective for domain adaptive semantic segmentation. In *NeurIPS*, 2020. 7

[13] Myeongjin Kim and Hyeran Byun. Learning texture invariant representation for domain adaptation of semantic segmentation. In *CVPR*, 2020. 5, 7

[14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 3

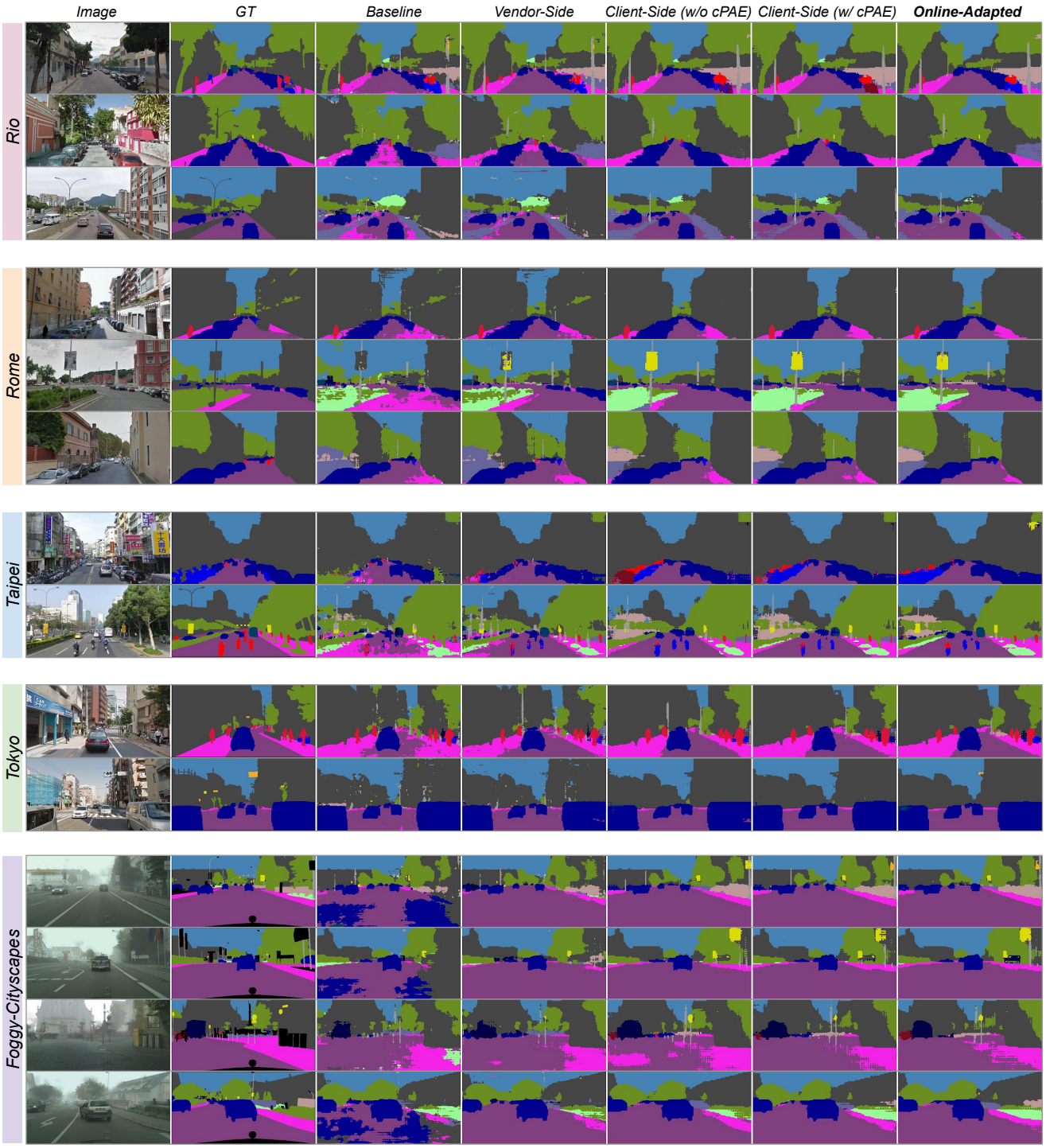

Figure 4. Qualitative evaluation of GTA5→Cityscapes and online adapted models on NTHU-Cross-City and Foggy-Cityscapes datasets. The performance generally improves from vendor-side to client-side to online-adapted model. *Best viewed in color.*

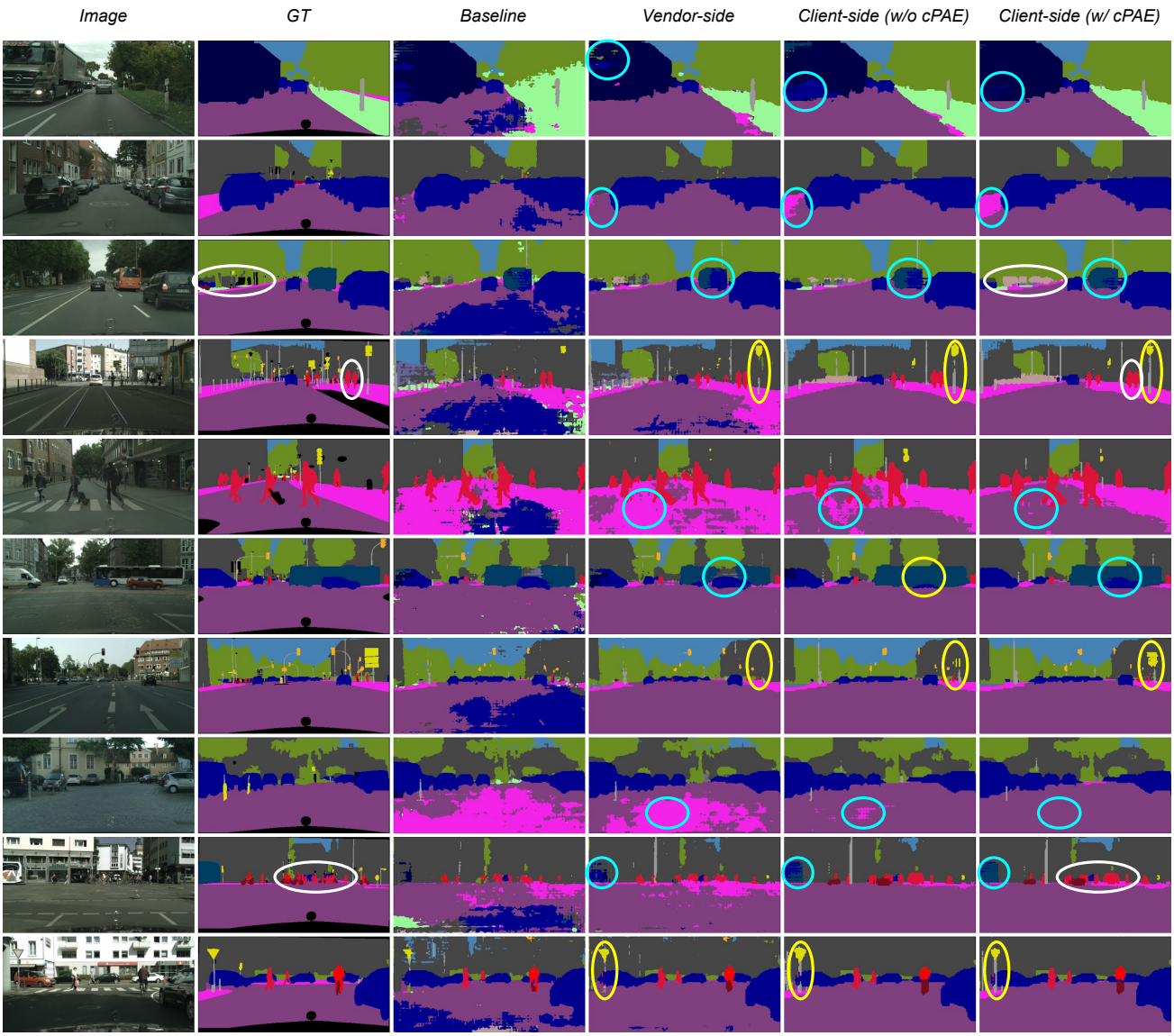

Figure 5. Qualitative evaluation of the proposed approach. Vendor-side model generalizes better than baseline but performs worse than client-side due to the domain gap. Inculcating prior knowledge from cPAE structurally regularizes the predictions and overcomes merged-region (yellow circle) and splitted-region (blue circle) problems. Some failure cases are also shown (white circle). *Best viewed in color.*

[15] Yunsheng Li, Lu Yuan, and Nuno Vasconcelos. Bidirectional learning for domain adaptation of semantic segmentation. In *CVPR*, 2019. 2, 3, 5, 7

[16] Qing Lian, Fengmao Lv, Lixin Duan, and Boqing Gong. Constructing self-motivated pyramid curriculums for cross-domain semantic segmentation: A non-adversarial approach. In *ICCV*, 2019. 7

[17] Yuang Liu, Wei Zhang, and Jun Wang. Source-free domain adaptation for semantic segmentation. In *CVPR*, 2021. 7

[18] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 2

[19] Fengmao Lv, Tao Liang, Xiang Chen, and Guosheng Lin. Cross-domain semantic segmentation via domain-invariant interactive relation transfer. In *CVPR*, 2020. 7

[20] Ke Mei, Chuang Zhu, Jiaqi Zou, and Shanghang Zhang. Instance adaptive self-training for unsupervised domain adaptation. In *ECCV*, 2020. 2, 7

[21] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019. 3

[22] Xingchao Peng, Zijun Huang, Ximeng Sun, and Kate Saenko. Domain agnostic learning with disentangled representations. In *ICML*, 2019. 2

[23] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *ECCV*, 2016. 2, 5

[24] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *CVPR*, 2016. 2, 5

[25] Christos Sakaridis, Dengxin Dai, Simon Hecker, and Luc Van Gool. Model adaptation with synthetic and real data for semantic dense foggy scene understanding. In *ECCV*, 2018. 4, 5

[26] Inkyu Shin, Sanghyun Woo, Fei Pan, and In So Kweon. Two-phase pseudo label densification for self-training based domain adaptation. In *ECCV*, 2020. 7

[27] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 2

[28] Prabhu Teja Sivaprasad and Francois Fleuret. Uncertainty reduction for model adaptation in semantic segmentation. In *CVPR*, 2021. 7

[29] Haoran Wang, Tong Shen, Wei Zhang, Lingyu Duan, and Tao Mei. Classes matter: A fine-grained adversarial approach to cross-domain semantic segmentation. In *ECCV*, 2020. 2, 7

[30] Jinyu Yang, Weizhi An, Sheng Wang, Xinliang Zhu, Chaochao Yan, and Junzhou Huang. Label-driven reconstruction for domain adaptation in semantic segmentation. In *ECCV*, 2020. 5, 7

[31] Jihan Yang, Ruijia Xu, Ruiyu Li, Xiaojuan Qi, Xiaoyong Shen, Guanbin Li, and Liang Lin. An adversarial perturbation oriented domain adaptation approach for semantic segmentation. In *AAAI*, 2020. 7

[32] Yanchao Yang, Dong Lao, Ganesh Sundaramoorthi, and Stefano Soatto. Phase consistent ecological domain adaptation. In *CVPR*, 2020. 5, 7

[33] Yanchao Yang and Stefano Soatto. Fda: Fourier domain adaptation for semantic segmentation. In *CVPR*, 2020. 2, 3, 5

[34] Qiming Zhang, Jing Zhang, Wei Liu, and Dacheng Tao. Category anchor-guided unsupervised domain adaptation for semantic segmentation. In *NeurIPS*, 2019. 7

[35] Yiheng Zhang, Zhaofan Qiu, Ting Yao, Chong-Wah Ngo, Dong Liu, and Tao Mei. Transferring and regularizing prediction for semantic segmentation. In *CVPR*, 2020. 7

[36] Zhedong Zheng and Yi Yang. Rectifying pseudo label learning via uncertainty estimation for domain adaptive semantic segmentation. *International Journal of Computer Vision (IJCV)*, 2020. 7

[37] Zhedong Zheng and Yi Yang. Unsupervised scene adaptation with memory regularization in vivo. In *IJCAI*, 2020. 7

[38] Yang Zou, Zhiding Yu, Xiaofeng Liu, BVK Kumar, and Jinsong Wang. Confidence regularized self-training. In *ICCV*, 2019. 2