Balancing Logit Variation for Long-tailed Semantic Segmentation
Supplementary Material

1. Overview

In this supplementary material, we first provide more details for reproducibility in Sec. 2. We further explore a potential improvement of BLV and corresponding preliminary results in Sec. 3. Then we demonstrate our BLV with more UDA methods on both the *GTA5* $\rightarrow$ *Cityscapes* and *SYNTHIA* $\rightarrow$ *Cityscapes* settings in Sec. 4. Intuitive feature space visualization is demonstrated by t-SNE method in Sec. 5. Pseudo-code for direct understanding of BLV is provided in Sec. 6. Information about computational overhead and distribution estimation is exhibited in Sec. 7 and Sec. 8.

2. More Details for Reproducibility

Details for parameters. As we mentioned in the paper, the only parameter for BLV is the σ in Eq. (3).

We set $\sigma = 4$ for unsupervised domain adaptive semantic segmentation task under the *SYNTHIA* $\rightarrow$ *Cityscapes* setting. For all the other tasks, we set $\sigma = 6$ consistently. Besides, the $\delta(\sigma)$ term is clamped into $[0, 1]$ to avoid particularly large values that makes training unstable.

Details for data augmentation. We follow DACS [12], using color jitter, Gaussian blur, and ClassMix [11] as the augmentation selections.

3. More Exploration of Variation

We explore the improvement over BLV. We set the σ in Eq. (3) as a temporal variable: $\sigma(t)$, where t denotes current iteration, t_{mid} and σ_0 are hyper-parameters with preset values. Fig. 1 depicts how σ changes with iterations.

The main idea is to let the perturbation increase gradually before t_{mid} to obtain an effective variation. After t_{mid}, we should let the variation decrease so that the model convergence is not affected. This exploration is easy to implement. We conducted the experiment under *GTA5* $\rightarrow$ *Cityscapes* benchmark. $t_{end} = 40k$, $t_{mid} = 30k$ and $\sigma_0 = 6$.

The result is demonstrated in Tab. 1. The baseline is DAFormer‡ model. This table suggests that this "temporal variable" does improve the original BLV. The overall result indicates that there is an opportunity to further improve our approach.

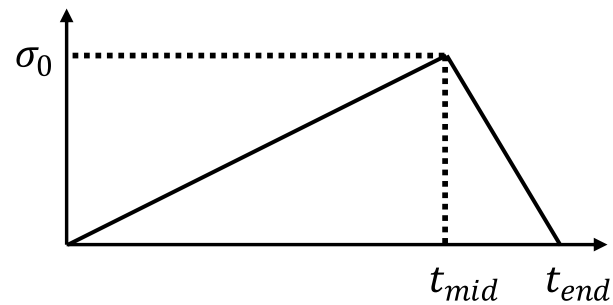


Figure 1. σ **that changes with training iterations.** t_{end} is the total iterations. t_{mid} is the turning point of σ with a corresponding maximum value σ_0.

Table 1. **Exploration on temporal variation of BLV.** "+tv" denotes our proposed "temporal variable".

Baseline	BLV	**BLV +tv**
68.3	69.6 ↑ 1.3	**70.0** ↑ 1.7

4. More Comparisons on UDA Benchmark

We add more comparisons of BLV with previous UDA methods for GTA5 $\rightarrow$ Cityscapes in Tab. 2 and for SYNTHIA $\rightarrow$ Cityscapes in Tab. 3.

We include following methods for comparision: AdaptSeg [13], CyCADA [5], ADVENT [15], FADA [16], CBST [21], IAST [10], CAG [19], ProDA [18], SAC [1], CPSL [7], PLCA [6], RCCR [20], and MCS [3]. All methods in Tab. 2 and Tab. 3 are based on ResNet-101 [4] + DeepLab V2 [2].

BLV surpasses other alternatives by a large margin, achieving mIoU of 59.0% on *GTA5* $\rightarrow$ *Cityscapes*, and 56.8% over 16 classes and 63.3% over 13 classes on *SYNTHIA* $\rightarrow$ *Cityscapes*, respectively.

5. Visualization on Feature Space

We use t-SNE [14] to visualize the logit feature space in Fig. 2. In terms of the degree of confusion in the feature space, BLV improves the baseline and proves its superiority.

Table 2. Comparison with state-of-the-art alternatives on *GTA5* → *Cityscapes* benchmark with ResNet-101 [4] and DeepLab-V2 [2]. The results are averaged over 3 random seeds. The top performance is highlighted in **bold** font and the second score is *underlined*.

Method	Road	S.walk	Build.	Wall*	Fence*	Pole*	T.light	Sign	Veget.	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	M.bike	Bike	mIoU
source only	70.2	14.6	71.3	24.1	15.3	25.5	32.1	13.5	82.9	25.1	78.0	56.2	33.3	76.3	26.6	29.8	12.3	28.5	18.0	38.6
AdaptSeg [13]	86.5	36.0	79.9	23.4	23.3	23.9	35.2	14.8	83.4	33.3	75.6	58.5	27.6	73.7	32.5	35.4	3.9	30.1	28.1	41.4
CyCADA [5]	86.7	35.6	80.1	19.8	17.5	38.0	39.9	41.5	82.7	27.9	73.6	64.9	19.0	65.0	12.0	28.6	4.5	31.1	42.0	42.7
ADVENT [15]	89.4	33.1	81.0	26.6	26.8	27.2	33.5	24.7	83.9	36.7	78.8	58.7	30.5	84.8	38.5	44.5	1.7	31.6	32.4	45.5
CBST [21]	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
PCLA [6]	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
FADA [16]	92.5	47.5	85.1	37.6	32.8	33.4	33.8	18.4	85.3	37.7	83.5	63.2	39.7	87.5	32.9	47.8	1.6	34.9	39.5	49.2
MCS [3]	92.6	54.0	85.4	35.0	26.0	32.4	41.2	29.7	85.1	40.9	85.4	62.6	34.7	85.7	35.6	50.8	2.4	31.0	34.0	49.7
CAG [19]	90.4	51.6	83.8	34.2	27.8	38.4	25.3	48.4	85.4	38.2	78.1	58.6	34.6	84.7	21.9	42.7	**41.1**	29.3	37.2	50.2
FDA [17]	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
PIT [9]	87.5	43.4	78.8	31.2	30.2	36.3	39.3	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
IAST [10]	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
DACS [12]	89.9	39.7	87.9	30.7	39.5	38.5	46.4	52.8	88.0	**44.0**	88.8	67.2	35.8	84.5	45.7	50.2	0.0	27.3	34.0	52.1
RCCR [20]	93.7	60.4	86.5	41.1	32.0	37.3	38.7	38.6	87.2	43.0	85.5	65.4	35.1	88.3	41.8	51.6	0.0	38.0	52.1	53.5
ProDA [18]	91.5	52.4	82.9	42.0	35.7	40.0	44.4	43.3	87.0	43.8	79.5	66.5	31.4	86.7	41.1	52.5	0.0	45.4	53.8	53.7
CPSL [7]	91.7	52.9	83.6	**43.0**	32.3	**43.7**	51.3	42.8	85.4	37.6	81.1	**69.5**	30.0	88.1	44.1	**59.9**	24.9	**47.2**	48.4	55.7
BLV (ours)	**94.9**	**68.2**	**88.8**	40.9	**37.1**	42.6	**52.1**	**62.1**	**88.3**	43.3	**89.3**	68.6	**44.5**	**88.9**	**56.0**	54.6	3.8	38.6	**58.3**	**59.0**

Table 3. Comparison with state-of-the-art alternatives on *SYNTHIA* → *Cityscapes* benchmark with ResNet-101 [4] and DeepLab-V2 [2]. The results are averaged over 3 random seeds. The mIoU and the mIoU* indicate we compute mean IoU over 16 and 13 categories, respectively. The top performance is highlighted in **bold** font and the second score is *underlined*.

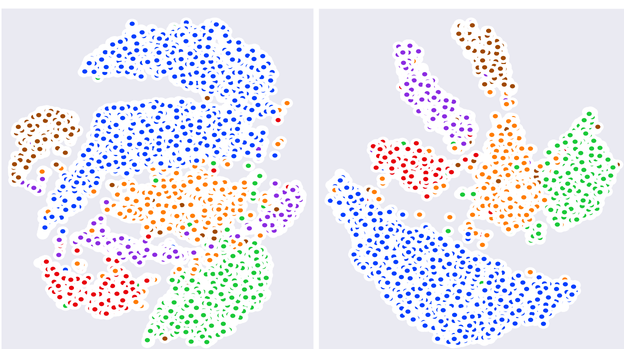
Method	Road	S.walk	Build.	Wall*	Fence*	Pole*	T.light	Sign	Veget.	Sky	Person	Rider	Car	Bus	M.bike	Bike	mIoU	mIoU*
source only†	55.6	23.8	74.6	9.2	0.2	24.4	6.1	12.1	74.8	79.0	55.3	19.1	39.6	23.3	13.7	25.0	33.5	38.6
AdaptSeg [13]	79.2	37.2	78.8	-	-	-	9.9	10.5	78.2	80.5	53.5	19.6	67.0	29.5	21.6	31.3	-	45.9
ADVENT [15]	85.6	42.2	79.7	8.7	0.4	25.9	5.4	8.1	80.4	84.1	57.9	23.8	73.3	36.4	14.2	33.0	41.2	48.0
CBST [21]	68.0	29.9	76.3	10.8	1.4	33.9	22.8	29.5	77.6	78.3	60.6	28.3	81.6	23.5	18.8	39.8	42.6	48.9
CAG [19]	84.7	40.8	81.7	7.8	0.0	35.1	13.3	22.7	84.5	77.6	64.2	27.8	80.9	19.7	22.7	48.3	44.5	51.5
PIT [9]	83.1	27.6	81.5	8.9	0.3	21.8	26.4	33.8	76.4	78.8	64.2	27.6	79.6	31.2	31.0	31.3	44.0	51.8
FDA [17]	79.3	35.0	73.2	-	-	-	19.9	24.0	61.7	82.6	61.4	31.1	83.9	40.8	38.4	51.1	-	52.5
FADA [16]	84.5	40.1	83.1	4.8	0.0	34.3	20.1	27.2	84.8	84.0	53.5	22.6	85.4	43.7	26.8	27.8	45.2	52.5
MCS [3]	88.3	**47.3**	80.1	-	-	-	21.6	20.2	79.6	82.1	59.0	28.2	82.0	39.2	17.3	46.7	-	53.2
PyCDA [8]	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
PLCA [6]	82.6	29.0	81.0	11.2	0.2	33.6	24.9	18.3	82.8	82.3	62.1	26.5	85.6	48.9	26.8	52.2	46.8	54.0
DACS [12]	80.6	25.1	81.9	21.5	2.9	37.2	22.7	24.0	83.7	**90.8**	67.6	38.3	82.9	38.9	28.5	47.6	48.3	54.8
RCCR [20]	79.4	45.3	83.3	-	-	-	24.7	29.6	68.9	87.5	63.1	33.8	87.0	51.0	32.1	52.1	-	56.8
IAST [10]	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
ProDA [18]	87.1	44.0	83.2	**26.9**	0.7	42.0	45.8	34.2	86.7	81.3	68.4	22.1	87.7	50.0	31.4	38.6	51.9	58.5
SAC [1]	**89.3**	47.2	85.5	26.5	1.3	**43.0**	45.5	32.0	**87.1**	89.3	63.6	25.4	86.9	35.6	30.4	53.0	52.6	59.3
CPSL [7]	87.3	44.4	83.8	25.0	0.4	42.9	47.5	32.4	86.5	83.3	69.6	29.1	**89.4**	**52.1**	42.6	54.1	54.4	61.7
BLV (ours)	70.4	28.9	**89.2**	25.2	**19.9**	40.2	**55.2**	**50.3**	86.9	84.2	**76.4**	**40.5**	79.6	51.3	**49.2**	**61.2**	**56.8**	**63.3**

6. Pseudo-code

To make BLV easy to understand, we provide pseudo-code in a Pytorch-like style in Algorithm 1.

Algorithm 1 Pseudo-code of BLV in a PyTorch-like style.

```
# cls_num_list: a list containing the number of pixels
     of each category.
# pred: model output logits
# target: ground-truch label
# sigma: hyper-paramter

def BLV_Loss(pred, target, sigma, cls_num_list):

   cls_num_list = torch.cuda.FloatTensor(cls_num_list)
   frequency_list = torch.log(torch.sum(cls_num_list))
        - torch.log(cls_num_list)

   sampler = torch.distributions.normal.Normal(0,
        sigma)

   noise = sampler.sample(pred.shape).clamp(0, 1).to(
        pred.device)

   pred = pred + (noise.abs().permute(0, 2, 3, 1) *
        frequency_list / frequency_list.max()).permute
        (0, 3, 1, 2)

   loss = torch.nn.functional.cross_entropy(pred,
        target)

   return loss
```



(a) *w/o* Logit Variation (b) *w/* Balanced Logit Variation

Figure 2. **t-SNE visualization from the logit feature space.** (a) Without logit variation, the spacing between instances of different categories is small, resulting in easy misclassification. (b) With balanced logit variation, instances are easier to distinguish.

7. Computational Overhead

We list parts of training time comparison in Tab. 4, which suggests the computational overhead introduced by BLV is limited and has a trivial impact on the overall training time. As a plug-in design, BLV demonstrates its superiority.

Table 4. Training time comparison (with 8 V100 GPUs).

Backbone	Decoder	*w/o* BLV	*w/* BLV
HRNet-18	OCRHead	20h11m	21h07m (+4.6%)
ResNet50	UperHead	16h20m	16h47m (+2.8%)

8. Estimation from the Labeled Data

Under semi-supervised settings, we have tried to estimate the distribution from the labeled data and found the overall performance improvement is limited. The results are presented in Tab. 5. We think this is due to the bias in estimating the full distribution from a small number of samples.

Table 5. Experiments under semi-supervised settings. ST indicates self-training baseline, † denotes estimation from the labeled data only, and ‡ means BLV estimation strategy described in the paper.

Partition	ST	ST+BLV†	ST+BLV‡
1/16	68.21	68.22 ↑ 0.01	**69.26 ↑ 1.05**
1/8	72.01	72.21 ↑ 0.20	**73.27 ↑ 1.26**

References

[1] Nikita Araslanov and Stefan Roth. Self-supervised augmentation consistency for adapting semantic segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021. 1, 2

[2] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(4):834–848, 2017. 1, 2

[3] Inseop Chung, Daesik Kim, and Nojun Kwak. Maximizing cosine similarity between spatial features for unsupervised domain adaptation in semantic segmentation. In *IEEE Winter Conf. Appl. Comput. Vis.*, 2022. 1, 2

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016. 1, 2

[5] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *Int. Conf. Mach. Learn.*, 2018. 1, 2

[6] Guoliang Kang, Yunchao Wei, Yi Yang, Yueting Zhuang, and Alexander Hauptmann. Pixel-level cycle association: A new perspective for domain adaptive semantic segmentation. In *Adv. Neural Inform. Process. Syst.*, 2020. 1, 2

[7] Ruihuang Li, Shuai Li, Chenhang He, Yabin Zhang, Xu Jia, and Lei Zhang. Class-balanced pixel-level self-labeling for domain adaptive semantic segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 11593–11603, 2022. 1, 2

[8] Qing Lian, Fengmao Lv, Lixin Duan, and Boqing Gong. Constructing self-motivated pyramid curriculums for cross-domain semantic segmentation: A non-adversarial approach. In *Int. Conf. Comput. Vis.*, 2019. 2

[9] Fengmao Lv, Tao Liang, Xiang Chen, and Guosheng Lin. Cross-domain semantic segmentation via domain-invariant interactive relation transfer. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020. 2

[10] Ke Mei, Chuang Zhu, Jiaqi Zou, and Shanghang Zhang. Instance adaptive self-training for unsupervised domain adaptation. In *Eur. Conf. Comput. Vis.*, 2020. 1, 2

[11] Viktor Olsson, Wilhelm Tranheden, Juliano Pinto, and Lennart Svensson. Classmix: Segmentation-based data augmentation for semi-supervised learning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021. 1

[12] Wilhelm Tranheden, Viktor Olsson, Juliano Pinto, and Lennart Svensson. Dacs: Domain adaptation via cross-domain mixed sampling. In *IEEE Winter Conf. Appl. Comput. Vis.*, 2021. 1, 2

[13] Yi-Hsuan Tsai, Wei-Chih Hung, Samuel Schulter, Kihyuk Sohn, Ming-Hsuan Yang, and Manmohan Chandraker. Learning to adapt structured output space for semantic segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018. 1, 2

[14] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. 1

[15] Tuan-Hung Vu, Himalaya Jain, Maxime Bucher, Matthieu Cord, and Patrick Pérez. Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019. 1, 2

[16] Haoran Wang, Tong Shen, Wei Zhang, Ling-Yu Duan, and Tao Mei. Classes matter: A fine-grained adversarial approach to cross-domain semantic segmentation. In *Eur. Conf. Comput. Vis.*, 2020. 1, 2

[17] Yanchao Yang and Stefano Soatto. Fda: Fourier domain adaptation for semantic segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020. 2

[18] Pan Zhang, Bo Zhang, Ting Zhang, Dong Chen, Yong Wang, and Fang Wen. Prototypical pseudo label denoising and target structure learning for domain adaptive semantic segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 12414–12424, 2021. 1, 2

[19] Qiming Zhang, Jing Zhang, Wei Liu, and Dacheng Tao. Category anchor-guided unsupervised domain adaptation for semantic segmentation. In *Adv. Neural Inform. Process. Syst.*, 2019. 1, 2

[20] Qianyu Zhou, Chuyun Zhuang, Xuequan Lu, and Lizhuang Ma. Domain adaptive semantic segmentation with regional contrastive consistency regularization. *arXiv preprint arXiv:2110.05170*, 2021. 1, 2

[21] Yang Zou, Zhiding Yu, BVK Kumar, and Jinsong Wang. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *Proceedings of the European conference on computer vision (ECCV)*, pages 289–305, 2018. 1, 2