Semi-Supervised Semantic Segmentation with Cross-Consistency Training: Supplementary Material

Yassine Ouali Céline Hudelot Myriam Tami
Université Paris-Saclay, CentraleSupélec, MICS, 91190, Gif-sur-Yvette, France.
{yassine.ouali,celine.hudelot,myriam.tami}@centralesupelec.fr

1. Comparison with Traditional Consistency Training Methods

In this section, we present the experiments to validate the observation that for semantic segmentation, enforcing a consistency over different perturbations applied to the encoder's outputs rather than the inputs is more aligned with the cluster assumption. To this end, we compare the proposed method with traditional consistency based SSL methods. Specifically, we conduct experiments using VAT [6] and Mean Teachers [9]. In VAT, at each training iteration, the unsupervised loss is computed as the KL-divergence between the model's predictions of the input $\mathbf{x}^u$ and its perturbed version $\mathbf{x}^u + r_{adv}$. For Mean Teachers, the discrepancy is measured using Mean Squared Error (MSE) between the prediction of the model and the prediction using an exponential weighted version of it. In this case, the noise is sampled at each training step with SGD.

Splits	n=500	n=1000
Baseline	51.4	59.2
Mean Teachers	51.3	59.4
VAT	50.0	57.9
CCT	**58.6**	**64.4**

Table 1. **CCT compared to traditional consistency methods.** We conduct an ablation study on PASCAL VOC, where we compare the performance of the baseline to the proposed method CCT, VAT and Mean Teachers. n represents the number of labeled examples.

The results are presented in Table 1. We see that applying the adversarial noise to inputs with VAT results in lower performance compared to the baseline. When using Mean Teachers, in which the noise is not implicitly added to the inputs, we obtain similar performance to the baseline. These results confirm our observation that enforcing a consistency over perturbations applied to the hidden representations is more aligned with the cluster assumption, thus yielding better results.

2. Additional Results and Evaluations

2.1. Distance Measures

In the experiments presented in the paper, MSE was used as a distance measure $\mathbf{d}(.,.)$ for the unsupervised loss $\mathcal{L}_u$, to measure the discrepancy between the main and auxiliary predictions. In this section, we investigate the effectiveness of other distance measures between the output probability distributions. Specifically, we compare the performance of MSE to the KL-divergence and the JS-divergence. For an unlabeled example $\mathbf{x}^u$, we obtain a main prediction y^u with the main decoder and an auxiliary prediction y_a^k with a given auxiliary decoder g_a^k. We compare the following distance measures:

$$\mathbf{d}_{\text{MSE}}(y^u, y_a^k) = \frac{1}{N}\sum_i^N (y^u(i) - y_a^k(i))^2 \tag{1}$$

$$\mathbf{d}_{\text{KL}}(y^u, y_a^k) = \frac{1}{N}\sum_i^N y^u(i) \log \frac{y^u(i)}{y_a^k(i)} \tag{2}$$

$$\mathbf{d}_{\text{JS}}(y^u, y_a^k) = \frac{1}{2}\mathbf{d}_{\text{KL}}(y^u, m) + \frac{1}{2}\mathbf{d}_{\text{KL}}(y_a^k, m) \tag{3}$$

where $m = \frac{1}{2}(y^u(i) + y_a^k(i))$ and $y^*(i)$ refers to the output probability distribution at a given spatial location i. The results of the comparison are shown in Table 2.

Splits	n=500	n=1000
Baseline	51.4	59.2
CCT KL	54.0	62.5
CCT JS	58.4	64.3
CCT MSE	**58.6**	**64.4**

Table 2. **CCT with different distance measures.** We compare the performance of MSE to the KL-divergence and the JS-divergence on PASCAL VOC dataset.

We observe similar performance with $\mathbf{d}_{\text{MSE}}$ and $\mathbf{d}_{\text{JS}}$, while we only obtain 2.6 and 3.3 points gain for $n = 500$ and $n = 1000$ respectively over the baseline when using

$\mathbf{d}_{\text{KL}}$. The low performance of $\mathbf{d}_{\text{KL}}$ might be due to its non-symmetric nature. With $\mathbf{d}_{\text{KL}}$, the auxiliary decoders are heavily penalized over sharp but wrong predictions, thus pushing them to produce uniform and uncertain outputs, and reducing the amount of training signal that can be extracted from the unlabeled examples. However, with $\mathbf{d}_{\text{JS}}$, which is a symmetrized and smoothed version of $\mathbf{d}_{\text{KL}}$, we can bypass the zero avoidance nature of the KL-divergence. Similarly, $\mathbf{d}_{\text{MSE}}$ can be seen as a multi-class Brier score [2] which is less sensitive to completely incorrect predictions, giving it similar properties to $\mathbf{d}_{\text{JS}}$ with a lower computational cost.

2.2. Confidence Masking and Pairwise Loss

Confidence Masking. (Conf-Mask) When training on the unlabeled examples, we use the main predictions as the source for consistency training, which may result in a corrupted training signal when based on uncertain predictions. A possible way to avoid this is masking the uncertain predictions. Given a main prediction y^u in the form of a probability distribution over the classes C at different spatial locations i. We compute the unsupervised loss $\mathcal{L}_u$ only over the pixels i with probability $\max_C y^u(i)$ greater than a fixed threshold β (*e.g.*, 0.5).

Pairwise Loss. (P-Wise) In CCT, we enforce the consistency of predictions only between the main and auxiliary decoders, without any pairwise consistency in between the auxiliary predictions. To investigate the effectiveness of enforcing such an additional pairwise consistency, we add the following an additional loss term $\mathcal{L}_{\text{P-Wise}}$ to the total loss in Paper Eq. (3) to penalize the auxiliary predictive variance:

$$\mathcal{L}_{\text{P-Wise}} = \frac{1}{K}\sum_{k=1}^{K}(y_a^k - \bar{y}_a)^2 \tag{4}$$

with $\bar{y}_a$ as the mean of the auxiliary predictions y_a^k. Given K auxiliary decoders, the computation of $\mathcal{L}_{\text{P-Wise}}$ is in the order of K^2. To reduce it, at each training iteration, we only compute $\mathcal{L}_{\text{P-Wise}}$ over a randomly chosen subset of the auxiliary predictions (*e.g.*, 8 out of $K = 30$).

Table 3 shows the results of the experiments when using CCT with Conf-Mask and P-Wise. Interestingly, we do not observe any gain over CCT when using Conf-Mask, indicating that using the uncertain main predictions to enforce the consistency does not hinder the performance. Additionally, adding a pairwise loss term results in lower performance compared to CCT, with 3 and 3.2 points difference in both settings, indicating that adding $\mathcal{L}_{\text{P-Wise}}$ can potentially compel the auxiliary decoders to produce similar predictions regardless of the applied perturbation, thus diminishing the representation learning of the encoder, and the performance of the segmentation network as a whole.

Splits	n=500	n=1000
Baseline	51.4	59.2
CCT +Conf-Mask	58.4	63.3
CCT +$\mathcal{L}_{\text{P-Wise}}$	55.6	61.2
CCT	**58.6**	**64.4**

Table 3. **CCT with** P-Wise **and** Conf-Mask. The results of the effect of adding a confidence masking over unsupervised loss and a pairwise loss between the auxiliary predictions on PASCAL VOC *val* set.

3. Algorithm

The proposed Cross-Consistency training method can be summarized by the following Algorithm:

Algorithm: Cross-Consistency Training (CCT).

Input: Labeled image $\mathbf{x}^l$, its pixel-level label y and unlabeled image $\mathbf{x}^u$

Require: Shared encoder h, main decoder g_m, K auxiliary decoders g_a^k

1) Forward $\mathbf{x}^l$ through the encoder and main decoder: $\hat{y}^l = g_m(h(\mathbf{x}^l))$
2) Forward the unlabeled input through the shared encoder: $\mathbf{z} = h(\mathbf{x}^u)$
3) Generate the main decoder's prediction for $\mathbf{x}^u$: $\hat{y}^u = g_m(\mathbf{z})$
4) Generate the aux. decoders predictions for $\mathbf{x}^u$:
   for k *in* $[1, K]$ **do**
      - Apply a given perturbation $\tilde{\mathbf{z}} = p_l(z)$
      - Forward through the aux. decoder k: $\hat{y}_a^i = g_a^k(\tilde{\mathbf{z}})$
   end
5) Training the network.
   $\mathcal{L}_s = \mathbf{H}(\hat{y}^u, y)$
   $\mathcal{L}_u = \frac{1}{K}\sum_k \mathbf{d}(\hat{y}^u, \hat{y}_a^k)$
   Update network by $\mathcal{L} = \mathcal{L}_s + \omega_u \mathcal{L}_u$

4. Further Investigation of The Cluster Assumption

The learned feature of a CNNs are generally more homogeneous, and at higher layers, the network learns to compose low level features into semantically meaningful representations while discarding high-frequency information (*e.g.*, texture). However, the leaned features in a segmentation network seem to have a unique property; the class boundaries correspond to low density regions, which are not observed in networks trained on other visual tasks (*e.g.*, classification, object detection). See Fig. 1 for an illustration of this difference.

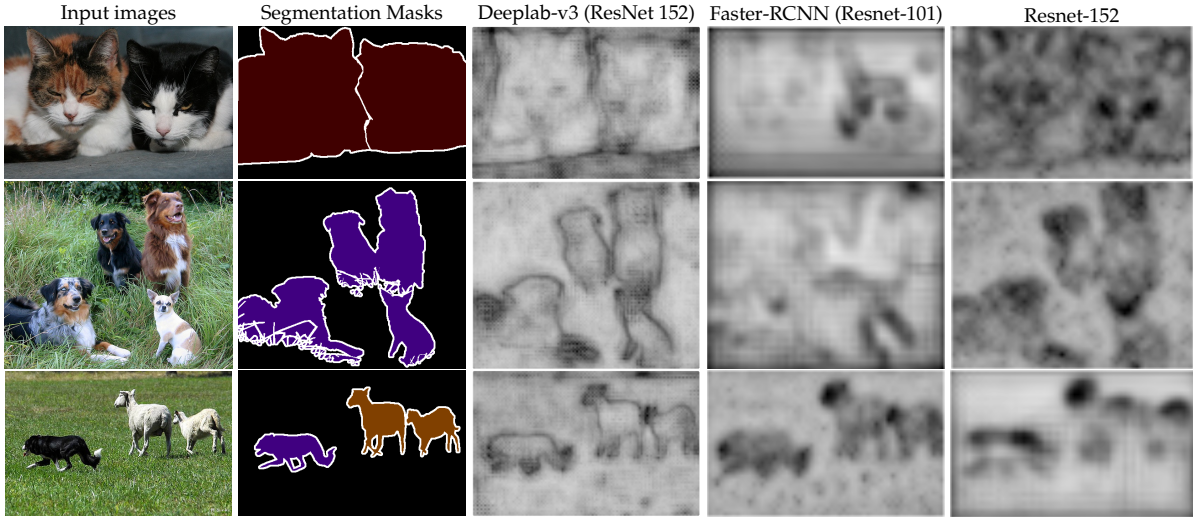


Figure 1. **The smoothness of CNNs features trained on different tasks.** (a) Examples from PASCAL VOC 2012 *train set*. (b) Results for a segmentation network. (c) Results for an object detection network. (b) Results for a classification network.

5. Adversarial Distribution Alignment

When applying CCT over multiple domains, and to further reduce the discrepancy between the encoder's representations of the two domains (*i.e.*, the empirical distribution mismatch measured by the $\mathcal{H}$-Divergence [1]), we investigate the addition of a discriminator branch g_d, which takes as input the encoder's representation $\mathbf{z}$, and predict 0 for examples from $\mathcal{D}^{(1)}$ and 1 for examples from $\mathcal{D}^{(2)}$. Hence, we add the following adversarial loss to the total loss in Eq. (1):

$$\begin{aligned}\mathcal{L}_{adv} = &\frac{1}{|\mathcal{D}^{(1)}|}\sum_{\mathbf{x}_i\in\mathcal{D}^{(1)}}\log(g_d(\mathbf{z}_i)) \\ &+\frac{1}{|\mathcal{D}^{(2)}|}\sum_{\mathbf{x}_i\in\mathcal{D}^{(2)}}(1-\log(g_d(\mathbf{z}_i)))\end{aligned} \tag{5}$$

The encoder and the discriminator branch are competitors within a min-max framework, *i.e.*, the training objective is $\max_{g_d}\min_{h}\mathcal{L}_{adv}$, which can be directly optimized using a gradient reversal layer as in [3]. The total loss in this case is:

$$\mathcal{L} = \mathcal{L}_s + \lambda_{adv}\mathcal{L}_{adv} + \omega_u\mathcal{L}_u \tag{6}$$

Method	n=50			n=100		
	CS	CVD	Avg.	CS	CVD	Avg.
Baseline	31.2	40.0	35.6	37.3	34.4	35.9
CCT	35.0	53.7	**44.4**	40.1	55.7	**47.9**
CCT +$\mathcal{L}_{avd}$	35.3	49.2	42.2	37.7	52.8	45.2

Table 4. **CCT applied to CS+CVD.**

For the discriminator branch, similar to [4], we use a fully convolutional discriminator, with a series of 3×3 convolutions and Leaky ReLU non-linearities as shown in Table 5. The outputs are of the same size as the encoder outputs (*i.e.* with an input image of spatial dimensions $H\times W$, the outputs of g_d are of size $2\times\frac{H}{8}\times\frac{W}{8}$).

Description	Resolution $\times$ channels
Conv $3\times3\times64$	$\frac{1}{8}\times64$
LeakyReLU	
Conv $3\times3\times128$	$\frac{1}{8}\times128$
LeakyReLU	
Conv $3\times3\times256$	$\frac{1}{8}\times256$
LeakyReLU	
Conv $3\times3\times512$	$\frac{1}{8}\times512$
LeakyReLU	
Conv $1\times1\times2$	$\frac{1}{8}\times2$

Table 5. **Discriminator Branch.** The added discriminator branch on top of the encoder, in order to further push towards an invariance of the encoder's representations between the different domains.

The results are shown in Table 5. Surprisingly, adding a discriminator branch diminishes the performance of the segmentation network, hinting to possible learning conflicts between CCT and the adversarial loss.

6. Multi-scale Inference

To further enhance the predictions of our segmentation network, we conduct additional evaluations on PASCAL VOC using multi-scale to simulate a similar situation to training where we apply random scaling between 0.5 and 2, random croping and random horizontal flip. We apply the same augmentations during test. For a given test image,

we create 5 versions using 5 scales: 0.5, 0.75, 1, 1.25 and 1.5, each image is also flipped horizontally, resulting in 10 versions of the test image. The model's prediction are computed for each image, rescaled to the original size, and are then aggregated by pixel-wise average pooling. The final result is obtain by taking the $argmax$ over the classes for each spatial location.

In Table 6, we report the results obtained with multi-scale inference.

	n	mIoU
CCT	1000	67.3 (+3.3)
CCT	1500	73.4 (+4)
CCT +9k Image-level labels	1500	75.1 (+2.9)

Table 6. **CCT results with multi-scale inference**. The mIoU when we apply multi-scale inference on PASCAL VOC *val* set.

7. Virtual Adversarial Training (VAT)

Without the label information in a semi-supervised setting, VAT [6] lends itself as a consistency regularization technique. It trains the output distribution to be isotropically smooth around each data point by selectively smoothing the model in its most anisotropic direction. In our case, we apply the adversarial perturbation r_{adv} to the encoder output $\mathbf{z} = h(\mathbf{x}^u)$. For a given auxiliary decoder g_a^k, we would like to compute the adversarial perturbation r_{adv} that will alter its predictions the most. We start by sampling a Gaussian noise r of the same size as $\mathbf{z}$, compute its gradients $grad_r$ with respect the loss between the two predictions, with and without the injections of the noise r (*i.e.*, KL-divergence is used as a distance measure $\mathbf{d}(.,.)$). r_{adv} can then be obtained by normalizing and scaling $grad_r$ by a hyperparameter ϵ. This can be written as follows:

$$r \sim \mathcal{N}\left(0, \frac{\xi}{\sqrt{dim(\mathbf{z})}} I\right) \tag{7}$$

$$grad_r = \nabla_r \mathbf{d}\left(g_a^k(\mathbf{z}), g_a^k(\mathbf{z}+r)\right) \tag{8}$$

$$r_{adv} = \epsilon \frac{grad_r}{\|grad_r\|} \tag{9}$$

Finally, the perturbed input to g_a^k is $\tilde{\mathbf{z}} = r_{adv} + \mathbf{z}$. The main drawback of such method is requiring multiple forward and backward passes for each training iteration to compute r_{adv}. In our case, the amount of computations needed are reduced given the small size of the auxiliary decoders.

8. Dataset sizes

For the size of each split of the datasets used in our experiments, see Table 7.

Splits	Train	Val	Test
PASCAL VOC	10582	1449	1456
Cityscapes	2975	500	1525
CamVid	367	101	233
SUN RGB-D	5285	-	5050

Table 7. **Semantic Segmentation Datasets.** The size of each split of the datasets used in the experiments.

9. Further Experimental Details

For the experiments throughout the paper, we used a ResNet 50 and a PSP module [10] for the encoder. As for the decoders, we used an initial 1×1 convolutions to adapt the depth to the number of classes C, followed by a series of 1×1 sub-pixel convolutions [8] (*i.e.*, PixelShuffle) to upsample the feature maps to the original size. For details see Table 8.

Encoder		Decoder	
Description	Resolution × channels	Description	Resolution × channels
ResNet 50	$\frac{1}{8} \times 2048$	Conv $1 \times 1 \times C$	$\frac{1}{8} \times C$
PSPModule [10]	$\frac{1}{8} \times 512$	Conv $1 \times 1 \times 4C$	$\frac{1}{8} \times 4C$
		PixelShuffle	$\frac{1}{4} \times C$
		Conv $1 \times 1 \times 4C$	$\frac{1}{4} \times 4C$
		PixelShuffle	$\frac{1}{2} \times C$
		Conv $1 \times 1 \times 4C$	$\frac{1}{2} \times 4C$
		PixelShuffle	$1 \times C$

Table 8. **Encoder-Decoder architecture.** Showing the layer type, the number of the outputs channels and the spatial resolution.

Inference Settings. For PASCAL VOC, during the ablation studies reported in Paper Fig. 6, in order to reduce the training time, we trained on smaller size image. Specifically, we resize the bigger side to 300 and randomly take crops of size 240×240. For the comparisons with state-of-the-art we resize the bigger side to 400 and take crops of size 321×321 and conduct the inference on the original sized images. For the rest of the datasets, the evaluation is conducted on the same sizes as the ones used during training.

10. Hyperparameters

In order to present a realistic evaluation of the proposed method, and following the practical considerations mentioned in [7]. We avoid any form of intensive hyperparameter search, be it that of the perturbation functions, model architecture or training settings. We choose the hyperparameters that resulted in stable training by hand, we do expect however that better performances can be achieved with a comprehensive search. The hyperparameters settings used in the experiments are summarized in Table 9.

Training	
SGD	
Learning rate	10^{-2}
Momentum	0.9
Weight Decay	10^{-4}
Number of training epochs	
PASCAL VOC	50
CamVid	50
Cityscapes & CamVid	50
Cityscapes & SUN RGB-D	100
Losses	
Unsupervised loss $\mathcal{L}_u$	
Rampup periode for $\mathcal{L}_u$	0.1
$\mathcal{L}_u$ weight λ_u	30
Weakly-supervised loss $\mathcal{L}_w$	
Rampup periode for $\mathcal{L}_w$	0.1
$\mathcal{L}_w$ weight λ_w	0.4
Annealed Cross-Entropy loss ab-CE	
Rampup periode	0.5
Final threshold	0.9
Adversarial loss $\mathcal{L}_{adv}$	
Weight λ_{adv}	2.10^{-2}
Perturbation Functions	
I-VAT	
VAT ϵ	2.0
VAT ξ	10^{-6}
DropOut	
Dropout rate p	0.5
G-Cutout	
Area of the dropped region	0.4
F-Drop	
Drop threshold range	$[0.6, 0.9]$
F-Noise	
The uniform noise range	$[-0.3, 0.3]$

Table 9. **Hyperparameters.** The hyperparameter settings used in our experiments.

11. Ramp-up functions

For the unsupervised loss in Paper Eq. (2), the weighting function w_u is gradually increased from 0 up to a fixed final weight λ_u. The rate of increase can follow many possible rates depending on the schedule used. Fig. 2 shows different ramp-up schedules. For our experiments, following [5], w_u ramps-up following an exp-schedule:

$$w_u(t) = \min(\lambda_u, e^{5(\frac{t}{T}-1)} \times \lambda_u) \quad (10)$$

with t as the current training iteration and T as the desired ramp-up length (*e.g.*, the first 10% of training time). Similarly, the threshold η in the ab-CE loss (Paper Eq. (4)) is gradually increased starting from $1/C$, with C as the number of classes, up to a final threshold α (*e.g.*, 0.9) within a ramp-up period T (*e.g.*, the first 40% of training time). For ab-CE, we use a log-schedule to quickly increase η in the beginning of training:

$$\eta(t) = \min(\alpha, (1 - e^{-5\frac{t}{T}}) \times (\alpha - 1/C) + 1/C) \quad (11)$$

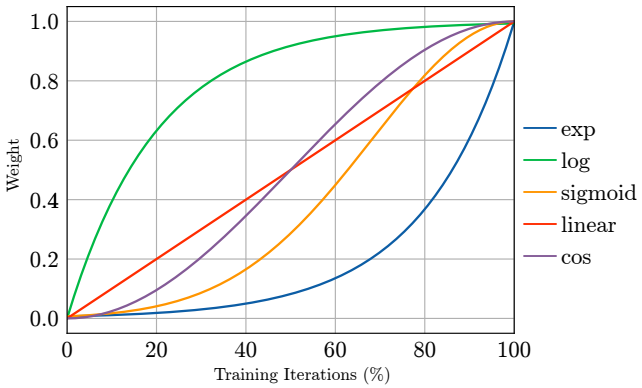


Figure 2. **Different ramp-up schedules.**

12. Computational Overhead

Decoders	Input size	GPU memory (MB)	GPU time (ms)
Main Decoder	96 × 96	139	2.0
DropOut		139	2.6
F-Drop		157	3.0
F-Noise		175	45.7
I-VAT		463	82.3
Obj-Msk		463	2.7
Con-Msk		463	2.4
G-Cutout		463	3.3
Main Decoder	256 × 128	457	4.0
DropOut		520	4.7
F-Drop		520	5.2
F-Noise		584	149.5
I-VAT		1592	176.0
Obj-Msk		1592	4.7
Con-Msk		1592	4.6
G-Cutout		1592	7.1

Table 10. **Computation and memory statistics.** Comparisons between the main and auxiliary decoders with different perturbation functions. The channel numbers of the input feature maps $\mathbf{z}$ is 512. The lower the values, the better.

In order to present a comparison between the computational overhead of the different types of auxiliary decoders, we present various computation and memory statistics in Table 10. We observe that for the majority of the auxiliary decoders, the GPU time is similar to that of the main decoder. However, we see a significant increase for I-VAT given the multiple forward and backward passes required to compute the adversarial perturbation. F-Noise also results in high GPU time due to the sampling procedure. To this end we reduce the number of I-VAT decoders (*e.g.*, $K = 2$

for our experiments). For F-Noise, for an input tensor of size $B \times C \times H \times W$ with B as the batch size, instead of sampling a noise tensor $\mathbf{N}$ of the same size, we sample a tensor of size $1 \times C \times H \times W$ and apply it over the whole batch. Significantly reduces the computation the computation time without impacting the performance.

13. Qualitative Results

Pseudo-Labels

Fig. 3 shows some qualitative results of the generated pseudo pixel-level labels using the available image-level labels. We observe that when considering regions with high attention scores (*i.e.* > 0.3), the assigned classes do correspond in most cases to true positives.

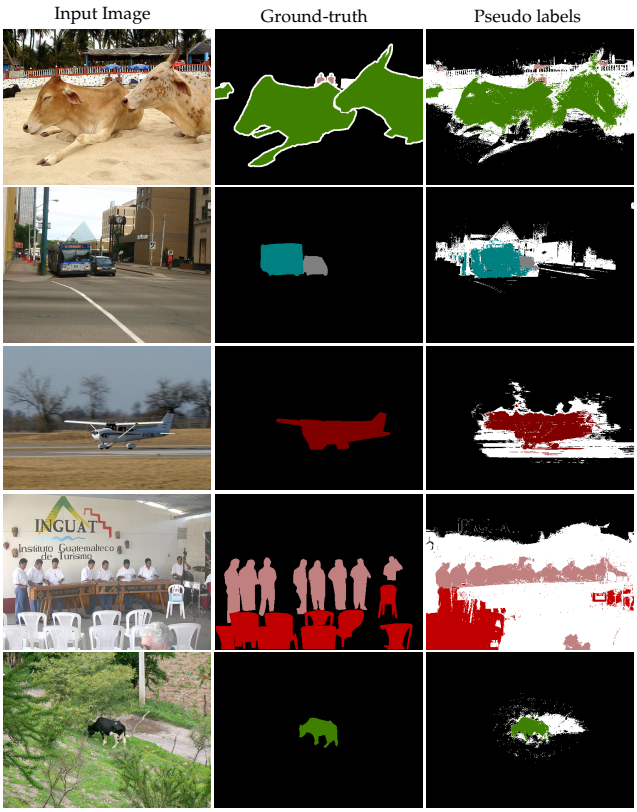
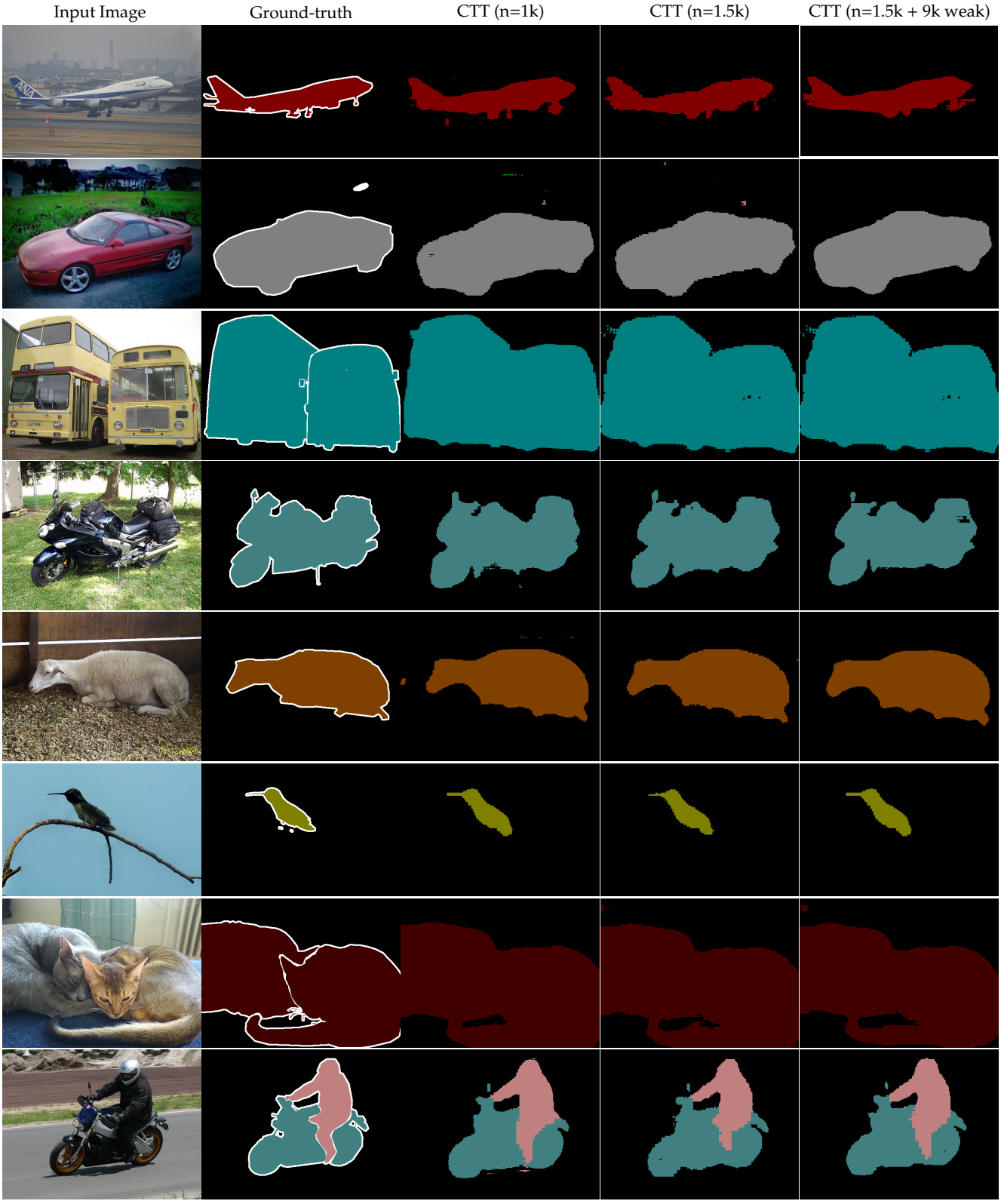


Figure 3. **The generated pseudo pixel-level labels.** Instances of the generated pseudo pixel-level labels from PASCAL VOC *train* set. The white regions correspond to the ignored pixels.

Predictions

Qualitative results of CCT on PASCAL VOC *val* images with different values of n are presented in Fig. 4.


Input Image
Ground-truth
CTT (n=1k)
CTT (n=1.5k)
CTT (n=1.5k + 9k weak)

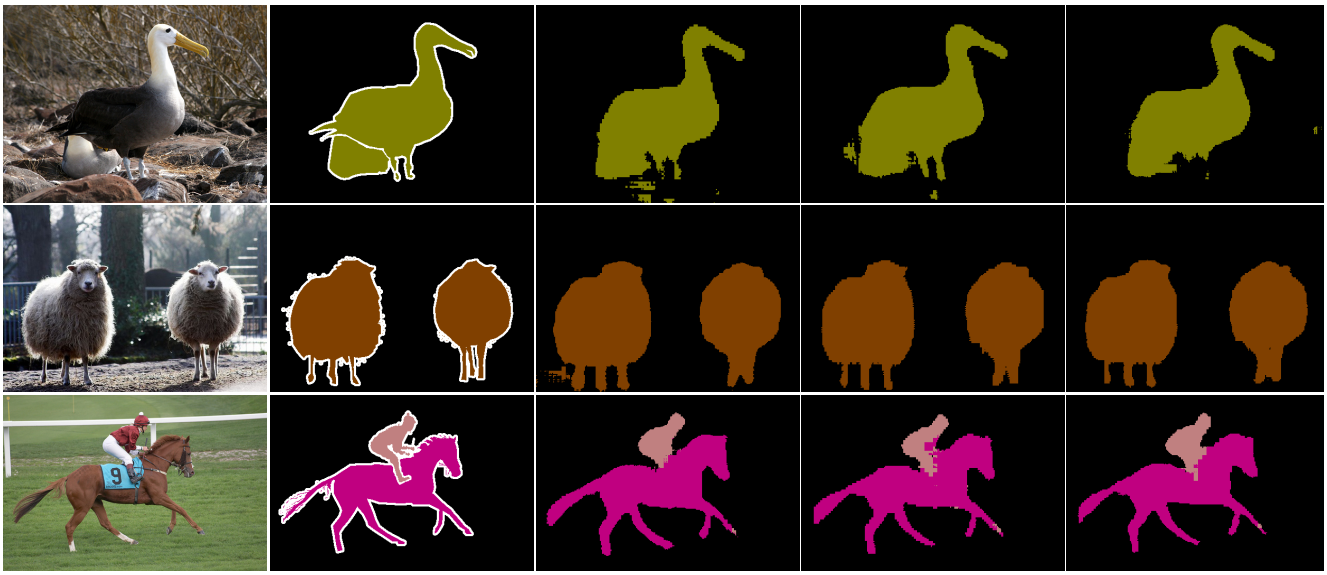


Figure 4. **CCT results.** Semantic Segmentation Results on the PASCAL VOC *val* images.

References

[1] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010.

[2] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. Mixmatch: A holistic approach to semi-supervised learning. *arXiv preprint arXiv:1905.02249*, 2019.

[3] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv:1409.7495*, 2014.

[4] Wei-Chih Hung, Yi-Hsuan Tsai, Yan-Ting Liou, Yen-Yu Lin, and Ming-Hsuan Yang. Adversarial learning for semi-supervised semantic segmentation. *arXiv preprint arXiv:1802.07934*, 2018.

[5] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning, 2016.

[6] Takeru Miyato, Shin-ichi Maeda, Shin Ishii, and Masanori Koyama. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[7] Avital Oliver, Augustus Odena, Colin A Raffel, Ekin Dogus Cubuk, and Ian Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. In *Advances in Neural Information Processing Systems*, pages 3235–3246, 2018.

[8] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1874–1883, 2016.

[9] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results, 2017.

[10] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.